SOCIETY FOR SCIENCE AND EDUCATION UNITED KINGDOM
TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
ISSN 2054-7390

Application of Genetic Algorithms Coupled with Neural Networks to Optimization of Reinforced Concrete Footings

[1]Jiin-Po Yeh and [2]Shu-Yu Yeh
Department of Civil and Ecological Engineering, I-Shou University, Kaohsiung, Taiwan
[1]jpyeh@isu.edu.tw, [2]abcd781031@gmail.com

ABSTRACT

This paper first applies genetic algorithms to optimally design reinforced concrete isolated footings subjected to concentric loading. Based on the ACI Building Code, constraints are built by considering wide-beam and punching shears, bending moment, allowable soil pressure, the development length for deformed bars and clear distance between deformed bars. Design variables consist of the width, length and thickness of the footing and the number of bars in the long and short directions, all of which are discrete. The objective function is to minimize the cost of steel and concrete in the footing. There are totally 144 cases of reinforced concrete isolated footings considered. The optimal results are randomly divided into three groups for the use of neural networks: training data, validation data and testing data. Two kinds of artificial neural networks are employed in this paper: two-layer feedforward backpropagation networks and radial basis networks. Linear regression analysis between the network outputs and targets of the testing data is performed to judge the accuracy of the neural networks. Numerical results show that the feedforward backpropagation network is very effective and has high accuracy with the correlation coefficients and the slope of the regression line being close to one and the y-intercept close to zero. Besides, it is better than the radial basis networks and needs much fewer neurons in the hidden layer.

Keywords: Reinforced Concrete Isolated Footings; Genetic Algorithms; Feedforward Backpropagation Networks; Radial Basis Networks.

1 Introduction

Genetic algorithms are a heuristic search that is based on natural selection and natural genetics. It was inspired by the evolution theory of "survival of the fittest," which can solve both constrained and unconstrained optimization problems according to the "natural selection." The constraints used in genetic algorithms can be in the form of linear equality or inequality with bounds on the optimization variables. The concept of genetic algorithms was formally introduced in 1970s by Professor John Holland at the University of Michigan, who in 1975 published the ground-breaking book "Adaptation in Natural and Artificial System" [1] that led to many important discoveries. In 1989, Goldberg described in more detail the theory of genetic algorithms and its applications [2]. From then on, the continuing price/performance improvements of computational systems have made genetic algorithms more attractive and popular. Genetic algorithms have successfully been applied to many fields like engineering, economics, chemistry,

DOI: 10.14738/tmlai.44.2150
Publication Date: 20th August, 2016
URL: http://dx.doi.org/10.14738/tmlai.44.2150

manufacturing, mathematics, physics and so on. In the civil engineering, there are a lot of applications, such as optimal design of reinforced concrete beams [3], optimal design of planar and space structures [4], multiobjective optimization of trusses [5], reliability analysis of structures [6], global optimization of grillages [7], global optimization of trusses with a modified genetic algorithm [8], optimization of pile groups using hybrid genetic algorithms [9] and optimization of grid shell topology and nodal positions [10].

Artificial neural networks are a computational tool based on the properties of biological neural systems, which have been considered to be simplified models of neural processing in the brain. The artificial neural network was originated by McCulloch and Pitts in 1943 [11], who claimed that neurons with binary inputs and a step-threshold activation function were analogous to first order systems. In 1986, Rumelhart *et al*. [12] proposed the theory of parallel distributed processing and developed the most famous learning algorithm in ANN-backpropagation, which uses a gradient descent technique to propagate error through a network to adjust the weights in an attempt to reach the global error minimum, marking a milestone in the current artificial neural networks. Since then, a huge proliferation in the ANN methodologies has been taking place. In particular there are many applications to the civil engineering, such as structural optimization [13-15], damage identification of structural elements [16], frame optimization [17], traffic sign classification [18], optimal design of continuous reinforced concrete beams [19], etc.

Owning to the abilities of genetic algorithms to deal with highly nonlinear constraints and neural networks to build very complicated nonlinear relationships between inputs and outputs, this paper combines these two techniques to optimally design the reinforced concrete isolated footings. Based on the provisions of the ACI Building Code Requirements for Structural Concrete and Commentary [20], the constraints of genetic algorithms are constructed, considering the wide-beam and punching shears, bending moment, allowable soil pressure and the development length for deformed bars. The design variables are the effective depth, width and length of the footing, the areas of bending reinforcements in the long and short directions; the object is to find the minimum cost of concrete and steel.

2 Discrete Optimization

Most optimization approaches were focused on and developed for continuous variables. However, the variables are usually discreet for design problems. Genetic algorithms are simple and extremely capable in solving discrete optimization problems. Hence, this paper adopts genetic algorithms provided by the MATLAB Global Optimization Toolbox [21] to optimally design reinforced concrete beams with discrete variables. There are three major components in the operation of genetic algorithms: (1) creating a random initial population of designs (individuals); (2) combining the individuals in the population in order to produce better individuals; (3) obtaining a new generation of designs and going to the next step. Each individual is real-coded in this paper, which is composed of the design variables. To create the new population, the algorithms performs the following steps: (1) Score each individual of the current population by computing its fitness value; (2) Scale the raw fitness scores to convert them into a more usable range of values; (3) Select individuals, called parents, based on their fitness. The lower the value of the fitness function, the more opportunity it has to be selected; (4) Choose some elites from the current population that have lower fitness function values. These elite individuals are just passed to the next population; (5) Produce children from the parents. Children are produced either by making random changes to a single parent—mutation—or by combining the vector entries of a pair of parents—crossover; (6) Replace the current population with the crossover and mutation children and elites to form the next

generation. The algorithm stops when one of the stopping criteria is met, such as the number of generation, the weighted average change in the fitness function value over some generations less than a specified tolerance, no improvement in the best fitness value for an interval of time, etc.

The optimization problem of the reinforced concrete isolated footing is constituted as follows:

Minimize the fitness function $f(\boldsymbol{x})$

such that

$$C_i(\boldsymbol{x}) \leq 0, \quad i=1,\ldots, m$$

$$Ci(\boldsymbol{x})=0, \quad i=m+1,\ldots, mt \tag{1}$$

$$\boldsymbol{LB} \leq \boldsymbol{x} \leq \mathbf{UB}$$

where $C_i(\boldsymbol{x})$ represents the nonlinear inequality and equality constraints, m is the number of nonlinear inequality constraints, mt is the number of nonlinear constraints, $f(\boldsymbol{x})$ is the total cost of concrete and tension steels, and ***LB*** and ***UB*** are the vectors of lower and upper bounds of design variables, respectively. The Global Optimization Toolbox based on MATLAB uses the augmented Lagrangian genetic algorithm [22, 23] to solve nonlinear constraint problems with bounds. A subproblem is formulated by combining the fitness function and nonlinear constraint functions using the Lagrangian and the penalty parameters. A sequence of such optimization problems are approximately minimized using the genetic algorithm such that the bounds are satisfied. A subproblem formulation is defined as

$$\Phi(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{s}, \rho) = f(\boldsymbol{x}) - \sum_{i=1}^{m} \lambda_i s_i \log(s_i - C_i(\boldsymbol{x})) + \sum_{i=m+1}^{mt} \lambda_i C_i(\boldsymbol{x}) + \frac{\rho}{2} \sum_{i=m+1}^{mt} C_i(\boldsymbol{x})^2 \tag{2}$$

where the components λ_i of the vector $\boldsymbol{\lambda}$ are nonnegative and known as Lagrange multiplier estimates. The elements s_i of the vector $\boldsymbol{s}$ are nonnegative shifts, and ρ is the positive penalty parameter. The algorithm begins by using initial values for the parameters. Parameters s_i and λ_i are updated based on the value of $C_i(\mathbf{x})$. The genetic algorithm minimizes a sequence of the subproblem, which is an approximation of the original problem. When the subproblem is minimized to a required accuracy, the Lagrangian multiplier estimates are updated, or the penalty parameter is increased by a penalty factor. These steps are repeated until the stopping criteria of the genetic algorithm are met.

3 Artificial Intelligence

An artificial neural network is an analytical system that addresses problems without explicit solutions or whose solutions are very difficult to explicitly formulate. The neural network is composed of some computational units, called neurons, which are highly interconnected. Every interconnection has strength, called weight, which is represented by a number. The basic capability of neural networks is to learn patterns from a large number of examples by adjusting the weights of each neuron. The learning can be supervised or unsupervised. In this paper, the Neural Network Toolbox based on MATLAB is employed [24] and two kinds of artificial neural networks are used: the feedforward backpropagation and the radial basis. Both of them are supervised neural networks, which are briefly illustrated as follows.

3.1 The Feedforward Backpropagation Network

The most commonly used neural network is the feedforward neural network with the backpropagation learning algorithm. The network discussed in this paper has two layers: one hidden layer and one output layer, whose structure is shown in Figure 1. The transfer function of the single hidden layer is the tan-sigmoid function defined by

$$a_i = f(n_i) = \frac{e^{n_i} - e^{-n_i}}{e^{n_i} + e^{-n_i}} \quad , \quad i = 1,2,3,\ldots,k \tag{3}$$

where k is the number of the artificial neurons, $n_i = w_{i1}P_1 + w_{i2}P_2 + \ldots + w_{iR}P_R + b_i$, P_1, P_2,…P_R are the inputs, R is the number of input elements, w_{i1} , w_{i2} ,…, w_{iR} are the weights connecting the input vector and the ith neuron, and b_i is the bias of the ith neuron in the hidden layer. The output

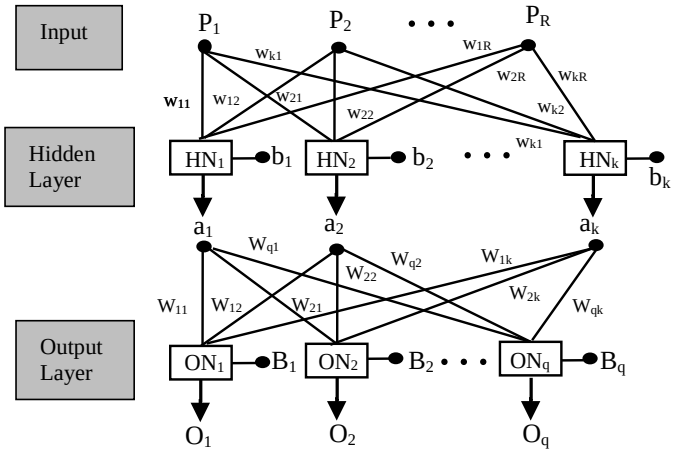

Figure 1 Two-layer feedforward backpropagation neural network with multiple outputs.

layer uses the linear transfer function defined by

$$Q_i = f(N_i) = N_i \quad , \quad i = 1,\ 2,\ldots,q \tag{4}$$

where $N_i = W_{i1}a_1 + W_{i2}a_2 + \ldots\ldots + W_{ik}a_k + B_i$, $W_{i1}, W_{i2}, \ldots, W_{is}$ are the weights connecting the neurons of the hidden layer and the ith neuron of the output layer, and B_i is the bias of the ith output neuron.

There are many variations of the backpropagation algorithm aiming to minimize the network performance function, i.e., the mean square error between the network outputs and the targets defined by

$$Minimize \quad MSE = \frac{1}{m}\sum_{j=1}^{m} (t_j - a_j)^2 \tag{5}$$

where t_j and a_j are the jth target and network output, respectively. Among many training functions, the Levenberg-Marquardt algorithm [25, 26] was chosen to minimize the network performance function. The formula to update the weights and biases is given by

$$\boldsymbol{X}_{k+1} = \boldsymbol{X}_k - [\boldsymbol{J}^T\boldsymbol{J} + \mu\ \boldsymbol{I}]^{-1}\boldsymbol{J}^T\boldsymbol{e} \tag{6}$$

where $\boldsymbol{X}_k$ is a vector of current weights and biases, $\boldsymbol{J}$ is the Jacobian matrix of the first derivative of the error vector $\boldsymbol{e}$ between the network outputs and target outputs with respect to the weights and biases, $\boldsymbol{I}$ is a unit matrix and μ is a parameter. If μ→0, Eq. (6) can be simplified as

$$\boldsymbol{X}_{k+1} = \boldsymbol{X}_k - [\boldsymbol{J}^T\boldsymbol{J}]^{-1}\boldsymbol{J}^T\boldsymbol{e} \approx \boldsymbol{X}_k - \boldsymbol{H}^{-1}\boldsymbol{J}^T\boldsymbol{e} \tag{7}$$

which is the quasi-Newton's method with the approximate Hessian matrix *H* [33]; if μ→∞, Eq. (6) turns out to be

$$\boldsymbol{X}_{k+1} = \boldsymbol{X}_k - \mu^{-1}\boldsymbol{J}^T\boldsymbol{e} \tag{8}$$

which is the gradient descent method with the learning rate μ^{-1} [24]. Therefore, this algorithm interpolates between the quasi-Newton's algorithm and the gradient descent method. If a tentative step increases the performance function, the parameter μ will be increased, causing this algorithm to act like the gradient descent method, while it shifts toward Newton's method if the reduction of the performance function is successful, i.e., the parameter μ will be decreased. In this way, the performance function will always be reduced at each iteration of the algorithm. To improve the network generalization, the error on the validation set is monitored simultaneously during the training process. When the network begins to overfit the training data, the error on the validation set typically begins to rise. Once the validation error increases for a specified number of iterations (The default value set by MATLAB is six), the training terminates and the weights and biases at the minimum of validation error are returned.

The number of neurons required in the hidden layer is usually unknown beforehand. Bayesian regularization [27] provides a measure of how many network parameters (weights and biases) are being effectively used by the network. According to this effective number of parameters, the number of neurons required in the hidden layer of the two-layer neural network can be estimated by the following equation

$$(Rk+k)+(kq+q)= Num \tag{9}$$

where *R* and *q* are the number of elements in the input and output vectors, respectively, *k* is the number of neurons to be determined in the hidden layer, and *Num* is the effective number of parameters found by the Bayesian regularization implemented by the function *trainbr* in MATLAB.

3.2 The Radial Basis Network

For comparison with the feedforward backpropagation network, this paper uses another network, the radial basis network that has two layers: the radial basis layer and output layer. The transfer function in the artificial radial basis neuron is the radial basis function defined by

$$a = radbas(n) = e^{-n^2} \tag{10}$$

as shown in Figure 2, where n= $\|\boldsymbol{w} - \boldsymbol{P}\|\ b$ is the vector distance (Euclidean distance) between the weight vector w and the input vector P multiplied by the bias b. As the distance between w and P decreases, the output increases. Thus the radial basis function acts as a detector that produces 1.0 whenever the input is identical to the weight vector. Each bias in the radial basis layer is set to be 0.8326/SPREAD, which causes radial basis function to output 0.5 when $\|\boldsymbol{w} - \boldsymbol{P}\|$ = +/- SPREAD. The larger the constant SPREAD is,

the smoother the radial basis function will be. The constant needs to be large enough so that several radial basis neurons respond to overlapping region of the input space and have fairly large output at any given instant, but not so large that all the neurons respond in essentially same manner [24]. Different *SPREADs* are usually tried to find the best value for a given problem. There are two functions in MATLAB to design the radial basis network: *newrb* and *newrbe.*

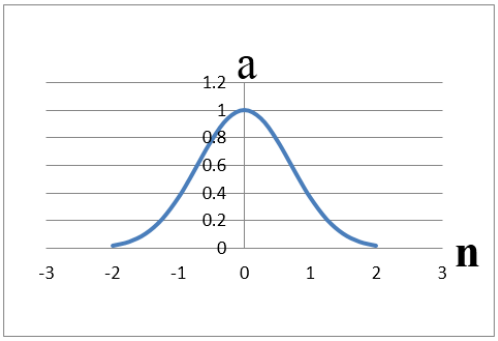

Figure 2 Radial basis function

3.2.1 The Design Function newrb

The function *newrb* iteratively creates one radial basis neuron at a time. At each iteration, the input vector that results in lowering the network error is used to create a radial basis neuron. Neurons are added to the network until the sum-squared error falls beneath an error goal or maximum number of neurons has been reached.

3.2.2 The Design Function newrbe

The function *newrbe* can produces a network with zero error on the training vectors. It creates as many radial basis neurons as there are input vectors, and each neuron acts as detector for a different input vector. The drawback to *newrbe* is that it produces a network with as many hidden neurons as there are input vectors. For this reason, it does not return an acceptable solution when many input vectors are needed to properly define a network, as is typically the case.

To make the above-mentioned neural networks more efficient, it is often useful to scale inputs and targets so that they will always fall within a specific range. For example, the following formula

$$y = 2(\frac{x - min}{max - min}) - 1 \tag{11}$$

is used in this paper to scale inputs and targets, where x is the original value, y is the scaled value, and *max* and *min* are the maximum and minimum of inputs or targets, respectively. Eq. (11) produces inputs and targets in the range [-1, 1]. To evaluate the performance of the trained network, this paper makes use of a regression analysis between the network outputs and the corresponding targets.

4 Design of Reinforced Concrete Isolated Footings

The reinforced concrete isolated footings considered in this paper are loaded concentrically, as shown in Figure 3, with width B, length L and thickness h. The dead and live loads transmitted by the column are denoted by P_D and P_L, respectively. The column size is $a \times b$. A variety of reinforced concrete isolated footings are optimally designed by the genetic algorithm. The objective function is to minimize the total

cost of the concrete and the tension reinforcements in the long and short directions of the rectangular isolated footing. All the constraints required to design the isolated footing comply with the ultimate-strength design of ACI 318-08 Code, considering wide-beam and punching shears, bending moment and the development length for deformed bars. The units of force and length in the following formulas are kgf (=9.81N) and cm, respectively.

4.1 Shear

The shear strength of the isolated footing in the vicinity of column reactions is governed by the more severe of the following two conditions:

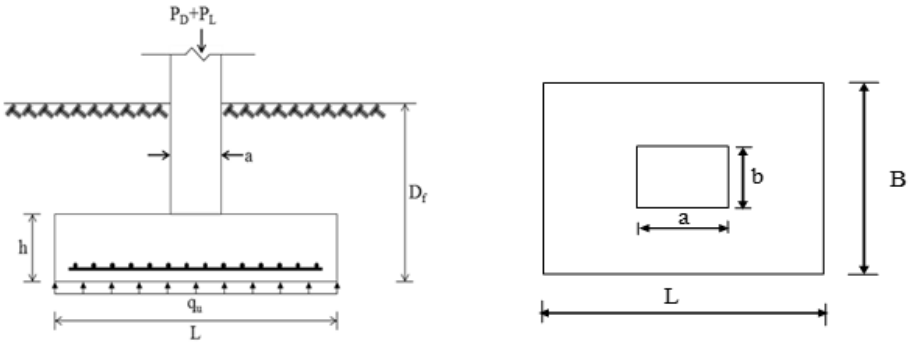

Figure 3 The isolated footing: (a) elevation and (b) plan.

4.1.1 Wide-beam Shear

The critical section is assumed to extend in a plane across the entire width and lies at a distance *d* from the face of the column, as shown in Figure 4(a). The nominal shear strength of this section is

$$V_{c1} = 0.53\sqrt{f_c'}Bd \tag{12}$$

or

$$V_{c2} = 0.53\sqrt{f_c'}Ld \tag{13}$$

where *d* is the effective depth of the footing. The constraints for wide-beam shear are

$$V_{u1} = q_u\left(\frac{L-a}{2} - d\right)B \le \phi V_{c1} \tag{14}$$

and

$$V_{u2} = q_u\left(\frac{B-b}{2} - d\right)L \le \phi V_{c2} \tag{15}$$

where $q_u = (1.2P_D+1.6P_L)/(BL)$ is the factored net soil pressure and ϕ =0.75 is the strength reduction factor for shear.

4.1.2 Punching Shear

The critical section occurs at a distance *d*/2 from the face of the column, as shown in Figure 4 (b). The maximum allowable nominal shear strength is the smallest of the following three equations

$$V_c = (0.53 + \frac{1.06}{\beta_c})\sqrt{f_c'}b_0 d \ ,$$
$$V_c = (0.53 + \frac{0.265\alpha_s d}{b_0})\sqrt{f_c'}b_0 d \quad (16)$$
$$V_c = 1.06\sqrt{f_c'}b_0 d$$

where β_c = long side *a*/short *b* of the column, b_0 = perimeter of the critical section *ijkl* and α_s = 40, 30 and 20 for interior, edge and corner columns, respectively. In this paper, interior columns are considered; therefore, α_s = 40. Similarly, the constraint for the punching shear is

$$V_u = q_u[BL - (a+d)(b+d)] \le \phi V_{c,\min} \quad (17)$$

where $V_{c,min}$ is the smallest of Eqs. (16). The area to be considered for factored shear V_u is equal to the total area of the footing less the area *ijkl*.

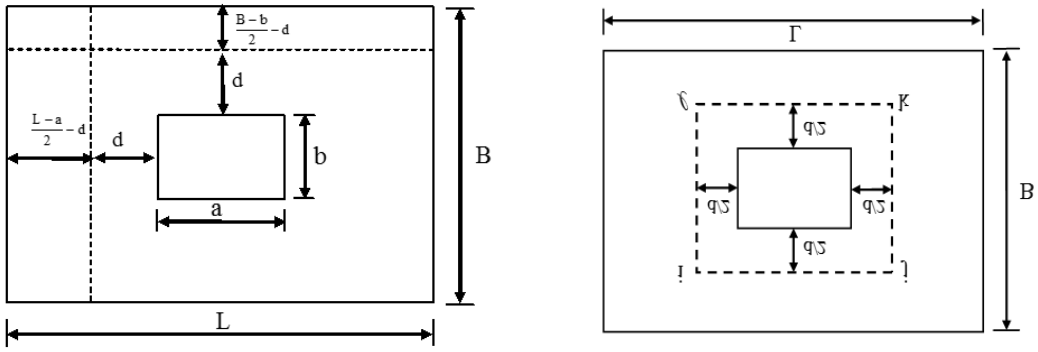

Figure 4 Critical sections: (a) wide-beam shear and (b) punching shear

4.2 Bending Moment

Let A_b be the cross-sectional area of the reinforcement steel and N_L and N_B be the number of steels in the long and short directions of the footing, respectively. The critical section for bending moment is at the face of the column and the constraints are

$$M_{uL} = q_u \frac{B}{2}(\frac{L-a}{2})^2 \le \phi_{mL} N_L A_b f_y (d - \frac{N_L A_b f_y}{2(0.85) f_c' B}) \quad (18)$$

and

$$M_{uB} = q_u \frac{L}{2}(\frac{B-b}{2})^2 \le \phi_{mB} N_B A_b f_y (d - \frac{N_B A_b f_y}{2(0.85) f_c' L}) \quad (19)$$

where ϕ_{mL} and ϕ_{mB} are the strength reduction factor for moment. Let ε_t be the tensile strain of the reinforcement, then

$$0.65 \le \phi_{mL} \text{ or } \phi_{mB} = 0.65 + 0.25\frac{\varepsilon_t - \varepsilon_y}{0.005 - \varepsilon_y} \le 0.9 \tag{20}$$

To prevent sudden failure with little or no warning when the beam cracks or fails in a brittle manner, the ACI code limits the minimum and maximum amount of steel to be

$$A_{sL,\min} \le N_L A_b \le A_{sL,\max} = \frac{0.85 f_c' \beta B d}{f_y}(\frac{3}{7}) \tag{21}$$

and

$$A_{sB,\min} \le N_B A_b \le A_{sB,\max} = \frac{0.85 f_c' \beta L d}{f_y}(\frac{3}{7}) \tag{22}$$

where β is the stress block depth factor,

$$A_{sL,\min} \ge \max\ (\frac{0.8\sqrt{f_c'}}{f_y} Bd,\ \frac{14Bd}{f_y}) \tag{23}$$

and

$$A_{sB,\min} \ge \max\ (\frac{0.8\sqrt{f_c'}}{f_y} Ld,\ \frac{14Ld}{f_y}) \tag{24}$$

The formula for $A_{sL,max}$ in Eq. (21) or $A_{sB,max}$ in Eq. (22) is derived based on the requirement that the tensile strain must be greater than or equal to 0.004. In addition, both the steel ratios $N_L A_b/(Bh)$ and $N_B A_b/(Lh)$ must exceed the minimum value required for temperature and shrinkage: 0.0018 for grade 60 deformed bars and 0.002 for grade 40 or 50 deformed bars.

4.3 Allowable Soil Pressure

Suppose that the allowable soil pressure under the base of the footing is q_a. The gross soil pressure must not exceed the allowable soil pressure, that is,

$$\frac{P_D + P_L}{BL} + hw_c + \gamma_s(D_f - h) \le q_a \tag{25}$$

where D_f is the distance from the base to the ground surface, as shown in Figure 3, w_c is the weight of concrete and γ_s is the unit weight of soil over the footing.

4.4 Development of Reinforcement

According to the ACI Code, the equation for the development length of bars in tension is

$$\ell_d = \frac{0.15 d_b f_y \psi_t \psi_e \lambda}{\sqrt{f_c'}} \tag{26}$$

for No. 6 and smaller bars with clear spacing not less than $2d_b$ and clear cover not less than d_b, where d_b is the bar diameter, and ψ_t and ψ_e are the bar location and coating factors, respectively. In this paper ψ_t and

ψ_e are assumed to be 1.0. For normal weight concrete, λ=1. The critical section for development-length of the bars in tension is the same as the critical section in flexure, that is, at the face of the column. Hence,

$$0.5(L\text{-}a) \text{ - concrete cover} \geq \ell_d \tag{27}$$

and

$$0.5(B\text{-}b) \text{ - concrete cover} \geq \ell_d \tag{28}$$

The equation for the development length of bars in compression is

$$\ell_{dc} = \max\ (\frac{0.075 d_b f_y}{\sqrt{f_c'}},\ 0.0043 d_b f_y) \tag{29}$$

The dowel bars stressed to f_y are required to transfer the axial compression force in the column into the footing, as shown in Figure 5; hence, there should be minimum extension of the dowels

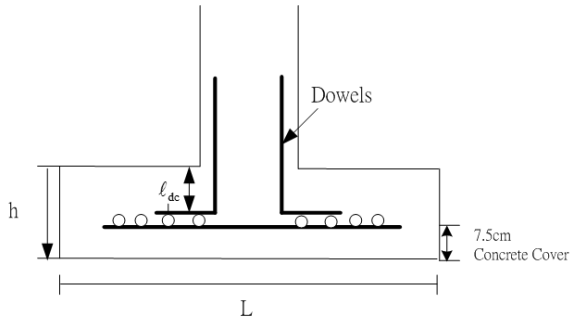

Figure 5 Dowels that transfer the column load to the footing slab.into the footing. Therefore, the thickness *h* of the footing must satisfy the following constraints:

$$h - \text{concrete cover} - 2d_b\,(\text{footing bars}) - d_b\,(\text{dowels}) \geq \ell_{dc} \tag{30}$$

4.5 Reinforcement Distribution and Clear Distance

In the short direction of the footing, a central band of width *B* shall contain the major portion of flexural reinforcement according to the formula $2N_B/(L/B+1)$ but rounded up to the nearest integer and uniformly distributed along the band width. The remainder is also uniformly distributed outside the central band. The reinforcement in the long direction is uniformly distributed across the entire width of the footing. The clear distance *s* between individual steel bars both in the long and short directions must satisfy

$$Min\ (3h,\ 45\ cm) \geq s \geq Max\ (2d_b,\ 2.5\ cm) \tag{31}$$

5 Numerical Results

The given design conditions for the isolated footing are as follows: the concentric dead load P_D, the live load P_L applied to the column, the allowable soil pressure q_a at the base of the footing, the distance from the footing bottom to the ground surface D_f, the compressive strength of concrete f_c', and the yield stress of steel f_y. In addition, the column size is assumed to be 0.45 m×0.45 m, the unit weight of soil over the footing is 2000 kgf/m^3, and the unit weight of concrete is 2400 kgf/m^3. In Taiwan, the unit prices of concrete and steel are 1900 NT$/m^3 and 18.4 NT$/kgf, respectively. The concrete cover for the

reinforcement of the footing is assumed to be 7.5 cm. The optimal results obtained from genetic algorithms consist of the minimum cost of the footing, the thickness h, width B and length L of the footing, and the number of steel bars in the long and short directions, N_L and N_B, respectively. Based on the often-used materials and customs in Taiwan, this paper selects two kinds of yield strength f_y of the tension reinforcement: 2800 kgf/cm^2 (40 ksi) and 4200 kgf/cm^2 (60 ksi) as well as two kinds of compressive strength f'_c of the concrete: 210 kgf/cm^2 (3000 psi) and 280 kgf/cm^2 (4000 psi). There are three kinds of q_a: 25 ton/m^2, 30 ton/m^2 and 35 ton/m^2; three kinds of D_f: 1.0 m, 1.5 m and 2.0 m; four kinds of dead load P_D: 60 ton, 80 ton, 100 ton and 120 ton as well as four kinds of P_L: 40 ton, 60 ton, 80 ton and 100 ton. There are totally144 kinds of footings being designed depending on the combination of the six given conditions. For the purpose of training, monitoring and testing the neural networks, the optimal data are divided into 3 groups: 100 training sets (70%), 22 validation sets (15%) and 22 testing sets (15%). The fitness function is the total cost in New Taiwan Dollars of the footing reinforcement and concrete. All the constraints are built according to the formulas discussed in Sec. 4.

5.1 Genetic Algorithms

This paper adopts the MATLAB global optimization toolbox to carry out genetic algorithms. To run genetic algorithms of the MATLAB software, some parameters need to be selected beforehand. After a number of trials, here are the values used in this paper: The population size is set to be 100, crossover rate 0.8, and elite number 6. Furthermore, all the individuals are encoded as integers; "Rank" is used as the scaling function that scales the fitness values based on the rank of each individual; "Roulette" is the selection function to choose parents for the next generation; "Two-point crossover" is used as the crossover method to form a new child for the next generation; The "Adaptive Feasible Function" is chosen as the mutation function to make small random changes in the individuals and ensure that linear constraints and bounds are satisfied. The steel bars used in the footing can usually range from No. 5 to No. 10. In order to decide the appropriate size of steel bars to be used in this paper, different sizes of reinforcement bars are tried for a variety of random combinations of f_y, f'_c, P_D, P_L, q_a and D_f. The results show that No. 5 bars always lead to the minimum cost. Therefore, No. 5 steel bars are used in each direction of the footing and as dowels that transfer the column load to the footing slab. Taken as an example, Table 1 shows the results for the case of f_y =2800 kgf/cm^2, f'_c =210 kgf/cm^2, P_D = 120 ton, P_L= 100 ton, q_a =35 ton/m^2 and D_f =1 m.

Table 1 The optimal results for different sizes of reinforcement bars for the case of f_y =2800 kgf/cm^2, f'_c =210 kgf/cm^2, P_D = 120 ton, P_L= 100 ton, q_a =35 ton/m^2 and D_f =1 m.

Size of bars	h(m)	B (m)	L(m)	N_B	N_L	cost(NT$10^3)
5	0.65	2.27	2.96	42	32	14.343
6	0.66	2.31	2.91	29	23	14.573
7	0.66	2.31	2.91	22	17	14.624
8	0.66	2.27	2.96	17	13	14.655
9	0.67	2.27	2.96	13	10	14.683
10	0.67	2.47	2.82	10	9	15.383

5.2 Feedforward Backpropagation Networks

The inputs of the artificial neural network consist of six elements: f_y, f'_c, P_D, P_L, q_a and D_f, and the targets also have six components: the minimum cost, h, B, L, N_B, N_L and the cost. To make the network more efficient, all the data are normalized by using Eq. (11) or the function *mapminmax* introduced in the MATLAB software for neural network. The function *trainbr* is first applied to find the number of effective parameters required for the network, which is found to be 118.9, as shown in Figure 6; therefore the number of neurons required in the hidden layer is 8.685 according to Eq. (9), which is then rounded up to 9. After the number of neurons needed in the hidden layer is determined, a very efficient training function *trainlm*, i.e., the Levenberg-Marquardt algorithm, is used to train the network. While the network is being trained, the training and validation data are both put into the network. During the training process, the network error for training data will decrease, while the network error for the validation data decreases first but increases later on. To avoid overfitting the data, the training will terminate when the network error of the validation data increases continuously for a number of epochs whose default value is six set by the MATLAB software. The training process is shown in Figure 7, where there are totally 23 epochs but the weights and biases at the epoch 17 are taken as those of the trained network. The value of the performance function at the epoch 17 is 0.0019623. After the normalized data are reversed to the original scale, the six network outputs and targets for the testing data are presented in Figs. 8-13. The 22 sets of testing data containing inputs and targets as well as outputs are shown in Tables 2(a) and 2(b), respectively. If more neurons in the hidden layer are used, for example, 12 neurons, the accuracy of the network can only improve a little bit, not significantly. Tables 3 shows the linear regression analysis results for the testing data when the hidden layer has 9 and 12 neurons, where parameters m, b and r stand for the slope of the straight line, the intercept with the vertical axis and correlation coefficient, respectively. Table 3 indicates that the networks with 9 neurons and 12 neurons in the hidden layer have almost the same accuracy. Based on the Figs. 8-13 and Table 3, the performance of the network is considered quite good on the whole, because the parameters m and r are close to one and parameter b also close to zero.

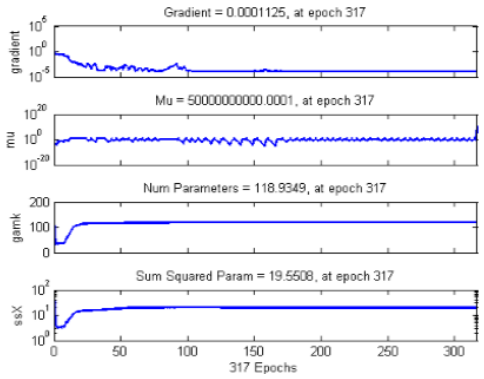

Figure 6 The number of effective parameters in the network obtained from the training function trainbr

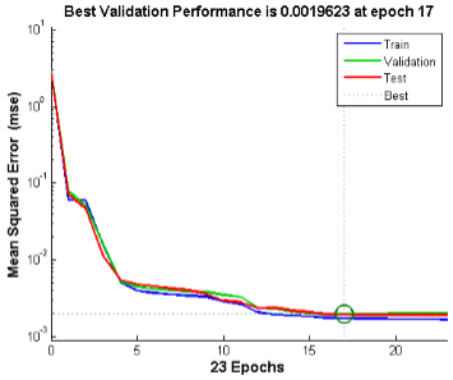

Figure 7 The training process for the feedforward backpropagation neural network with 9 neurons in the hidden layer

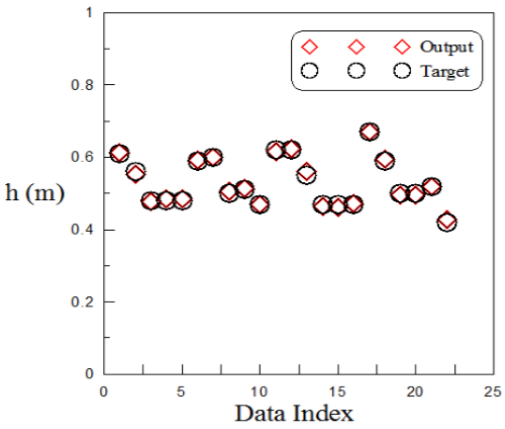

Figure 8 Outputs and targets of the footing thickness h for the testing data with 9 neurons in the hidden layer of the feedforward backpropagation network

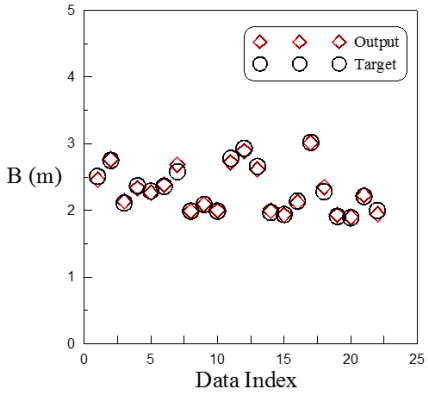

Figure 9 Outputs and targets of the footing width *B* for the testing data with 9 neurons in the hidden layer of the feedforward backpropagation network

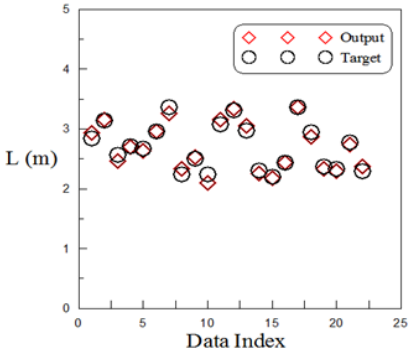

Figure 10 Outputs and targets of the footing Length *L* for the testing data with 9 neurons in the hidden layer of the feedforward backpropagation network

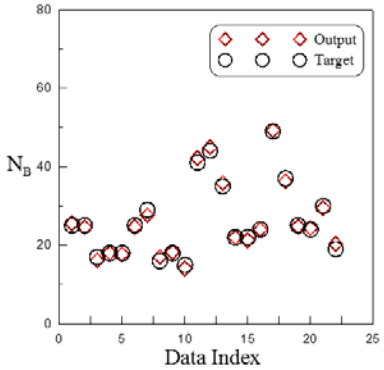

Figure 11 Outputs and targets of the number of steel bars in the short direction for the testing data with 9 neurons in the hidden layer of the feedforward backpropagation network

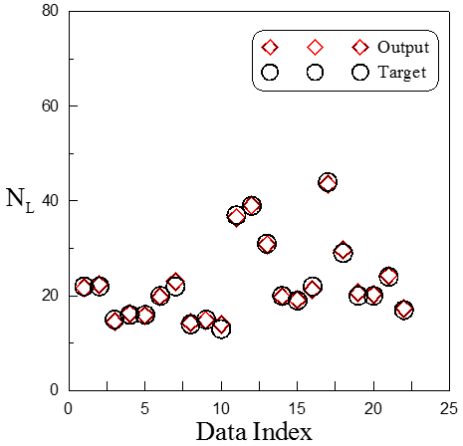

Figure 12 Outputs and targets of the number of steel bars in the long direction for the testing data with 9 neurons in the hidden layer of the feedforward backpropagation network

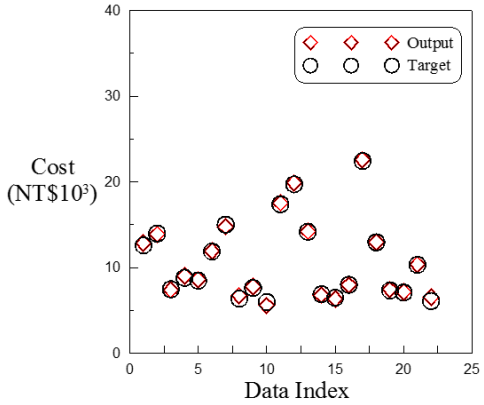

Figure 13 Outputs and targets of the total cost for the testing data with 9 neurons in the hidden layer of the feedforward backpropagation network.

Table 2(a) Inputs and targets of the testing data with 9 neurons in the hidden layer of the feedforward backpropagation network.

Data index	Inputs						Targets					
	f_y (kg/cm^2)	f_c' (kg/cm^2)	P_D (ton)	P_L (ton)	q_a (ton/m^2)	D_f (m)	h (cm)	B (cm)	L (cm)	N_B	N_L	Cost (NT$10^3)
1	4200	280	120	100	35	2	61	251	285	25	22	12.585
2	4200	280	100	80	25	2	56	275	315	25	22	13.972
3	4200	280	80	60	30	2	48	211	257	17	15	7.478
4	4200	280	80	60	25	1.5	48	237	271	18	16	8.812
5	4200	280	80	60	25	1	48	229	268	18	16	8.469
6	4200	210	100	80	30	2	59	236	296	25	20	11.876
7	4200	210	100	80	25	2	60	258	336	29	22	15.000
8	4200	210	80	60	35	1.5	50	199	224	16	14	6.377
9	4200	210	80	60	30	1.5	51	209	250	18	15	7.624
10	4200	210	60	40	35	1.5	47	199	224	15	13	5.982
11	2800	280	120	100	30	2	62	278	308	41	37	17.397
12	2800	280	120	100	25	1	62	293	331	44	39	19.723
13	2800	280	100	80	25	1	55	266	297	35	31	14.189
14	2800	280	80	60	35	2	47	197	231	22	20	6.897
15	2800	280	80	60	35	1	47	194	220	22	19	6.476
16	2800	280	80	60	30	1.5	47	214	244	24	22	7.990
17	2800	210	120	100	25	1.5	67	302	336	49	44	22.422
18	2800	210	100	80	30	1.5	59	228	295	37	29	12.957
19	2800	210	80	60	35	2	50	191	238	25	20	7.333
20	2800	210	80	60	35	1.5	50	189	233	24	20	7.090
21	2800	210	80	60	25	1	52	221	278	30	24	10.319
22	2800	210	60	40	25	1.5	42	200	229	19	17	6.106

Table 2(b) Network outputs of the testing data with 9 neurons in the hidden layer of the feedforward backpropagation network

Data index	Network outputs					
	h (cm)	B (cm)	L (cm)	N_B	N_L	Cost (NT$10^3)
1	61.3	246.1	293.2	25.5	21.6	12.860
2	55.4	276.5	315.4	24.5	22.4	13.893
3	47.8	213.5	247.1	16.2	14.6	7.411
4	48.6	233.4	269.9	18.0	16.3	9.038
5	48.5	226.7	263.2	17.7	15.9	8.591
6	59.1	238.6	295.9	24.6	19.8	11.933
7	60.1	268.6	325.8	27.6	23.0	14.771
8	50.5	198.8	233.8	17.0	14.4	6.745
9	51.4	209.0	253.3	18.2	14.9	7.804
10	46.9	199.8	209.5	14.1	14.0	5.582
11	61.6	272.2	315.5	42.2	36.5	17.563
12	62.3	288.8	334.6	45.0	39.1	19.888
13	56.0	261.3	305.4	35.8	30.8	14.250
14	46.5	199.4	225.4	21.9	19.8	6.793
15	46.3	193.2	218.4	21.3	19.2	6.338
16	47.3	212.3	243.9	24.0	21.3	7.960
17	67.0	301.3	336.5	49.3	43.5	22.571
18	59.4	234.8	287.6	36.3	29.7	12.965
19	49.6	193.5	233.5	24.7	20.7	7.431
20	49.5	190.2	230.2	24.3	20.3	7.184
21	51.9	223.0	273.9	29.5	24.3	10.355
22	42.6	194.2	237.2	20.3	17.3	6.541

Table 3 The linear regression analysis of targets and outputs for the testing data with 9 and 12 neurons in the hidden layer of the feedforward backpropagation network.

No. of neurons in the hidden layer	Outputs (targets)	m (slope)	b (y-intercept)	r (correlation coefficient)
9	h	1.0008	0.0044	0.9975
	b	0.9767	-0.0029	0.9933
	L	1.0168	-0.0069	0.9861
	N_B	1.0191	0.0048	0.9973
	N_L	0.9825	-0.00001	0.9985
	Cost	1.0031	0.0073	0.9992
12	h	0.9755	-0.001	0.9983
	b	0.9687	-0.0001	0.9908
	L	1.0113	0.0089	0.9906
	N_B	0.9787	-0.0036	0.9966
	N_L	0.9965	-0.0001	0.9986
	Cost	1.0019	0.0025	0.9996

5.3 Radial Basis Networks

There are two kinds of design functions for the radial basis network: *newrb* and *newrbe*. The *newrbe* produces as many number of radial basis neurons as the input vectors; therefore, it has zero error between the network output and target, while the *newrb* creates one neuron at a time until the preset mean square error between the network outputs and the targets or the number of epochs are reached. Because the radial basis network does not require the validation data, only training and testing data are considered. For comparison, these two sets of data are intended to be the same as those of feedward backpropagaton with 9 neurons in the hidden layer.

5.3.1 The Function *newrb*

The mean square error between the network outputs and targets is set to be 0.0019623, which is the same as the result of the feedforward backpropagation. Let the parameter *SPREAD* change from 1.0 to 2.0 and train the network for each case. By observing the regression analysis of the network outputs and targets, all of them show very good performance, while the network with *SPREAD*=1.6 is a little bit better than other values; therefore, *SPREAD*=1.6 is selected for the radial basis layer. From the training process, it is found that 44 epochs are required for the mean square error to fall beneath the goal of 0.0019623; therefore, 44 radial basis neurons are created for the network. The testing data are then substituted into the trained network. The linear regression analysis of the six network outputs and targets is shown in Table 4, where the correlation coefficients are between 0.9801 and 0.9939. Tables 3 and 4 indicate that the performance of the radial basis network is a little inferior to the feedforward backpropagation network.

Table 4 The linear regression analysis of the outputs and targets for the testing data using *newrb* with SPREAD=1.6.

Outputs (Targets) / Parameters	h	B	L	N_B	N_L	Cost
m	0.9971	0.9488	1.0194	0.9775	0.9676	0.9747
b	-0.029	-0.0303	-0.0225	-0.0297	-0.0301	-0.0269
r	0.9913	0.9826	0.9801	0.9899	0.9930	0.9939

5.3.2 The Function *newrbe*

The default value for the mean square error is set to be zero. In order to compare with the *newrb* function, let *SPREAD* =1.6. Because the network can achieve zero error, all the training data will lead to the exactly fitting regression model with the parameters $m = r = 1$ and $b = 0$. Then, substitute the testing data into the trained network. The linear regression analysis of the six network outputs and targets for the testing data is shown in Table 5, where the correlation coefficients are between 0.9656 and 0.9983.

Table 5 The linear regression analysis of the outputs and targets for the testing data using *newrbe* with SPREAD=1.6.

Outputs (Targets) / Parameters	h	B	L	N_B	N_L	Cost
m	1.0406	0.9772	1.1108	1.0522	0.9823	1.0433
b	-0.0042	-0.0073	0.0241	0.0182	-0.013	0.0152
r	0.9942	0.9819	0.9656	0.9903	0.9944	0.9983

6 Conclusions

This paper first applies the genetic algorithm to engage in the optimal design of the reinforced concrete isolated footings. There are 144 different isolated footings being designed, the results of which are used as the training, validation and testing data of the artificial neural networks. From the numerical results, the principal conclusions may be summarized as follows:

(1) According to the effective number of parameters obtained from the *trainbr* function, this paper only uses 9 neurons in the hidden layer for the feedforward backpropagation network. Even if more neurons are used in the hidden layer, the accuracy does not improve significantly. The correlation coefficients between the network outputs and targets range from 0.9861 to 0.9992 for testing data. In addition, the slope of the regression line is close to 1 and the intercept with the vertical axis close to zero. The results suggest very good performance of the feedforward backpropagation network.

(2) The parameter *SPREAD*=1.6 is the most suitable value for the *newrb* design function of the radial basis network. The correlation coefficients between the network outputs and targets range from 0.9801 to 0.9939 for testing data. On the whole, its performance is a little inferior to the feedforward backpropagation network. Besides it needs more radial basis neurons in the hidden layer. If the *newrbe* function is used, the results of the regression analysis are similar to *newrb*, although it can reach zero error during the training process.

(3) After the artificial neural network is trained, it can serve as a model to optimally design the reinforced concrete isolated footings effectively and efficiently. For practical purposes, the outputs of the neural network can be rounded up to the nearest whole number.

REFERENCES

[1] Holland, J. H. (1975), *Adaptation in natural and artificial systems*, The University of Michigan Press, Ann Arbor, MI, USA.

[2] Goldberg, D. E. (1989), *Genetic algorithms in search, optimization and machine learning*, Addison Wesley, Reading, MA, USA.

[3] Coello, C. C., Hernandez, F. S., and Farrera, F. A. (1997), "Optimal design of reinforced concrete beams using genetic algorithms," *Expert Systems with Applications*, Vol. 12, No. 1, pp. 101-108.

[4] Erbatur, F, Hasancebi, O., Tutuncu, I, and Kilic, H. (2000), "Optimal design of planar and space structures with genetic algorithms," *Computers and Structures*, Vol. 75, No. 2, pp. 209-224.

[5] Coello, C. A. and Christiansen, A. D. (2000), "Multiobjective optimization of trusses using genetic algorithms," *Computers and Structures*, Vol. 75, pp. 647-660.

[6] Cheng, J and Li, Q. S. (2008), "Reliability analysis of structures using artificial neural network based genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, Vol. 197, No. 45. pp. 3742-3750.

[7] Belevičius, R. and Šešok, D. (2008), "Global optimization of grillages using genetic algorithms," *Mechanika*, Nr. 6(74), pp. 38-44.

[8] Šešok, D. and Belevičius, R. (2008), "Global optimization of trusses with a modified genetic algorithm," *Journal of Civil Engineering Management*, Vol. 14, No. 3, pp. 147-154.

[9] Chan, C. M., Zhang, L. M. and Jenny, T. N. (2009), "Optimization of pile groups using hybrid genetic algorithms, "*Journal of Geotechnical and Geoenvironmental Engineering*, Vol. 135, Issue 4, pp. 497-505.

[10] Richardson, J. N., Adriaenssens, S., Coelho, R. F. and Bouillard, P. (2013), "Coupled form-finding and grid optimization approach for single layer grid shells," *Engineering Structures*, Vol. 52, pp. 230-239.

[11] McCulloch, W.S. and Pitts, W. (1943), "A logical calculus of ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133.

[12] Rumelhart, D. E., McClelland, J. L. and the PDP Research Group (1986), *Parallel distributed processing: explorations in the microstructure of cognition. volume 1: foundations*, MIT Press, Cambridge, MA, USA.

[13] Cheng, J. and Li, Q. S. (2009), "A hybrid artificial neural network method with uniform design for structural optimization," *Computational Mechanics*, Vol. 44, No. 1, pp. 61-71.

[14] Möller, O., Foschi, R. O., Quiroz, L. M., and Rubinstein, M. (2009), "Structural optimization for performance-based design in earthquake engineering: applications of neural networks," *Structural Safety*, Vol. 31, No. 6, pp. 490–499.

[15] Gholizadeh, S. and Salajegheh, E. (2010), "Optimal design of structures for earthquake loading by self organizing radial basis function neural networks," *Advances in Structural Engineering*, Vol. 13, No. 2, pp. 339-356.

[16] Nazarko, P. and Ziemiański, L. (2011), "Application of artificial neural networks in the damage identification of structural elements," *Computer Assisted Mechanics and Engineering Sciences*, Vol. 18, No. 3, pp. 175–189.

[17] Meon, M. S., Anuar, M. A., Ramli, M. H. M., Kuntjoro, W., and Muhammad, Z. (2012), "Frame optimization using neural network", *International Journal Advanced Science Engineering Information Technology*, Vol. 2, No. 1, pp. 28-33.

[18] Ciresan, D. C., Meier, U., Masci, J. and Schmidhuber, J. (2012), "Multi-column deep neural network for traffic sign classification," *Neural Networks*, Vol. 32, pp. 333-338.

[19] Yeh, J.-P. and Yang, R.-P. (2015), "Optimal Design of Continuous Reinforced Concrete Beams Using Neural Networks," *Transactions on Machine Learning and Artificial Intelligence*, Vol. 3, No. 4, pp. 1-16.

[20] ACI (2008), *Building code requirements for structural concrete* (*ACI 318-08*) *and commentary* (*ACI 318R-08*), American Concrete Institute, Farminton Hills, MI, USA.

[21] The MathWorks (2015), *Global optimization toolbox: user's guide*, The MathWorks, Inc., Natick, MA, USA.

[22] Conn, A. R., Gould, N. I. M., and Toint, Ph. L. (1991), "A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds," *SIAM Journal on Numerical Analysis*, Vol. 28, No. 2, pp. 545-572.

[23] Conn, A. R., Gould, N. I. M., and Toint, Ph. L. (1997), "A globally convergent augmented Lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds," *Mathematics of Computation*, Vol. 66, No. 217, pp. 261-288.

[24] Demuth, H., Beale, M. and Hagan, M. (2009), *Neural network toolbox 6: user's guide*, The MathWorks, Inc., Natick, MA, USA.

[25] Hagan, M. T., Demuth, H. B. and Beale, M. H. (1996), *Neural network design*, PWS Publishing, Boston, MA, USA.

[26] Pujol, J (2007), "The solution of nonlinear inverse problems and the Levenberg-Marquardt method". *Geophysics*, Vol. 72, No. 4, W1–W16.

[27] MacKay, D. J. C. (1992), "Bayesian interpolation," *Neural Computation*, Vol. 4, No. 3, pp. 415-447.