Tuning of neural networks using particle swarm optimization to model MIG welding process

Rakesh Malviya, Dilip Kumar Pratihar

Soft Computing Laboratory, Department of Mechanical Engineering, Indian Institute of Technology, Kharagpur, Kharagpur-721302, West Bengal, India

ARTICLE INFO

Article history:
Received 11 March 2011
Received in revised form
30 June 2011
Accepted 4 July 2011
Available online 22 August 2011

Keywords:
Particle swarm optimization
Neural networks
MIG welding
Forward and reverse mappings
Clustering
Hybrid algorithm

ABSTRACT

Particle swarm optimization technique has been used for tuning of neural networks utilized for carrying out both forward and reverse mappings of metal inert gas (MIG) welding process. Four approaches have been developed and their performances are compared to solve the said problems. The first and second approaches deal with tuning of multi-layer feed-forward neural network and radial basis function neural network, respectively. In the third and fourth approaches, a back-propagation algorithm has been used along with particle swarm optimization to tune radial basis function neural network. Moreover, in these two approaches, two different clustering algorithms have been utilized to decide the structure of the network. The performances of hybrid approaches (that is, the third and fourth approaches) are found to be better than that of the other two.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

In order to automate any process, it might be required to know its input–output relationships both in forward and reverse directions, on-line. In forward mapping, each response of a process is expressed as the function of input parameters, whereas reverse mapping aims to determine the said relationship in opposite direction.

Artificial neural networks (ANNs) can be used to tackle the said problem of modeling the manufacturing processes involving multiple inputs and outputs. In this regard, Feed-forward neural networks (FFNNs) may be suitable to establish input–output relationships of a process. In FFNNs, information is passed into one direction, that is, starting from the input layer toward the output layer through the hidden layer. A few FFNNs, such as Multi-layer feed-forward neural networks (MLFFNNs) and Radial basis function neural networks (RBFNNs) have been used to establish the said relationship. It is to be noted that the RBFNN may be a better option for learning and adaptation of the input–output relationships, because of its less computational complexity compared to MLFFNNs, although they can ensure almost the same level of accuracy in predictions. The performance of RBFNN depends upon its structure decided by the number of its hidden neurons, parameters of radial basis functions used, connecting weights between the hidden and output layers of the networks and others. It is important to mention that various methods have been adopted to decide an appropriate number of hidden neurons for the RBFNNs. The number of hidden neurons of an RBFNN may also be kept equal to the number of clusters formed by the input–output data to be modeled. In order to improve the performance of neural network, its parameters are to be properly tuned using an optimizer with the help of some training scenarios. Particle swarm optimization (PSO) [1] is one of the population-based evolutionary computing techniques, where the population is called swarm. It can carry out both global and local searches simultaneously, whereas a Genetic algorithm (GA) [2] concentrates mainly on the global search. However, most of the stochastic optimization algorithms (including PSO and GA) may suffer from the curse of dimensionality [3], which simply implies that their performances degrade with the dimension of search space. To avoid this difficulty, a hybrid computing scheme may be adopted, in which a part of the optimization task will be carried out using a global cum local optimizer like PSO and the remaining part will be handled using conventional local optimizer named back-propagation algorithm.

2. Literature review

The concept of PSO algorithm was introduced by Kennedy and Eberhart [1] in 1995. It is a population-based search algorithm, which is initialized with the population of random solutions, called particles, and the population is known as swarm.
Several modifications in the PSO algorithm had been done by various researchers. Shi and Eberhart [4] introduced a new parameter called inertia weight into the original PSO algorithm, which played an important role of balancing the global and local searches. Clerc and Kennedy [5] analyzed how a particle carries out search in a complex problem space and modified the original PSO on the basis of this analysis. Zhenya et al. [6] extracted rules for the welding. Fig. 3 shows the photograph of macro-etched sample steelplates of dimension: 150 mm × 8 mm and having the composition (in %): C—0.10, Mn—0.9, S—0.032, P—0.032. Direct current electrode positive (DCEP) polarity was used to perform the welding. It was observed that a combined optimization scheme of one global (that is, GA) and one local (back-propagation algorithm) could be an efficient technique to optimize both the structure and parameters of the ANNs. It is important to mention that GA is a powerful tool for global search but its local search capability is poor. On the other hand, PSO can carry out both the local and global searches simultaneously. The objective of the present study is to conduct the input-output modeling of an MIG welding process in both forward and reverse directions with the help of neural networks tuned by the modified-PSO.

### 3. Data collection

Input and output parameters of MIG welding process considered in the present study are shown in Fig. 1. Input-output data of this process were experimentally collected by Ganjigatti [18], which have been used in the present study. Table 1 displays the ranges of various input parameters considered in this study. As six input process parameters and two levels for each of them were considered, there were 26 = 64 combinations of input process parameters, for which the responses, namely bead height (BH), bead width (BW) and bead penetration (BP) had been obtained experimentally.

A single-pass bead-on-plate welding was conducted on mild steel plates of dimension: 150 mm × 75 mm × 8 mm and having the composition (in %): C—0.10, Mn—0.9, S—0.032, P—0.032. Direct current electrode positive (DCEP) polarity was used to perform the welding. Fig. 3 shows the photograph of macro-etched sample.
### Table 1
Ranges of input parameters of MIG welding process.

<table>
<thead>
<tr>
<th>Input parameter</th>
<th>Unit</th>
<th>Notation</th>
<th>Minimum value</th>
<th>Maximum value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Welding speed</td>
<td>cm/min</td>
<td>S</td>
<td>25</td>
<td>45</td>
</tr>
<tr>
<td>Arc voltage</td>
<td>V</td>
<td>V</td>
<td>26</td>
<td>30</td>
</tr>
<tr>
<td>Wire feed rate</td>
<td>m/min</td>
<td>F</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Gas flow rate</td>
<td>1/min</td>
<td>G</td>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>Nozzle-to-plate distance</td>
<td>mm</td>
<td>D</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>Torch angle</td>
<td>degree</td>
<td>A</td>
<td>70</td>
<td>100</td>
</tr>
</tbody>
</table>

Non-linear regression analysis was carried out with the help of the said 64 combinations of input process parameters and the following response equations were obtained [18]:

\[
BH = 10^{1.1215 \times F^{0.3416} \times D^{0.1027} / S^{0.4034} \times V^{1.3077} \times G^{0.0266} \times A^{0.2509}} 
\]

\[
BW = 10^{1.2239 \times S^{0.3566} \times G^{0.0666} / V^{2.6382} \times F^{0.9482} \times A^{0.13767}} 
\]

\[
BP = 10^{0.97682 \times D^{0.26514} \times G^{0.01051} \times A^{0.55368} / V^{1.26382} \times S^{0.94822} \times G^{0.13767}} 
\]

A batch mode of training has been provided to the neural networks in the present study, which requires a large number (either equal to or more than the number of variables considered during updating of the networks) of training data. In the present work, a set of 1000 training scenarios containing 64 data collected through experiment (with the help of which the regression model had been built) and 936 data obtained artificially utilizing the above regression equations, has been utilized.

4. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO), introduced by Kennedy and Eberhart [1] in 1995, is a stochastic population-based evolutionary computation technique. It has similarity with bird flocking, fish schooling or sociological behavior of a group of people. It has been used to solve a variety of optimization problems. In PSO, the population of solution is known as swarm, which is composed of a number of agents called particles moving around the search space to look for the best solution. Each particle is treated as a point in an N-dimensional space, which modifies its flying according to its own flying experience and that of other particles. Each particle keeps a track of its coordinates in the solution space, which are associated with the best solution in terms of fitness achieved so far by it. This value is called personal best, Pbest. Another best value that is tracked by the PSO is that obtained so far by any particle lying in its neighborhood. This value is called the global best, Gbest. The basic concept of PSO lies in accelerating each particle toward its Pbest and Gbest locations, with a random weighted acceleration at each time step.

PSO algorithm

Similar to other population-based optimization methods, such as GAs, the PSO starts with the random initialization of population particles in the search space. The PSO algorithm works based on the social behavior of particles in the swarm. Therefore, it finds the global best solution by simply adjusting the trajectory of each individual toward its own best location and toward the best particle of the entire swarm at each time step (generation) [1,5]. In PSO algorithm, the trajectory of each individual is adjusted...
Fig. 4. A flowchart of PSO algorithm.

\begin{align*}
v_{id}(t+1) &= w v_{id}(t) + c_1 \text{rand}(\cdot) (p_{id} - x_{id}(t)) \\
&\quad + c_2 \text{Rand}(\cdot) (p_{gd} - x_{id}(t)), \quad (4a) \\
x_{id}(t+1) &= x_{id}(t) + v_{id}(t+1), \quad (4b)
\end{align*}

where $v_{id}$ is the velocity of $d$-th dimension of the $i$-th particle, $w$ is a constant known as inertia weight [20] $c_1$ and $c_2$ denote the acceleration coefficients, and rand$(\cdot)$ and Rand$(\cdot)$ are two separately generated uniformly distributed random numbers lying in the range of $[0, 1]$. The first part of Eq. (4a) represents the previous velocity, which provides the necessary momentum to the particles to move across the search space. The second part of this equation, known as the cognitive component, represents the personal thinking of each particle. The cognitive component encourages the particles to move toward their respective best positions found so far. The third part is known as the social component, which indicates the collaborative effect of the particles in finding the global optimal solution. The social component always pulls the particles toward the global best particle found so far. The PSO algorithm has become popular due to its simple architecture, ease of implementation and ability to quickly converge to a reasonably good solution. The flowchart of PSO algorithm is shown in Fig. 4.

5. Clustering algorithms used

Clustering is a powerful tool for data mining, whose aim is to extract useful information from a set of data. A clustering technique analyzes the pattern of the data set and groups the data into several clusters based on their similarity among each other.

5.1. Fuzzy C-means clustering

Fuzzy C-means (FCM) clustering [21] is one of the popular fuzzy clustering techniques, in which a particular data of the set may be the member of several clusters with different values of membership. In this algorithm, an attempt has been made to minimize the dissimilarity measure (expressed in terms of Euclidean distance) of the data with pre-defined clusters. Let us consider $N$ data points having $L$ dimensions, which are to be clustered.

dynamically in the search space by changing the velocity of each particle, according to its own flying experience and that of other particles. The position and velocity vectors of $i$-th particle in the $d$-dimensional search space can be represented as $X_i = (x_{i1}, \ldots, x_{id})$ and $V_i = (v_{i1}, \ldots, v_{id})$, respectively. The value of $V_i$ vector is kept fixed in the range of $[-v_{\text{max}}, v_{\text{max}}]$ to reduce the tendency of particles to leave the search space. The value of $v_{\text{max}}$ is usually chosen to be equal to $k \times x_{\text{max}}$, where $0.1 \leq k \leq 1.0$ [19]. According to a user defined fitness function, let us say that the best position of each particle (which corresponds to the best fitness value obtained by that particle at time $t$) is $P_i = (p_{i1}, \ldots, p_{id})$ and the fittest particle found so far at time $t$ is $P_{\text{gd}} = (p_{g1}, \ldots, p_{gd})$. The new velocities and positions of the particles for the next fitness evaluation are then calculated as follows:

\begin{align*}
v_{id}(t+1) &= w v_{id}(t) + c_1 \text{rand}(\cdot) (p_{id} - x_{id}(t)) \\
&\quad + c_2 \text{Rand}(\cdot) (p_{gd} - x_{id}(t)), \quad (4a) \\
x_{id}(t+1) &= x_{id}(t) + v_{id}(t+1). \quad (4b)
\end{align*}
The algorithm consists of the following steps [22]:

- **Step 1**: Assume the number of clusters is to be made \( C \), where \( 2 \leq C \leq N \).
- **Step 2**: Choose an appropriate level of cluster fuzziness \( f > 1 \).
- **Step 3**: Initialize the \( N \times C \) sized membership matrix \([U]\) at random, such that \( U_{ij} \in [0.0, 1.0] \) and \( \sum_{j=1}^{C} U_{ij} = 1.0 \), for each \( i \).
- **Step 4**: Calculate the \( k \)-th dimension of \( j \)-th cluster center \( CC_{jk} \) using the expression given below:
  \[
  CC_{jk} = \frac{\sum_{i=1}^{N} U_{ij} x_{ik}}{\sum_{i=1}^{N} U_{ij}}.
  \]
- **Step 5**: Calculate the Euclidean distance between \( i \)-th data point and \( j \)-th cluster center like the following:
  \[
  d_{ij} = \| (C_j - x_i) \|.
  \]
- **Step 6**: Update fuzzy membership matrix according to \( d_{ij} \). If \( d_{ij} > 0 \), then
  \[
  U_{ij} = \frac{1}{\sum_{m=1}^{C} \left( \frac{d_{ij}}{d_{im}} \right)^{2f-1}}.
  \]
  If \( d_{ij} = 0 \), then the data point coincides with the \( j \)-th cluster center \( C_j \) and it will have the full membership value, that is, \( U_{ij} = 1.0 \).
- **Step 7**: Repeat the steps 4 through 6 until changes in \([U]\) come out to be less than some pre-specified value.

### 5.2. Entropy-based fuzzy clustering

The FCM algorithm cannot determine the number of clusters, into which the data set is to be divided. On the other hand, Entropy-based Fuzzy Clustering (EFC) [23] identifies the number of clusters and their locations by itself. Here, entropy (probability) values of the points are determined based on a similarity measure. The similarity of two points depends on the Euclidean distance between them. The point having the minimum entropy value is selected as the cluster center.
Let us suppose that the input dataset \([T]\) contains \(N\) points having \(L\) dimensions. Thus, \([T]\) is an \(N \times L\) matrix. The algorithm consists of the following steps [22]:

- **Step 1**: Arrange the data set \(N\) rows and \(L\) columns.
- **Step 2**: Calculate the Euclidean distances between the points \(i\) and \(j\) as follows:
  \[
  d_{ij} = \sqrt{\sum_{k=1}^{L} (X_{ik} - X_{jk})^2}.
  \]
- **Step 3**: Determine the similarity \(S_{ij}\) between two points \(i\) and \(j\) by using the expression given below.
  \[
  S_{ij} = e^{-ad_{ij}},
  \]
  where \(a\) is a constant. It is important to note that similarity \(S_{ij}\) varies in the range of \((0.0, 1.0)\).
- **Step 4**: Calculate total entropy value of a data point \(X_i\) with respect to all other points using the expression given below.
  \[
  E_i = -\sum_{j \neq i} \left( S_{ij} \log_2 S_{ij} + (1 - S_{ij}) \log_2 (1 - S_{ij}) \right).
  \]
- **Step 5**: Identify \(X_i\) that has the minimum \(E_i\) value and select it (\(X_{i,\text{min}}\)) as the cluster center.
- **Step 6**: Put \(X_{i,\text{min}}\) and the data point having similarity with \(X_{i,\text{min}}\) greater than \(\beta\) (which is a threshold value of similarity) in a cluster and remove them from \([T]\).
- **Step 7**: Check if \([T]\) is empty. If yes, terminate the program, else go to Step 6.

6. Developed approaches

Four approaches have been developed using ANNs to model input–output relationships of MIG welding process in both forward and reverse directions, as discussed below.

**Approach 1: PSO-based tuning of multi-layer feed-forward neural network parameters**

A multi-layer feed-forward neural network (MLFFNN) has been used to capture input–output relationships of the MIG welding process (refer to Fig. 5). The number of neurons of the input layer has been kept equal to the number of input process parameters (that is, six) and that of neurons of output layer is kept fixed to three (that is, number of outputs of the process). Thus, the structure
of the network is dependent on the number of neurons lying in the hidden layer, which has been decided in the present approach through a parametric study by varying one parameter at a time and keeping the others unaltered. A PSO algorithm has been utilized to update the parameters like connecting weights between input and hidden layers and that between hidden and output layers, coefficients of transfer functions of the hidden and output layers.

A batch mode of training [22] has been adopted with the help of 1000 training scenarios and the fitness (goodness) of a solution is calculated as mean squared error in predictions, which is to be minimized. Once trained, the performance of the network has been evaluated with the help of 27 test data obtained through real experiments.

**Approach 2: PSO-based tuning of radial-basis function neural network parameters**

A Radial-basis function neural network (RBFNN) has been used in this approach, in place of MLFFNN because of its less computational complexity as compared to the latter after ensuring almost a same level of prediction capability to capture input–output relationships of the MIG welding process (refer to Fig. 6). The number of nodes has been kept equal to that of input process parameters (that is, six) and that of output neurons is kept fixed to three. The performance of RBFNN with Gaussian transfer functions at the hidden layer depends upon the number of hidden neurons (obtained through a parametric study) and their centers and radii values, which are optimized using the PSO. The fitness values of the solutions have been determined as explained in approach 1 and a batch mode of training has been adopted in this approach.

**Approach 3: Determination of optimal structure of the RBFNN using PSO-driven FCM algorithm and optimization of connecting weights utilizing back-propagation algorithm**

In this approach, input–output data have been clustered using the FCM algorithm, the performance of a clustering algorithm depends on predefined number of cluster to be made and level of fuzziness. The structure of RBFNN is dependent on the number of hidden neurons, which is kept equal to the number of clusters formed by the training data. Here, the parameters like number of cluster centers (C), level of fuzziness (f), learning rate (η) and momentum factor (α) of back-propagation algorithm, and constant of log-sigmoid of transfer function (λ) have been optimized by the PSO. As the GA has to handle a large number of variables, it may suffer from the well-known permutation problem [22] to optimize the connecting weights, centers and other parameters, and consequently, it may not be able to reach the globally optimal solution. However, the parameters like the connecting weights, mean and standard deviation values of the Gaussian distributions have been updated using the back-propagation algorithm. Thus, a hybrid optimization scheme has been adopted in this approach.
7. Results and discussion

Forward and reverse mappings of MIG welding process have been carried out using the neural networks developed through the above four approaches. Results of the said mappings are stated and discussed as follows.

7.1. Results of forward mapping

In forward mapping, the neural network has been designed and developed in order to get the optimum outputs corresponding to the set of input parameters. In approach 1, the number of hidden neurons is obtained first through a thorough parametric study (by varying one parameter at a time and keeping the others fixed) and it is found to be equal to ten. During the PSO-based tuning of the network, the parameters, namely connecting weights between input-hidden layers, and hidden-output layers and coefficients of log-sigmoid and tan-sigmoid transfer function are varied in the ranges of \((-1.0, 1.0), (-1.0, 1.0), (0.5, 5)\) and \((0.5, 5)\), respectively. The performance of PSO depends on the selection of its parameters, namely number of executions in each run and number of runs, while the other parameters like \(w, c_1\) and \(c_2\) are kept fixed to 0.721348, 1.93147 and 1.93147, respectively. Through a thorough study, the optimal values of PSO-parameters are found to be as follows: number of executions in each run \(= 40,000\) and number of runs \(= 100\). The performance of the PSO-tuned MLPNN has been tested on 27 test cases and the value of average absolute percent deviation in predictions is found to be equal to 15.89.

In approach 2, the number of neurons has been kept fixed to ten. Through an exhaustive study by varying one parameter at a time, the optimal PSO-parameters are found to be as follows: number of executions in each run \(= 50,000\) and number of runs \(= 200\). During the PSO-based tuning of neural network, the parameters, namely cluster centers, standard deviations of Gaussian distributions, connecting weights of hidden-output layers and coefficients of log-sigmoid transfer functions are varied in the ranges of \((0.0, 1.0), (0.01, 1.0), (-1.0, 1.0)\) and \((0.5, 5.0)\), respectively. The performance of the PSO-tuned RBFNN has been tested on 27 test cases and it has yielded the value of average absolute percent deviation in predictions as 7.98.

In approach 3, during the PSO-based tuning of neural network, the parameters, namely \(C, f, \eta, \alpha\) and \(\lambda\) have been varied in the ranges of \((2, 12), (1.01, 2.0), (0.4, 1.0), (0.4, 1.0)\) and \((0.5, 5.0)\) respectively and the corresponding optimal values are found to be equal to 12, 1.09074, 0.985767, 0.560896 and 3.232785, respectively. The best results are obtained with the following PSO-parameters: number of executions in each run \(= 20,000\) and number of runs \(= 50\). The performance of the PSO-tuned RBFNN has been verified on 27 test cases and the value of average absolute percent deviation in predictions has turned out to be equal to 8.81.

In approach 4, the values of \(\gamma, \beta, \eta, \alpha\) and \(\lambda\) have been varied in the ranges of \((0.4, 1.0), (0.2, 0.45), (0.4, 1.0), (0.4, 1.0)\) and \((0.5, 5.0)\), respectively, during optimization and their optimal values are...
Table 2
Comparisons of the results of forward mapping obtained using four developed approaches in the present study with those of Ganjigatti [18].

<table>
<thead>
<tr>
<th>Approaches developed in present study</th>
<th>Avg. absolute % deviation in predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach 1</td>
<td>15.89</td>
</tr>
<tr>
<td>Approach 2</td>
<td>7.98</td>
</tr>
<tr>
<td>Approach 3</td>
<td>8.81</td>
</tr>
<tr>
<td>Approach 4</td>
<td>6.85</td>
</tr>
</tbody>
</table>

Approaches developed by Ganjigatti [18]

| Approach 1                            | 9.18                                     |
| Approach 2                            | 10.15                                    |
| Approach 3                            | 8.21                                     |

Fig. 13. Comparison of different approaches in terms of % deviation in prediction of BP on the test data set.

Fig. 14. Target vs. predicted values of S.

found to be equal to 0.984644, 0.395346, 1.000000, 0.881018 and 0.667510, respectively. The following PSO-parameters are seen to yield the best results: number of executions in each run = 20,000 and number of runs = 50. The PSO-tuned RBFNN has yielded the value of average absolute percent deviation in predictions for 27 cases as 6.85.

The predicted and target values of BH for 27 test cases as obtained by the above four approaches have been compared in Fig. 8, and their corresponding values of percent deviation in predictions have been displayed in Fig. 9. The similar comparisons of the developed approaches have also been made for other responses, namely BW and BP, as shown in Figs. 10–13, respectively. A close watch on these figures and the above values of average absolute percent deviation in predictions indicates that approach 4 has outperformed the other approaches in terms of accuracy in predictions. The performance of approach 3 is also found to be reasonably good. Both in approaches 3 and 4, the RBFNs have been designed in a more systematic and scientific way of data mining using the concept of clustering. The better performance of approaches 4 compared to approach 3 may be due to the use of EFC algorithm in the place of FCM algorithm (as used in approach 3). In general, EFC algorithm is seen to be more flexible and adaptive in comparison with the FCM algorithm.

Comparisons have been made of the above approaches in terms of average absolute % deviation in predictions of the responses with three fuzzy logic-based approaches used by Ganjigatti [18] on the same data set (refer to Table 2).

It is important to mention that the three approaches developed by Ganjigatti [18] were fuzzy logic-based and a GA was used for their tuning in different ways. Approach 4 of the present study is found to outperform all three approaches of Ganjigatti [18] in terms of average absolute percent deviation in predictions.

7.2. Results of reverse mapping

In reverse mapping, the neural networks have been designed to get input process parameters corresponding to a set of desired responses.

In approach 1, the number of hidden neurons is found to be equal to ten through a detailed parametric study. The parameters, namely connecting weights between input-hidden layers, that between hidden-output layers and coefficients of log-sigmoid and tan-sigmoid transfer functions are varied in the ranges of (−1.0, 1.0), (−1.0, 1.0), (0.5, 5) and (0.5, 5), respectively. The optimized values of PSO-parameters are found to be as follows: number of executions in each run = 50,000 and number of runs = 100.
The performance of PSO-tuned MLFFNN has been tested on 27 test cases and the value of average absolute percent deviation in predictions is found to be equal to 11.91.

In approach 2, the following PSO-parameters are seen to yield the best results: number of executions in each run = 50,000 and number of runs = 100. During the PSO-based tuning of neural network, the parameters, namely cluster centers, standard deviation of Gaussian distributions, connecting weights of hidden-output layers and coefficient of log-sigmoid transfer functions are...
Fig. 19. Comparison of different approaches in terms of % deviation in prediction of $G$ on the test data-set.

Fig. 20. Target vs. predicted Values of $D$.

Fig. 21. Comparison of different approaches in terms of % deviation in prediction of $D$ on the test data-set.

Fig. 22. Target vs. predicted values of $F$.

In approach 3, the parameters, such as $C$, $f$, $\eta$, $\alpha$ and $\lambda$, have been varied in the ranges of $(0,0.1)$, $(0.01, 1.0)$, $(-1.0, 1.0)$ and $(0.5, 5.0)$, respectively. The performance of PSO-tuned RBFNN has been tested on 27 test cases and the value of average absolute percent deviation in predictions has turned out to be equal to 10.27.

varied in the ranges of $(0.0, 1.0)$, $(0.01, 1.0)$, $(-1.0, 1.0)$ and $(0.5, 5.0)$, respectively. The performance of PSO-tuned RBFNN has been tested on 27 test cases and the value of average absolute percent deviation in predictions has turned out to be equal to 10.27.
In approach 4, the parameters, namely $\gamma$, $\beta$, $\eta$, $\alpha$ and $\lambda$ have been varied in the ranges of $(0.4, 1.0), (0.2, 0.45), (0.4, 1.0), (0.4, 1.0)$ and $(0.5, 5.0)$ and their optimal values are obtained as 0.884058, 0.265994, 0.933121, 0.825914 and 3.133114, respectively. The optimized PSO-parameters are found to be as follows: number of executions in each run = 20,000 and number of runs = 50. The value of average absolute percent deviation in predictions of the parameters is seen to be equal to 9.07 for the same 27 test cases.

The predicted values of the process parameter $S$ (as obtained by four approaches) have been compared with its corresponding target values for 27 cases as shown in Fig. 14. Moreover, the values of % deviation in the prediction of $S$ have been displayed in Fig. 15 for all four approaches. Similar comparisons have also been made for other process parameters as displayed in Figs. 16 through 25. Once again, approaches 3 and 4 are found to perform better than approaches 1 and 2. It has happened due to the reasons explained earlier. The performances of approaches 3 and 4 are seen to be comparable. A slightly better performance of approach 3 compared to approach 4 may be due to the use of a different clustering algorithm. The performances of clustering algorithms are, in general, data-dependent.

In reverse mapping also, the performances of four developed approaches have been compared with two fuzzy logic-based approaches utilized by Ganjigatti [18] on the same data set. Approach 3 of the present study is seen to perform better than...
both the approaches of Ganjigatti [18] in terms of average absolute percent deviation in predictions (refer to Table 3).

8. Concluding remarks

In the present study, PSO has been used to tune various neural networks assigned to carry out input–output modeling of MIG welding process in both forward and reverse directions. Four approaches have been developed and their performances have been compared in terms of percent deviation in predictions of different parameters. The first and second approaches deal with PSO-tuned MLFFNN and RBFNN, respectively, whereas some hybrid computing schemes have been adopted in approaches 3 and 4. Both the approaches: 3 and 4 have shown either comparable or a slightly better performance compared to the approaches: 1 and 2. Moreover, the performances of approaches 3 and 4 are found to be comparable. It has happened so, because in approaches 3 and 4, a local optimizer like back-propagation algorithm has been used along with the PSO algorithm, although the latter alone is a potential tool for optimization. Moreover, PSO itself has both the local and global search capabilities. In order to optimize a function involving a large number of variables, the PSO may suffer from the well-known permutation problem, and to avoid it, a back-propagation algorithm has been used in approaches 3 and 4. Both the approaches 3 and 4 are able to conduct the forward and reverse modeling efficiently.

9. Scope for future work

Attempts will be made in future to modify the PSO algorithm further and its performance will be compared with that of the present PSO algorithm. In fact, a few stronger PSO-versions have been proposed [24,25] recently, and the authors will be working on these algorithms in order to tune the above neural networks.

References