J Intell Manuf (2019) 30:1137–1154
https://doi.org/10.1007/s10845-017-1313-7

Combining SOM and evolutionary computation algorithms for RBF neural network training

Zhen-Yao Chen[1] · **R. J. Kuo**[2]

Received: 30 October 2014 / Accepted: 17 February 2017 / Published online: 27 February 2017

Abstract This paper intends to enhance the learning performance of radial basis function neural network (RBFnn) using self-organizing map (SOM) neural network (SOMnn). In addition, the particle swarm optimization (PSO) and genetic algorithm (GA) based (PG) algorithm is employed to train RBFnn for function approximation. The proposed mix of SOMnn with PG (MSPG) algorithm combines the automatically clustering ability of SOMnn and the PG algorithm. The simulation results revealed that SOMnn, PSO and GA approaches can be combined ingeniously and redeveloped into a hybrid algorithm which aims for obtaining a more accurate learning performance among relevant algorithms. On the other hand, method evaluation results for four continuous test function experiments and the demand estimation case showed that the MSPG algorithm outperforms other algorithms and the Box–Jenkins models in accuracy. Additionally, the proposed MSPG algorithm is allowed to be embedded into business' enterprise resource planning system in different industries to provide suppliers, resellers or retailers in the supply chain more accurate demand information for evaluation and so to lower the inventory cost. Next, it can be further applied to the intelligent manufacturing system to cope with real situation in the industry to meet the need of customization.

✉ Zhen-Yao Chen
keyzyc@gmail.com

R. J. Kuo
rjkuo@mail.ntust.edu.tw

1 Department of Business Administration, DE LIN Institute of Technology, New Taipei City, Taiwan

2 Department of Industrial Management, National Taiwan University of Science and Technology, Taipei, Taiwan

Keywords Hybrid algorithm · Radial basis function neural network · Self-organizing map neural network · Particle swarm optimization · Genetic algorithm

Introduction

In general, statistical methods can obtain reasonable prediction accuracy for future demand conditions, they have two common limitations: (1) it is difficult to specify the most suitable model without human expertise; (2) the models generated by these methods may not be able to capture some strongly nonlinear characteristics of short-term demand data (Chan et al. 2012). Also, the widely used time series models for forecasting purpose especially auto-regressive (AR) integrated moving average (MA) (ARIMA) model (Box and Jenkins 1976) is generally applicable to linear modeling and it hardly captures the non-linearity inherent in time series data (Jaipuria and Mahapatra 2014). Further, it is very important for decision-makers to focus on alternative models when non-stationary and non-linearity play a significant role in the forecasting (Sattari et al. 2012). Therefore, according to reliable sales forecasting methods, decision-makers can response quickly to market change, maintain the inventory in a relatively low level, and control the cost of production (Du et al. 2015).

Various meta-heuristic optimization algorithms have been applied to solve different optimization problems by researchers in many different scientific areas. The main objective all of the optimization algorithms is to be operated to search the global optimum to the all optimization problems (Duman et al. 2015). After that, neural network (NN) is an artificial intelligence (AI) system, which converts information of different space into the same space through simulating human intelligence behaviors. It is effectively applied to many application fields (Lu et al. 2015). For forecasting purpose, NN neither requires any statistical information nor stationary nature of data series (Jaipuria and Mahapatra 2014). For example, the self-organizing mapping (SOM) algorithm (Kohonen 1987) is one of the most popular NN model based on the unsupervised competitive learning paradigm (Yadav and Srinivasan 2011). Next, radial basis function neural networks (RBFnns) have a number of advantages over other types of NNs and these include better approximation capabilities, simpler network structures and faster learning algorithms (Qasem et al. 2012).

Additionally, due to the omnipresence of constraints in real-world optimization problems, constrained optimization problems have received considerable attentions over recent years. There is an increasing number of nature-inspired meta-heuristic algorithms (Mezura-Montes and CoelloCoello 2011) proposed, such as genetic algorithm (GA) (Holland 1975), particle swarm optimization (PSO) (Kennedy and Eberhart 1995), and artificial bee colony (ABC) (Karaboga and Basturk 2008) algorithms.

Prediction under soft-computing models, such as NNs, intelligent algorithms and hybrid intelligent approaches were used (Anbazhagan and Kumarappan 2014). The previous researchers have adopted the RBFnn structure along with other single approaches such as PSO (Feng 2006) and GA (Sarimveis et al. 2004), to implement the learning of the network. However, as every single technique always exists with some drawbacks, hybridizing is a reasonable way to take strengths and avoid weakness. Therefore, the hybrid methods become very popular for the combinatorial optimization problem (Qiu and Lau 2014). As such, this study intends to propose a mix of SOMnn with PSO and GA based (MSPG) algorithm for training RBFnn and make suitable performance verification and comparison. The evolutionary learning mechanism of the MSPG algorithm can be used to train and find out the optimal network parameters within the solution space of the individuals generated population in RBFnn. Further, we can utilize this verified MSPG algorithm, in terms of forecasting accuracy, to make predictions in the demand estimation problem.

In summary, this study starts from the idea of "clustering first then classification (2-stage)" to integrate SOMnn and evolutionary computation algorithms (ECAs) and then proposes the MSPG algorithm. Further, the MSPG algorithm is applied on the RBFnn training to obtain the better learning performance with much higher forecasting accuracy. This is considered the contribution to the theoretical methodology in this paper. In addition, the proposed MSPG algorithm is allowed to be further embedded into business' information system to perform its forecasting capability with high accuracy. Therefore, it can be applied to the enterprise resource planning (ERP) system in different industries to provide suppliers, resellers or retailers in the supply chain more accurate demand information for evaluation and so to lower the inventory cost. This is considered the contribution to the practical domain in this paper.

The rest of this paper is organized as follows. "Literature review" section summarizes RBFnn, SOMnn, and several ECAs. The proposed MSPG algorithm is presented in "Methodology" section. The results of experimental simulation, model evaluation, and comparison with relevant algorithms are illustrated in "Experimental simulation results" and "Model evaluation results" sections. Finally, concluding remarks are made in "Conclusions" section.

Literature review

This section presents general backgrounds associated to this research, including RBFnn, SOMnn, and evolutionary computation algorithms. The theories and applications pertaining to this study will be discussed in detail.

Radial basis function (RBF) neural network (RBFnn)

RBFnn was proposed by Duda and Hart (1973), it provides an alternative to accomplish the same work as NN does (Huang and Wang 2007). Next, the transfer function of the hidden layer is generally a non-linear Gaussian function (Wu and Liu 2012). Further, the mathematical equation which defines a RBFnn is given as (Ayala and Coelho 2016)

$$\hat{y}(t) = \sum_{m=1}^{M} w_m \phi(r(t), c_m, \sigma_m), \tag{1}$$

where $M \in \mathrm{N}^+$ is the number of neurons in the hidden layer, $\hat{y}(t) \in \mathrm{R}$ and $r(t) \in \mathrm{R}^{\mathrm{n_r}}$ are, respectively, the network predicted output and the input vector at a given instant t; $c_m \in \mathrm{R}^{\mathrm{n_r}}$ and $\sigma_m \in \mathrm{R}^+$ are, respectively, the center and the width of the mth hidden node of RBFnn. Each of the output weights is given by $w_m \in \mathrm{R}$. The Gaussian RBF is defined as (Ayala and Coelho 2016)

$$\begin{aligned}\phi(r, c, \sigma) &= \exp\left(-\frac{\|r-c\|^2}{2\sigma^2}\right)\\ &= \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n_r}(r_i - c_i)^2\right)\end{aligned} \tag{2}$$

Afterward, because of the simple topological structure and the ability to reveal how learning proceeds in an explicit manner, the RBFnn has been widely used as the universal function approximator to solve nonlinear problems (Lin and Wu 2011). In the field of prediction, Yu et al. (2010) proposes an RBFnn-ensemble forecasting model to obtain accurate prediction results and improve prediction quality further. In addition, Shafie-khah et al. (2011) proposed a novel hybrid model to forecast day-ahead electricity price, based on the wavelet transform, ARIMA models and RBFnn.

The difficulty of applying the RBFnns is in network training which should select and estimate properly the input parameters including centers and widths of the basis functions and the neuron connection weights (Tsekouras and Tsimikas 2013). Next, feature pre-processing technique in forecasting model also influences the forecasting accuracy significantly. Especially, a NN combined with pre-processed input feature data will achieve better prediction accuracy (Anbazhagan and Kumarappan 2014).

Self-organizing map (SOM) neural network (SOMnn)

A self-organizing map (SOM) neural network (SOMnn) is a nonlinear NN paradigm (Kohonen 1987). Next, learning in the SOM is unsupervised, making it useful in a variety of situations and easily modified to suit a variety of purposes (Rumbell et al. 2014). Contrary to the supervised clustering algorithms, unsupervised clustering algorithms do not need prior information that makes these algorithms more acceptable in the literature (Ozturk et al. 2015).

The key to a successful implementation of SOMnn is to find suitable centers for the Gaussian functions (Kurt et al. 2008). The SOMnn consists of M neurons arranged in a 2-D rectangular or hexagonal grid (Hadavandi et al. 2012).

Each neuron i is assigned a weight vector, $w_i \in R^n$ (index $i = (p, q)$ for 2-D map). At each training step t, a training data $x^t \in R^n$ is randomly drawn from data set and calculates the Euclidean distances between x^t and all neurons. A winning neuron with weight of w_j can be found according to the minimum distance to x^t:

$$j = \arg\min_i \left\| x^t - w_i^t \right\|, \quad i \in \{1, 2, \ldots, M\} \tag{3}$$

Then, the SOM adjusts the weight of the winner neuron and neighborhood neurons and moves closer to the input space according to:

$$w_t^{t+1} = w_i^t + \alpha^t \times h_{ji}^t \times \left[x^t - w_i^t \right], \tag{4}$$

where α^t and h_{ji}^t are the learning rate and neighborhood kernel at time t, respectively. Both α^t and h_{ji}^t decrease monotonically with time and within [0, 1]. The neighborhood kernel h_{ji}^t is a function of time and distance between neighbor neuron i and winning neuron. A widely applied neighborhood kernel can be written in terms of Gaussian function:

$$h_{ji}^t = \exp\left(-\frac{\left\| r_j - r_i \right\|^2}{2\sigma_t^2} \right), \tag{5}$$

where r_j and r_i are the position of winner neuron and neighborhood neuron on map. σ_t is kernel width and decreasing with time. This process of weight-updating will be performed for a specified number of iterations (Hadavandi et al. 2012).

SOM networks' ability to associate new data with similar previously learnt data can be applied to forecasting applications (Lopez et al. 2012). For example, Hsu et al. (2009) showed that SOM outperforms the hierarchical methods in clustering messy data and has better accuracy and robustness. Also, Lin and Wu (2009) proposed a hybrid NN model to forecast the typhoon rainfall using the SOMs and the feed-forward NNs. Further, feature pre-processing technique in forecasting model influences the forecasting accuracy significantly. Especially, a NN combined with pre-processed input feature data will achieve better prediction accuracy (Anbazhagan and Kumarappan 2014).

Evolutionary computation algorithms (ECAs)

Evolutionary computations (ECs) inherit the principles of biological evolution. This is stochastic in nature and stronger as compared to traditional optimization methods (Dey et al. 2014). Further, considering the drawbacks of traditional optimization techniques, attempts are being made to solve the optimization problems by using meta-heuristics, which are mostly nature inspired, such as PSO, artificial immune algorithm (AIA), and GA algorithms (Savsani et al. 2014). After that, PSO is a novel multi-agent optimization system inspired by social behavior metaphor (Kennedy and Eberhart 1995), while GAs are a family of computational models developed by Holland in 1975. Thus, PSO and GA are two intelligent optimization algorithms that are widely and successfully applied in various types of model parameter estimation because of their outstanding optimization capability (Yu et al. 2015b).

PSO is a swarm intelligence-based optimization technique inspired by social behavior and dynamic movement of a flock of insects, birds, and fish, which was developed by Kennedy and Eberhart (1995) (Ketabchi and Ataie-Ashtiani 2015). In a PSO system, each particle is 'flown' through the multidimensional search space, adjusting its position in search space according to its own experience and that of neighboring particles (Wang et al. 2014). Further, the velocities and positions of particles are updated in each time step according to the following equations (Kennedy and Eberhart 1995):

$$\begin{aligned} V_{id}(t+1) = V_{id}(t) + C_1 r_{1d}(P_{id} - X_{id}) \\ + C_2 r_{2d}(P_{gd} - X_{id}); \end{aligned} \tag{6}$$

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1), \tag{7}$$

where C_1 and C_2 are called cognitive and social acceleration coefficients respectively, r_{1d} and r_{2d} are two random numbers in the interval [0, 1] (Kennedy and Eberhart 1995). Afterward, PSO is based on social adaptation of knowledge for working, and all individuals are considered to be of the same generation. The particles with higher degree of constraint violation fly by the search space according to the information exchanged by their P_{id} (i.e., local best, Lbest) and P_{gd} (i.e., global best, Gbest) to search the better positions (Deng et al. 2012).

PSO shares many similarities with other EC techniques. The system is initialized with a population of random solutions and searches for optima by updating generations (Katherasan et al. 2014). Unlike GA, PSO has no evolution operators such as crossover and mutation. Compared to GA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust (Katherasan et al. 2014). However, although the original PSO presents a high convergence velocity, it does not present the capability to escape from local minima. It occurs because the original PSO is not able to maintain diversity within the swarm whenever it is necessary during the search process. These issues affect the PSO performance, mainly in dynamic problems or high dimensional multimodal search spaces (Vitorino et al. 2015). In addition, an adaptive self-generating RBFnn model with mixed encoding PSO is utilized to optimize the RBF's structure and parameters (Yu et al. 2009) and applied to predict the primary energy consumption of China (Yu et al. 2012b). Recently, researchers working in this area have started taking some interest on some promising approaches to numerical optimization.

On the other hand, the GA is referred to as EC technique and was proposed by Holland (1975) as an algorithm for searching an optimal solution based on survival of the fittest. The GA searches for an optimal solution through generations. Typically, the search starts with a population believed to possess the required best solution to the problem to be solved. Survival of the fittest is responsible for fostering evolution in the population to create the fittest chromosomes (Chiroma et al. 2015). The chromosomes with the best fitness values are selected for crossover and mutation whereas those with lower fitness are ignored for the reproduction. The fitness values are determined by the objective cost function of the problem. The fittest chromosomes are then selected for recombination through mutation and crossover, which is typically performed with probabilities (Chiroma et al. 2015). Next, GA is an effective optimization method for large and complex problems to escape the local optima and acquire a global optimal solution (Bagheri et al. 2015). Also, GAs present many advantages that have led to an increasing use, particularly with the rise in the processing power of computers. As they work with a set of potential solutions, GAs do not easily get trapped in local minima (Bagheri et al. 2015).

PSO and GA never require initial guesses and only the upper and lower bound must be defined (Garcia-Gonzalo and Fernandez-Martinez 2012). The superiority of such methods over other statistical/engineering ones is their ability to handle local minima/maxima points efficiently (Rezaee-Jordehi and Jasni 2013). Next, to avoid the particle to be stuck in the local minimum, Kuo and Han (2011) integrated the mutation mechanism of GA with PSO. In addition, although a great research effort has been put forward to obtain an ideal, accurately constructed and visually meaningful image clustering performance by Kuo et al. (2012), it still remains a challenge (Ozturk et al. 2015). Further, the hybrid PSO–GA algorithm has been applied for optimization in some fields, such as in primary energy demand prediction (Yu et al. 2012a) and curve fitting of manufacturing (Galvez and Iglesias 2013; Yu et al. 2015a). In addition, in the PSO–GA algorithm (Yu et al. 2012a), PSO first transforms the population into certainty generations. The best particles are retained, whereas the other particles are removed. Second, new individuals are generated by implementing the selection, crossover, and mutation operators of GA according to the remaining best particles. Third, the generated new individuals are placed in the remaining best particles to form a new population for the next generation. During the evolution process, the algorithm exchanges information several times to fully exploit the combination (Yu et al. 2012a). Recently, to strike a right balance between the performance and time complexity, a handful of meta-heuristics emerge as useful and powerful approaches to solve a wide range of optimization problems (Zhang et al. 2015a).

Methodology

The main idea underlying SOMnn is that RBFnns are local approximations, and the centers of local units (RBF neurons) are adjusted to move to the real center in the sense of feature representation (Er et al. 2005). The SOM is selected for this study since it is a fast, easy and reliable unsupervised clustering technique. SOM is used to divide the data into sub-population and hopefully reduce the complexity of the data space to more homogeneous sub-classes (Chang and Liao 2006). Further, the traditional SOM formulation includes a decaying neighborhood width over time to produce a more finely tuned output mapping (Rumbell et al. 2014).

As such, combining the automatically clustering ability of SOMnn (Kohonen 1987) with the PG algorithm, we proposed the mix of SOMnn with PG (MSPG) algorithm to improve the accuracy of function approximation by RBFnn. It provides the settings of some parameters, such as the neuron, width, and weight within RBFnn. During the process of the MSPG algorithm, SOMnn determines the number of center and its position values at first through its automatically clus-

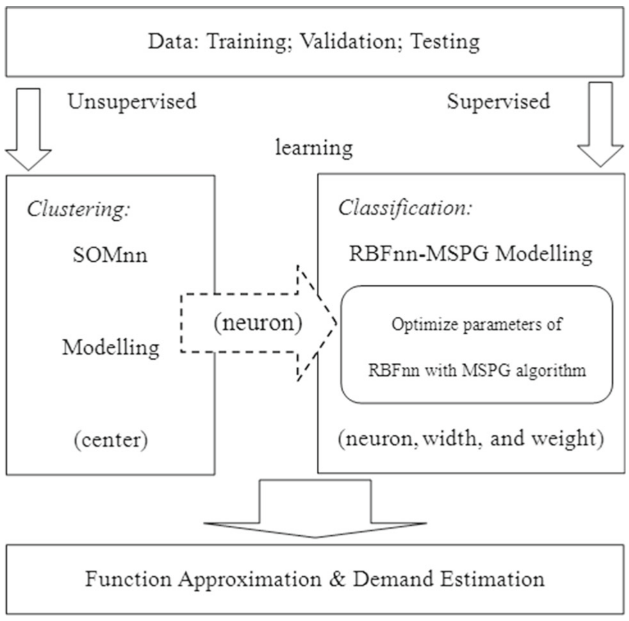
Data: Training; Validation; Testing
Unsupervised
Supervised
learning
Clustering:
SOMnn
Modelling
(center)
(neuron)
Classification:
RBFnn-MSPG Modelling
Optimize parameters of RBFnn with MSPG algorithm
(neuron, width, and weight)
Function Approximation & Demand Estimation

Fig. 1 The framework of the proposed MSPG algorithm

tering ability. The results are used as the number of neuron in RBFnn. The algorithm for the PG algorithm provides the settings of some parameters, such as the width and weight in RBFnn. The framework for the MSPG algorithm is illustrated in Fig. 1.

The analysis of the MSPG algorithm

The proposed MSPG algorithm, which combines SOMnn and the evolutionary learning approaches of the PSO and GA, was designed to resolve the problem of network parameters training and solving with RBFnn. The MSPG algorithm applies PSO and GA approaches as the learning mechanism in PG algorithm, respectively. The pseudo code for the SOMnn method of the MSPG algorithm is illustrated in Fig. 2.

Through the approaches such as PSO and GA within PG algorithm of the MSPG algorithm, we intend to solve proper values of the parameters from the setting domain in the experiment. The pseudo code for the PG algorithm of the MSPG algorithm is illustrated in Fig. 3.

The MSPG algorithm integrates SOMnn and virtues of PSO and GA approaches to enhance learning efficiency of RBFnn. The optimal values of parameters solution can be obtained and used in the MSPG algorithm with RBFnn to solve the problem for function approximation. This solution will enable RBFnn to make the most exact approximation toward the test functions in the experiment.

The inverse of root mean squared error (RMSE) is used as fitness function (i.e., Fitness $=$ RMSE^{-1}) (DelaOssa et al. 2006). Next, the nonlinear function that the RBFnn hidden layer adopted is the Gaussian function shown in Eq. (2), and the fitness value of individuals in population is calculated by Eq. (8). The fitness values for relevant algorithms in the experiment are computed by maximizing the RMSE^{-1} (Lee 2008) defined as:

$$Fitness = \sqrt{\frac{N}{\sum_{j=1}^{N}(y_j - \hat{y}_j)^2}}, \qquad (8)$$

where N is the number of the testing set, $\hat{y}_j$ is the predicted output of the learned RBFnn model for the jth training pattern, and y_j is the actual output.

The detailed description of the MSPG algorithm

A population of individuals undergoes a sequence of transformation by means of genetic operators to form a new population (Qasem et al. 2012). Further, each solution is called a 'particle' in PSO and 'chromosome' in GA where on the contrary to the former new solutions are not created from the parents within the evolution process (Yousefi et al. 2012). In which, the data are divided in three subsets $(X_1, Y_1), (X_2, Y_2), (X_3, Y_3)$ of size M_1, M_2 and M_3, which are the training (65%), testing (25%) and validation (10%) sets respectively (Looney 1996). Therefore, the evolutionary procedures for the PG algorithm of the MSPG algorithm was performed and summarized as follows.

(1) **Initialization:** The initialization corresponding to nature random selection ensures the diversity among individuals (i.e., particles in PSO approach or chromosomes in GA approach) and benefits the evolutionary procedure afterwards. An initial population with a number of individuals is generated and the initializing procedures are as follows.

(a) Each individual in the initial population is the set of positions of neuron (i.e., $c^s_{i,j}$) and width (i.e., wd^s_i) on RBFnn, defined as a matrix form. Figure 4 presents the design of decoding routine for the matrix form. Meanwhile, the embedded values of C_s are equivalent to RBFnn hidden nodes which include the $c^s_{ij}(i = \{1, \ldots, H\}, j = \{1, \ldots, N\})$ and $wd^s_i(i = \{1, \ldots, H\})$ of parameters solution (i.e., individuals) such as positions of neuron and width. S matrices $C_1, C_2, \ldots, C_s$ (i.e., population size) of size $H \times (N + 1)$ are created by setting all their elements equal to zero. For each $C_s(s = 1, 2, \ldots, S)$, a random integer $h_s \in \{1, \ldots, H\}$ from the number of center generated in SOMnn is selected.
The results are used as the number of neuron in RBFnn. The $\{1, \ldots, h_s\}$ rows of the C_s are replaced by an equal number of row vectors of size $1 \times (N+1)$

Fig. 2 The pseudo code for the SOMnn method of the MSPG algorithm. *Note* Initially, trainingData and testingData contains X[] and Y[], where elements of X[] and Y[] are sampled from the input and output space, respectively

```
SOMnn ( trainingData, testingData ){
set somData to the X[] part of trainingData and testingData; // see: Note.
// train SOMnn with somData
Initial a 10x10 SOMnn by setting the weight vector of each center randomly;
set learning rate $\varepsilon$ to 0.8;   set radius $\sigma$ to 10;
for i = 1 to 100000
// In the first 1000 generations, both $\varepsilon$ and $\sigma$ decrease linearly
    if ( i <= 1000 )
        decrease $\varepsilon$ linearly;   decrease $\sigma$ linearly;
    select randomly one element from somData as x;
    select the best center according to x;
    for each center $c$ in the neighborhood region of the current best center
    //update the weight vector of $c$
      $c = c + \varepsilon(x - c)$;   // refer Kohonen (1990)
    end for
end for
//choose centers from the trained SOMnn
for each $c$ in the trained SOMnn
 select $c$ as a center if there is at least one element in somData that regards $c$ as its best center;
end for
Use the PG algorithm to choose the widths of those selected centers; }
```

that are the neurons of RBFnn associated with this individual. The $\{h_s, \ldots, H\}$ rows remain equal to zero and do not correspond to a neuron.

(b) When the width parameter is fixed and a set of RBF neurons is provided, a RBFnn which has such a structure and an orthogonal least squares (OLS) algorithm (Chen et al. 1991) is developed for constructing parsimonious RBFnn (Chen et al. 1999) (i.e., RBFnn algorithm). Then the Gram–Schmidt scheme (Golub and Loan 1996) and Moore Penrose pseudo-inverse (Denker 1986) methods of the basis matrix are used to calculate the weights.
For each C_s, the output weights of the respective RBFnn are calculated by Eq. (9) (Denker 1986):

$$w_s = (B_s^T \cdot B_s)^{-1} \cdot (B_s^T \cdot Y_1) = B_s^{-1} Y_1, \qquad (9)$$

where w_s is the pseudo-inverse of the design matrix B_s; B_s is the $M_1 \times h_s$ matrix containing the responses of the hidden layer to the X_1 subset of examples; Y_1 is the desired response vector in the training set. The number of columns of the B_s equals the number of neurons at the hidden layer and the number of rows equals the number of training samples. Each column of B_s corresponds to the response of the respective hidden neuron to all input data (Barra et al. 2006). The calculation of the output weights completes the formulation of h_s RBFnns, which can be represented by the pairs $(C_1, w_1), (C_2, w_2), \ldots, (C_{h_s}, w_{h_s})$.

(c) The fitness value of individual matrix in population is calculated by Eq. (8) (i.e., RMSE^{-1}).

(2) **PSO approach:** The maximum value of Minimum selection type PSO learning method (Feng 2006) (i.e., PSO approach) will be considered the active number of RBFs for all particles and ensure that the same vector length is achieved. The solution of RBFnn correlated values of parameters that are included in the individuals of particle population, is equivalent to a set of the RBFnn solution. The PSO approach is one step which will be executed in one epoch and continue in the following process with the PG algorithm of the MSPG algorithm. This

Fig. 3 The pseudo code for the PG algorithm of the MSPG algorithm

```
Procedure PG algorithm;
Begin
    Initialize population by encoding routine;
    Calculate the weights and fitness values of population by decoding routine;
    Preserve the best chromosome;
    for ( i = 0 ; i < number of epochs ; i++ ) {
        Perform PSO Learning;
        Duplicate current chromosomes to [PSO-only] chromosomes
        Perform uniform crossover operator on chromosomes;
        Perform one-cut point mutation operator on chromosomes;
        Perform addition or deletion operator on chromosomes (Sarimveis et al., 2004);
        Calculate the weights and fitness values of chromosomes;
        Select a new population from both [PSO + GA] and [PSO-only] populations
          with roulette wheel selection;
        Replace the weakest chromosome with the previous best chromosome;
        Preserve the best chromosome;
        Decrease linearly operators of the updated probabilities of $P_c$, $P_m$, and $k$ by degrees
          with the number of epoch;
End
PSO Learning:
Begin
    Calculate the local and global optimal values through Eqs. (6) and (7);
    Perform Minimum selection type PSO learning method (Feng, 2006);
    Update velocities and positions of particles through Eq. (7);
    Calculate the weights and fitness values of particles by decoding routine;
End
```

step can update the values of velocity and the embedded values of all particle matrices to record the Lbest values through Eqs. (6) and (7). The procedures of the PSO approach are as follows.

(a) The number of the RBFnn hidden node neurons that use C_s of initialize population is regarded as the number of the neurons for each particle with PSO learning population, and thus is called particle matrix to progress the evolutionary process afterwards.
(b) The particles in population don't move toward any particular direction by Eq. (7) until the Lbest and Gbest of the present particle are calculated by Eq. (6).

(3) **Duplication:** The population enhanced by the Minimum selection type PSO learning method (Feng 2006) is duplicated and called [PSO-only] population.
(4) **GA approach:** The approach of GA evolution that includes one-cut point mutation, addition/deletion (Sarimveis et al. 2004), and uniform crossover (Syswerda 1989) operators in the population of PSO enhanced learning is called [PSO + GA] population. The operators used in GA approach are as follows.

(a) Figure 5 illustrates the idea of uniform crossover idea schematically. Later, each row of the selected paired C_s will have equal probability to precede uniform

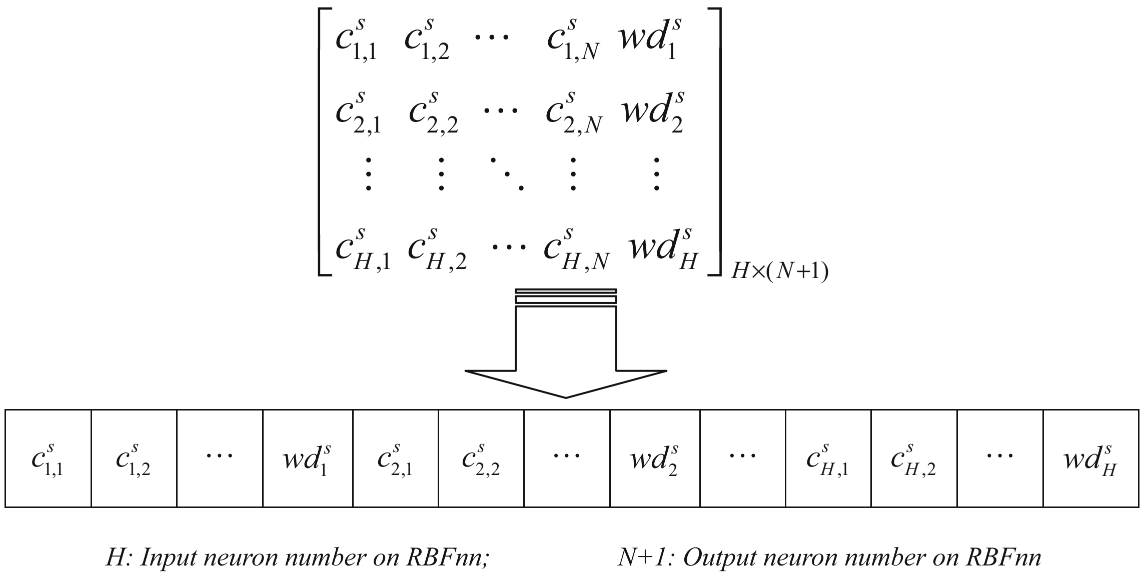
H×(N+1)
H: Input neuron number on RBFnn;
N+1: Output neuron number on RBFnn

Fig. 4 The design of decoding routine for the matrix form

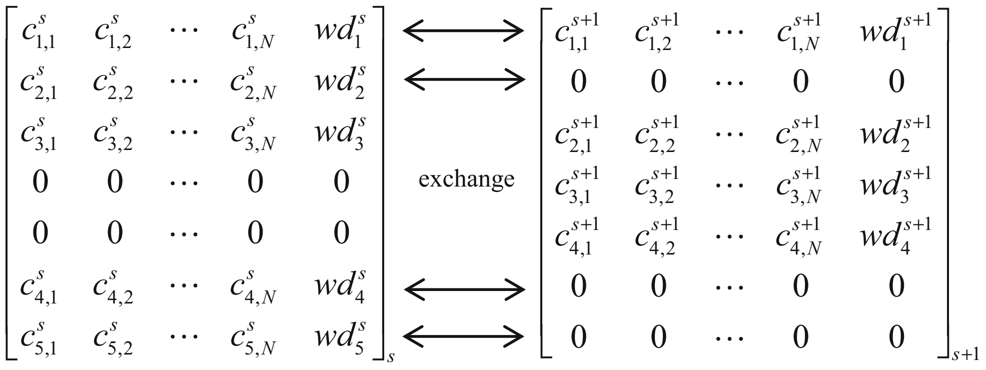

Fig. 5 Schematic illustration of uniform crossover between C_s and C_{s+1} with each pair of rows independently exchanging their values with probability 0.5

crossover (Syswerda 1989) operator, so as to conform to the spirit of GA.

(b) The mutation operator is one of the strategies used to ensure variability within the population and design space exploration (Rocha et al. 2014). Through the mutation operator, the values are replaced by randomly selected values from the range of the search domain in each dimension, which maintains the diversity and generates new solutions.

(5) **Reproduction:** In order to force the GA to propagate more intensely the genetic material from the best parents, the roulette wheel selection (Goldberg 1989) role was used for formation of the mating pairs (Kuzmanovski et al. 2007). The [PSO-only] and [PSO + GA] populations are combined after evolutionary process. Same amount of individuals from the initial population are randomly selected by the roulette wheel (proportional) selection for the evolution afterwards.

(a) The (X_2, Y_2) subset is used in this step as a testing set, in the following manner. First, the predictions $\hat{Y}_{2,1}, \hat{Y}_{2,2}, \ldots, \hat{Y}_{2,S}$ of the S RBFnn formulated in the previous step and the corresponding $RMSE_s$ are computed as follows:

$$RMSE_s = \sqrt{\frac{\sum_{s=1}^{S} (Y_2 - \hat{Y}_{2,s})^2}{S}} \tag{10}$$

(b) The pair (C_s, w_s) associated with the maximum error is replaced by the best RBFnn of the previous epoch so that the optimal solution survives in all epochs (this replacement will not take place in the first epoch). The network associated with the minimum error is stored for later use. The objective is to give more chances of survival to the network associated with smaller error values. Therefore, the probability of selection p_s of every C_s is calculated by Eq. (11), and the cumulative probability q_s is computed by Eq. (12):

$$p_s = \frac{RMSE_s^{-1}}{\sum_{s=1}^{S} (RMSE_s^{-1})}, \tag{11}$$

$$q_s = \sum_{i=1}^{s} p_i \tag{12}$$

(6) **Termination:** The PG algorithm of the MSPG algorithm will not stop returning to step (2) unless a specific number of epochs has been achieved.

We focus on how to combine SOMnn and two evolutionary approaches to obtain the complementary learning effect. In the evolutionary process of PG (i.e., PSO + GA) algorithm, our differences with other evolutionary algorithms are: (1) The PSO and GA approaches in PG algorithm are able to take their own best calculated results to do cross learning in the next generation, and then gradually obtain the optimal solution in the whole population; (2) PG algorithm has the capability to dynamically adjust relevant parameters (i.e., inertia weight, mutation probability, and crossover probability) by decreasing linearly in a certain range. And with such way of having multiple learning factors for cooperation, it facilitates the PG algorithm to converge and further solve the optimal solution.

For population in PSO approach, the particle figures out the best solution after consulting itself and other particles, and decides the proceeding direction and velocity. Also, the memory mechanism (Xu et al. 2007) implemented in PSO approach can retain the information of previous best solutions that may get lost during the population evolution. Through the memory mechanism, the obtained parameter solution in the population will be more advanced than the initial ones to facilitate the evolutionary process afterwards. Thus, executing an evolutionary computation through the PSO approach would obtain an enhanced evolution population, which is better than the initial population.

As the algorithm proceeds, the members of the population improve gradually. Due to the property of global search with GA approach, no matter what the fitness values of the individuals in population are, they all have the chances to proceed with some genetic operators and enter the next generation of population to evolve. In this way, the PG algorithm of the MSPG algorithm meets the spirit of GA approach and ensures the genetic diversity in the future evolution process, and proceeds to obtain a new enhanced population. In addition, through the GA approach within the PG algorithm to estimate the fitness values of individual parameter solution in the population, the better solutions will be obtained gradually. Thus, the solution space in population could be changed gradually and converge toward the optimal solution. The algorithm stops after a specific number of epochs have been completed.

In the latter experiment, the MSPG algorithm stops and the RBFnn corresponding to the maximum fitness value is selected. Finally, it is validated by using the (X_3, Y_3) subset, which has not been utilized throughout the entire learning procedure. After those critical parameter values are set, RBFnn can initiate the training of approximation and learning through four benchmark problems. The above mentioned are our contribution to provide theoretical development.

Experimental simulation results

In this study, 1000 randomly generated data sets are divided into three parts to train RBFnn: 65% training set, 25% testing set, and 10% validation set (Looney 1996), in which we can examine the learning status and adjust the parameter setting.

Four benchmark problems

There are several test functions with many local minima, thus they can be used for comparison (Tsai et al. 2006). Continuous test function leads to excellent approximation to compensate RBFnn for the outcome of nonlinear mapping relation. The MSPG algorithm has better performance among other algorithms through the experiment in four benchmark problems, including Rosenbrock, Griewank, B2 (Shelokar et al. 2007), and Mackey-Glass time series (Whitehead and Choate 1996) continuous test functions, which are defined in "Appendix".

In the experiment, it was performed on a PC with Intel Xeon$^{\text{TM}}$ CPU, running at 3.40 GHz symmetric multi-processing (SMP), and 2 GB of RAM. Simulation were programmed in the Java 2 platform, standard edition (J2SE) 1.5. In addition, Gnuplot version 4.2 open source software was also used in the analysis to present the drafting results. In this study, the search domain is two-dimensional (2D) and the unit of output is amplitude. The maximum number of epochs is set at 1000 to take as termination condition in the experiment.

Parameters setup

There are several values of parameters within RBFnn that must be set up in advance to perform training for function approximation. The proposed MSPG algorithm is better than trial and error way in the literature in that it determines the appropriate values of parameters from the verified domain to train RBFnn. Relevant algorithms start with the selection of

Table 1 Parameter setup for four benchmark problems

Parameter	Description	Continuous test function			
		Rosenbrock	Griewank	B2	Mackey-Glass time series
D_s	Search domain	[−5, 5]	[−300, 600]	[−100, 100]	[0.4, 1.6]
wd_i^s	The widths of RBFnn hidden layer	[300, 600]	[43,300, 44,000]	[24,000, 25,000]	[0.1, 0.2]

the parameters setting for four benchmark problems shown in Table 1.

Furthermore, the Taguchi (robust design) method (Taguchi and Yokoyama 1993) (which used in this experiment for parameter setup) is a powerful experimental design tool (Olabi 2008) for solving the problems of optimizing the performance, quality and cost of a product or process in a simpler, more efficient and systematic manner than traditional trial-and-error processes (Lin et al. 2009). As such, the parameters setting for the MSPG algorithm in this study is obtained according to Taguchi experiment design (Taguchi et al. 2005) and several literatures.

In the SOMnn method of the MSPG algorithm, the maximum number of center C is 100, the learning rate ε is 0.8, the radius σ is 10, and the maximum number of generation G is 100,000 (Kohonen 1990). According to 4, a suitable population size is about 20–30 chromosomes. Thus, S is assigned as 25 in this study.

In the PSO approach, Shi and Eberhart (1998) introduced the parameter inertia weight k into the PSO equation to improve its performance. Suitable selection of k provides a balance between global and local explorations, and thus requiring fewer generation on average to identify a sufficiently optimal solution. The k in Eq. (13) can be expressed by the inertia weights approach (Kennedy and Eberhart 2001), as given below:

$$k = k_{\max} - \frac{k_{\max} - k_{\min}}{E} \times n, \tag{13}$$

where $k_{\max}$ and $k_{\min}$ are the maximum and minimum value, respectively; n is the current number of epochs, and E represents the maximum number of epochs. The parameter selection problem is formulated as a searching problem and a method based on a PSO evolutional learning method which is applied to select a parameter set R in the searching space, in which the scaling factor is 0.75 (Feng 2006). As originally developed, k often decreases linearly from approximately 0.9 to 0.4 during a run (Amraee et al. 2007). Thus, this study adopted the k decreased linearly from 0.75 to 0.4 with the increase of epochs. Further, c_1 and c_2 in the PSO approach are assigned as 2, which represent the same weight of stochastic terms pulling the particle toward Pbest and Gbest (Wang and Lu 2006).

Table 2 Parameters setup for the proposed MSPG algorithm in the experiment

Parameter	Description	Value
E	The maximum number of generations	1000
C	The number of centers of SOMnn	[1, 100]
ε	The learning rate of SOMnn	0.8
σ	The radius of SOMnn	10
G	The maximum number of generations of SOMnn	100,000
δ	The learning rate of RBFnn	0.42
S	Population size	25
k	$\text{Weight}_{\text{Inertia}}$	[0.4, 0.75]
c_1, c_2	Acceleration constants	2
P_m	Mutation probability (one-cut point mutation)	[0.02, 0.04]
P_c	Crossover probability (uniform crossover)	[0.5, 0.9]
P_a, P_d	Addition/deletion probability	0.005

In the GA approach, according to Azadeh and Tarverdian (2007), P_m shows best while varying between 1 and 5%. In addition, the P_c is recommended from Holland (1975). In the meanwhile, the P_m and P_c are decreased linearly from 0.04 to 0.02 and 0.9 to 0.5 with epochs respectively in this study. Moreover, the P_a and P_d are assigned as 0.005 from Sarimveis et al. (2004).

Since the drawback of soft computing techniques is the parameter setup, this study applies Taguchi method (Taguchi and Yokoyama 1993) for experimental design. Consequently, the statistical software MINITAB 14 was used in the analysis of parameter design for algorithm, where the signal-to-noise (S/N) ratio (Lin et al. 2009) is used to evaluate the stability of system quality in the experiment. Afterward, the Taguchi trials (Taguchi et al. 2005) were configured in an L_9 (3^4) orthogonal array for the MSPG algorithm after the experiment was implemented thirty times. Finally, the MSPG algorithm starts with the selection of the parameters setting shown in Table 2 to ensure consistent basis in the experiment.

Performance analysis of experimental results

The best approximation results of the MSPG algorithm for RBFnn trained in the experiment are showed in Figs. 6 and 7. As shown in Fig. 6, the best approximation results indicate

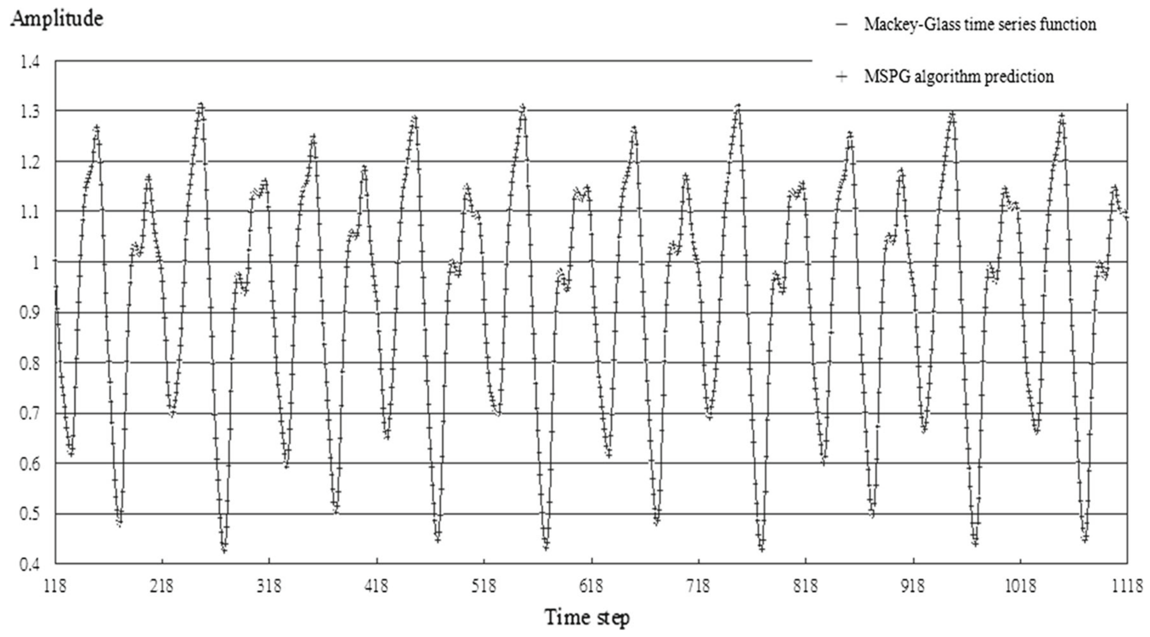

Fig. 6 The best approximation result obtained over 1000 trials of the MSPG algorithm with RBFnn trained in Mackey-Glass time series function prediction

that by using the MSPG algorithm under the circumstances of Mackey-Glass time series function, the performance resulted from effective learning of RBFnn can approximately accord with the curved functions.

The learning of MSPG algorithm on several RBFnn parameters solutions, which are generated by the population during the operation of evolutionary procedure in the experiment, is implemented. The MSPG algorithm is used to solve the optimal RBFnn parameters solution. It generates unrepeated 65% random training set from 1000 generated data and inputs the set to network for learning. With the same method, it then generates another unrepeated 25% random testing set to verify individual parameters solution in population and calculates the fitness value. Up to this learning stage, 90% dataset has been used by RBFnn. Once the evolutionary process has progressed for 1000 epochs, the optimal RBFnn parameters solution is obtained. Lastly, unrepeated random 10% validation set is generated to prove how the individual parameter solution approximates four benchmark problems, and the RMSE values are recorded to confirm the learning situation of RBFnn.

The learning and validation stages mentioned above were implements for 50 runs. The average RMSE values were calculated and are shown in Table 3 along with their standard deviations (SD). The results indicate that the smallest values are acquired by the MSPG algorithm with stable performance during the whole training process in the experiment, and RBFnn is able to obtain the parameters solution from the evolutionary learning process in population, which has achieved the optimal function approximation situation. Once the training of RBFnn by the MSPG algorithm is finished, the individual value of parameters (i.e., neuron, width, and weight) with its optimal solution is then the exact setting of network.

As shown in Table 3, the trends of training and validation performance are consistently small, which means RBFnn trained through the MSPG algorithm provides certain stability. Such result not only suffices for the training set and validation set, a generalization could also be made with regards to other unseen dataset. It may thus be known that over-fitting and over-training problems do not exist in the experiment adopting the MSPG algorithm.

Additionally, as for the verification of statistical significant difference, we obtained the results significantly while conducting the matched paired sample tests of t test with the absolute error from the predicted dataset of the source data in each algorithm. Next, the estimation verification and the t test results among relevant algorithms are shown in Table 4, the MSPG algorithm is not statistical significant (p value larger than 0.05, i.e., there is no significant deviation between the predicted values and actual values). Thus, the statistical results indicate that the MSPG algorithm has the best performance in terms of prediction accuracy among relevant algorithms.

In the next section, a real-world demand estimation case is applied to verify the accuracy and practicality for the proposed MSPG algorithm.

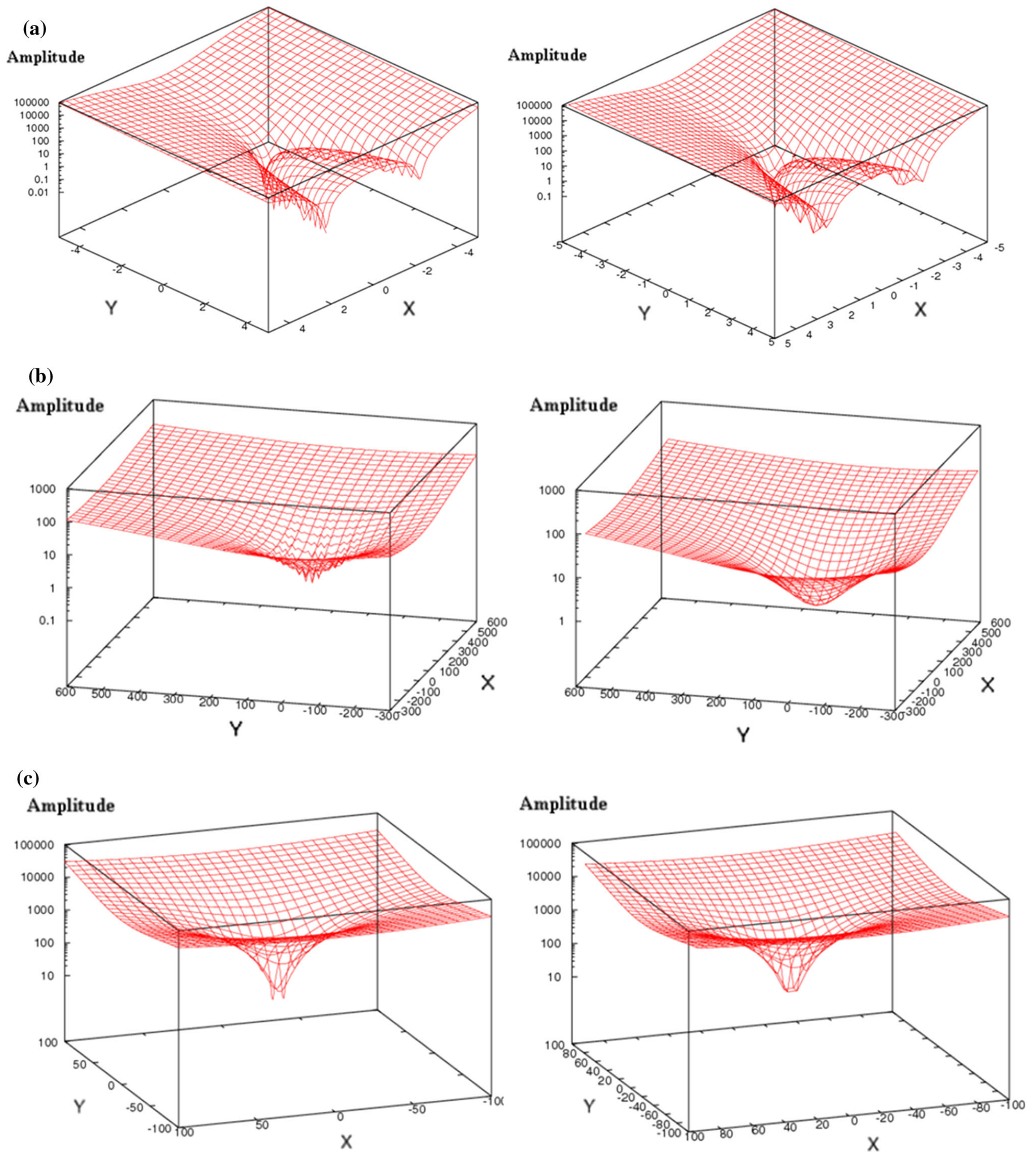

Fig. 7 The best approximation result obtained over 1000 trials of the proposed MSPG algorithm with RBFnn trained in continuous test functions: **a** original and predicted Rosenbrock function with RBFnn trained by the MSPG algorithm, **b** original and predicted Griewank function with RBFnn trained by the MSPG algorithm, **c** original and predicted B2 function with RBFnn trained by the MSPG algorithm

Model evaluation results

RBFnn has already been verified to be able to generate an accurate approximation on four benchmark problems through the proposed MSPG algorithm. The results are compared with other algorithms, illustrating the accuracy of the MSPG algorithm. Afterward, the daily sales observations of 500 cm^3 containers of papaya milk were offered by a com-

Table 3 Comparison results for relevant algorithms in the experiment

Experiment	Rosenbrock function		Griewank function	
Algorithm	Training set	Validation set	Training set	Validation set
RBFnn (Chen et al. 1999)	11,880 ± 1343.4	12,731 ± 2555.7	26.394 ± 2.359	27.85 ± 3.777
PSO (Feng 2006)	62.05E−4 ± 35.68E−4	89.63E−4 ± 52.15E−4	6.90E−1 ± 1177.99E−4	7.39E−1 ± 1129.46E−4
GA (Sarimveis et al. 2004)	7.28E−4 ± 1.12E−4	9.16E−4 ± 2.40E−4	5.20E−1 ± 117.08E−4	5.55E−1 ± 160.69E−4
PSO–GA (Yu et al. 2012a)	4.02E−4 ± 0.35E−4	4.91E−4 ± 1.04E−4	5.05E−1 ± 67.27E−4	5.11E−1 ± 86.23E−4
MSPG	1.08E−4 ± 0.55E−4	1.20E−4 ± 0.21E−4	4.02E−1 ± 10.36E−4	4.09E−1 ± 22.15E−4
Experiment	**B2 function**		**Mackey-Glass time series**	
Algorithm	Training set	Validation set	Training set	Validation set
RBFnn (Chen et al. 1999)	5791.5 ± 403.7	5848.8 ± 673.66	691 ± 45.7	705 ± 59.2
PSO (Feng 2006)	24.79E−2 ± 3.86E−3	39.40E−2 ± 683.88E−3	4.33E−3 ± 5.56E−4	4.05E−3 ± 5.84E−4
GA (Sarimveis et al. 2004)	24.18E−2 ± 4.19E−3	30.24E−2 ± 12.74E−3	2.51E−3 ± 1.43E−4	2.48E−3 ± 1.77E−4
PSO–GA (Yu et al. 2012a)	18.17E−2 ± 1.79E−3	16.54E−2 ± 5.70E−3	2.14E−3 ± 1.26E−4	2.97E−3 ± 1.83E−4
MSPG	6.74E−2 ± 1.27E−3	5.43E−2 ± 2.03E−3	1.80E−3 ± 0.72E−4	2.09E−3 ± 0.93E−4

Table 4 The statistical results for *t* test among relevant algorithms

Algorithm	Paired differences			
	Mean	SD	T	Significant (2-tailed)
RBFnn (Chen et al. 1999)	−502.394	439.074	−3.140	0.011*
GA (Sarimveis et al. 2004)	−227.466	428.782	−1.429	0.147*
PSO (Feng 2006)	−438.594	433.705	−2.174	0.036*
PSO–GA (Yu et al. 2012a)	−352.181	304.391	−2.285	0.023*
MSPG	−112.371	214.572	−0.628	0.705

Mean is equal to arithmetic average
* 5% significance level

pany of chain convenience stores in Taiwan's retail industry. The analysis had assumed that the influence of external experimental factors did not exist. The trend of papaya milk sales was not interfered by any special events. Moreover, there are several values of parameters within RBFnn that must be set up in advance to perform training for the case of estimation analysis. Relevant algorithms start with parameters setting shown in Table 5 and are meant to ensure consistent basis in this case.

Input data and RBFnn learning

Most studies in the literatures use convenient ratio of splitting for in- and out-of-samples such as 70:30, 80:20, or 90:10% (Zou et al. 2007). We use the ratio of 90:10% (Zou et al. 2007) here as the basis of division. The detailed data distribution of the case is shown in Table 6.

The application example with papaya milk for historical sales is based on time series data distribution and applied to estimation analysis. In order to obtain convergence within a reasonable number of cycles, the input and output data should be normalized and scaled to the range of 0–1 by Eq. (14) (Jin et al. 2011) for data of papaya milk.

$$x_{ni} = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}, \tag{14}$$

where x_i is the actual value of the observed data, $x_{\max}$ and $x_{\min}$ are the maximum and minimum observation values of the dataset, and x_{ni} is the normalized value of the observed data. The first 90% of the observations were used for model estimation while the remaining 10% were used for validation and one-step-ahead forecasting. This study elaborates how data is input to RBFnn for estimation through relevant algorithms, and comparison with Box–Jenkins models.

Building the Box–Jenkins models

EViews$^{\text{TM}}$ 6.0 software was used the analysis of Box–Jenkins models to calculate the numerical results. If the data is stationary, model estimation can be implemented directly. This research precedes the data identification of Box–Jenkins models through augmented Dickey–Fuller (Dickey

Table 5 Parameters setting for relevant algorithms in the demand estimation case

Parameter	Description	Value
E	The maximum number of generations	1000
C	The number of centers of SOMnn	[1, 100]
ε	The learning rate of SOMnn	0.8
σ	The radius of SOMnn	10
G	The maximum number of generations of SOMnn	100,000
S	Population size	30
wd_i^s	The width of RBFnn hidden layer	[1000, 40,000]
k	$\text{Weight}_{\text{Inertia}}$	0.5
c_1, c_2	Acceleration coefficients	2
P_m	Mutation probability (one-cut point mutation)	[0.01, 0.05]
P_c	Crossover probability (uniform crossover)	[0.5, 0.8]
P_a, P_d	Addition/deletion probability	0.005

Table 6 The observations data distribution in the demand estimation case

Case study	The observations: month-day-year (number of samples)		
	Total number of data	Learning set (90%)	Forecasting set (10%)
Papaya milk sales	01/01/1995–01/14/1996	01/01/1995–12/07/1995	12/08/1995–01/14/1996

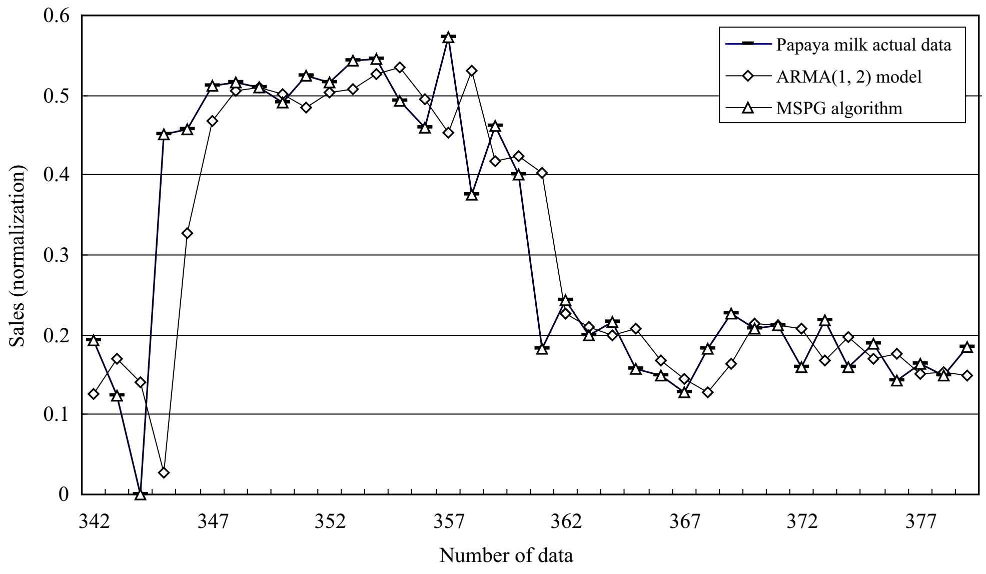

Fig. 8 The estimation results comparison of the proposed MSPG algorithm and ARMA (1, 2) model for the demand estimation case

and Fuller 1981) testing. Next, the study carries out demand estimation based on ARIMA (p, d, q) models, the procedures can be divided into three steps (Babu and Reddy 2014): (1) identifying the model order (i.e., identifying p and q)—Akaike (1974) information criterion (i.e., AIC value $= -2.3062$) were employed to sift the optimal model out (Engle et al. 1987) (i.e., ARMA (1, 2) model); (2) estimating the model coefficients—the results of model diagnosis reveal that the values of Ljung–Box statistic (i.e., Q-statistic) (Kmenta 1986) are greater than 0.05 in result of Box–Jenkins models, in which the results are white noise (i.e., serial non-correlation) and it had been suitable fitted; (3) forecasting the data.

Error measure of the estimation performance in the case

The mean absolute error (MAE), RMSE, and mean absolute percentage error (MAPE) are applied to evaluate the forecasting accuracy (Zhang et al. 2015b). The results of forecasting

Table 7 The estimation errors comparison for relevant algorithms using the demand estimation case

Algorithm error	PSO (Feng 2006)	GA (Sarimveis et al. 2004)	PSO–GA (Yu et al. 2012a)	MSPG	ARMA (1, 2) model
RMSE	3.68E−2	78.32E−2	1.97E−2	1.62E−2	0.0946
MAE	5.98E−2	53.60E−2	4.15E−2	1.32E−2	0.0556
MAPE	27.12E−1	192E−1	16.8E−1	5.44E−1	18.9163

set for the case is shown in Fig. 8. Also, the estimation performances of above mentioned algorithms with the case data is presented in Table 7.

Among these algorithms, the results derived from RMSE, MAE, and MAPE of the proposed MSPG algorithm were the smallest ones. According to the obtained numerical results, we know that compared to traditional Box–Jenkins models, the MSPG algorithm can substantially improve the accuracy of practical demand estimation.

Moreover, the proposed MSPG algorithm in this study can be applied to the company's internal information system in the case study to forecast the product demands. It can be further applied to the intelligent manufacturing system of the production line to generate different product demand data forecasts for different clients to manage their supply. Therefore, it is able to cope with real situation in the industry to meet the need of product demand customization (e.g., few items with large demand, many items with less demand, and many items with large demand), which adjusts supply dynamically.

Conclusions

Our study for the proposed MSPG algorithm combines the automatically clustering ability of SOMnn with the PG algorithm, which provides the settings of RBFnn parameters. The complementation of some evolutionary operations that improves the diversity of populations also increases the precision of the results. In addition, a case study and the tuning values of parameters with RBFnn using the trained algorithm has been given. The MSPG algorithm has better parameter setting of network and consequently enables RBFnn to perform better learning and approximation in four benchmark problems and application in the demand estimation case.

In the future, it may be promising to employ different evolutionary computation algorithms, such as ant colony optimization (ACO), artificial immune system (AIS), and ABC algorithms and further training different type NNs. Afterward, perhaps the sales data in the short term would be more stable and could be more beneficial to improve demand estimation. Thus, the accuracy of prediction in product sales data within shorter period can be further compared in the future. Additionally, it is common to have significant fluctuation and change in general sales prediction, and it could be the result of exogenous variables or unexpected variances such as sales force, promotional campaign, and exposure in international exhibitions. These exogenous variables were not considered in this study and thus could be considered for future work.

Appendix: Four continuous test functions (Shelokar et al. 2007; Whitehead and Choate 1996)

The first experiment, Rosenbrock function (Shelokar et al. 2007) is expressed as follows:

$$RS(x_j, x_{j+1}) = \sum_{j=1}^{n-1} [100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2], n = 2 \tag{15}$$

(a) search domain: $-5 \leqq x_j \leqq 5$, $j = 1$;
(b) one global minimum: $(x_1, x_2) = (1, 1)$; $RS(x_1, x_2) = 0$.

In the second experiment, Griewank function (Shelokar et al. 2007) is expressed as follows:

$$GR(x_j, x_{j+1}) = \sum_{j=1}^{n} \frac{x_j^2}{4000} - \prod_{j=1}^{n} \cos\left(\frac{x_{j+1}}{\sqrt{j+1}}\right) + 1, n = 1 \tag{16}$$

(a) search domain: $-300 \leqq x_j \leqq 600$, j= 1;
(b) one global minimum: (x_1, x_2) = (0, 0); $GR(x_1, x_2)$= 0.

In the third experiment, B2 function (Shelokar et al. 2007) is expressed as follows:

$$B2(x_j, x_{j+1}) = x_j^2 + 2x_{j+1}^2 - 0.3\cos(3\pi x_j) \\ -0.4\cos(4\pi x_{j+1}) + 0.7 \tag{17}$$

(a) search domain: $-100 \leqq x_j \leqq 100$, $j = 1$;
(b) one global minima: $(x_1, x_2) = (0, 0)$; $B2(x_1, x_2) = 0$.

In the fourth experiment, the Mackey-Glass time series (Whitehead and Choate 1996) is expressed as follows:

$$\frac{dx(t)}{d(t)} = 0.1x(t) + 0.2 \cdot \frac{x(t-17)}{1 + x(t-17)^{10}} \tag{18}$$

The research for the retrieved time step was in the range from 118 to 1118 with the Mackey-Glass time series function, from which 1000 samples were generated randomly.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, *19*, 716–723.

Amraee, T., Ranjbar, A. M., Mozafari, B., & Sadati, N. (2007). An enhanced under-voltage load-shedding scheme to provide voltage stability. *Electric Power Systems Research*, *77*, 1038–1046.

Anbazhagan, S., & Kumarappan, N. (2014). Day-ahead deregulated electricity market price forecasting using neural network input featured by DCT. *Energy Conversion and Management*, *78*, 711–719.

Ayala, H. V. H., & Coelho, L. D. S. (2016). Cascaded evolutionary algorithm for nonlinear system identification based on correlation functions and radial basis functions neural networks. *Mechanical Systems and Signal Processing*, *68–69*, 378–392.

Azadeh, A., & Tarverdian, S. (2007). Integration of genetic algorithm, computer simulation and design of experiments for forecasting electrical energy consumption. *Energy Policy*, *35*, 5229–5241.

Babu, C. N., & Reddy, B. E. (2014). A moving-average-filter-based hybrid ARIMA–ANN model for forecasting time series data. *Applied Soft Computing*, *23*, 27–38.

Bagheri, M., Mirbagheri, S. A., Bagheri, Z., & Kamarkhani, A. M. (2015). Modeling and optimization of activated sludge bulking for a real wastewater treatment plant using hybrid artificial neural networks–genetic algorithm approach. *Process Safety and Environmental Protection*, *95*, 12–25.

Barra, T. V., Bezerra, G. B., & de Castro, L. N. (2006). An immunological density-preserving approach to the synthesis of RBF neural networks for classification. In *International joint conference on neural networks* (pp. 929–935).

Box, G. E. P., & Jenkins, G. (1976). *Time series analysis, forecasting and control*. San Francisco: Holden-Day.

Chan, K. Y., Dillon, T. S., Singh, J. S., & Chang, E. (2012). Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and Levenberg–Marquardt algorithm. *IEEE Transaction on Intelligent Transportation Systems*, *13*(2), 644–654.

Chang, P. C., & Liao, T. W. (2006). Combining SOM and fuzzy rule base for flow time prediction in semiconductor manufacturing factory. *Applied Soft Computing*, *6*, 198–206.

Chen, S., Cowan, C. F. N., & Grant, P. M. (1991). Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transaction on Neural Networks*, *2*(2), 302–309.

Chen, S., Wu, Y., & Luk, B. L. (1999). Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks. *IEEE Transactions on Neural Networks*, *10*(5), 1239–1243.

Chiroma, H., Abdulkareem, S., & Herawan, T. (2015). Evolutionary neural network model for West Texas Intermediate crude oil price prediction. *Applied Energy*, *142*, 266–273.

DelaOssa, L., Gamez, J. A., & Puetra, J. M. (2006). Learning weighted linguistic fuzzy rules with estimation of distribution algorithms. In *IEEE congress on evolutionary computation* (pp. 900–907). Vancouver, BC: Sheraton Vancouver Wall Centre Hotel.

Deng, W., Chen, R., Gao, J., Song, Y., & Xu, J. (2012). A novel parallel hybrid intelligence optimization algorithm for a function approximation problem. *Computers and Mathematics with Applications*, *63*, 325–336.

Denker, J. S. (1986). Neural network models of learning and adaptation. *Physica D*, *22*, 216–232.

Dey, S., Bhattacharyya, S., & Maulik, U. (2014). Quantum inspired genetic algorithm and particle swarm optimization using chaotic map model based interference for gray level image thresholding. *Swarm and Evolutionary Computation*, *15*, 38–57.

Dickey, D. A., & Fuller, W. A. (1981). Likelihood ration statistics for autoregressive time series with a unit root. *Econometrica*, *49*(4), 1057–1072.

Du, W., Leung, S. Y. S., & Kwong, C. K. (2015). A multiobjective optimization-based neural network model for short-term replenishment forecasting in fashion industry. *Neurocomputing*, *151*, 342–353.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley.

Duman, S., Yorukeren, N., & Altas, I. H. (2015). A novel modified hybrid PSOGSA based on fuzzy logic for non-convex economic dispatch problem with value-point effect. *Electrical Power and Energy Systems*, *64*, 121–135.

Engle, R. F., Robert, F., & Yoo, B. S. (1987). Forecasting and testing in cointegrated systems. *Journal of Econometrics*, *35*, 588–589.

Er, M. J., Li, Z., Cai, H., & Chen, Q. (2005). Adaptive noise cancellation using enhanced dynamic fuzzy neural network. *IEEE Transactions on Fuzzy Systems*, *13*(3), 331–342.

Feng, H. M. (2006). Self-generation RBFNs using evolutional PSO learning. *Neurocomputing*, *70*, 241–251.

Galvez, A., & Iglesias, A. (2013). A new iterative mutually coupled hybrid GA–PSO approach for curve fitting in manufacturing. *Applied Soft Computing*, *13*, 1491–1504.

Garcia-Gonzalo, E., & Fernandez-Martinez, J. L. (2012). A brief historical review of particle swarm optimization (PSO). *Journal of Bioinformatics and Intelligent Control*, *1*(1), 3–16.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization & machine learning*. Reading, MA: Addison-Wesley.

Golub, G. H., & Loan, C. F. V. (1996). *Matrix computations* (3rd ed.). Baltimore, MD: Johns Hopkins Univ. Press.

Hadavandi, E., Shavandi, H., Ghanbari, A., & Naghneh, S. A. (2012). Developing a hybrid artificial intelligence model for outpatient visits forecasting in hospitals. *Applied Soft Computing*, *12*, 700–711.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: MIT Press. Reprinted in 1998.

Hsu, S. H., Hsieh, J. P. A., Chih, T. C., & Hsu, K. C. (2009). A two-stage architecture for stock price forecasting by integrating self-organizing map and support vector regression. *Expert Systems with Applications*, *36*, 7947–7951.

Huang, C. M., & Wang, F. L. (2007). An RBF network with OLS and EPSO algorithms for real-time power dispatch. *IEEE Transaction on Power Systems*, *22*(1), 96–104.

Jaipuria, S., & Mahapatra, S. S. (2014). An improved demand forecasting method to reduce bullwhip effect in supply chains. *Expert Systems with Applications*, *41*, 2395–2408.

Jin, D., Wang, P., Bai, Z., Wang, X., Peng, H., Qi, R., et al. (2011). Analysis of bacterial community in bulking sludge using culture-dependent and-independent approaches. *Journal of Environmental Sciences*, *23*, 1880–1887.

Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing*, *8*(1), 687–697.

Katherasan, D., Elias, J. V., Sathiya, P., & Haq, A. N. (2014). Simulation and parameter optimization of flux cored arc welding using artificial neural network and particle swarm optimization algorithm. *Journal of Intelligent Manufacturing*, *25*, 67–76.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks* (pp. 1942–1948). Perth: IEEE Service Center.

Kennedy, J., & Eberhart, R. C. (2001). *Swarm intelligence*. San Mateo, CA: Morgan Kaufmann.

Ketabchi, H., & Ataie-Ashtiani, B. (2015). Evolutionary algorithms for the optimal management of coastal groundwater: A comparative study toward future challenges. *Journal of Hydrology*, *520*, 193–213.

Kmenta, J. (1986). *Elements of econometrics* (2nd ed.). New York: Macmillan Publishing Co.

Kohonen, T. (1987). *Self-organizing and associative memory* (2nd ed.). Berlin: Springer.

Kohonen, T. (1990). The self-organizing map. *Proceedings of IEEE*, *78*(9), 1464–1480.

Kuo, R. J., & Han, Y. S. (2011). A hybrid of genetic algorithm and particle swarm optimization for solving bi-level linear programming problem: A case study on supply chain mode. *Applied Mathematical Modelling*, *35*(8), 3905–3917.

Kuo, R. J., Syu, Y. J., Chen, Z. Y., & Tien, F. C. (2012). Integration of particle swarm optimization and genetic algorithm for dynamic clustering. *Information Sciences*, *195*, 124–140.

Kurt, I., Ture, M., & Kurum, A. T. (2008). Comparing performances of logistic regression, classification and regression tree, and neural networks for predicting coronary artery disease. *Expert Systems with Applications*, *34*, 366–374.

Kuzmanovski, I., Lazova, S. D., & Aleksovska, S. (2007). Classification of perovskites with supervised self-organizing maps. *Analytica Chimica Acta*, *595*, 182–189.

Lee, Z. J. (2008). A novel hybrid algorithm for function approximation. *Expert Systems with Applications*, *34*, 384–390.

Lin, C. F., Wu, C. C., Yang, P. H., & Kuo, T. Y. (2009). Application of Taguchi method in light-emitting diode backlight design for wide color gamut displays. *Journal of Display Technology*, *5*(8), 323–330.

Lin, G., & Wu, M. (2009). A hybrid neural network model for typhoon-rainfall forecasting. *Journal of Hydrology*, *375*, 450–458.

Lin, G. F., & Wu, M. C. (2011). An RBF network with a two-step learning algorithm for developing a reservoir inflow forecasting model. *Journal of Hydrology*, *405*, 439–450.

Looney, C. G. (1996). Advances in feedforward neural networks: Demystifying knowledge acquiring black boxes. *IEEE Transactions on Knowledge and Data Engineering*, *8*(2), 211–226.

Lopez, M., Valero, S., Senabre, C., Aparicio, J., & Gabaldon, A. (2012). Application of SOM neural networks to short-term load forecasting: The Spanish electricity market case study. *Electric Power Systems Research*, *91*, 18–27.

Lu, J., Hu, H., & Bai, Y. (2015). Generalized radial basis function neural network based on an improved dynamic particle swarm optimization and AdaBoost algorithm. *Neurocomputing*, *152*, 305–315.

Mezura-Montes, E., & CoelloCoello, C. A. (2011). Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm Evol. Comput.*, *1*(22), 173–194.

Olabi, A. G. (2008). Using Taguchi method to optimize welding pool of dissimilar laser-welded components. *Optics & Laser Technology*, *40*, 379–388.

Ozturk, C., Hancer, E., & Karaboga, D. (2015). A novel binary artificial bee colony algorithm based on genetic operators. *Information Sciences*, *297*, 154–170.

Qasem, S. N., Shamsuddin, S. M., & Zain, A. M. (2012). Multi-objective hybrid evolutionary algorithms for radial basis function neural network design. *Knowledge-Based Systems*, *27*, 475–497.

Qiu, X., & Lau, H. Y. K. (2014). An AIS-based hybrid algorithm for static job shop scheduling problem. *Journal of Intelligent Manufacturing*, *25*, 489–503.

Rezaee-Jordehi, A., & Jasni, J. (2013). Parameter selection in particle swarm optimisation: A survey. *Journal of Experimental & Theoretical Artificial Intelligence*, *25*(4), 527–542.

Rocha, I. B. C. M., Parente, E, Jr., & Melo, A. M. C. (2014). A hybrid shared/distributed memory parallel genetic algorithm for optimization of laminate composites. *Composite Structures*, *107*, 288–297.

Rumbell, T., Denham, S. L., & Wennekers, T. (2014). A spiking self-organizing map combining STDP, oscillations, and continuous learning. *IEEE Transaction on Neural, Networks and Learning Systems*, *25*(5), 894–907.

Sarimveis, H., Alexandridis, A., Mazarakis, S., & Bafas, G. (2004). A new algorithm for developing dynamic radial basis function neural network models based on genetic algorithms. *Computers and Chemical Engineering*, *28*, 209–217.

Sattari, M. T., Yurekli, K., & Pal, M. (2012). Performance evaluation of artificial neural network approaches in forecasting reservoir inflow. *Applied Mathematical Modelling*, *36*, 2649–2657.

Savsani, P., Jhala, R. L., & Savsani, V. (2014). Effect of hybridizing biogeography-based optimization (BBO) technique with artificial immune algorithm (AIA) and ant colony optimization (ACO). *Applied Soft Computing*, *21*, 542–553.

Shafie-khah, M., Moghaddam, M. P., & Sheikh-El-Eslami, M. K. (2011). Price forecasting of day-ahead electricity markets using a hybrid forecast method. *Energy Conversion and Management*, *52*, 2165–2169.

Shelokar, P. S., Siarry, P., Jayaraman, V. K., & Kulkarni, B. D. (2007). Particle swarm and colony algorithms hybridized for improved continuous optimization. *Applied Mathematics and Computation*, *188*, 129–142.

Shi, Y., & Eberhart, R. C. (1998). Parameter selection in particle swarm optimization. In *Evolutionary programming VII: Proceedings of EP98* (pp. 591–600). New York: Springer.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In J. D. Sehaffer (Ed.), *Proceedings of the third international conference on genetic algorithms and their applications* (pp. 2–9). San Mateo: CA Morgan Kaufmann Publishers.

Taguchi, G., Chowdhury, S., & Wu, Y. (2005). *Taguchi's quality engineering handbook*. Hoboken, NJ: Wiley.

Taguchi, G., & Yokoyama, T. (1993). *Taguchi methods: Design of experiments*. Dearbon, MI: ASI Press.

Tsai, J. T., Chou, J. H., & Liu, T. K. (2006). Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm. *IEEE Transactions on Neural Networks*, *17*(1), 69–80.

Tsekouras, G. E., & Tsimikas, J. (2013). On training RBF neural networks using input–output fuzzy clustering and particle swarm optimization. *Fuzzy Sets and Systems*, *221*, 65–89.

Vitorino, L. N., Ribeiro, S. F., & Bastos-Filho, C. J. A. (2015). A mechanism based on artificial bee colony to generate diversity in particle swarm optimization. *Neurocomputing*, *148*, 39–45.

Wang, D., & Lu, W. Z. (2006). Forecasting of ozone level in time series using MLP model with a novel hybrid training algorithm. *Atmospheric Environment*, *40*, 913–924.

Wang, W. M., Peng, X., Nhu, G. N., Hu, J., & Peng, Y. H. (2014). Dynamic representation of fuzzy knowledge based on fuzzy petri net and genetic-particle swarm optimization. *Expert Systems with Applications*, *41*, 1369–1376.

Whitehead, B. A., & Choate, T. D. (1996). Cooperative–competitive genetic evolution of radial basis function centers and widths for time series prediction. *IEEE Transaction on Neural Networks*, *7*(4), 869–880.

Wu, J. D., & Liu, J. C. (2012). A forecasting system for car fuel consumption using a radial basis function neural network. *Expert Systems with Applications*, *39*, 1883–1888.

Xu, R., Venayagamoorthy, G. K., & Wunsch, D. C. (2007). Modeling of gene regulatory networks with hybrid differential evolution and particle swarm optimization. *Neural Networks*, *20*, 917–927.

Yadav, V., & Srinivasan, D. (2011). A SOM-based hybrid linear-neural model for short-term load forecasting. *Neurocomputing*, *74*, 2874–2885.

Yousefi, M., Enayatifar, R., & Darus, A. N. (2012). Optimal design of plate-fin heat exchangers by a hybrid evolutionary algorithm. *International Communications in Heat and Mass Transfer*, *39*, 258–263.

Yu, L., Wang, S., Lai, K. K., & Wen, F. (2010). A multiscale neural network learning paradigm for financial crisis forecasting. *Neurocomputing*, *73*, 716–725.

Yu, S., Wang, K., & Wei, Y. M. (2015a). A hybrid self-adaptive particle swarm optimization-genetic algorithm-radial basis function model for annual electricity demand prediction. *Energy Conversion & Management*, *91*, 176–185.

Yu, S., Wei, Y. M., & Wang, K. (2012a). A PSO–GA optimal model to estimate primary energy demand of China. *Energy Policy*, *42*, 329–340.

Yu, S., Wei, Y. M., & Wang, K. (2012b). China's primary energy demands in 2020: Predictions from an MPSO–RBF estimation model. *Energy Conversion & Management*, *61*, 59–66.

Yu, S., Zhang, J., Zheng, S., & Sun, H. (2015b). Provincial carbon intensity abatement potential estimation in China: A PSO-GA-optimized multi-factor environmental learning curve method. *Energy Policy*, *77*, 46–55.

Yu, S., Zhu, K., & Gao, S. (2009). A hybrid MPSO-BP structure adaptive algorithm for RBFNs. *Neural Computing and Applications*, *18*, 769–779.

Zhang, Z., Su, S., Lin, Y., Cheng, X., Shuang, K., & Xu, P. (2015a). Adaptive multi-objective artificial immune system based virtual network embedding. *Journal of Network and Computer Applications*, *53*, 140–155.

Zhang, J. L., Zhang, Y. J., & Zhang, L. (2015b). A novel hybrid method for crude oil forecasting. *Energy Economics*, *49*, 649–659.

Zou, H. F., Xia, G. P., Yang, F. T., & Wang, H. Y. (2007). An investigation and comparison of artificial neural network and time series models for Chinese food grain price forecasting. *Neurocomputing*, *70*, 2913–2923.