Tolerance synthesis by neural learning and nonlinear programming

Mu-Chen Chen*

Institute of Commerce Automation and Management, National Taipei University of Technology, 1, Section 3,
Chung-Hsiao E. Road, Taipei 106, Taiwan

Received 8 December 1999; accepted 10 April 2000

Abstract

The tolerance design directly influences the functionality of products and related production costs. Tolerance synthesis is a procedure that distributes assembly tolerances between components or distributes final part design tolerances between related tolerances. In order to make a reliable trade-off between design tolerances and costs, it is necessary to obtain the cost–tolerance relationships. Various operations such as turning, milling, drilling, grinding, casting, etc., have different cost–tolerance relationships. Previous studies have usually established cost–tolerance functions for various manufacturing operations by regression analysis using the empirical data. Using traditional methods of regression analysis, people must make assumptions about the form of the regression equation or its parameters, which may not be valid. The neural network recently has been reported to be an effective statistical tool for determining the relationships between input factors and output responses. This study deals with the optimal tolerance design for an assembly simultaneously considering manufacturing cost and quality loss. In this paper, a backpropagation (BP) network is applied to fit the cost–tolerance relationship. Once the cost–tolerance functions have been generated, mathematical models for tolerance synthesis can be built. By solving the formulated mathematical models, the optimal tolerance allocation can be generated. An optimization method based on simulated annealing (SA) is then used to locate the combination of controllable factors (tolerances) to optimize the output response (manufacturing cost plus quality loss) using the equations stemming from the trained network. A tolerance synthesis problem for a motor assembly is used to investigate the effectiveness and efficiency of the proposed methodology. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Tolerance synthesis; Cost–tolerance relationship; Optimization; Neural network

1. Introduction

Nominal dimensions specify the ideal geometry for size, location and form. The range between the upper and lower limits of the variation from the nominal dimension is called tolerance [1]. Design procedure mainly includes two phases: functional design (product design) and manufacturing design (process design). The tolerance design directly influences the functionality of parts and costs. Tolerance synthesis (optimal tolerance allocation) is a procedure that distributes assembly tolerances between components or distributes final part

*Corresponding author. Tel.: +886-2-2771-2171 ext. 3417; fax: +886-2-2776-3964.
E-mail address: f10937@ntut.edu.tw (M.-C. Chen).
design tolerances between related tolerances [2]. Tolerance synthesis is an essential step in both design phases to assure quality conformity and economic manufacturing. A tighter tolerance is normally preferred for product functionality; however, manufacturing cost usually increases due to the requirement of more rigorous operations. Contrarily, a loose tolerance is less costly; however, it may cause inferior quality. Therefore, it is necessary to make trade-off between tolerances and costs. Traditionally, designers allocate tolerance based on their experience and information contained in design handbooks or standards [3]. This approach does not guarantee functionality or assembly; nor does it minimize costs.

Tolerance synthesis problems have attracted attention for many years. Instead of using traditional methods, recently researchers have developed tolerance synthesis approaches for optimal tolerance designs. Applications of tolerance synthesis require mathematical modeling of cost–tolerance relationships. The trade-off between specification and realization illustrates the traditional conflict between design and manufacturing [4]. Distinct operations have different cost–tolerance relationships. In order to make a reliable trade-off between design tolerances and costs, it is necessary to determine the cost–tolerance relationships. Most researchers agree that there is an inverse relationship between tolerance and cost [3]. Numerous cost–tolerance functions for various manufacturing operations, which include turning, milling, drilling, grinding, casting, etc., are given in the literature [3, 5–9]. They include exponential, reciprocal squared, reciprocal power, reciprocal, discrete, polynomial, B-spline and hybrid form, etc., functions. These functions are established by regression analysis using empirical data from the real manufacturing. Tolerance synthesis is more complicated in an assembly due to the fact that a manufacturing process for an assembly consists of placing various components and subassemblies jointly to establish a finished product with an expected functionality. Once the cost–tolerance relationships have been generated, mathematical models for tolerance synthesis can be built to obtain the optimal tolerance design.

The tolerance allocation among the assembly components is vital to ensure that functionality and design quality are satisfied [10]. In real manufacturing environments, the cost–tolerance relationship exists. However, it is quite difficult to obtain the parameters of cost–tolerance functions. Using traditional methods of regression analysis, one must make assumptions about the form of the regression equation or its parameters, which may not be valid in practice. In addition, the previously developed forms of cost–tolerance relationships may not be suitable for considering the quality loss. The costs incurred in a product life cycle are categorized as manufacturing costs that occur before the product is sold to the customer, and quality losses that occur after the product is sold [11]. The quality loss is also an ideal function for establishing practical manufacturing tolerances. Previously, tolerance synthesis has been formulated as an optimization problem by treating manufacturing cost as an objective function. More recently, researchers [10, 12, 13] adjusted the design tolerances to reach an economic balance between manufacturing cost and quality loss for product tolerance design. The total cost under such a situation takes the form [10, 12]

$$\text{TC}_i = \sum_{j=1}^{q} K_j [(U_{ij} - T_j)^2 + \sigma_{ij}^2] + \sum_{k=1}^{m} C_M(t_{ik}), \quad (1)$$

where $m$ is the total number of components from $q$ assembly dimensions in a finished product, $K_j$ the cost coefficient of the $j$th resultant dimension for quadratic loss function, $U_{ij}$ the $j$th resultant dimension from the $i$th experimental results, $\sigma_{ij}$ the $j$th resultant variance of statistical data from the $i$th experimental results, $T_j$ the design nominal value for the $j$th assembly dimension, $t_{ik}$ the tolerance established in the $i$th experiment for the $k$th component, and $C_M(t_{ik})$ the manufacturing cost for the tolerance $t_{ik}$.

Tolerance synthesis problems are usually formulated as nonlinear programming models. To deal with discrete tolerance optimization, Chase et al. [6] proposed three solution procedures: exhaustive search, univariate search and sequential quadratic programming (SQP). Dong et al. [7] applied the constrained variable matrix (CVM) method to resolve their formulated tolerance allocation problems with hybrid cost–tolerance
functions. Lee et al. [9] transformed their tolerance optimization model to a specific form and applied the Lagrangean multiplier approach to locate the optimal solution. Lee and Wei [13] used the general interactive optimizer (GINO) to allocate tolerances under minimal manufacturing loss criterion. In nonlinear programming, no existing algorithm guarantees a global optimum unless the objective and the constraints are of certain forms (e.g., the model presented by Lee et al. [9]). In the optimization algorithm set, the simulated annealing (SA) algorithm and genetic algorithms (GA) have been reported to be the reliable global optimization methods. Zhang and Wang [2,3] have employed SA, and Al-Ansary and Deiab [5] have used GA to resolve tolerance synthesis problems. From their computational results, SA and GA perform well in complex tolerance optimization problems.

The experimental design incorporating response surface methodology (RSM) also provides an approach for dealing with tolerance synthesis. RSM is a mixture of mathematical and statistical techniques. It applies regression analysis to fit the relationship between input factors (tolerances) and an output response (cost). The gradient search is then used to locate the optimal setting of input factors in which an objective is related to the cost–tolerance relationship.

As mentioned previously, it is quite difficult to obtain the parameters of cost–tolerance functions. The major disadvantage of regression analysis for function fitting is that one must make assumptions about the form of the regression equation. Additionally, regression analysis may be inclined to generate numerous tables of results. These results are frequently difficult for design engineers to interpret without a statistics background. Generally, design tolerances are of a high degree of complexity due to their interrelated features, particularly for an assembly with several components. Therefore, the nonlinear programming models for tolerance synthesis have a high degree of computational complexity.

In this paper, a tolerance synthesis procedure based on backpropagation (BP) neural networks and simulated annealing (SA) is developed. The neural network approach can be regarded as a statistical method. The feature hidden within the designed experiment can be learned by the neural network approach based on the collected data. In addition, a neural network can be constructed without requiring any assumptions concerning the functional form of the relationship between predictors and responses [14]. Therefore, the neural network approach outperforms the conventional statistical modeling approach in terms of analyzing experimental data [15]. The solution surface from a trained neural network may become very noisy. A stochastic local search, SA, is then applied for tolerance optimization using the cost–tolerance functions generated from the trained network. The SA-based optimization algorithm has several promising properties: ease of implementation, wide applicability, high-quality solution and robustness.

The remainder of this paper is organized as follows. Section 2 introduces the application of a BP network for building cost–tolerance functions. Section 3 presents the proposed optimization procedure based on SA. In Section 4, a motor assembly is taken as a design example for tolerance synthesis using the proposed methodology. Finally, conclusions are made in Section 5.

2. Neural network-based cost–tolerance functions

2.1. Overview of the backpropagation network

Neural networks have received a lot of attention in many research and application areas. One of the major benefits of neural networks is the adaptive ability of their generalization of data from the real world. Exploiting this advantage, many researchers apply neural networks for nonlinear regression analysis and have reported positive experimental results in their applications [14]. Recently, neural networks have received a great deal of attention in manufacturing areas. Zhang and Huang [16] presented an extensive review of neural network applications in manufacturing.

Neural networks are defined by Rumelhart and McClelland [17] as "massively parallel interconnected networks of simple (usually adaptive) elements and their hierarchical organizations which are intended to interact with objects of the real world in the same way as biological nervous
systems do". The approach towards constructing the cost–tolerance relationships is based on a supervised backpropagation (BP) neural network. Among several well-known supervised neural networks, the BP model is the most extensively used and can provide good solutions for many industrial applications [18].

A BP network is a feed-forward network with one or more layers of nodes between the input and output nodes. An imperative item of the BP network is the iterative method that propagates the error terms required to adopt weights back from nodes in the output layer to nodes in lower layers. The training of a BP network involves three stages: the feed forward of the input training pattern, the calculation and BP of the associated error, and the adjustment of the weights. After the network reaches a satisfactory level of performance, it will learn the relationships between input and output patterns and its weights can be used to recognize new input patterns.

Fig. 1 depicts a BP network with one hidden layer. The hidden nodes of the hidden layer perform an important role in creating internal representation. The following nomenclatures are used for describing the BP learning rule.

\[ \text{net}_{pi} = \sum_j w_{ij}a_{pj}, \quad (2) \]
\[ a_{pi} = \frac{1}{1 + \exp(\text{net}_{pi})}. \quad (3) \]

The net input and the activation values of the middle processing nodes are calculated as follows:

The net input is the weighted sum of activation values of the connected input units plus a bias value. Initially, the connection weights are assigned randomly and are varied continuously. The activation values are in turn used to calculate the net inputs and the activation values of the output processing units using the same Eqs. (2) and (3).

Once the activation values of the output units are calculated, we compare the target value with the activation value of each output unit. The discrepancy is propagated backward using

\[ \delta_{pi} = (g_{pi} - a_{pi})f'(\text{net}_{pi}). \quad (4) \]

For the hidden processing units in which the target values are unknown, instead of Eq. (4), the following equation is used to calculate the discrepancy. It takes the form

\[ \delta_{pi} = f'(\text{net}_{pi})\sum_k \delta_{pk}w_{ki}. \quad (5) \]

From the results of Eqs. (4) and (5), the weights between processing units are adjusted using

\[ \Delta w_{ij} = \varepsilon \delta_{pi}a_{pj}. \quad (6) \]

The detailed BP training algorithm can be found in Rumelhart and McClelland [17]. The basic step of tolerance synthesis is to provide the appropriate fitting for cost–tolerance functions. Neural networks have several unique characteristics that allow them to perform the complex function modeling: (1) neural networks can be used to accurately represent response surface models of complex systems [16]; (2) neural networks can emulate human knowledge fusion capabilities by forming a single coherent model from a variety of partial knowledge sources.

2.2. Constructing cost–tolerance functions

This subsection presents the application of BP neural network techniques for constructing
Table 1

<table>
<thead>
<tr>
<th>Tolerance $t$</th>
<th>Cost $C_M(t)$</th>
<th>$C_{RG}(t)$</th>
<th>$C_{NN}(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.670</td>
<td>2.897</td>
<td>3.379</td>
</tr>
<tr>
<td>2</td>
<td>2.370</td>
<td>1.864</td>
<td>2.289</td>
</tr>
<tr>
<td>3</td>
<td>1.070</td>
<td>1.120</td>
<td>1.016</td>
</tr>
<tr>
<td>4</td>
<td>0.500</td>
<td>0.772</td>
<td>0.381</td>
</tr>
<tr>
<td>5</td>
<td>0.390</td>
<td>0.497</td>
<td>0.290</td>
</tr>
<tr>
<td>6</td>
<td>0.290</td>
<td>0.333</td>
<td>0.291</td>
</tr>
<tr>
<td>7</td>
<td>0.240</td>
<td>0.215</td>
<td>0.201</td>
</tr>
<tr>
<td>8</td>
<td>0.190</td>
<td>0.138</td>
<td>0.138</td>
</tr>
</tbody>
</table>

The results of experiments are taken as training patterns to train the BP neural network. The influencing factors (tolerances) are fed into the input nodes of the neural network, and the response of interest (total cost) is the target associated with the output node. As the training procedure begins, the input-target patterns from empirical results are presented to the networks. If the trained result is satisfactory, the cost–tolerance functions can be generated. The functions represented by the trained neural network, which represent the relationships between the controllable factors (tolerances) and response (total cost), are nonlinear functions for the empirical data set.

A simple data set [19] is taken as an example to illustrate the application of the BP network for generating the cost–tolerance function. The data set is shown in Table 1. The cost–tolerance function is assumed a priori as the inverse power form [19] with the parameters determined by the regression analysis. The cost–tolerance function takes the form of Eq. (7). The estimated costs $C_{RG}(t)$ with respect to various tolerances are listed in Table 1:

$$C_{RG}(t) = 0.0083t^{-0.6357} \quad (R^2 = 0.9394). \quad (7)$$

The tolerance–cost pairs are used as training patterns for the BP network. The tolerance is taken as the input value; the cost is the target value. The architecture of this BP network is 1–3–1, i.e., one input node (tolerance), three hidden nodes in the hidden layer, and one output node (cost). The BP-specific parameters are set as follows: learning rate = 0.6, momentum = 0.9, and training epochs = 2000. The weights (and biases) are randomly initialized between −0.5 and 0.5. The cost–tolerance relationship is generated by using the eight empirical data as training patterns, as shown in Table 1. Using the BP learning, the estimated costs with respect to various tolerances $C_{NN}(t) (R^2 = 0.9892)$ are listed in Table 1. In view of the above results, the BP network obviously has better cost–tolerance fitting results than that of regression analysis, even when there are only eight training patterns. Fig. 2 graphically illustrates the results for comparison purpose.

Using the BP network, we can construct the nonlinear functions, which depict the relationships between the input variables (tolerances) and the response (total cost). The mathematical models can then be constructed according to the primary objective of the tolerance synthesis.

3. The optimization approach

Metropolis et al. [20] proposed an algorithm that is used to simulate the cooling of a solid to reach a new energy state. In 1983, Kirkpatrick et al. [21] applied the Metropolis criterion to minimize the objective function in a combinatorial optimization problem and developed an optimization algorithm called simulated annealing (SA). Recently, SA has emerged as a foremost approach for large and complex optimization algorithms. The successful applications of SA to operations research problems have been reported continuously [22].

Although a descent algorithm is simple and efficient to execute, the main disadvantage of the
method is that the local optimum found may be far from the global optimum [22]. SA is a local search algorithm. The SA algorithm attempts to avoid entrapment in a local optimum by accepting a neighborhood move that increases the value of the objective function with respect to a controlled probability. The probability of accepting a move which causes an increase in objective function is normally set to \( \exp(-\Delta E/\Gamma) \) where \( \Gamma \) is a temperature similar to the temperature in physical annealing. The following is a general procedure of SA in pseudo-code [23]:

Select an initial state \( X^0 \in \mathcal{S} \);
Select an initial temperature \( \Gamma^0 > 0 \); Repeat
Set repetition counter = count
Repeat
  Generate state \( X^* \), a neighbor of \( X \);
  Calculate \( \Delta E = F(X^*) - F(X) \);
  If \( \Delta E < 0 \) then \( X = X^* \)
  else if random \( (0,1) < \exp(-\Delta E/\Gamma) \) then \( X = X^* \);
  count = count + 1;
  until count = \( M \);

The annealing schedule includes (1) the initial temperature \( \Gamma^0 \), (2) a cooling function, (3) the number of iterations to be performed at each temperature, and (4) a terminating criterion.

SA was reported to be a powerful approach for difficult combinatorial optimization problems [22]. Corana et al. [24] have reported that SA can generate nearly optimal solutions for nonlinear programs. However, the approach proposed by Corana et al. [24] requires huge computational time. Chen and Tsai [25] developed an effective and efficient SA-based optimization algorithm to solve complex constrained nonlinear programs. This paper modifies the SA-based optimization algorithm developed by Chen and Tsai [25] to solve the tolerance synthesis problem addressed herein. Interested readers are referred to Chen and Tsai [25] for a detailed discussion of this methodology.

The SA-based optimization algorithm for tolerance synthesis is presented as follows. Table 2 lists the notations used in the proposed optimization algorithm.

The SA-based optimization algorithm for tolerance synthesis

**Step 1:** Initialize the search procedure.

(a) Obtain an initial solution \( X^0 \), an initial control temperature \( \Gamma^0 \) and initial step sizes \( U^0 \).

(b) Set \( X = X^0 \), \( U = U^0 \), \( \Gamma = \Gamma^0 \), \( n_K = 0 \), \( n_M = 0 \), \( n_I = 0 \), \( n_P = 0 \). Evaluate \( F(X) \).

**Step 2:** Exploratory move. Set \( \Delta X = 0, X^* = X \).

For \( j = 1 \) to \( m \)

(a) Set \( x_j = x_j + u_j, \Delta E = F(X) - F(X) \).
  If \( \Delta E < 0 \) (downhill move),
  set \( X = X^* \), \( \Delta x_j = u_j \), improvement = 1.

(b) If \( \Delta E \geq 0 \) (uphill move), set \( X = X^* \),
  \( \Delta x_j = -u_j \), improvement = 1 with probability \( e^{-\Delta E/\Gamma} \).

(c) Perform sub-procedure CHECK.
  If frozen = 1, go to Step 5.

(d) If \( \Delta x_j = 0 \), set \( x_j = x_j - u_j, \Delta E = F(X) - F(X) \).
  Otherwise, return to (a).

(e) If \( \Delta E < 0 \) (downhill move),
  set \( X = X^* \), \( \Delta x_j = -u_j \), improvement = 1.
Table 3
Summary of the controllable factors

<table>
<thead>
<tr>
<th>Component</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Illustration</td>
<td>x-base</td>
<td>Motor base</td>
<td>Motor shaft</td>
<td>Motor shaft</td>
</tr>
<tr>
<td>Tolerance</td>
<td>Surface on x-base</td>
<td>Surface on the bottom of motor base</td>
<td>Size of shaft (target value 20 mm)</td>
<td>Perpendicularity of shaft</td>
</tr>
<tr>
<td>feature</td>
<td>Flatness</td>
<td>Flatness</td>
<td>Size</td>
<td>Perpendicularity</td>
</tr>
<tr>
<td>Tolerance</td>
<td>0.100</td>
<td>0.050</td>
<td>0.050</td>
<td>0.040</td>
</tr>
<tr>
<td>levels</td>
<td>0.150</td>
<td>0.075</td>
<td>0.075</td>
<td>0.060</td>
</tr>
<tr>
<td></td>
<td>0.200</td>
<td>0.100</td>
<td>0.100</td>
<td>0.080</td>
</tr>
</tbody>
</table>

(f) If $\Delta E \geq 0$ (uphill move), set $X = X'$, $\Delta x_j = -u_j$, improvement = 1 with probability $e^{-\Delta E/\Gamma}$.

(g) Perform sub-procedure CHECK.
   If frozen = 1, go to Step 5.

Step 3: Check pattern direction and adjust step sizes.
   (a) If $\Delta X \neq 0$, go to Step 4.
   (b) If $n_I \leq I$, set $n_I = n_I + 1$ (increase step sizes).
      For $i = 1$ to $m$
      Set $u_i = (r_I)^{n_I} \times u_i^0$.
      Go to Step 2.
   (c) Otherwise, set $n_I = n_I + 1$ (decrease step sizes).
      For $i = 1$ to $m$
      Set $u_i = (r_I)^{n_I} \times u_i^0$.
      Go to Step 2.

Step 4: Pattern move.
   (a) Set $X' = X + \Delta X$, $\Delta E = F(X') - F(X)$.
   (b) If $\Delta E < 0$ (downhill move), set $X = X'$, improvement = 1 and go to (d).
   (c) If $\Delta E \geq 0$ (uphill move), set $X = X'$, improvement = 1 with probability $e^{-\Delta E/\Gamma}$.
   (d) Perform sub-procedure CHECK.
      If frozen = 1, go to Step 5.
   (e) If $X = X'$, return to (a) (continue pattern move).
   (f) Otherwise, return to Step 2 with $X$.

Step 5: Termination.
Return $X^*$ and terminate search.

SUB-PROCEDURE CHECK (Check improvement in current best solution during $M$ moves and lower control temperature)

Step 1: Set $n_M = n_M + 1$.
   If $n_M = M$, go to Step 2.
   Otherwise, go to Step 4.

Step 2: (Check improvement during $M$ moves) Set $n_M = 0$.
   (a) If improvement = 1 (current best solution improved)
      Set $n_K = 0$.
   (b) Otherwise, set $n_K = n_K + 1$.
      If $n_K = K$ (frozen state achieved)
      Set frozen = 1.
      Otherwise, set frozen = 0.

Step 3: (Lower control temperature)
Set $\Gamma = c \times \Gamma^0$.

Step 4: Return.

4. The tolerance design example

A motor assembly [10] consisting of an x-base, a motor, a shaft, a motor base and a crank is investigated using the proposed tolerance synthesis approach discussed previously. Fig. 3 represents
Table 4  
Tolerance costs for each factor at various levels

<table>
<thead>
<tr>
<th>x</th>
<th>Lower level</th>
<th>Middle level</th>
<th>Upper level</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$18.065</td>
<td>$13.626</td>
<td>$12.815</td>
</tr>
<tr>
<td>2</td>
<td>$35.178</td>
<td>$24.681</td>
<td>$21.903</td>
</tr>
<tr>
<td>3</td>
<td>$279.612</td>
<td>$170.394</td>
<td>$108.574</td>
</tr>
<tr>
<td>4</td>
<td>$29.874</td>
<td>$19.622</td>
<td>$17.983</td>
</tr>
</tbody>
</table>

the motor assembly graphically. The four features of x-base flatness, motor base flatness, motor shaft size, and motor shaft perpendicularity affect the clearance measurement. These four features are treated as controllable factors. The number of levels for each factor is three. The dimensioning and tolerancing schemes and tolerance levels are summarized in Table 3. Table 4 shows the costs for each component tolerance at various levels. The output response in this example is the total cost, consisting of manufacturing cost and quality loss as expressed in Eq. (1).

Due to the complexity of tolerance allocation to the motor assembly, a CAD system, VSA-3D/Pro, has been applied by Jeang [10] for tolerance analysis. Table 5 illustrates the controllable factors, levels and response for this experiment with 27 runs. Each run is repeated 400 times to obtain adequate samples for accurate results [10]. Jeang applied regression analysis to estimate the relationship function between the four tolerance features and total cost. The estimated costs from the regression analysis are shown in Table 5.

For neural fitting, 3 runs and 24 runs out of the 27 runs are randomly chosen for testing and training patterns, respectively. The weights (and biases) are commonly initialized to random values between −0.5 and 0.5. The learning rate, momentum and learning epochs are set to 0.6, 0.9 and 3000, respectively. To ensure efficient convergence of the network training and the desired performance of the trained network, several network architectures are investigated. The RMSEs of training and testing under various network architectures are all smaller than 0.001. Also, the $R^2$ values of all architectures for fitting 27 runs are all larger than 0.99 and are listed in Table 6. Observing the neural fitting results ($R^2$ values), the BP network is very suitable for building cost–tolerance functions.

The CPU times (based on a PC 586) for various network architectures are between [8.0, 14.0] seconds. In Table 6, the 4−6−1 architecture has the best performance; thus, it is adopted to generate the neural network-based cost–tolerance function under this case study. The estimated total cost of each treatment from the trained neural network is listed in Table 7.

Jeang [10] applied a regression method to estimate the total cost of the studied problem. In order to illustrate the accuracy of different prediction methods, both the responses from regression and from the BP network are compared with simulation values. Fig. 4 graphically demonstrates that
Table 6
$R^2$ value for each network architecture

<table>
<thead>
<tr>
<th>Network architecture</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-4-1</td>
<td>0.9926</td>
</tr>
<tr>
<td>4-5-1</td>
<td>0.9993</td>
</tr>
<tr>
<td><strong>4-6-1</strong></td>
<td><strong>0.9997</strong></td>
</tr>
<tr>
<td>4-7-1</td>
<td>0.9991</td>
</tr>
<tr>
<td>4-8-1</td>
<td>0.9985</td>
</tr>
<tr>
<td>4-9-1</td>
<td>0.9983</td>
</tr>
</tbody>
</table>

Table 7
The estimated total cost from the trained network (4-6-1)*

<table>
<thead>
<tr>
<th>Run</th>
<th>$TC_{NN}(X)$</th>
<th>Run</th>
<th>$TC_{NN}(X)$</th>
<th>Run</th>
<th>$TC_{NN}(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>228.300</td>
<td>10</td>
<td>362.360</td>
<td>19</td>
<td>229.676</td>
</tr>
<tr>
<td>2</td>
<td>238.389</td>
<td>11</td>
<td>240.052</td>
<td>20</td>
<td>362.243</td>
</tr>
<tr>
<td>3</td>
<td>360.572</td>
<td>12</td>
<td>372.033</td>
<td>21</td>
<td>273.806</td>
</tr>
<tr>
<td>4</td>
<td>372.633</td>
<td>13</td>
<td>274.470</td>
<td>22</td>
<td>289.768</td>
</tr>
<tr>
<td>5</td>
<td>265.495</td>
<td>14</td>
<td>286.141</td>
<td>23</td>
<td>267.172</td>
</tr>
<tr>
<td>6</td>
<td>277.141</td>
<td>15</td>
<td>278.091</td>
<td>24</td>
<td>277.589</td>
</tr>
<tr>
<td>7</td>
<td>277.812</td>
<td>16</td>
<td>278.091</td>
<td>25</td>
<td>269.962</td>
</tr>
<tr>
<td>8</td>
<td>289.664</td>
<td>17</td>
<td>236.881</td>
<td>26</td>
<td>269.962</td>
</tr>
<tr>
<td>9</td>
<td>227.803</td>
<td>18</td>
<td>369.887</td>
<td>27</td>
<td>269.962</td>
</tr>
</tbody>
</table>

* $TC_{NN}$ is the estimated total cost from the neural network.

The BP network (4-6-1 architecture) outperforms the regression analysis [10] for fitting the cost-tolerance relationship.

At this point, the relationship between input factors $X = (x_1, x_2, x_3, x_4) = (x$-base flatness, motor base flatness, motor shaft size, motor shaft perpendicularity), and output response $F(X)$ (total cost defined by Eq. (1)) can be revealed from the constructed neural network. The solution of the motor assembly case can be found by solving the following mathematical models:

Maximize $F(X) = F(x_1, x_2, x_3, x_4)$
subject to $0.1 \leq x_1 \leq 0.2,$
        $0.05 \leq x_2 \leq 0.1,$
        $0.05 \leq x_3 \leq 0.1,$
        $0.04 \leq x_4 \leq 0.08.$

Accordingly, the neural network-based prediction provides a better basis for solving the tolerance synthesis problem. Problem (8) is solved by the proposed SA-based optimization method discussed in Section 3. The study example is solved on an IBM PC 586 compatible computer using the C programming language. The initial solution is randomly selected within the variable bounds. The SA algorithm-specific parameters for the tolerance synthesis problem addressed herein are listed in Table 8. The initial step sizes can be reasonably determined by using the ranges of variable bounds. The resolution of each variable in the proposed SA-based algorithm is relatively precise because the number of allowed step size reductions is not limited. The least total cost is found to be $225.819 with a tolerance allocation of $x$-base flatness = 0.117513, motor base flatness = 0.098011, motor shaft size = 0.100000, and motor shaft perpendicularity = 0.073709.

Since the final solution may fluctuate due to the stochastic nature of simulated annealing, the optimization procedures are run 50 times. The average CPU time per optimization procedure is only 0.3 seconds. The resulting average cost and standard deviation of cost are $225.8193 and 0.00097, respectively. Since the variation is quite
insignificant, the proposed SA-based optimization method is consistent for the motor assembly example. Jeang [10] applied RSM to allocate tolerances for minimizing the total cost. The least cost is $228.206, and the optimal tolerance allocation is x-base flatness = 0.14181, motor base flatness = 0.080894, motor shaft size = 0.098334, and motor shaft perpendicularity = 0.064308 [10]. The above computational results indicate the SA-based optimization method can locate high-quality solutions.

It can be concluded that the proposed hybrid methodology with BP and SA can solve a complex tolerance synthesis problem effectively. The BP network fits cost–tolerance relationships more precisely than regression analysis, and the SA-based optimization method allocates tolerances effectively.

5. Conclusions

Tolerance synthesis is an essential step in design phases related to quality conformity and economic manufacturing. In this research, the proposed approach provides better formulation of cost–tolerance relationships from empirical data, and generates more robust outcomes of tolerance synthesis. Traditionally, designers allocate tolerance based on their experience, and on the information contained in design handbooks or standards. This approach does not guarantee functionality or assembly; nor does it minimize costs. In practice, the cost–tolerance relationship is quite difficult to obtain. This study proposes a tolerance synthesis based on the BP learning and a SA-based optimization algorithm. The mathematical modeling of cost–tolerance relationships using the BP network outperforms the regression analysis in terms of fitting quality. However, the solution surface from the trained BP network may become very noisy and complicated. An SA-based optimization algorithm is utilized to resolve the formulated complex mathematical models for tolerance synthesis problems. The SA-based algorithm can obtain optimal tolerance allocation effectively and efficiently. Owing to its simplicity, the proposed methodology can be integrated into CAD/CAM/CAE for optimal tolerance allocation.

References


