Neural Computing and Applications
https://doi.org/10.1007/s00521-020-05163-4

ORIGINAL ARTICLE

Time series forecasting with feedforward neural networks trained using particle swarm optimizers for dynamic environments

Salihu A. Abdulkarim[1] · Andries P. Engelbrecht[2,3]

Received: 4 February 2019 / Accepted: 24 June 2020

Abstract
Several studies have applied particle swarm optimization (PSO) algorithms to train neural networks (NNs) for time series forecasting and the results indicated good performance. These studies, however, assumed static environments, making the PSO trained NNs unsuitable for forecasting many real-world time series which are generated by non-stationary processes. This study formulates training of a NN forecaster as a dynamic optimization problem, to investigate the application of a dynamic PSO algorithm to train NNs in forecasting time series in non-stationary environments. For this purpose, a set of experiments were conducted on three simulated and seven real-life time series forecasting problems under four different dynamic scenarios. Results obtained are compared to the results of NNs trained using a standard PSO and resilient backpropagation (Rprop). The results show that the NNs trained using dynamic PSO algorithms outperform the NNs trained using PSO and Rprop. These findings highlight the potential of using dynamic PSO in training NNs for real-world forecasting applications.

Keywords Time series forecasting · Neural networks · Particle swarm optimization · Cooperative quantum particle swarm optimization

1 Introduction

Forecasting is an essential task in dealing with uncertainties. Most decisions taken by businesses, governments, and people are based on forecasts. Generally, forecasting involves estimating or predicting future events (values) of a time based sequence of historical data (time series). The oldest and simplest method used in forecasting is human judgment and is often subject to errors. These errors may be within acceptable limits when the data to be predicted are well understood. When the time series problem is relatively more complex and less understood, other methods that require little or no human judgment become more appropriate. The time series problem becomes increasingly more complex in an environment with continuously shifting conditions.

Due to the importance of time series forecasting in many application areas, much research effort has gone into the development of forecasting models and in improving prediction accuracies. Statistical methods were the largely used time series forecasting tools [14]. Statistical methods are, however, known to have two main drawbacks, namely high mathematical complexity and dependence on prior knowledge of how the time series was generated [1]. Artificial neural networks (NNs), as an alternative approach to time series modeling, address these issues and have shown to be more efficient in terms of prediction accuracy [30, 33]. The performance of NNs, however, depends on the selection of the right network architecture and an appropriate training algorithm [62].

Particle swarm optimization (PSO) is now an established method for training NNs and was shown in several studies to outperform the classical back-propagation training algorithm [51, 52, 60]. Due to the successful application of PSO to train NNs in a number of problems,

✉ Salihu A. Abdulkarim
sakwami@fud.edu.ng

[1] Department of Computer Science, Federal University Dutse, Dutse, Nigeria

[2] Department of Industrial Engineering, Stellenbosch University, Stellenbosch 7600, South Africa

[3] Computer Science Division, Stellenbosch University, Stellenbosch 7600, South Africa

Published online: 07 July 2020

many researchers and analysts applied PSO in time series forecasting [21, 23, 40, 61]. PSO was also combined with other learning algorithms in order to improve forecasting accuracy. For example, [39] and [5] applied a hybrid of a genetic algorithm (GA) and a PSO to train a feedforward NN in predicting reservoir permeability and the power of solar stirling heat engines, respectively.

Forecasting accuracy of NNs trained using PSO was evaluated in several studies, such as in [4, 28], and the results showed that PSO outperformed backpropagation. [28], however, concluded that PSO was not able to track non-stationary data. In a separate study, PSO was also shown to outperform a GA for a load forecasting problem [22].

Even though several studies have applied PSO to the task of training NN forecasters, producing favorable results, these studies have assumed that the data generating process of the time series is constant. For many real-world time series, this assumption is often incorrect as shifting environmental conditions may cause the underlying data generating process to change. Thus, there is a need to improve the PSO-based NN forecater to effectively handle non-static environments.

Since PSO was developed for optimization in a stationary environment, it seems more appropriate to use PSO variants designed for solving problems in dynamic environments to train NN forecasters in non-static environments. These dynamic PSO algorithms address the problems of outdated memory and diversity loss experienced by PSO in dynamic environments [11, 12, 29].

Dynamic PSOs have been used to train NNs on classification problems under non-stationary environments [16, 44, 46]. These studies showed that dynamic PSOs are indeed applicable to NN training on classification problems in dynamic environments, by yielding significantly better or similar results compared to the classic NN training algorithm.

This study formulates training of a NN forecaster as a dynamic optimization problem to investigate the applicability and efficiency of a dynamic PSO algorithm as training algorithm for FNN forecasters under non-stationary environments, and to compare the performance against standard PSO and back-propagation (Rprop) algorithms. To the knowledge of the authors, this is the first study that applies dynamic PSO algorithms to train NN forecasters in non-static environments.

In the remainder of the paper, Sect. 2 presents the background information for the study. The experimental procedure is given in Sect. 3. Section 4 presents and discusses the results obtained. Section 5 summarizes the findings and concludes the paper.

2 Background

This section provides the necessary background information for the remainder of the paper. Section 2.1 briefly discusses NNs, Sect. 2.2 describes NN training using PSO, Sect. 2.3 discusses the challenges faced by PSO in dynamic environments, while Sect. 2.4 discusses a cooperative quantum PSO (CQSO) algorithm.

2.1 Neural network

NNs are powerful computational models inspired from structural/functional studies of the human central nervous system. Basically, a NN is a complex function for nonlinear mapping from a given input pattern to an output value. A traditional NN has an architecture composed of interconnected artificial neurons (nodes) organized in three layers. The layers of a NN are the input, hidden and output layers. A NN with enough hidden layer nodes is capable of approximating any nonlinear function [26]. The number of nodes in the hidden layer is problem dependent and must be optimized in order to avoid under/over fitting problems [18].

Even though NNs are capable of mapping a given input pattern to an output value, they are required to be trained to learn patterns contained in a data set for accurate approximation of the mapping. The objective of training a NN is to find an optimal set of weights (including biases) that minimizes the NN error. The training can be via a supervised, unsupervised or reinforcement process. This paper deals with supervised training only. Choosing an appropriate training algorithm is a very important component of NN implementation [38].

2.2 Training neural networks with particle swarm optimizers

PSO was first introduced by Eberhart and Kennedy in 1995 [17]. It is a population-based search algorithm based on the social dynamics of group interactions in bird flocks. In PSO, individuals called particles traverse a hyper-dimensional search space searching for an optimum position (solution). As they move around, each particle keeps track of the best solution it has found so far. The particles also have social contact with other particles in their neighborhood, where they can query for the best solution found within their neighborhood. The direction and step size taken by each particle as it travels in the search space are influenced by the best solution it found so far, the best solution found in its neighborhood, and the momentum (inertia). Through this simple social behavior, the collective effort results in finding an optimal solution in the high-

dimensional search space. See [18] for a detailed description of the algorithm.

To train a NN using PSO, particles are randomly initialized in an n-dimensional search space, where n is the total number of weights and biases of the NN. The position vector of each particle represents a solution (weight vector) of the NN, and the quality of each particle's position is determined by constructing the NN from the weight vector and calculating the sum square error (SSE) over the training set. Using this representation and fitness function, particles follow the PSO procedure described above to determine their direction and step size as they travel searching to find the best set of weights that minimizes the error function.

2.3 Particle swarm optimizers for dynamic environments

In spite of PSO's strength in solving stationary optimization problems, it faces two main challenges in dynamic environments [11, 12]:

- *Outdated memory* Any change in the environment renders the personal and global best positions of PSO invalid, thereby misleading the swarm in its search process to regions that contain poor solutions that may no longer be good solutions.
- *Loss of swarm diversity* As the swarm starts to converge, all particles are concentrated around one point in the search space. Any environmental change that causes the optimum to move outside of this region cannot be tracked due to lack in swarm diversity.

To address these issues, different PSO variants for dynamic environments were developed, such as the charged PSO [8], quantum PSO (QSO) [6], cooperative charged PSO [45] and CQSO [54]. For a comprehensive review of dynamic PSO algorithms, refer to [16, 29]. CQSO was employed in this work because empirical evidence has suggested its superiority over a number of other PSO variants for dynamic environments [54].

2.4 Cooperative quantum particle swarm optimization

The CQSO was conceptualized based on the principle of the cooperative split PSO [57], but using QSO in the subswarms. The two algorithms are briefly described below, and the CQSO algorithm is summarized in Algorithm 1.

2.4.1 Cooperative split particle swarm optimization

The cooperative split PSO partitions the search space dimension-wise into k disjoint groups where each group is assigned to an individual subswarm to optimize. The algorithm maintains a complete solution (or context) vector that individual subswarms uses to evaluate the quality of its particles. Quality is evaluated for each particle in a subswarm by substituting values of the dimensions the subswarm is accountable for into the context vector, while keeping constant the remaining values in the vector (which are the best values from other subswarms). The particle that, when substituted into the context vector, had the best quality is returned as the best solution in the subswarm. After all subswarms evaluate their particles' quality, the context vector contains the optimal solution discovered so far.

2.4.2 Quantum particle swarm optimization

The QSO algorithm [7] is inspired by the quantum model of an atom. A portion of the particles implement the standard PSO velocity and position updates. The rest of the particles, referred to as quantum particles, have their positions sampled from a probability distribution centered on the global best position. Positions of the quantum particles are calculated as,

$$\mathbf{x}_i \sim d(\hat{\mathbf{y}}, r_{cloud}) \tag{1}$$

where d is the probability distribution and r_{cloud} is the quantum radius. The control parameter, r_{cloud}, is problem and environment dependent [7, 16].

Recently, [25] proposed using a parent centric crossover (PCX) operator to generate the positions of quantum particles instead of using the radius and probability distribution parameters used in standard QSO. In an empirical study using single-peak dynamic environment types, the new variant (i.e., QSO-PCX) was shown to be superior to QSO on progressive and chaotic environments, while no superior approach on quasi-static and abrupt environments was found [25]. The performance of the QSO-PCX algorithm was, however, not evaluated on real-world and multi-peak environments.

Algorithm 1 Pseudo code for CQSO, assuming L particles per swarm

```
Create and randomly initialize k d-dimensional QSOs S_1 - S_K;
while stopping condition(s) is (are) not true do
  Select best particle b_1 - b_K from each of S_1 - S_K respectively
  for i ∈ [1...L], j ∈ [1...K] do
    Select p = i^th particle from swarm S_j
    Construct a vector w = (b_1, b_2, ..., p, ..., b_K)
    F(w) = error function at w
    Set fitness of particle i in swarm S_j to F(w)
    Update best fitness of b_1 - b_K if necessary
    Perform QSO update on S_1 - S_K
  end for
end while
```

3 Experimental procedure

This section describes the various data sets, algorithm setup, performance metrics and statistical methods used in the study. All algorithms used were implemented in the Computational Intelligence library (CIlib) version 0.9. CIlib is an open source collective project aimed at building a generic framework, designed to accommodate the implementation and execution of computational intelligence algorithms [42]. CIlib is available at http://github.com/cirg-up/cilib.

3.1 Datasets

A set of ten different well-studied time series with varying complexities were used in the study. Seven were real-world time series obtained online from the time series data library at http://robjhyndman.com/TSDL, and the other three were artificially generated. The time series studied are described in the subsections that follow.

3.1.1 Sunspot annual measure time series

The sunspot annual measure (SAM) dataset consists of 289 points representing the total annual measure of sunspots from 1770 to 1988. Illustrated in Fig. 1a, the SAM series has a strong seasonal pattern with somewhat constant trend. The SAM time series has been extensively studied in statistical literature and is often used as a standard for evaluating and comparing new forecasting methods. The data are known to be nonlinear, non-stationary and non-Gaussian [62].

3.1.2 International airline passengers time series

The international airline passengers (IAP) dataset has a total of 144 observations representing the monthly number of international airline passengers from January 1949 to December 1960. The IAP time series is a well-known dataset and was used in classical work by Box and Jenkins [9]. As shown in Fig. 1b, the series follows a multiplicative seasonal pattern with an upward trend and no outliers. The IAP dataset is non-stationary due to the presence of strong seasonal variation.

3.1.3 Australian wine sales time series

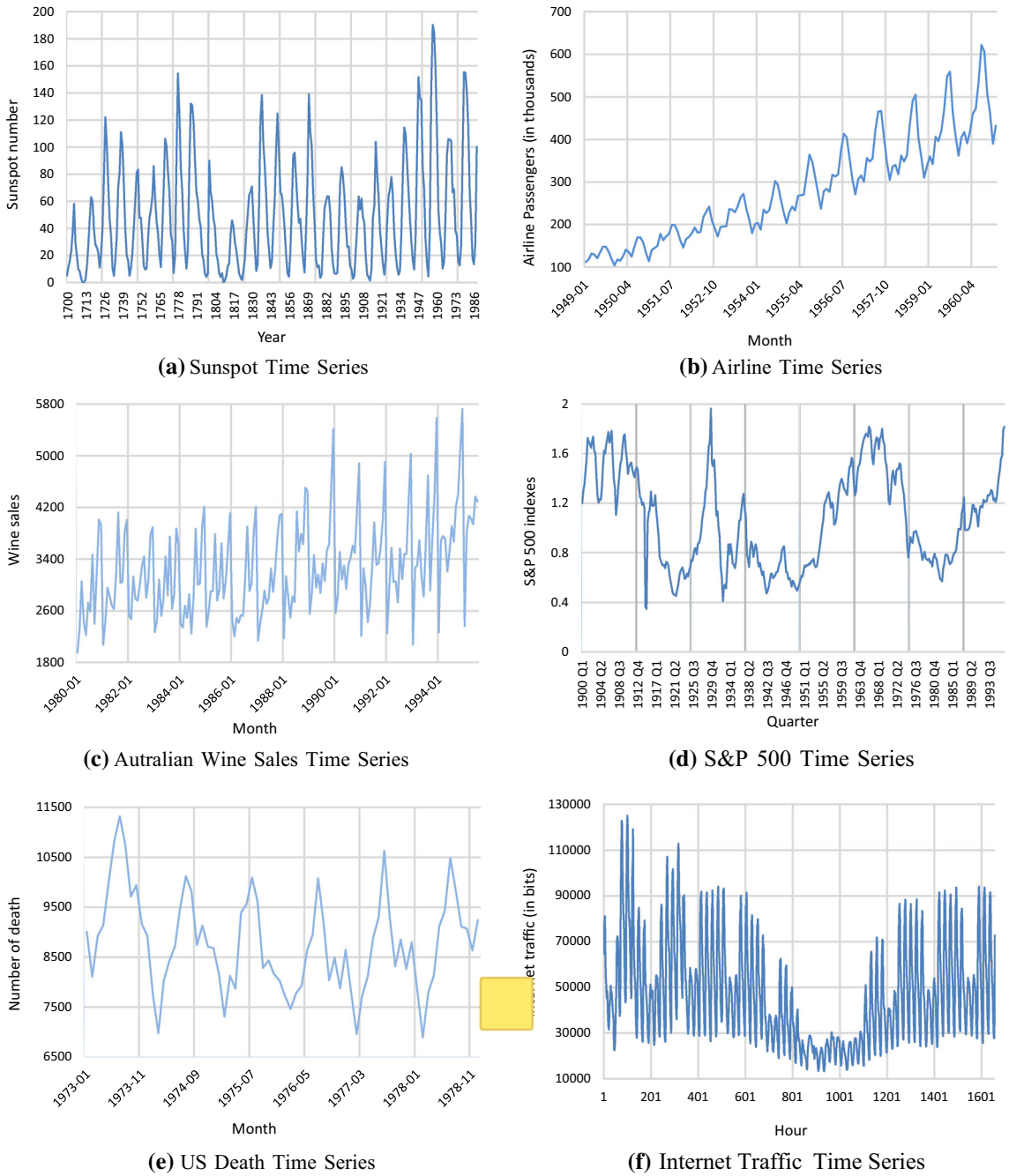
The Australian wine sales (AWS) series contains 187 observations of monthly wine sales (in thousands of liters) in Australia from January 1980 to July 1995. Illustrated in Fig. 1c, the AWS time series has a slightly increasing trend and a strong seasonal pattern (peaks in July, and troughs in January) with no obvious outliers. Another distinct feature of the series is increase in variability.

3.1.4 Standard and poor 500 indexes

The Standard and Poor (S&P) series has 388 values of historical quarterly S&P 500 indexes from 1990 to 1996. It is a very common financial time series used as a benchmark by many researchers in testing forecasting models [13, 43, 50]. A plot of the dataset is given in Fig. 1d, which reveals a constant trend with long-run cycles.

3.1.5 US death time series

The US death (USD) time series contains the total monthly deaths caused by accidents in the USA between January 1973 and December 1978, making a total of 72 observations. Shown in Fig. 1e, the USD series reveals a slightly constant trend with seasonal variations (peaks in January).



(a) Sunspot Time Series

(b) Airline Time Series

(c) Autralian Wine Sales Time Series

(d) S&P 500 Time Series

(e) US Death Time Series

(f) Internet Traffic Time Series

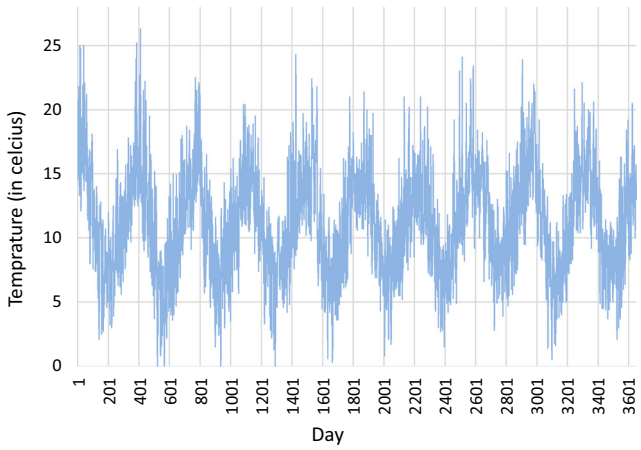
Fig. 1 Time plot of six datasets considered

3.1.6 Hourly internet traffic time series

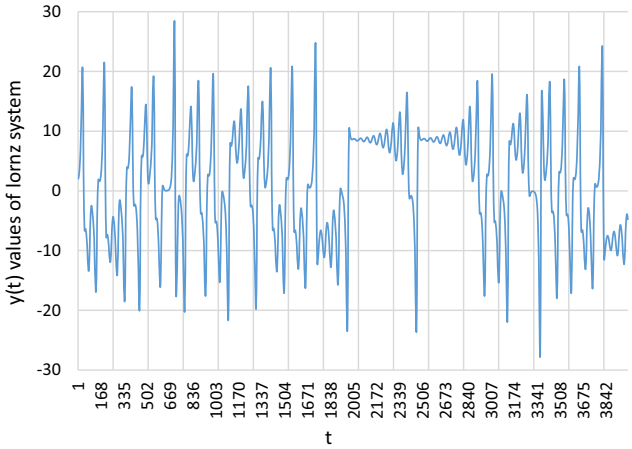
The hourly internet traffic (HIT) time series has 1657 data points representing aggregated hourly internet traffic data (in bits) from an internet service provider in the United Kingdom academic network backbone. It was collected between 19 November 2004, at 09:30 hours and 27 January 2005, at 11:11 hours. As shown in Fig. 1f, the HIT time series is non-stationary and with a slightly downward trend.

3.1.7 Daily minimum temperature time series

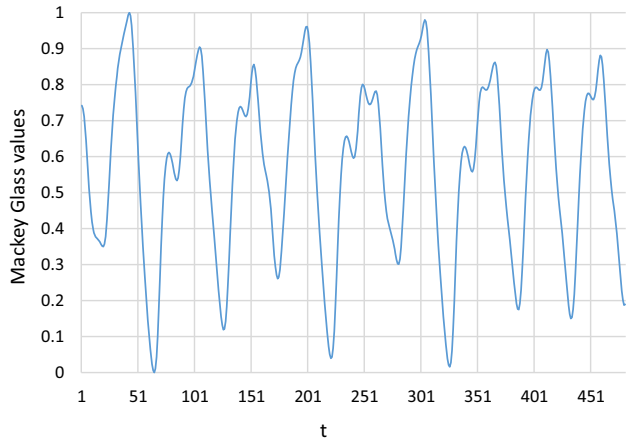
The daily minimum temperature (DMT) time series contains a total of 3650 observations representing daily minimum temperatures in Melbourne, Australia, 1981-1990. Figure 2a displays the DMT time series graphically. The series is highly noisy with a constant trend over the entire time span.



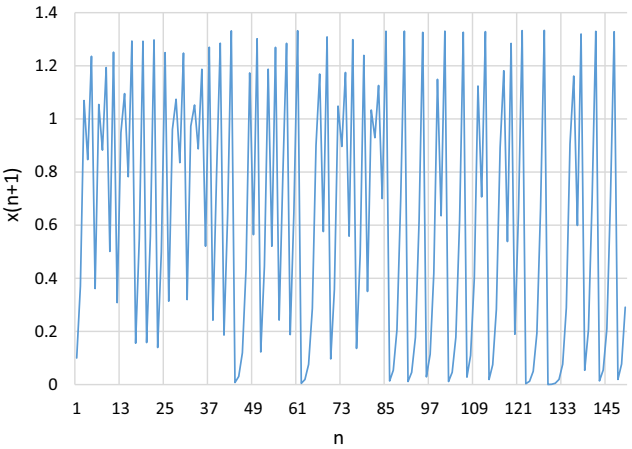
(a) Daily Temprature Time Series



(b) Lorenz Time Series



(c) Mackey Glass Time Series



(d) Logistic Map Time Series

Fig. 2 Time plot of four datasets considered

3.1.8 Mackay glass time series

The Mackay glass (MG) time series is a chaotic series generated from a solution of the Mackay-Glass delay-differential equation [32],

$$\frac{dx}{dt} = \frac{ax(t-\tau)}{1-x^c(t-\tau)} - bx(t) \tag{2}$$

using $\tau = 30, a = 0.2, b = 0.1, c = 10$, and initial condition $x(t) = 0.9$ for $0 \leq t \leq \tau$. A 500 points dataset was generated for this study, where 480 data points after the initial transients were used for training and testing. A plot of the MG series is given in Fig. 2c.

3.1.9 Lorenz time series

The Lorenz time series is a widely studied chaotic series generated from a Lorenz system. Described in the equations below, the Lorenz system is known to be a simplified model of various physical systems [35],

$$\begin{aligned} x(t+1) &= x(t) + K\sigma[y(t) - x(t)] \\ y(t+1) &= y(t) + K(x(t)[r - z(t)] - y(t)) \\ z(t+1) &= z(t) + K(x(t)y(t) - bz(t)) \end{aligned} \tag{3}$$

where $x(t), y(t), z(t)$ are states of the Lorenz system, r, σ, b are constant parameters of the system, and K is the sampling time. A total of 5000 samples were generated using

$$\begin{aligned} &\sigma = 10.500001, r = 28.200001, b = 2.700001, \\ &K = 0.0130001, x(0) = 1.200001, y(0) = 1.500001, \\ &z(0) = 1.600001 \end{aligned}$$

. The $y(t)$ state data were chosen for the study, and only the last 4000 samples were used to avoid a transient mode (i.e., prevailing influence of parameters used in generating the series). A plot of the time series is given in Fig. 2b.

3.1.10 Logistic map time series

The logistic map (LM) time series is a chaotic series generated from the logistic map equation [27],

$$x(t+1) = x(t) + Gx(t)(1 - x(t)) \tag{4}$$

For this study, data points were generated by iterating the equation 150 times starting from a random initial value set to 0.1 and $G = 3$. The equation behaves chaotically when G is set to 3. The LM time series as illustrated in Fig. 2d is chaotic, nonlinear, and non-stationary.

To keep the datasets within the active range of the activation functions used in the NNs, the datasets were scaled to [-1, 1]. For faster convergence as suggested in [34], the datasets are further normalized to set the mean of the training data close to zero. Normalization was done using,

$$x'_n = \frac{x_n}{\sqrt{N}} \tag{5}$$

where x_n is an observation, x'_n is the normalized observation, and N is the number of observations in the dataset. Each of the datasets was divided into two independent subsets. The first 80% of the dataset was used for training and the remaining 20% for testing. Since the work deals with time series, the datasets were not shuffled. For parameter optimization purposes only, the training subset was split further into a 70:30 ratio for training and validation, respectively.

For each dataset, performance was investigated under four different dynamic environmental scenarios. The scenarios were simulated using a sliding time window technique. This involves choosing a window of size w and a step value s for sliding the window over the dataset. The window is used to train the NNs for f number of iterations (i.e., change frequency) before the window slides over the dataset. The sliding process involves throwing away the s values at the beginning of the window and adding the next s values in the data series to the end of the window. The training and sliding process is repeated until all of the dataset is used. An example to illustrate this process with $w = 6$ and $s = 4$ is given in Fig. 3. When the window slides, four values, {X1, X2, X3, X4}, are discarded and new values, {X7, X8, X9, X10}, are added, while X5 and X6 remain in the window.

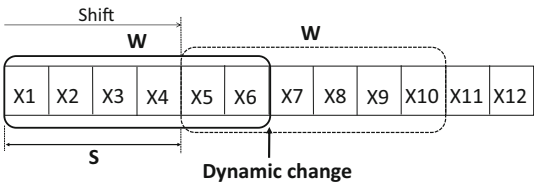


Fig. 3 Sliding time window

Table 1 Dynamic scenarios settings for each dataset

Dataset	Parameters	Scenario			
		I	II	III	IV
SAM	Window size	60	60	60	60
	Step size	10	20	40	60
	Change frequency	50	100	150	100
HIT	Window size	584	584	584	584
	Step size	100	250	500	584
	Change frequency	50	100	150	100
DMT	Window size	510	510	510	510
	Step size	100	200	400	510
	Change frequency	50	100	150	100
MG	Window size	84	84	84	84
	Step size	10	30	60	84
	Change frequency	50	100	200	100
Lorenz	Window size	330	330	330	330
	Step size	50	100	250	330
	Change frequency	50	100	150	100
IAP	Window size	32	32	32	32
	Step size	5	10	25	32
	Change frequency	50	100	150	100
AWS	Window size	42	42	42	42
	Step size	5	20	35	32
	Change frequency	50	100	150	100
S&P	Window size	58	58	58	58
	Step size	10	20	40	58
	Change frequency	50	100	150	100
USD	Window size	20	20	20	20
	Step size	2	8	16	20
	Change frequency	50	100	150	100
LM	Window size	31	31	31	31
	Step size	5	10	25	31
	Change frequency	50	100	50	100

The step size determines the spatial severity of the change. A small value for s implies a slight change, while a large value implies a drastic change. The number of iterations, f, an algorithm runs on a window before the window slides, controls the temporal severity. Table 1 presents the parameter setup used to simulate the dynamic scenarios for all the problems. As shown in the table, the combination of s and f differs under different scenarios and therefore requires different numbers of iterations to traverse the entire dataset, I_t, calculated as

$$I_t = f * \frac{N - w}{s} + f \tag{6}$$

where N is the total number of observations in the dataset.

3.2 Performance metric

This study uses the collective mean fitness (CMF) proposed by Morrison [41] as the performance measure for all experiments. The CMF reflects algorithm performance across the entire range of landscape dynamics and is given as

$$CMF = \sum_{t=1}^{T} \frac{F(t)}{T} \tag{7}$$

where $F(t)$ is a measure of quality of the solution and T is the total number of iterations. CMF is quite interesting for dynamic problems, because it captures an algorithm's entire performance history. This measure allows for convenient statistical comparison between algorithms [44]. The mean square error (MSE) calculated over the data set during each epoch was used as the algorithm fitness at each iteration.

The generalization factor ρ proposed in [49] was used to check the overfitting behavior of the algorithms used in this study. Overfitting is a phenomenon where a NN performs well on training data but poorly on generalization data. The ρ is defined as G_E/T_E, where G_E and T_E are the generalization and training errors, respectively. A $\rho < 1$ is an indication of good generalization performance, while $\rho > 1$ is an indication of overfitting.

3.3 Statistical methods

A two-tailed non-parametric Mann–Whitney U test [37] was used to determine whether the difference in performance between two algorithms is statistically significant or not. The choice of the significance test is based on [15], where the authors showed that the Mann–Whitney U test is safer than parametric tests such as the t-test, since the Mann–Whitney U test assumes neither normal distributions of data nor homogeneity of variance. The null hypothesis, $H_0 : \mu_1 = \mu_2$, where μ_1 and μ_2 are the means of the two samples being compared, was evaluated at a significance level of 95%. The alternative hypothesis was defined as $H_1 : \mu_1 \neq \mu_2$. Thus, any p value less than 0.05 corresponds to rejection of the null hypothesis that there was no statistically significant difference between the sample means. For the sake of convenience, all p values were bounded below by 0.0001.

3.4 Experimental setup

The study aimed to investigate the applicability of PSO algorithms designed to solve optimization problems in dynamic environments, as NN training algorithm in time series modeling. For this purpose, a set of experiments were conducted on the ten forecasting problems described in Sect. 3.1. Each experiment entails training a NN using a dynamic PSO, standard PSO, or Rprop algorithm under the four dynamic settings given in Table 1 to forecast one of the problems. For each problem, the performance of a NN trained with a dynamic PSO algorithm was compared with the performance of a NN trained with backpropagation and standard PSO. For effective evaluation of the performance, 30 independent runs for each experiment were carried out and averages over these 30 runs were computed. The number of iterations each algorithm ran was determined using equation (6).

These set of experiments were repeated under stationary environments (i.e., no sliding window technique used) to determine how these algorithms perform in a static environments.

A description of the NN architectures, training algorithms, and problem parameters used in the experiments are outlined below.

3.4.1 Neural networks setup

The NN architecture for each dataset was determined as follows:

- Input layer: For datasets collected annually, 10 input nodes were used, each representing a year in the decade. For monthly datasets, 12 input nodes were used, each representing a month of the year. For quarterly/weekly datasets, four input nodes were used, each representing a quarter of the year or a week of the month, respectively. For data collected hourly, 24 input nodes were used. This intuitive method was used by a number of analysts such as [2, 3, 24, 53], and has been effective in constructing optimal NN structures. For the synthetic datasets, the paper adopts the number of input nodes from previous studies as shown in Table 2.
- Hidden layer: A single hidden layer was used for all the NNs, and the number of hidden nodes was iteratively optimized on the training set, where discrete numbers in the range [2, 25] were considered. For every value within the range, 30 independent runs were conducted and the value that yielded the minimum average training and validation errors was chosen as optimal.

Table 2 Number of input nodes for synthetic datasets adopted from previous studies

Dataset	Input nodes	Researchers
MG	4	[10]
Lorenz	5	[19]
LM	3	[20]

For all the NNs, linear activation functions were used in the hidden layer nodes. Bounded functions were not used in the hidden layer because they lead to overfitting and non-convergent swarm behavior during NN training [47, 58].

- Output layer: A single-output node was used for all the NNs (one step ahead forecasting was considered). A modified hyperbolic tangent function was employed in the output layer nodes, defined as [34]

$$f(net) = 1.7159 tanh\left(\frac{2}{3} net\right) \tag{8}$$

where *net* is the weighted sum over all input signals.

All NN weights were randomly initialized in the range $[-\frac{1}{\sqrt{F}}, \frac{1}{\sqrt{F}}]$, where F is the number of incoming connections for a specific node. [59] has shown this to be a good initialization range.

For all the problems, the NN dimension was determined as

$$N_w = (I+1)J + (J+1)K \tag{9}$$

where N_w is the total number of NN weights (taking bias into account), I is the number of inputs, J is the number of hidden units, and K is the number of outputs.

3.4.2 Backpropagation setup

A variant of backpropagation algorithm, called resilient propagation (Rprop), was employed due to its computational simplicity and fast convergence rate. The key advantage of Rprop is that, for many problems, no user defined parameter is needed to obtain optimal convergence [48].

3.4.3 Particle swarm optimization setup

A linearly decreasing inertia weight was used for all the experiments, with an initial value of 0.9 and a final value of 0.5, while acceleration coefficients were fixed at $c_1 = c_2 = 1.49$ as suggested in [55]. The swarm size for each problem was determined as equal to the total number of particles used in CQSO to facilitate fair comparison. Velocity was not constrained and the Von Neumann topology was used since it facilitates diversity [31, 36], which is good for dynamic problems.

Table 3 PSO parameter ranges considered

Parameter	Range
Quantum radius, r	[0.2, 0.5, 0.8, 1, 2]
% of quantum particles	[10, 20, 30, 40, 50]
Number of dimensions per group, d	[4, 6, 8, 10,12]

3.4.4 Cooperative quantum particle swarm optimization

In addition to the PSO parameter setup given in Sect. 3.4.3, the radius of the quantum cloud and the percentage of quantum particles per swarm were iteratively selected from ranges given in Table 3, as suggested in [6].

For each dataset, the number of subswarms, k, in the CQSO was determined as the ratio $[N_w/d]$, where N_w is the total number of weights and biases in the NN and d is the number of weights grouped together. The value of the parameter d was iteratively optimized from a range of values given in Table 3. The size of each subswarm was set to 10 particles based on [56, 57].

4 Experimental results and discussion

Tables 4 and 5 summarize the CMF T_E and G_E with their corresponding confidence intervals in parenthesis obtained from the experiments. Minimum error values are displayed in bold. Also reported in the table is ρ. The ρ reported was calculated using to the CMF [Eq. (7)]. Thus, all reported values of ρ reflect the generalization factor across the entire algorithm run. Table 6 presents an overall algorithm ranking per forecasting problem based on the CMF T_E and G_E values, taking into account the p values listed in Table 7. In addition to these, a sample of 10 graphs are drawn to visualize the results obtained. The graphs illustrate the performance progression of the algorithms for the SAM problem under scenarios I to IV and for the AWS problem under scenario IV (see Fig. 4, 5).

The error values in Table 4 show that for static environment, Rprop outperformed both PSO and CQSO for all the ten problems. Comparing the performance of PSO to CQSO, PSO yielded the lowest errors in terms of both training and generalization for all the problems. All p values of pairwise comparisons of the algorithms' performance show significant difference in performance for all problems.

Performance of the algorithms on the problems considered under different dynamic environments is discussed below. Results from different problems are, however, discussed in groups, based on the similarity in performance.

1. Under all four dynamic scenarios considered, training with CQSO yielded the lowest CMF T_E and G_E, significantly outperforming both PSO and Rprop in forecasting the SAM, HIT, DMT, MG, Lorenz, and S&P problems. The CQSO is therefore ranked 1st for these problems. Figure 4 visually illustrates performance progression of the algorithms over time, obtained on the SAM problem during

Table 4 Results of the algorithms' performance in stationary environments: means over 30 samples reported with confidence interval in parenthesis

Problem	Rprop			PSO			CQSO		
	TE	GE	GF	TE	GE	GF	TE	GE	GF
IAP	**1.51E−04**	**1.74E−05**	0.12	1.53E−04	2.33E−05	0.15	3.61E−04	8.02E−05	0.22
	(8.20E−07)	**(1.24E−07)**	(0.10)	(2.69E−06)	(1.07E−06)	(0.40)	(3.53E−05)	(8.88E−06)	(0.25)
LM	**2.70E−03**	**2.33E−03**	0.86	2.70E−03	2.33E−03	0.86	2.70E−03	2.33E−03	0.86
	(2.86E−09)	**(1.20E−12)**	(0.00)	(3.06E−09)	(1.66E−12)	(0.00)	(1.94E−06)	(1.50E−07)	(0.08)
Lorenz	**5.06E−08**	**3.59E−08**	0.71	1.91E−06	1.52E−06	0.80	2.57E−06	2.08E−06	0.81
	(5.89E−09)	**(4.21E−09)**	(0.00)	(3.04E−07)	(2.47E−07)	(0.81)	(2.85E−07)	(2.40E−07)	(0.84)
MG	**9.80E−07**	**9.85E−07**	1.01	4.97E−06	5.15E−06	1.04	6.38E−05	6.73E−05	1.05
	(1.02E−07)	**(1.03E−07)**	(0.00)	(1.79E−06)	(1.86E−06)	(1.04)	(8.78E−06)	(9.79E−06)	(1.12)
S&P	**1.54E−05**	**3.85E−05**	2.50	1.68E−05	4.42E−05	2.63	6.28E−05	1.31E−04	2.09
	(9.90E−10)	**(1.44E−11)**	(0.00)	(3.17E−07)	(7.82E−07)	(2.47)	(1.18E−05)	(2.08E−05)	(1.76)
SAM	**1.50E−04**	**8.50E−05**	0.57	2.07E−04	1.13E−04	0.55	3.90E−04	1.80E−04	0.46
	(7.39E−08)	**(6.50E−10)**	(0.00)	(5.62E−06)	(2.49E−06)	(0.44)	(3.40E−05)	(8.74E−06)	(0.26)
DMT	**7.71E−06**	**9.44E−06**	1.22	9.41E−06	1.11E−05	1.18	7.13E−05	7.13E−05	1.00
	(2.99E−10)	**(3.42E−12)**	(0.00)	(1.59E−07)	(1.67E−07)	(1.05)	(6.28E−05)	(6.16E−05)	(0.98)
HIT	**1.71E−06**	**1.85E−06**	1.08	9.55E−06	9.30E−06	0.97	3.49E−05	3.14E−05	0.90
	(1.35E−08)	**(7.26E−09)**	(0.00)	(6.02E−07)	(6.66E−07)	(1.11)	(3.80E−06)	(3.49E−06)	(0.92)
USD	**5.27E−04**	**3.05E−04**	0.58	5.30E−04	3.27E−04	0.62	6.99E−04	4.29E−04	0.61
	(1.90E−08)	**(1.04E−11)**	(0.00)	(5.75E−06)	(5.12E−06)	(0.89)	(2.12E−05)	(1.15E−05)	(0.54)
AWS	**5.52E−04**	**1.90E−04**	0.34	7.28E−04	2.13E−04	0.29	1.23E−03	4.19E−04	0.34
	(1.39E−07)	**(1.52E−11)**	(0.00)	(3.37E−05)	(6.11E−06)	(0.18)	(8.02E−05)	(1.83E−05)	(0.23)

training and generalization. As visualized in the figures, CQSO took about ten algorithm iterations to locate a minimum under all scenarios, and once it did, it constantly tracked the minimum and successfully recovered from changes throughout the experiment. The Rprop and PSO algorithms took about 100 algorithm iterations to locate a minimum.

Comparing the performance of PSO to Rprop, PSO produced significantly better results both on T_E and G_E under scenarios I to IV in forecasting the HIT, DMT, MG and Lorenz time series. Therefore, PSO was ranked 2nd and Rprop 3rd for these four problems. For the S&P problem, Rprop yielded lower errors than PSO under all scenarios. However, under scenario IV (where there is an abrupt change) the difference in performance was not significant. Thus, Rprop was ranked 2nd and PSO 3rd. For the SAM problem, Rprop significantly outperformed PSO under scenarios I and II, while PSO outperformed Rprop under scenarios III and IV. This implies that Rprop performed better than PSO when the change was slight and gradual, and worse when the change was severe and abrupt. Thus, there is no overall winner between Rprop and PSO across the four scenarios, as they had the same average G_E ranking.

2. In forecasting the IAP time series, CQSO yielded the lowest G_E across all four dynamic scenarios, significantly outperforming the other algorithms. Thus, in terms of G_E, the CQSO, Rprop and PSO were ranked 1st, 2nd and 3rd, respectively. In terms of T_E, Rprop produced the lowest error under scenario I, while CQSO had the lowest error under scenario IV. This indicates that CQSO trains better when the change is severe and abrupt. Under scenarios II and III, the difference in performance between Rprop and CQSO was not significant. Hence, Rprop and CQSO achieved equal ranking in terms of T_E.

3. In predicting the USD series, CQSO yielded significantly the lowest G_E under all scenarios, with the exception of scenario I where the difference was not significant compared to Rprop. Thus, CQSO is the overall winner in terms of G_E, outperforming the other two algorithms, with PSO ranked 2nd and Rprop 3rd. Considering the training performance result, Rprop produced the lowest error under scenarios I and II, and while CQSO had the lowest error under scenarios III and IV, the difference in performance with Rprop was not significant. Therefore, Rprop was ranked 1st, with CQSO and PSO as 2nd and 3rd respectively.

Table 5 Results of the algorithms performance in non-stationary environments: means over 30 samples reported with confidence interval in parenthesis

Problem	Algorithm	Scenario I			Scenario II			Scenario III			Scenario IV		
		T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
SAM	Rprop	1.88E−04	2.17E−04	1.17	1.77E−04	2.05E−04	1.18	2.17E−04	2.73E−04	1.29	3.87E−04	3.92E−04	1.02
		(2.53E−05)	(2.39E−05)	(0.02)	(2.49E−05)	(2.36E−05)	(0.02)	(3.45E−05)	(3.32E−05)	(0.03)	(5.56E−05)	(5.42E−05)	(0.02)
	PSO	2.46E−04	3.27E−04	1.33	2.37E−04	2.78E−04	1.17	2.77E−04	1.44E−04	0.52	2.62E−04	2.26E−04	0.86
		(9.07E−06)	(1.80E−05)	(0.05)	(7.00E−06)	(2.00E−05)	(0.07)	(1.17E−05)	(1.11E−05)	(0.03)	(9.74E−06)	(1.51E−05)	(0.04)
	CQSO	**1.37E−04**	**1.61E−04**	1.18	**1.32E−04**	**1.19E−04**	0.91	**1.44E−04**	**8.09E−05**	0.56	**1.29E−04**	**9.36E−05**	0.73
		(3.74E−06)	**(5.17E−06)**	(0.01)	**(2.76E−06)**	**(2.18E−06)**	(0.00)	**(2.01E−06)**	**(6.72E−07)**	(0.01)	**(1.97E−06)**	**(1.56E−06)**	(0.00)
HIT	Rprop	2.42E−04	2.41E−04	1.00	1.88E−04	1.84E−04	0.99	3.52E−04	3.49E−04	0.99	6.06E−04	6.06E−04	1.00
		(8.36E−05)	(8.33E−05)	(0.01)	(4.78E−05)	(4.59E−05)	(0.01)	(1.10E−04)	(1.09E−04)	(0.01)	(1.94E−04)	(1.95E−04)	(0.01)
	PSO	2.83E−05	2.78E−05	0.99	3.11E−05	2.70E−05	0.86	3.37E−05	2.91E−05	0.86	5.24E−05	4.69E−05	0.89
		(4.38E−06)	(4.36E−06)	(0.02)	(5.90E−06)	(5.44E−06)	(0.02)	(2.84E−06)	(2.95E−06)	(0.02)	(5.85E−06)	(5.63E−06)	(0.02)
	CQSO	**4.59E−06**	**5.62E−06**	1.22	**5.01E−06**	**4.67E−06**	0.93	**5.58E−06**	**5.84E−06**	1.05	**6.75E−06**	**6.38E−06**	0.95
		(2.25E−07)	**(3.05E−07)**	(0.01)	**(2.11E−07)**	**(1.86E−07)**	(0.00)	**(2.07E−07)**	**(2.01E−07)**	(0.01)	**(5.10E−07)**	**(4.26E−07)**	(0.01)
DMT	Rprop	4.28E−04	4.27E−04	1.00	1.46E−04	1.47E−04	1.01	8.54E−05	8.77E−05	1.04	2.76E−04	2.75E−04	1.00
		(9.89E−05)	(9.92E−05)	(0.01)	(3.82E−05)	(3.80E−05)	(0.01)	(2.07E−05)	(2.07E−05)	(0.01)	(9.55E−05)	(9.49E−05)	(0.01)
	PSO	2.02E−05	2.06E−05	1.02	1.59E−05	1.61E−05	1.02	1.75E−05	2.00E−05	1.16	2.01E−05	2.06E−05	1.03
		(2.52E−06)	(2.60E−06)	(0.02)	(2.89E−06)	(2.62E−06)	(0.01)	(2.94E−06)	(2.91E−06)	(0.02	(2.45E−06)	(2.46E−06)	(0.02)
	CQSO	**1.14E−05**	**1.22E−05**	1.07	**1.10E−05**	**1.17E−05**	1.06	**1.07E−05**	**1.28E−05**	1.20	**1.09E−05**	**1.19E−05**	1.10
		(1.60E−07)	**(2.05E−07)**	(0.01)	**(1.29E−07)**	**(1.39E−07)**	(0.00)	**(1.29E−07)**	**(1.62E−07)**	(0.00)	**(1.60E−07)**	**(2.10E−07)**	(0.00)
MG	Rprop	9.17E−05	8.93E−05	0.97	2.55E−04	2.46E−04	0.96	1.09E−04	1.05E−04	0.96	2.84E−04	2.73E−04	0.96
		(9.53E−06)	(9.21E−06)	(0.01)	(2.58E−05)	(2.64E−05)	(0.01)	(9.43E−06)	(9.53E−06)	(0.02)	(3.01E−05)	(2.97E−05)	(0.01)
	PSO	3.13E−05	1.39E−05	0.42	3.89E−05	1.88E−05	0.48	3.82E−05	2.24E−05	0.58	8.47E−05	3.97E−05	0.47
		(3.73E−06)	(2.91E−06)	(0.05)	(3.93E−06)	(2.60E−06)	(0.04)	(3.68E−06)	(2.78E−06)	(0.05)	(6.22E−06)	(3.80E−06)	(0.03)
	CQSO	**3.74E−06**	**3.02E−06**	0.81	**4.22E−06**	**2.74E−06**	0.65	**4.01E−06**	**3.12E−06**	0.78	**5.93E−06**	**5.22E−06**	0.89
		(1.96E−07)	**(1.26E−07)**	(0.02)	**(2.65E−07)**	**(1.53E−07)**	(0.02)	**(1.98E−07)**	**(1.23E−07)**	(0.02)	**(5.81E−07)**	**(4.58E−07)**	(0.02)
Lorenz	Rprop	1.69E−04	1.69E−04	1.00	6.71E−05	6.72E−05	1.00	7.13E−05	7.09E−05	1.00	1.97E−04	1.97E−04	1.00
		(3.20E−05)	(3.17E−05)	(0.01)	(1.28E−05)	(1.30E−05)	(0.01)	(8.60E−06)	(8.61E−06)	(0.01)	(3.26E−05)	(3.32E−05)	(0.01)
	PSO	2.00E−05	1.88E−05	0.91	1.03E−05	8.73E−06	0.80	1.58E−05	1.46E−05	0.90	2.94E−05	2.92E−05	0.94
		(5.53E−06)	(5.76E−06)	(0.03)	(3.20E−06)	(3.06E−06)	(0.06)	(4.96E−06)	(4.77E−06)	(0.05)	(1.06E−05)	(1.17E−05)	(0.06)
	CQSO	**1.78E−06**	**1.26E−06**	0.70	**8.40E−07**	**4.47E−07**	0.53	**1.02E−06**	**6.87E−07**	0.68	**2.08E−06**	**1.53E−06**	0.74
		(1.93E−07)	**(1.44E−07)**	(0.02)	**(1.60E−07)**	**(8.65E−08)**	(0.02)	**(1.31E−07)**	**(8.21E−08)**	(0.01)	**(2.32E−07)**	**(1.61E−07)**	(0.03)

Table 5 (continued)

Problem	Algorithm	Scenario I			Scenario II			Scenario III			Scenario IV		
		T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ	T_E	G_E	ρ
S&P	Rprop	1.96E−04	2.99E−04	1.56	1.91E−04	3.31E−04	1.76	2.49E−04	3.76E−04	1.54	4.26E−04	5.53E−04	1.32
		(1.54E−05)	(1.59E−05)	(0.07)	(1.47E−05)	(1.80E−05)	(0.07)	(2.29E−05)	(2.79E−05)	(0.06)	(4.09E−05)	(4.51E−05)	(0.04)
	PSO	5.88E−04	7.55E−04	1.34	6.06E−04	7.68E−04	1.29	4.74E−04	7.38E−04	1.56	4.63E−04	6.00E−04	1.66
		(9.24E−05)	(1.04E−04)	(0.05)	(7.21E−05)	(8.13E−05)	(0.03)	(3.96E−05)	(5.61E−05)	(0.02)	(9.37E−05)	(9.00E−05)	(0.31)
	CQSO	**7.60E−05**	**1.91E−04**	2.58	**8.82E−05**	**2.21E−04**	2.66	**1.08E−04**	**2.45E−04**	2.39	**6.91E−05**	**1.75E−04**	3.67
		(6.88E−06)	**(1.08E−05)**	(0.11)	**(1.09E−05)**	**(1.31E−05)**	(0.18)	**(2.24E−05)**	**(3.01E−05)**	(0.11)	**(1.75E−05)**	**(8.25E−06)**	(0.66)
IAP	Rprop	**9.65E−05**	1.26E−04	1.33	9.80E−05	1.36E−04	1.53	**1.41E−04**	2.11E−04	1.71	2.61E−04	3.58E−04	1.52
		(1.04E−05)	(1.05E−05)	(0.05)	(2.27E−05)	(2.10E−05)	(0.11)	**(3.53E−05)**	(3.12E−05)	(0.17)	(5.72E−05)	(5.60E−05)	(0.12)
	PSO	2.19E−03	2.32E−03	1.07	1.44E−03	1.57E−03	1.08	9.54E−04	1.33E−03	1.45	1.16E−03	2.33E−03	2.02
		(5.08E−04)	(5.30E−04)	(0.01)	(2.03E−04)	(2.30E−04)	(0.02)	(1.76E−04)	(2.41E−04)	(0.09)	(1.56E−04)	(2.88E−04)	(0.03)
	CQSO	1.02E−04	**1.21E−04**	1.22	**8.84E−05**	**9.78E−05**	1.08	1.43E−04	**1.61E−04**	1.14	**8.91E−05**	**1.51E−04**	1.59
		(4.22E−05)	**(4.62E−05)**	(0.03)	**(2.50E−05)**	**(2.97E−05)**	(0.06)	(4.59E−05)	**(5.49E−05)**	(0.08)	**(2.04E−05)**	**(4.75E−05)**	(0.18)
AWS	Rprop	**2.79E−04**	**4.72E−04**	1.69	**3.85E−04**	**5.44E−04**	1.42	3.89E−04	7.29E−04	1.89	4.95E−04	8.67E−04	1.77
		(9.25E−06)	**(9.70E−06)**	(0.02)	**(1.98E−05)**	**(1.95E−05)**	(0.03)	(1.89E−05)	(1.70E−05)	(0.05)	(3.02E−05)	(3.18E−05)	(0.04)
	PSO	5.58E−04	6.90E−04	1.24	5.45E−04	9.02E−04	1.65	6.48E−04	6.75E−04	1.04	6.99E−04	1.15E−03	1.64
		(1.92E−05)	(2.57E−05)	(0.02)	(1.49E−05)	(4.70E−05)	(0.07)	(2.81E−05)	(4.24E−05)	(0.04)	(3.57E−05)	(8.83E−05)	(0.06)
	CQSO	3.78E−04	5.56E−04	1.46	4.07E−04	7.05E−04	1.73	**3.51E−04**	**6.10E−04**	1.73	**4.10E−04**	**8.35E−04**	2.02
		(1.66E−05)	(3.58E−05)	(0.05)	(1.88E−05)	(4.08E−05)	(0.02)	**(1.40E−05)**	**(2.89E−05)**	(0.02)	**(2.22E−05)**	**(5.73E−05)**	(0.03)
USD	Rprop	**1.88E−04**	1.16E−03	7.33	**3.18E−04**	1.19E−03	4.25	4.60E−04	9.25E−04	2.22	5.64E−04	1.19E−03	2.69
		(3.87E−05)	(8.27E−05)	(0.91)	**(5.05E−05)**	(5.67E−05)	(0.46)	(8.59E−05)	(9.42E−05)	(0.17)	(1.43E−04)	(1.37E−04)	(0.34)
	PSO	5.64E−04	7.92E−04	1.47	6.02E−04	1.01E−03	1.71	5.45E−04	8.79E−04	1.61	6.42E−04	1.26E−03	1.97
		(5.92E−05)	(5.54E−05)	(0.12)	(4.50E−05)	(7.57E−05)	(0.11)	(2.34E−05)	(4.22E−05)	(0.03)	(2.96E−05)	(5.19E−05)	(0.07)
	CQSO	3.57E−04	**5.98E−04**	1.81	4.61E−04	**9.05E−04**	1.98	**3.84E−04**	**6.15E−04**	1.61	**4.97E−04**	**1.05E−03**	2.13
		(5.03E−05)	**(2.66E−05)**	(0.15)	(1.92E−05)	**(5.39E−05)**	(0.12)	**(1.80E−05)**	**(2.40E−05)**	(0.03)	**(2.47E−05)**	**(2.81E−05)**	(0.09)
LM	Rprop	**2.11E−03**	3.73E−03	1.77	**2.06E−03**	3.72E−03	1.81	2.58E−03	2.98E−03	1.16	2.73E−03	2.68E−03	0.98
		(2.15E−05)	(2.33E−05)	(0.01)	**(2.06E−05)**	(2.06E−05)	(0.01)	(3.86E−05)	(3.98E−05)	(0.00)	(5.68E−05)	(5.65E−05)	(0.00)
	PSO	2.99E−03	**2.81E−03**	0.94	2.75E−03	**3.58E−03**	1.31	2.39E−03	**2.53E−03**	1.06	**2.39E−03**	**2.23E−03**	0.93
		(7.44E−05)	**(4.77E−05)**	(0.02)	(8.53E−05)	**(2.67E−05)**	(0.04)	(5.24E−06)	**(1.11E−05)**	(0.00)	**(1.30E−05)**	**(1.89E−05)**	(0.01)
	CQSO	2.89E−03	2.85E−03	1.00	2.39E−03	3.78E−03	1.60	**2.38E−03**	2.54E−03	1.07	2.41E−03	2.26E−03	0.94
		(1.12E−04)	(9.34E−05)	(0.06)	(8.84E−05)	(4.56E−05)	(0.06)	**(5.57E−06)**	(1.15E−05)	(0.01)	(1.14E−05)	(1.77E−05)	(0.01)

Table 6 Algorithm ranking for scenario I to IV

Time series	Algorithm	Scenario I		Scenario II		Scenario III		Scenario IV		Average error	
		$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$	$R(T_E)$	$R(G_E)$
SAM	Rprop	2	2	2	2	2	3	3	3	2.25	2.5
	PSO	3	3	3	3	3	2	2	2	2.75	2.5
	CQSO	1	1	1	1	1	1	1	1	1	1
HIT	Rprop	3	3	3	3	3	3	3	3	3	3
	PSO	2	2	2	2	2	2	2	2	2	2
	CQSO	1	1	1	1	1	1	1	1	1	1
DMT	Rprop	3	3	3	3	3	3	3	3	3	3
	PSO	2	2	2	2	2	2	2	2	2	2
	CQSO	1	1	1	1	1	1	1	1	1	1
MG	Rprop	3	3	3	3	3	3	3	3	3	3
	PSO	2	2	2	2	2	2	2	2	2	2
	CQSO	1	1	1	1	1	1	1	1	1	1
Lorenz	Rprop	3	3	3	3	3	3	3	3	3	3
	PSO	2	2	2	2	2	2	2	2	2	2
	CQSO	1	1	1	1	1	1	1	1	1	1
S&P	Rprop	2	2	2	2	2	2	2.5	2.5	2.13	2.13
	PSO	3	3	3	3	3	3	2.5	2.5	2.88	2.88
	CQSO	1	1	1	1	1	1	1	1	1	1
IAP	Rprop	1	2	1.5	2	1.5	2	2	2	1.5	2
	PSO	3	3	3	3	3	3	3	3	3	3
	CQSO	2	1	1.5	1	1.5	1	1	1	1.5	1
AWS	Rprop	1	1	1.5	1	2	2.5	2	1.5	1.62	1.5
	PSO	3	3	3	3	3	2.5	3	3	3	2.88
	CQSO	2	2	1.5	2	1	1	1	1.5	1.38	1.63
USD	Rprop	1	3	1	3	1.5	2.5	1.5	1.5	1.25	2.5
	PSO	3	2	3	2	3	2.5	3	3	3	2.38
	CQSO	2	1	2	1	1.5	1	1.5	1.5	1.75	1.13
LM	Rprop	1	3	1	2	3	3	3	3	2	2.75
	PSO	2.5	1.5	3	1	2	1.5	1.5	1	2.25	1.25
	CQSO	2.5	1.5	2	3	1	1.5	1.5	2	1.75	2
Overall rank	Rprop	1.91	2.55	2.00	2.36	2.45	2.73	2.64	2.59		
	PSO	2.55	2.27	2.64	2.18	2.45	2.18	2.23	2.14		
	CQSO	**1.50**	**1.10**	**1.35**	**1.40**	**1.10**	**1.10**	**1.15**	**1.25**		

Bold values indicate key values of interest

4. For the AWS problem, Rprop produced the lowest T_E and G_E under scenarios I and II, while CQSO had the lowest errors under scenarios III and IV as shown in Table 5. PSO yielded the worst results across all scenarios. Overall, CQSO is ranked first in terms of T_E, while Rprop takes the lead in terms of G_E. Figure 5 shows the progression of the performance of the algorithms over time under scenario IV.

5. A slightly different result was obtained in forecasting the LM problem compared to the other nine problems studied. PSO yielded the lowest G_E under all four scenarios. However, the difference was not significant compared to CQSO under scenarios I and III. PSO emerged as the overall winner in terms of generalization. With respect to T_E, CQSO was the overall winner, with Rprop and PSO ranked as 2nd and 3rd, respectively.

In the overall performance ranking of the algorithms shown in Table 6, the CQSO algorithm was ranked 1st in terms of T_E and G_E under all four dynamic scenarios. The PSO algorithm was ranked 2nd in terms of G_E under the four dynamic scenarios. In terms of T_E, Rprop was ranked 2nd under scenarios I and II, while PSO was ranked 2nd under scenario IV.

Table 7 Mann–Whitney U p values obtained for the average training and generalization errors comparisons with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%

Problem	Algorithm	Training				Generalization			
		Senario				Senario			
		I	II	III	IV	I	II	III	IV
SAM	Rprop versus PSO	0.0001	0.0001	0.0001	0.0005	0.0001	0.0001	0.0001	0.0001
	Rprop versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	PSO versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
HIT	Rprop versus PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	Rprop versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	PSO versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
DMT	Rprop versus PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	Rprop versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	PSO versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
MG	Rprop versus PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	Rprop versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	PSO versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
Lorenz	Rprop versus PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	Rprop versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	PSO versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
S&P	Rprop versus PSO	0.0001	0.0001	0.0001	**0.1646**	0.0001	0.0001	0.0001	**0.3835**
	Rprop versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	PSO versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
IAP	Rprop versus PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	Rprop versus CQSO	0.0054	**0.1984**	**0.3750**	0.0001	0.0020	0.0002	0.0002	0.0001
	PSO versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
AWS	Rprop versus PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	**0.0537**	0.0001
	Rprop versus CQSO	0.0001	**0.1103**	0.0030	0.0001	0.0001	0.0001	0.0001	**0.1515**
	PSO versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0173	0.0001
USD	Rprop versus PSO	0.0001	0.0001	0.0047	0.0256	0.0001	0.0001	**0.6744**	0.0134
	Rprop versus CQSO	0.0001	0.0001	**0.9411**	**0.1515**	0.0001	0.0001	0.0001	**0.9058**
	PSO versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0139	0.0001	0.0001
LM	Rprop versus PSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	Rprop versus CQSO	0.0001	0.0001	0.0001	0.0001	0.0001	0.0068	0.0001	0.0001
	PSO versus CQSO	**0.2142**	0.0001	0.0371	**0.1039**	**0.9876**	0.0001	**0.2495**	0.0415

Bold values indicate key values of interest

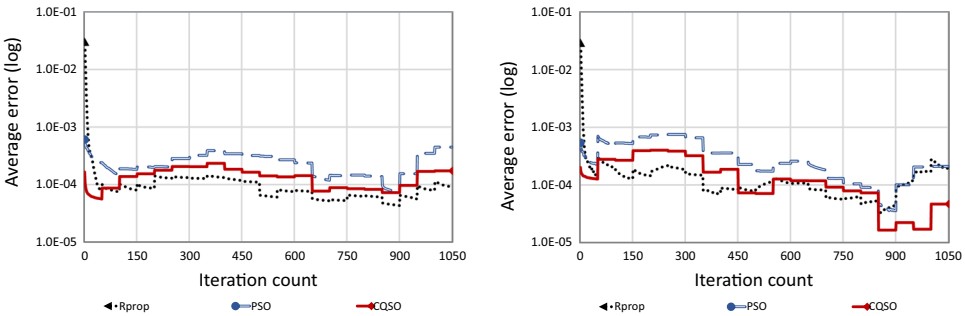
The CQSO benefited from the component-wise optimization for all the problems considered, as evident under scenario I (where change was gradual and not quite significant) by outperforming PSO which ordinarily can handle such slight changes.

The problems with more than 400 time series points indicated no sign of overfitting for all the algorithms studied under the four dynamic scenarios, while those with less than 400 points indicated some sign of overfitting. The USD problem which has only 72 data points had the worst overfitting performance (even though the number of hidden nodes was optimized).

5 Conclusion

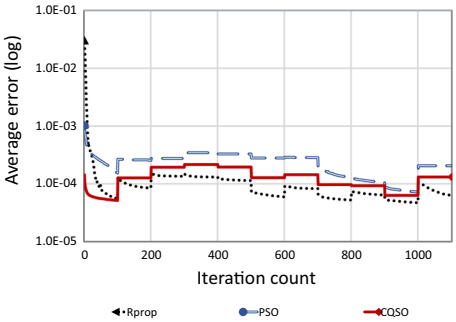
The aim of the paper was to investigate the applicability and efficiency of a dynamic PSO algorithm in training NN forecasters under non-static environments, and to compare the performance against standard PSO and backpropagation (Rprop).

Experiments were conducted by training a NN using the algorithms investigated to forecast ten problems under four different dynamic scenarios. Analysis of the results obtained indicates that dynamic PSO is quite applicable in training a NN time series forecaster in dynamic environments. Overall ranking of the algorithms on all the

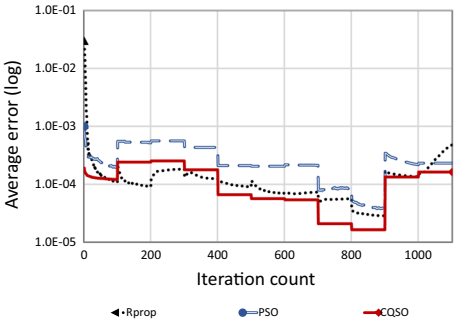


(a) Scenario I: Training Error

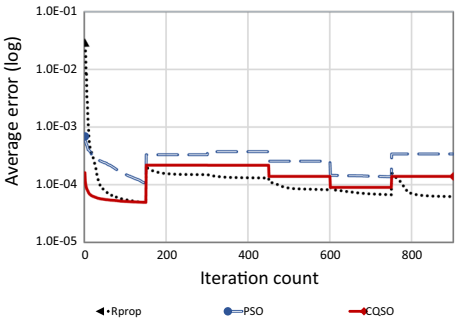
(b) Scenario I: Generalisation Error



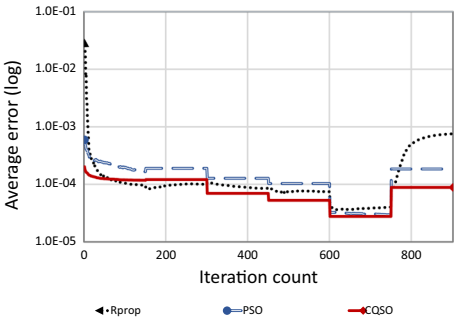
(c) Scenario II: Training Error



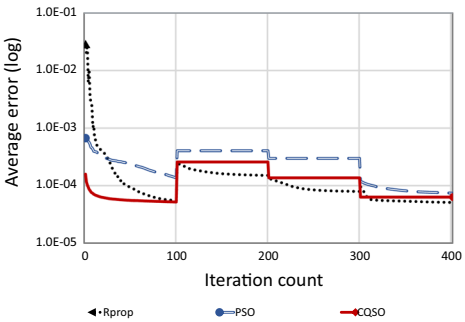
(d) Scenario II: Generalisation Error



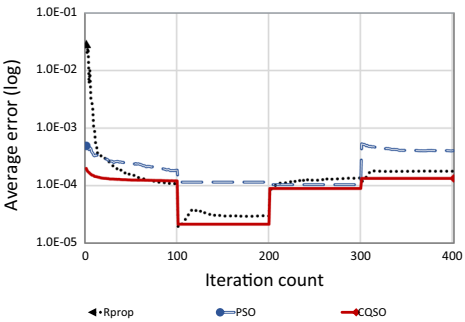
(e) Scenario III: Training Error



(f) Scenario III: Generalisation Error



(g) Scenario IV: Training Error

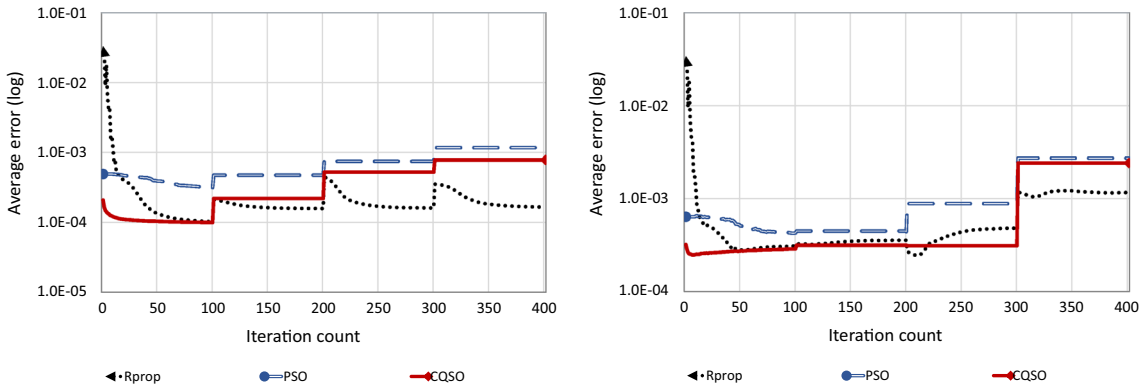


(h) Scenario IV: Generalisation Error

Fig. 4 Training and generalization error results for SAM under scenarios I to IV

problems studied showed that the dynamic PSO (specifically CQSO) outperformed both the standard PSO and Rprop algorithms in terms of T_E and G_E under all four dynamic scenarios.

Further studies will include a detailed study of dynamic CPSO for variable dependencies applied to different recurrent neural networks. Other properties of training algorithms not explored in this paper, e.g., recovery speed after a change, will be investigated.



(a) Scenario IV: Training Error

(b) Scenario IV: Generalisation Error

Fig. 5 Training and generalization error results for AWS under scenario IV

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interest.

References

1. Abdulkarim S, Garko A (2016) Effectiveness of firefly algorithm based neural network in time series forecasting. Bayero J Pure Appl Sci 9(1):6–10
2. Abdulkarim SA (2018) Time series forecasting using dynamic particle swarm optimizer trained neural networks. Phd thesis, University of Pretoria
3. Abdulkarim SA, Engelbrecht AP (2019) Time series forecasting using neural networks: Are recurrent connections necessary? Neural Process Lett 50:2763–2795
4. Adhikari R, Agrawal RK (2011) Effectiveness of PSO based neural network for seasonal time series forecasting. In: Proceedings of the fifth Indian international conference on artificial intelligence, pp 231–244
5. Ahmadi MH, Aghaj SSG, Nazeri A (2013) Prediction of power in solar stirling heat engine by using neural network based on hybrid genetic algorithm and particle swarm optimization. Neural Comput Appl 22(6):1141–1150
6. Blackwell T, Branke J (2006) Multiswarms, exclusion, and anti-convergence in dynamic environments. IEEE Trans Evol Comput 10(4):459–472
7. Blackwell T, Branke J, Li X (2008) Particle swarms for dynamic optimization problems. Swarm intelligence. Springer, Berlin pp 193–217
8. Blackwell TM, Bentley PJ (2002) Dynamic search with charged swarms. In: Proceedings of the genetic and evolutionary computation conference, pp 9–16
9. Box GEP, Jenkins GM, Reinsel GC, Ljung GM (2015) Time series analysis: forecasting and control. Wiley, New York
10. Brezak D, Bacek T, Majetic D, Kasac J, Novakovic B (2012) A comparison of feed-forward and recurrent neural networks in time series forecasting. In: Proceedings of IEEE conference on computational intelligence for financial engineering and economics, pp 1–6
11. Carlisle A, Dozier G (2000) Adapting particle swarm optimization to dynamic environments. Int Conf Artif Intell 1:429–434
12. Carlisle A, Dozler G (2002) Tracking changing extrema with adaptive particle swarm optimizer. In: Automation congress, 2002 proceedings of the 5th biannual world, vol 13. IEEE, pp 265–270
13. Chatfield C (2016) The analysis of time series: an introduction. CRC Press, Boca Raton
14. Deb C, Zhang F, Yang J, Lee SE, Shah KW (2017) A review on time series forecasting techniques for building energy consumption. Renew Sustain Energy Rev 74:902–924
15. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7(Jan):1–30
16. Duhain JGOL (2011) Particle swarm optimization in dynamically changing environment an empirical study. MSc Thesis, University of Pretoria
17. Eberhart RC, Kennedy J (1995) Particles swarm optimization. In: Proceedings of IEEE international conference on neural networks, Perth, Australia, pp 1942–1948
18. Engelbrecht AP (2007) Computational intelligence: an introduction. Wiley, New York
19. Frank RJ, Davey N, Hunt SP (2001) Time series prediction and neural networks. J Intell Robot Syst 31(1–3):91–103
20. Galvan IM, Isasi P (2001) Multi-step learning rule for recurrent neural models: an application to time series forecasting. Neural Process Lett 13(2):115–133
21. Gordan B, Armaghani DJ, Hajihassani M, Monjezi M (2016) Prediction of seismic slope stability through combination of particle swarm optimization and neural network. Eng Comput 32(1):85–97
22. Grimaldi EA, Grimaccia F, Mussetta M, Zich RE (2004) PSO as an effective learning algorithm for neural network applications. In: 2004 3rd International Conference on proceedings of the computational electromagnetics and its applications, 2004 (ICCEA 2004). IEEE, pp 557–560
23. Hajihassani M, Armaghani DJ, Monjezi M, Mohamad ET, Marto A (2015) Blast-induced air and ground vibration prediction: a particle swarm optimization-based artificial neural network approach. Environ Earth Sci 74(4):2799–2817
24. Hamzacebi C (2008) Improving artificial neural networks performance in seasonal time series forecasting. Inf Sci 178(23):4550–4559
25. Harrison KR, Ombuki-berman BM, Engelbrecht AP (2016) A radius-free quantum particle swarm optimization technique for dynamic optimization problems. In: Proceedings of IEEE congress on evolutionary computation, pp 578–585
26. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2(5):359–366

27. Iqbal S, Zang X, Zhu Y, Liu X (2014) Introducing undergraduate electrical engineering students to chaotic dynamics: computer simulations with logistic map and buck converter. In: IEEE modelling symposium, pp 47–52
28. Jha GK, Thulasiraman P, Thulasiram RK (2009) PSO based neural network for time series forecasting. In: Proceedings of IEEE international joint conference on neural networks, pp 1422–1427
29. Jordehi AR (2014) Particle swarm optimisation for dynamic optimisation problems: a review. Neural Comput Appl 25:1507–1516
30. Kedrowski R, Nelson J, Nair AS, Ranganathan P (2018) Short-term seasonal energy forecasting. In: 2018 IEEE international conference on electro/information technology (EIT). IEEE, pp 696–700
31. Kennedy J, Mendes R (2002) Population structure and particle swarm performance. Proc IEEE Congr Evol Comput 2:1671–1676
32. Laepes A, Farben R (1987) Nonlinear signal processing using neural networks: prediction and system modelling. In: Technical Report, Los Alamos National Laboratory, Los Alamos, NM
33. Lasheras FS, de Cos Juez FJ, Sánchez AS, Krzemień A, Fernández PR (2015) Forecasting the comex copper spot price by means of neural networks and arima models. Resour Policy 45:37–43
34. LeCun YA, Bottou L, Orr GB, Müller KR (2012) Efficient backprop. Neural networks: tricks of the trade. Springer, Berlin, pp 9–48
35. Li C, Sprott JC, Thio W (2015) Linearization of the Lorenz system. Phys Lett A 379(10):888–893
36. Li X, Dam KH (2003) Comparing particle swarms for tracking extrema in dynamic environments. IEEE Proc Evol Comput 3:1772–1779
37. Mann HB, Whitney DR (1947) On a test of whether one of two random variables is stochastically larger than the other. Ann Math Stat 18(1):50–60
38. Milano P, Elettrotecnica D (2004) PSO as an effective learning algorithm for neural network applications. In: Proceedings of the international conference on computational electronics and its applications, pp 557–560
39. Mohammad AA, Sohrab Z, Ali L, Ali E, Ioannis E (2013) Reservoir permeability prediction by neural networks combined with hybrid genetic algorithm and particle swarm optimization. Geophys Prospect 61(3):582–598
40. Momeni E, Armaghani DJ, Hajihassani M, Amin M (2015) Prediction of uniaxial compressive strength of rock samples using hybrid particle swarm optimization-based artificial neural networks. Measurement 60:50–63
41. Morrison RW (2003) Performance measurement in dynamic environments. In: Proceedings of the GECCO workshop on evolutionary algorithms for dynamic optimization problems, pp 5–8
42. Pampara G, Engelbrecht AP (2015) Towards a generic computational intelligence library: preventing insanity. In: Proceedings of the IEEE symposium series on computational intelligence, pp 1460–1467
43. Qi M, Zhang GP (2001) An investigation of model selection criteria for neural network time series forecasting. Eur J Oper Res 132(3):666–680
44. Rakitianskaia A (2011) Using particle swarm optimisation to train feedforward neural networks in dynamic environments. MSc. thesis, University of Pretoria
45. Rakitianskaia A, Engelbrecht AP (2008) Cooperative charged particle swarm optimiser. In: IEEE congress on evolutionary computation, pp 933–939
46. Rakitianskaia A, Engelbrecht AP (2009) Training neural networks with PSO in dynamic environments. In: IEEE congress on evolutionary computation, 2009 (CEC'09), pp 667–673
47. Rakitianskaia A, Engelbrecht AP (2015) Saturation in PSO neural network training: good or evil? IEEE Congr Evol Comput 2:125–132
48. Riedmiller M, Braun H (1993) A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In: IEEE international conference on neural networks, pp 586–591
49. Röbel A (1994) The dynamic pattern selection algorithm: effective training and controlled generalization of backpropagation neural networks. In: Technical Report, Tech- nische Universität Berlin
50. Sitte R, Sitte J (2000) Analysis of the predictive ability of time delay neural networks applied to the S&P 500 time series. IEEE Trans Syst Man Cybern C Appl Rev 30(4):568–572
51. Soodi HA, Vural AM (2018) STATCOM estimation using back-propagation, PSO, shuffled frog leap algorithm, and genetic algorithm based neural networks. Comput Intell Neurosci 2018:6381610. https://doi.org/10.1155/2018/6381610
52. Suresh A, Harish K, Radhika N (2015) Particle swarm optimization over back propagation neural network for length of stay prediction. Procedia Comput Sci 46:268–275
53. Tang Z, Fishwick PA (1993) Feedforward neural nets as models for time series forecasting. ORSA J Comput 5(4):374–385
54. Unger NJ, Ombuki-Berman BM, Engelbrecht AP (2013) Cooperative particle swarm optimization in dynamic environments. In: Proceedings of IEEE symposium on swarm intelligence, pp 172–179
55. Van den Bergh F (2001) An analysis of particle swarm optimizers. Phd Thesis, University of Pretoria
56. Van den Bergh F, Engelbrecht AP (2001) Effects of swarm size on cooperative particle swarm optimisers. In: Proceedings of the genetic and evolutionary computation conference, pp 892–899
57. Van den Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput 8(3):225–239
58. Van Wyk AB, Engelbrecht AP (2010) Overfitting by PSO trained feedforward neural networks. In: IEEE congress on evolutionary computation, pp 1–8
59. Wessels LFA, Barnard E (1992) Avoiding false local minima by proper initialization of connections. IEEE Trans Neural Netw 3(6):899–905
60. Yalcin N, Tezel G, Karakuzu C (2015) Epilepsy diagnosis using artificial neural network learned by PSO. Turk J Electr Eng Comput Sci 23(2):421–432
61. Yeh W, Yeh Y, Chang P, Ke Y, Chung V (2014) Forecasting wind power in the Mai Liao wind farm based on the multi-layer perceptron artificial neural network model with improved simplified swarm optimization. Int J Electr Power Energy Syst 55:741–748
62. Zhang G, Patuwo BE, Hu MY (1998) Forecasting with artificial neural networks: the state of the art. Int J Forecast 14(1):35–62