Front. Struct. Civ. Eng. 2023, 17(3): 378–395
https://doi.org/10.1007/s11709-022-0899-9

RESEARCH ARTICLE

Optimal design of double-layer barrel vaults using genetic and pattern search algorithms and optimized neural network as surrogate model

Reza JAVANMARDI, Behrouz AHMADI-NEDUSHAN[*]

Department of Civil Engineering, Yazd University, Yazd 89195, Iran
[*]*Corresponding author. E-mail: behrooz.ahmadi@yazd.ac.ir*

ABSTRACT This paper presents a combined method based on optimized neural networks and optimization algorithms to solve structural optimization problems. The main idea is to utilize an optimized artificial neural network (OANN) as a surrogate model to reduce the number of computations for structural analysis. First, the OANN is trained appropriately. Subsequently, the main optimization problem is solved using the OANN and a population-based algorithm. The algorithms considered in this step are the arithmetic optimization algorithm (AOA) and genetic algorithm (GA). Finally, the abovementioned problem is solved using the optimal point obtained from the previous step and the pattern search (PS) algorithm. To evaluate the performance of the proposed method, two numerical examples are considered. In the first example, the performance of two algorithms, OANN + AOA + PS and OANN + GA + PS, is investigated. Using the GA reduces the elapsed time by approximately 50% compared with using the AOA. Results show that both the OANN + GA + PS and OANN + AOA + PS algorithms perform well in solving structural optimization problems and achieve the same optimal design. However, the OANN + GA + PS algorithm requires significantly fewer function evaluations to achieve the same accuracy as the OANN + AOA + PS algorithm.

KEYWORDS optimization, surrogate models, artificial neural network, SAP2000, genetic algorithm

1 Introduction

Optimization achieves the best results for a system while satisfying the constraints imposed [1]. Optimization methods have been used in various engineering fields [2–11], and several techniques have been proposed to solve optimization problems. Some algorithms require an initial population, such as the arithmetic optimization algorithm (AOA) [12] and genetic algorithms (GA) [13], whereas others perform optimization based on an initial point, such as the pattern search (PS) algorithm [14]. The former typically results in better solutions than the latter. However, initial population-based techniques incur higher computational costs than initial point-based techniques.

Commercial software, such as Abaqus, ANSYS, COMSOL, and SAP2000, are employed to solve most engineering problems. They have been developed for decades and used as reliable solutions. Researchers typically focus on the optimization of actual structures modeled using these software packages. Because the problem-solving process using these software packages is time-consuming, the use of optimization algorithms, which are typically processed using multiple calls of the objective function, poses significant challenges in solving real optimization problems. For example, the finite element analysis of a tie-arch bridge, which comprises 259 members, requires 60 s, but its optimization process requires approximately 133 h [15].

Hence, machine learning (ML) has been developed in recent years. Using ML-based surrogate models to predict the behavior of structures reduces the number of finite element analyses significantly and hence the overall computational time [16].

These models can learn the problem environment based on experiences obtained from problem solving. Different types of surrogate models exist, such as neural networks, kriging, support vector machines, and decision trees,

Article history: Received May 26, 2022; Accepted Aug 31, 2022

which can be used to solve various problems. Among the surrogate models, neural networks offer three distinct advantages, which have garnered the interest of researchers. First, neural networks can learn any complex, nonlinear environment. Second, they do not require any initial assumptions regarding the data distribution. Third, they can accommodate incomplete and lost data [17].

Surrogate models have been used to assist the metaheuristic algorithms in several studies [3,18–21]. In this study, a three-step approach is used for optimization. First, an optimized neural network is calculated. Subsequently, the main optimization problem is solved using the surrogate model and a population-based algorithm. The algorithms considered in this step are the AOA and GA. Finally, the abovementioned problem is solved using the optimal point obtained from the previous step and the PS algorithm based on the initial point. In this step, the optimal point obtained from the previous step is used as the start point. Despite all the improvements mentioned, these models are approximations of the actual model; therefore, their output cannot be used with certainty as the final output of the problem. Additionally, a surrogate model with a fixed topology was used in the aforementioned studies, which may cause an overfitting of the surrogate model.

One of the most significant issues in learning neural networks is the overfitting of learning data, where the neural network no longer improves its ability to solve the problem at a certain time during the learning period. However, it learns the random regularities in the training patterns [22–26]. Notably, determining the network size for a specific application is difficult. If the number of parameters in the network is much smaller than the total number of points in the learning set, then overfitting will not or is less likely to occur [27].

Herein, a three-step hybrid algorithm is proposed to address the time-consuming nature of optimization problems. The main objective of this method is to exploit all the advantages of optimization algorithms, commercial software, and surrogate models to solve the problem. First, a surrogate model is proposed based on optimized multilayer neural networks to minimize the risk of overfitting. Second, the optimization problem is solved using this model and a population-based algorithm. Third, the problem is solved using an actual model (i.e., using SAP2000 results) and an algorithm based on the initial point. In this step, the PS algorithm defined in the MATLAB optimization toolbox is employed; the PS algorithm uses the optimal point provided in Step 2 as the starting point.

2 Metamodels

The metamodels used were based on an approximate mathematical model of a detailed simulation model. This model predicts the output data (objective response) from the input data (design variables) in the entire design space, which is more efficient than the process of detailed simulation models [21]. The following three aspects are prioritized in the metamodel construction process: a) obtaining the initial input dataset points inside the design space; b) selecting the metamodel type to construct the approximate mathematical model; c) selecting the fitting model. These steps can be performed using various approaches [28]. Several mathematical formulations are available for constructing metamodels with different characteristics [28,29]. Although the performances of these metamodels have been compared [30–32], the most appropriate metamodel could not be determined because the comparison was based on different problems. The most typically used metamodels are polynomial regression, neural networks, and kriging [28,29].

Several types of surrogate models have been applied in structural engineering [33]. The use of kriging and support vector regression for structural engineering applications has been investigated [34]. The most typically used methods are polynomial response surfaces [32,35], polynomial chaos expansions [36], kriging or Gaussian process models [37], support vector machines [38], and artificial neural networks [39].

Neural networks are a form of computational method based on the structure of neurons in the nervous system. A neural network can detect the design of a database using learning data. The most typically used neural network is the “multilayer feedforward neural network” which includes an input layer, one or more hidden layers, and an output layer. In this network, neurons are organized into multiple layers, including the input, hidden, and output layers. The outputs from the previous neurons become the inputs of the current neuron after scaling is performed with the corresponding weights. All inputs are summed and then transformed using an activation function [16].

The backpropagation (BP) network is an outstanding representative of a multilayer feedforward neural network that uses an error BP algorithm. The error BP algorithm comprises two stages: forward propagation of the signal and BP of the error. The learning process of a neural network is based on a learning set that adjusts the connection weights between neurons and the biases of each functional neuron. The neurons of the hidden layer weigh the sum of the input variables (Eq. (1)), whereas those of the output layer weighs the sum of the input variables (Eq. (2)) [40].

$$O_i = f\left(\sum_{j=1}^{M} w_{ij}x_j + \theta_i\right), \tag{1}$$

$$O_k = \varphi\left[\sum_{i=1}^{q} w_{ki} f\left(\sum_{j=1}^{M} w_{ij}x_j + \theta_i\right) + a_k\right], \tag{2}$$

where f and φ are the activation functions of the hidden and output layers, respectively; M and q are the vector dimensions of the input and output layers, respectively; w_{ij} and w_{ki} are the weight of the input layer to the hidden layer and that of the hidden layer to the output layer, respectively; θ and a_k are the biases of the hidden and output layers, respectively.

The output error of the entire network is calculated using the error gradient descent method to adjust the weights and biases of each layer until the output of the modified network becomes similar to the expected value.

For a sample p, the error criterion function is defined as follows [41]:

$$E_p = \frac{1}{2}\sum_{k=1}^{L}(T_k^p - O_k^p)^2. \tag{3}$$

The total mean square error (mse) criterion function for learning sample P is estimated as follows [42]:

$$E_P = \frac{1}{2P}\sum_{p=1}^{P}\sum_{k=1}^{L}(T_k^p - O_k^p)^2, \tag{4}$$

where T_k^p and O_k^p are target and predicted values, respectively, L is the number of outputs.

The learning process terminates when the total error function reaches the desired value or satisfies the desired requirements [43].

3 SM toolbox

Numerous studies using artificial intelligence and optimization have been conducted in various engineering fields, including structural and earthquake engineering. In these studies, researchers used various programming methods to investigate structures. Among the methods, calling the SAP2000 software using MATLAB is one of the most effective ones. SAP2000 was developed in 1975 by Computers and Structures, Inc., which is a Berkeley University affiliate, and has been used by engineers in 160 countries [44]. In this study, the SM Toolbox, which was developed to call the SAP2000 software using MATLAB, was employed [45].

4 Parallel and serial processing

Parallel programming involves executing one or more programs simultaneously on multiple processors. In this method, a problem is segmented into several subproblems such that they can be solved simultaneously. Subsequently, each subproblem is converted into a series of parallel commands that are executed in parallel on processors. In general, parallel programming involves the use of at least two microprocessors to perform a task. Hence, scientists segment a specific problem into several components using specific software. Subsequently, each component of the problem is sent to a dedicated processor. Each processor then performs a task assigned to the problem solver, after which the software compiles the results for the final solution of the initial complex problem [46]. Parallel programming allows problems to be solved in less time, large and complex problems to be solved consistently, and several operations to be performed simultaneously. Furthermore, it allows the use of non-local resources such as computers on a network, which can significantly improve the programming capability.

5 GA and PS algorithms

The population-based algorithms considered in this study were the AOA and GA. In addition, the PS algorithm was used in the final step of the proposed method.

The GA is one of the first population-based stochastic algorithms [47] and is inspired by the Darwinian theory of evolution [48,49]. The GA assesses the suitability of each chromosome in a population using a user-defined cost function. This algorithm relies on biologically inspired natural selection, crossover, and mutation operators to achieve the optimal solution. The natural selection operator is the process by which the next generation of parents are selected. The crossover operator creates the next generation by combining the parents transferred from the previous generation. The mutation operator randomly changes the genes of the chromosomes.

The AOA is a fundamental component in number theory. It is vital to modern mathematics, along with geometry, algebra, and analysis. This algorithm is a population-based metaheuristic algorithm that can solve optimization problems without the calculation of their derivatives. The proposed AOA is primarily inspired by the use of arithmetic operators in solving arithmetic problems, in particular the addition, subtraction, multiplication, and division of numbers [12].

The PS algorithm is an effective search method for solving engineering problems that require a significant number of function evaluations (NFES) [14]. This algorithm utilizes direct search for solving optimization problems, does not require the function gradient, and aims to obtain favorable results around the current point. Direct search can be performed to solve problems in which the objective function is not differentiable or continuous [27].

6 Double-layer barrel vaults

A skeletal space frame refers to a system of individual members connected in three dimensions. It can support loads at any point or angle relative to the structure surface and direction. Furthermore, it is suitable for shielding large surfaces without internal support [50]. Space frames are widely used in designs that prioritize low weight, such as those of structural and automotive engineering, space, microlattice structures, and microelectromechanical systems [51]. One of the most important advantages of space frames is their appropriate seismic behavior arising from their light weight and high degree of static indeterminacy, which affords adequate stability in local failures. Additionally, space frames can span relatively large areas without requiring internal support. Different types of skeletal space frames are often used in the form of grids, domes, and barrel vaults. As they can span large areas without requiring internal support, they are primarily utilized in constructing sports stadiums, exhibition centers, halls, swimming pools, shopping malls, and industrial buildings. The barrel vault, which is a trendy design, is one of the oldest structures developed in the nineteenth century using iron bars [52]. In the previous century, barrel vault structures have been considered as customized roofs for shielding large areas using steel structural elements; in fact, they can span lengths exceeding 100 m [53,54]. The early types of barrel vaults were single-layer barrel vaults. Compared with single-layer barrel vaults, double-layer barrel vaults can span larger areas. These structures are considered highly stable owing to their rigidity [55].

7 Proposed algorithm

The main components of the proposed algorithm are illustrated in Fig. 1. First, the algorithm identifies the best surrogate model and the number of calls required to achieve this surrogate model. Second, the algorithm identifies the optimal point using the surrogate model obtained from the previous step and the GA. Finally, the optimal point is calculated using the PS algorithm, which is the most time-consuming function, and the optimal point obtained in the previous step. The optimal point is regarded as the starting point for the search.

In Fig. 1, Ds_{ii} is the ith data set; C and $C_{\max}$ are the neural network quality index and generated dataset, respectively; d_0 is the optimal point obtained using the surrogate model; GA is the optimization algorithm; N_S is the number of samples required to construct the surrogate

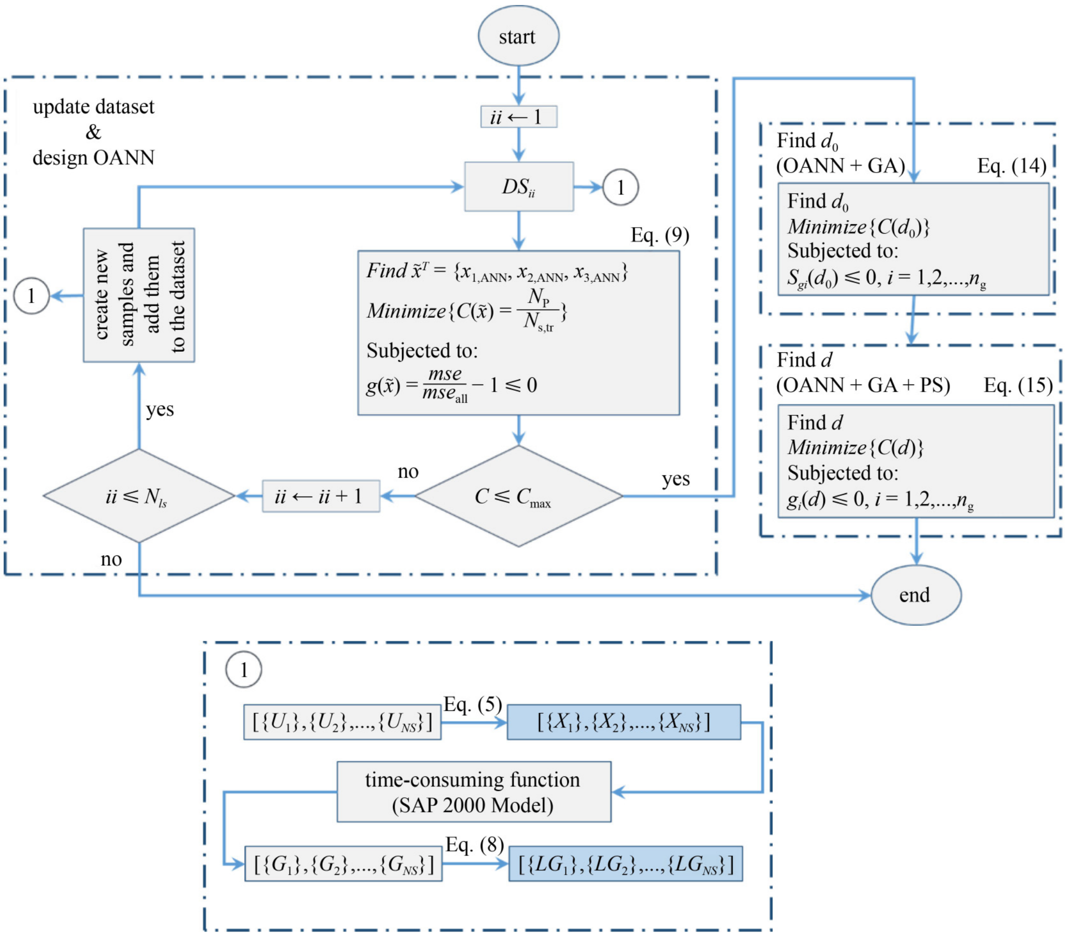

start
update dataset & design OANN
$ii \leftarrow 1$
DS_{ii}
1
Eq. (9)
Find $\bar{x}^T = \{x_{1,\text{ANN}}, x_{2,\text{ANN}}, x_{3,\text{ANN}}\}$
Minimize$\{C(\bar{x}) = \frac{N_P}{N_{s,tr}}\}$
Subjected to:
$g(\bar{x}) = \frac{mse}{mse_{all}} - 1 \leq 0$
create new samples and add them to the dataset
1
yes
$ii \leq N_{ls}$
$ii \leftarrow ii + 1$
no
$C \leq C_{\max}$
yes
no
Find d_0 (OANN + GA)
Eq. (14)
Find d_0
Minimize$\{C(d_0)\}$
Subjected to:
$S_{gi}(d_0) \leq 0, i = 1,2,...,n_g$
Find d (OANN + GA + PS)
Eq. (15)
Find d
Minimize$\{C(d)\}$
Subjected to:
$g_i(d) \leq 0, i = 1,2,...,n_g$
end
1
$[\{U_1\},\{U_2\},...,\{U_{NS}\}]$
Eq. (5)
$[\{X_1\},\{X_2\},...,\{X_{NS}\}]$
time-consuming function (SAP 2000 Model)
$[\{G_1\},\{G_2\},...,\{G_{NS}\}]$
Eq. (8)
$[\{LG_1\},\{LG_2\},...,\{LG_{NS}\}]$

Fig. 1 Flowchart of proposed algorithm.

model; N_{ls} is the number of data production processes; $\{G\}$ and $\{LG\}$ are the vectors that contain the principal constraints of the problem and substituted constraints, respectively. The final dataset is considered between vectors $\{X\}$ and $\{LG\}$. Discrete design variables are considered continuously during the data-creation stage. After obtaining the surrogate model, the discrete values are used in the final step.

7.1 Creating and updating dataset

In the first stage, i.e., the dataset-update stage, the Latin hypercube sampling method is used. The Latin hypercube method reduces the number of simulations required to obtain reasonable results. In this method, the range of possible values of each random input variable is partitioned into "strata", and a value from each stratum is randomly selected as a representative value. Subsequently, the representative values for each random variable are combined such that each representative value is considered once only in the simulation process [56]. Hence, a more uniform distribution of variables in the design space is ensured. This implies that the obtained surrogate model affords reliable performance. The sampling method for the two variables u_1 and u_2, which have values between 0 and 1, is shown in Fig. 2.

Because the input variables of the Latin hypercube sampling method have values between 0 and 1, the following linear conversion is used to convert these variables into data input variables:

$$x_i = lb_i + u_i(u_ib_i - lb_i),\ i = 1,2,\ldots n_x, \tag{5}$$

where u_i is a variable whose value is between 0 and 1, x_i is the principal variable, and n_x is the total number of variables. At this stage, the design variables are assumed to be uniformly distributed variables whose lower and upper limits change randomly. Subsequently, the finite element model of the problem is created using the SM toolbox commands, and the outputs of the problem,

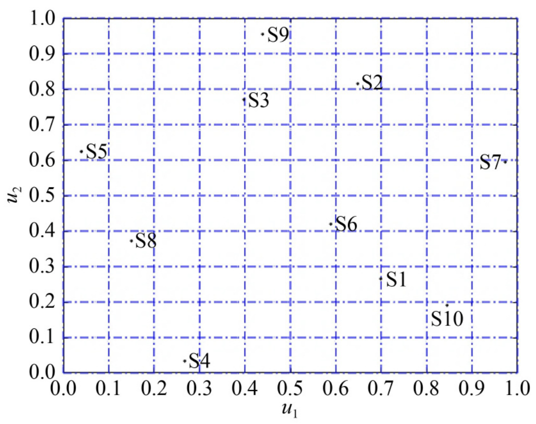


Fig. 2 Latin hypercube sampling ($n = 10$).

which are the values of the constraints, are calculated.

Based on the outputs of the SAP2000 software, the constraints of this problem are formulated as follows:

$$g_i(X) = R_i(X) - 1 \leqslant 0,\ i = 1,2,\ldots,n_g, \tag{6}$$

where n_g is the number of constraints defined in the problem, and $R_i(X)$ is the stress control coefficient or the capacity of the constituent elements of the structure, which must be less than or equal to 1. This coefficient may be expressed as different types, such as the PMM, major shear, minor shear, major beam–column capacity ratio, and minor beam–column capacity ratio [57], and always assumes a value greater than or equal to 1. Because the SAP2000 software returns a capacity factor for each load combination, the maximum value of these factors is used to calculate the constraints. The constraints of the problem (Eq. (6)) may have values from −1 to +∞, which complicates the learning process of the neural network. In this study, the following alternative constraint was utilized instead of defined constraints:

$$L_{gi} \leqslant 0,\ i = 1,2,\ldots,n_x, \tag{7}$$

where L_{gi} is calculated as

$$L_{gi} = \begin{cases} g_i, & -1 \leqslant g_i \leqslant 1, \\ 1 + \ln(g_i), & g_i > 1. \end{cases} \tag{8}$$

This function returns the same constraints for values between –1 and +1 and scales the constraints for values greater than +1.

7.2 Optimized artificial neural network

The algorithm proposed herein identifies the best surrogate model based on multilayer neural networks such that the ratio of the number of effective parameters in the neural network to the number of dataset members is in the appropriate range, and the learned network error is lower than the allowed error. Specifically, the algorithm uses a two-step approach. In the first step, a dataset of a specific size is created using the Latin hypercube sampling method. In the second step, the algorithm mimics an optimized neural network based on the existing dataset. If the quality index of the neural network is less than the desired value, then the algorithm proceeds to the next step; otherwise, the size of the dataset increases, and the problem is repeated. To calculate the quality index, the following constrained optimization problem is solved using the PS algorithm:

$$Find\ \tilde{x}^{\mathrm{T}} = \{x_{1,\mathrm{ANN}},\ x_{2,\mathrm{ANN}},\ x_{3,\mathrm{ANN}}\}$$

$$Minimize\{C(\tilde{x})\}$$

$$\text{Subjected to: } g(\tilde{x}) = \frac{mse}{mse_{\mathrm{all}}} - 1 \leqslant 0,\ \tilde{x}_{\mathrm{low}} \leqslant \tilde{x} \leqslant \tilde{x}_{\mathrm{up}}, \tag{9}$$

where $x_{1,\mathrm{ANN}}$ is the design variable representing the number of layers (Fig. 3), $x_{2,\mathrm{ANN}}$ is the design variable representing the number of neurons in each layer, $x_{3,\mathrm{ANN}}$ is the design variable representing the type of transfer function (the range of changes of this variable is shown in Table 1), $C(\tilde{x})$ is an objective function (network quality index), *mse* is the mean squared normalized error performance, and mse_{all} is the allowable mean-squared normalized error performance of the neural network.

The objective function, i.e., the network quality index, is the ratio of the number of neural network parameters to the number of members of the dataset used in the network learning operations. Thus, by minimizing this function, the surrogate neural network will contain the appropriate function (the chance of overfitting is minimized), and the number of dataset members will be sufficiently high.

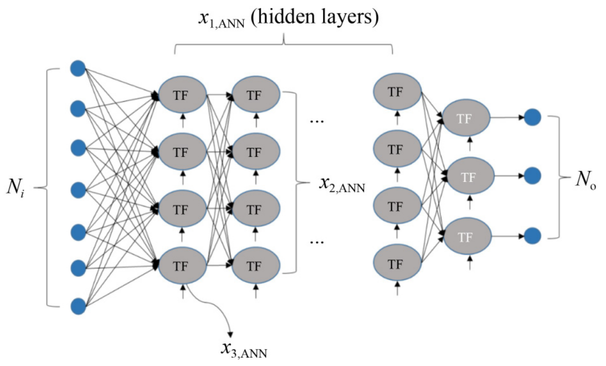
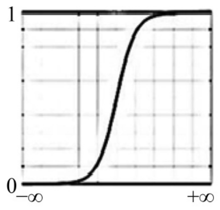
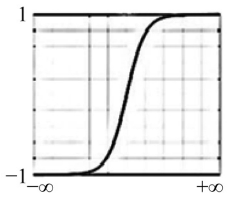
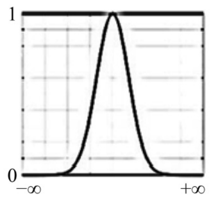


Fig. 3 Variables used in the defined neural network.

Table 1 Different types of transmission functions used in the neural network

function	figure
1: $logsig(n) = \frac{1}{1+\mathrm{e}^{-n}}$	
2: $tansig(n) = \frac{2}{1+\mathrm{e}^{-2n}} - 1$	
3: $radbas(n) = \mathrm{e}^{-n^2}$	

$$C(\tilde{x}) = \frac{N_{\mathrm{P}}}{N_{\mathrm{s,tr}}}, \tag{10}$$

where N_{P} is the number of parameters used in the neural network, and $N_{\mathrm{s,tr}}$ the number of dataset members used. The number of parameters used in the neural network is equal to the number of members in the coefficient matrices and the number of elements in the bias vectors. Each layer of a multilayer neural network contains an $S \times R$ coefficient matrix an $S \times 1$ bias vector, where S is the number of output parameters in the layer, and R is the number of input parameters. Therefore, the ith layer of a neural network exhibits the following relationship:

$$N_{\mathrm{P},i} = S\,(1+R), \tag{11}$$

where $N_{\mathrm{P},i}$ denotes the number of parameters in the ith layer. As mentioned, the number of parameters used in the neural network in this study is obtained using the following equation:

$$N_{\mathrm{P}} = x_{2,\mathrm{ANN}}\left((x_{1,\mathrm{ANN}}-1)(x_{2,\mathrm{ANN}}+1)+N_{\mathrm{i}}+N_{\mathrm{o}}+1\right)+N_{\mathrm{o}}, \tag{12}$$

where N_{i} is the number of input variables, N_{o} the number of output variables, $x_{1,\mathrm{ANN}}$ the number of hidden layers, $x_{2,\mathrm{ANN}}$ the number of neurons in each hidden layer, and $x_{3,\mathrm{ANN}}$ the type of transfer function (Table 1). Subsequently, the following equation is used to calculate the *mse*:

$$mse = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2, \tag{13}$$

where t_i is the output value in the dataset, and a_i is the corresponding output of the neural network.

7.3 Calculation of starting search point

In this step, the surrogate optimization problem is solved using the optimal neural network obtained in the previous step and the GA. The response obtained in this step is used as the starting point for the next step.

$$\text{Find } d_0$$

$$Minimize\{C(d_0)\}$$

$$\text{Subjected to: } S_{gi}(d_0) \leqslant 0,\ i = 1,2,\ldots,n_{\mathrm{g}},\ d_{\mathrm{low}} \leqslant d_0 \leqslant d_{\mathrm{up}}, \tag{14}$$

where $S_{gi}(d_0)$ is the constraint replaced by the optimized neural network; d_{low} and d_{up} are vectors containing the lower and upper limits of the design variables, respectively.

7.4 Calculation of optimal point

In this step, the original optimization problem is solved using the PS algorithm. The response obtained in this step is regarded as the final optimal point.

$$\text{Find } d$$

$$Minimize\{C(d)\}$$

$$\text{Subjected to}: \; g_i(d) \leqslant 0,\; i = 1,2,\ldots,n_g,\; d_{\text{low}} \leqslant d \leqslant d_{\text{up}}, \tag{15}$$

where $g_i(d)$ is the main constraint of the problem.

7.5 Main pseudocodes of proposed algorithm

In this section, two main pseudocodes used in the proposed algorithm are presented: Listing 1 presents the pseudocode for generating and updating the dataset and optimizing the neural network; Listing 2 presents the pseudocode of the objective function used to optimize the neural network of the proposed method.

```
1 for ii=1:Nls
2 Ut=lhsdesign(Ns,Nd)
3 for jj=1:Ns
4 [x,g]=Constraint(u);
5 T(:,end+1)=g;
6 I(:,end+1)=x;
7 end;
8 [T1]=ScalingTargets(T);
9 func=@(X)MyNet(X,I,T1,...);
10 [XBest,C,...]=patternsearch(func,...);
11 if C<=Cmax
12 break;
13 end
14 end
```

Listing 1. Pseudocode for generating and updating the dataset and optimizing the neural network.

In line 1, Nls represents the number of steps required to increase the dataset size. In line 2, the lhsdesign function generates a random sample using Latin hypercube sampling. The output of this function is matrix Ut. This matrix is composed of u vectors with values between 0 and 1. Ns and Nd represent the number of design variables and dataset records each time the dataset size is increased, respectively. In line 4, the constraint function calculates the constraints of the problem and converts vector u into the vector x. In lines 5 and 6, matrices I and T store the simulation results obtained from the Latin hypercube sampling method. In line 8, the function ScalingTargets is used. The output of this function is matrix T1, whose values are scaled based on Eq. (8). In line 9, the MyNet function handle is stored in the func variable. The input of this function is the vector of the neural network variables and the variables associated with the dataset. In line 10, the PS function returns the optimization results to determine the optimal neural network. In lines 11–14, the stopping condition of the first step of the proposed algorithm is implemented.

```
1 function [ C ]=MyNet(X,I,T,...)
2 D=round(X(1));
3 W=round(X(2));
4 Itf=round(X(3));
5 Ni=size(1,1);
6 No=size(T,1);
7 Np=W*((D−1)*(W+1)+Ni+No+1)+No;
8 Ns=size(I,2);
9 [TF]=GetTF(Itf);
10 HL=ones(1,D);
11 HL=W*HL;
12 net = feedforwardnet(HL,...);
13 net.trainParam.goal=Prf;
14 tr=net.divideParam.trainRatio;
15 Ntr=tr*Ns;
16 for ii=1:numel(HL)
17 net.layers{ii}.transferFcn=TF;
18 end
19 net = train(net,I,T,'useParallel','yes');
20 Y = net(I);
21 prf = perform(net,T,Y);
22 g=(prf/Prf)−1;
23 C=Np/Ntr;
24 C=C*(1+r*max(0,g));
25 function [ TF ]=GetTF(Itf)
26 TF=‘logsig’;
27 switch Itf
28 case 1
29 TF=‘logsig’;
30 case 2
31 TF=‘tansig’;
32 case 3
33 TF=‘radbas’;
34 end
35 end
```

Listing 2. Pseudocode of objective function used in neural network optimization.

In the pseudocode of the function above, X is the vector of variables associated with the neural network; I and T are the matrices pertaining to the dataset, respectively; C is the quality index of the network. In lines 2–4, D is the depth of the network (number of hidden layers), W the width of the network (number of neurons in each layer), and Itf a parameter specifying the transfer function type. In lines 6–8, Ni is the number of input variables, No the number of outputs, and Np the number of neural network parameters. In line 9, the transfer function type is specified using the subfunction defined in lines 25 to 35. In lines 10–11, HL is the vector that specifies the hidden layers of the neural network. In line 13, Prf denotes the target performance of the network. In line 14, tr is the percentage that determines the number of members in the dataset used in the training process. In line 15, Ntr denotes the number of records used in the training process. The for loop in lines 16–18 defines the transfer functions in the layers. In line 19, the training operation is performed using parallel processing. The value of the network quality index is calculated in lines 20–24, where prf denotes the network performance, and r the penalty factor.

8 Numerical examples

Two examples are presented in this section. In the first example, a 10-member truss is considered. The purpose of this example is to compare the performances of the OANN + AOA + PS and OANN + GA + PS algorithms. In the second example, a double-layer barrel with the scale of a real structure is investigated using a more appropriate algorithm.

In both problems, the frame elements were modeled as truss elements. In the second example, shell elements were used to model the roof-shielding elements. In both examples, the deformations were assumed to be insignificant and reflect those of linear elastic materials. Static loading was considered as well.

The approach presented herein this article is universal and thus applicable to any type of structure. For modeling, the SM toolbox commands were used to call the SAP2000 software [45].

8.1 Example1: 10-member, two-dimensional (2D) truss

In this example, the optimization of a 2D truss (Figs. 4 and 5) is presented. This example was presented in previous publications [58–60] and has been modified for the present study. Specifically, circular sections were used in the design of the structural members. The AISC 360-10 standard was employed to design the steel members. The following design load combinations were considered:

$$1)\ 1.4DL,$$

$$2)\ 1.2DL + 1.6LL. \tag{16}$$

Therefore, the 10-member truss problem is a problem affected by dead and live loads with circular cross sections and was used to evaluate the performance of the proposed algorithm. The characteristics of the constants and the design variables are listed in Table 2. The members of this structure featured circular cross sections specified in two groups, G_1 and G_2. The cross-sectional dimensions of these groups (thickness and diameter) were considered as design variables.

Based on the outputs of the SAP2000 software, the constraints of this problem are formulated as follows:

$$g_1(X) = R_1(X) - 1 \leqslant 0, \tag{17}$$

$$g_2(X) = R_2(X) - 1 \leqslant 0, \tag{18}$$

where $R_1(X)$ and $R_2(X)$ are the maximum controlled stress coefficients or the capacities of the structural elements in groups G_1 and G_2, respectively, which must be less than or equal to 1. The objective function is assumed to be equal to the weight of the truss and is calculated as follows:

$$W_t = \pi\gamma\left(2S + H + 2\sqrt{S^2 + H^2}\right)(D_{G_1}t_{G_1} + D_{G_2}t_{G_2}), \tag{19}$$

where W_t is the weight of the structure.

The change ranges of the design variables associated with the neural network are as follows:

$$X_{1,\mathrm{ANN}} \in \{1, 2, 3\}, \tag{20}$$

$$X_{2,\mathrm{ANN}} \in \{1, 3, \ldots, +\infty\}, \tag{21}$$

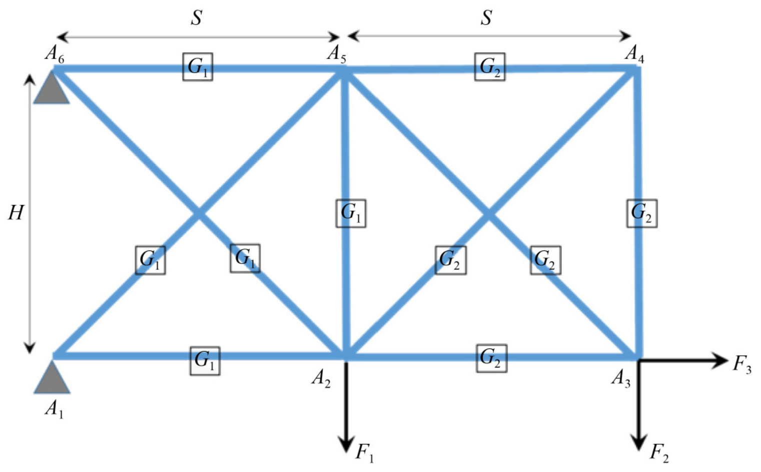


Fig. 4 10-member, 2D truss.

$$X_{3,\mathrm{ANN}} \in \{1,2,3\}. \tag{22}$$

This example was solved using the OANN+GA+PS and OANN+AOA+PS algorithms, and $C_{\max}$ was assumed to be 0.2. In both examples, the same settings were used in the optimal neural network design section, as listed in Table 3.

The performance and regression of the optimized neural network are shown in Figs. 6–10. The architecture of optimal neural network is depicted in Fig. 11.

Table 4 lists the optimal parameters used in the neural network.

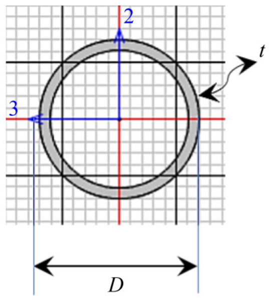


Fig. 5 Parameters D,t.

As shown in Table 4, the surrogate model can achieve a quality index of 0.176 via 300 function calls and using one hidden layer, including four neurons and a logsig transfer function.

8.1.1 OANN + GA + PS

Table 5 lists the parameters used in the GA.

Crossoverscattered creates a random binary vector and selects the genes whose vector is 1 from the first parent, and the genes whose vector is 0 from the second parent, and then combines the genes to form a child. Mutationgaussian adds a random number obtained from a Gaussian distribution with a mean of 0 for each entry of the parent vector [27].

Figure 12 shows the convergence diagram (semi-logarithmic) for the first step (OANN + GA), and Fig. 13 illustrates the final step (OANN + GA + PS) of the algorithm.

Table 6 lists the optimal parameters obtained using the OANN + GA + PS algorithm.

As shown in Table 6, the optimal point is obtained with a slight change in the initial point, indicating the proper

Table 2 Design constants and variables used for solving 2D truss problem

items	remarks	type	value
E	modulus of elasticity of steel	constant	210×10^3 N/mm^2
F_y	steel yield stress	constant	355 N/mm^2
γ	specific gravity of steel	constant	7850 kg·f·m^{-3}
$F_{1,\mathrm{DL}}$	dead load applied to the structure	constant	18 kN
$F_{1,\mathrm{LL}}$	live load applied to the structure	constant	10 kN
$F_{2,\mathrm{DL}}$	dead load applied to the structure	constant	18 kN
$F_{2,\mathrm{LL}}$	live load applied to the structure	constant	10 kN
$F_{3,\mathrm{DL}}$	dead load applied to the structure	constant	18 kN
$F_{3,\mathrm{LL}}$	live load applied to the structure	constant	10 kN
S	dimensional parameter	constant	3000 mm
H	dimensional parameter	constant	3000 mm
$d_1\left(D_{\mathrm{G}_1}\right)$	circular cross-section diameter, group 1	design variable	20 : 0.1 : 150 mm
$d_2\left(t_{\mathrm{G}_1}\right)$	circular cross-section thickness, group 1	design variable	2 : 0.1 : 10 mm
$d_3\left(D_{\mathrm{G}_2}\right)$	circular cross-section diameter, group 2	design variable	20 : 0.1 : 150 mm
$d_4\left(t_{\mathrm{G}_2}\right)$	circular cross-section thickness, group 2	design variable	2 : 0.1 : 10 mm

Table 3 Parameters of the neural networks

items	remarks	values
trainfcn	the algorithm used in network learning	Levenberg–Marquardt backpropagation
dividefcn	a function that determines the manner by which the members are divided	random
trainratio	the ratio of the number of dataset members used to learn the network to the total members of the dataset	0.7
valratio	the ratio of the number of dataset members used to validate the network to the total dataset members	0.15
testratio	the ratio of the number of dataset members used to test the network to the total members of the dataset	0.15

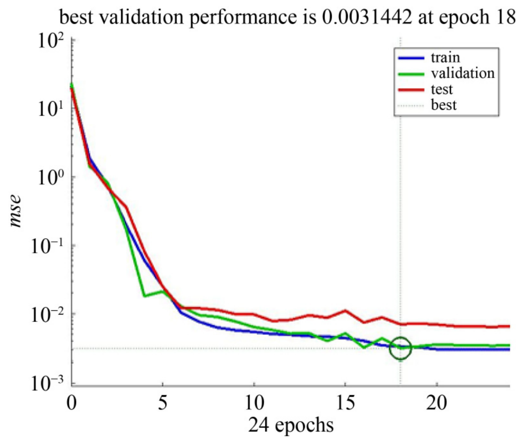


Fig. 6 Error vs. epoch for the training, validation, and test performances of the optimized neural network for 10-member truss.

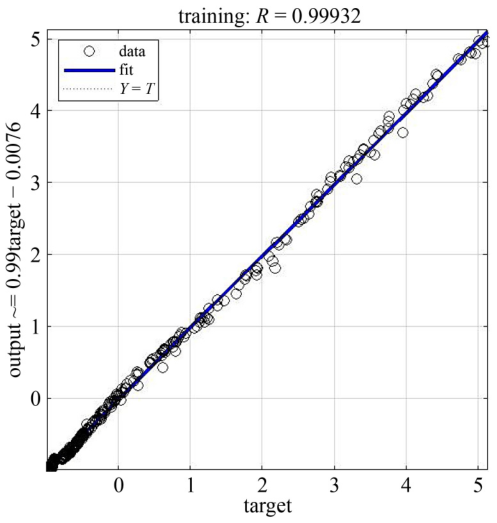


Fig. 7 Regression of learning data in optimized neural network for 2D, 10-member truss.

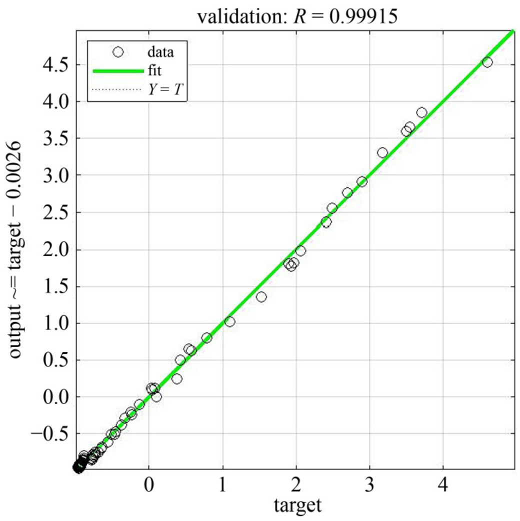


Fig. 8 Regression of validation data in optimized neural network for 2D, 10-member truss.

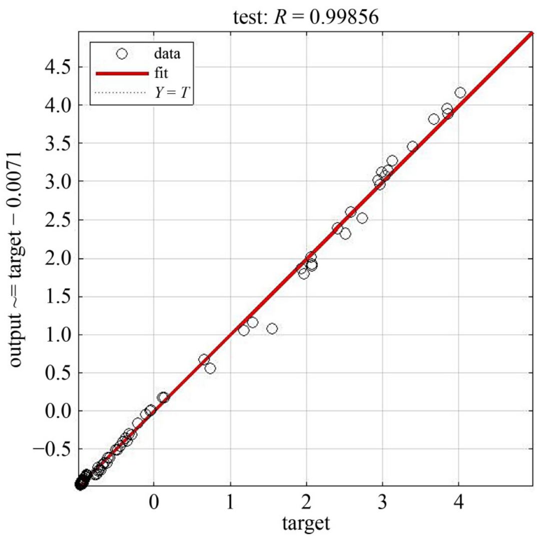


Fig. 9 Regression of test data in optimized neural network for 2D, 10-member truss.

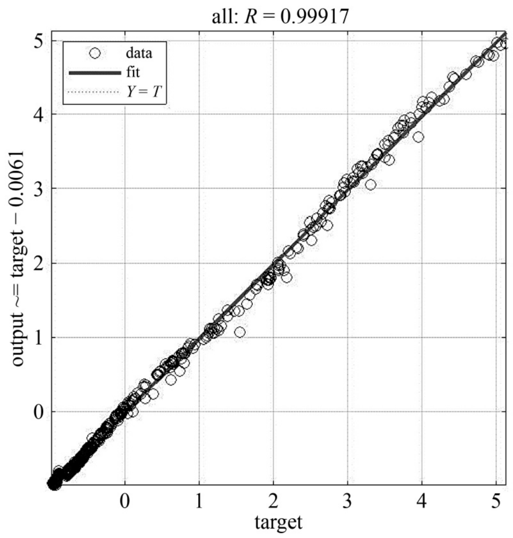


Fig. 10 Regression of all data in optimized neural network for 2D, 10-member truss.

performance of the first two parts of the algorithm (OANN + GA).

8.1.2 OANN + AOA + PS

Table 7 lists the parameters used in the AOA.

Figures 14 and 15 show the convergence diagrams in the first (OANN + AOA) and final (OANN + AOA + PS) stages, respectively.

Table 8 presents the optimal parameters obtained using the OANN + AOA + PS algorithm.

As shown in Table 8, the optimal point is obtained with a slight change in the initial point, indicating the proper performance of the first two parts of the algorithm (OANN + AOA).

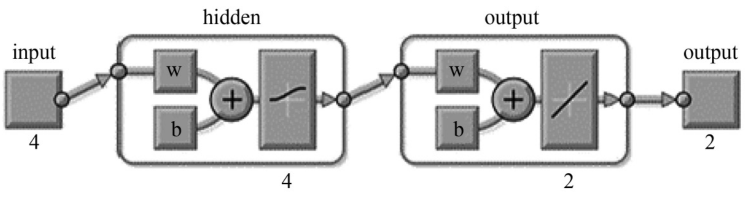


Fig. 11 Neural network optimized to solve problem of 2D, 10-member truss.

Table 4 Optimal parameters used in neural network for solving 10-member truss

item	remark	value
N_{st}	number of records required to create dataset	300
X_{OANN}	optimal variables associated with the neural network	[1,4,1]
C	quality index of surrogate model	0.176

Table 5 Parameters used in the GA

item	remark	value
populationsize	specifies the number of individuals in each generation [27]	50
crossoverfcn	the function used by the algorithm to create crossover children [27]	crossoverscattered
crossoverfraction	the fraction of the population in the next generation, not including elite children, that the crossover function creates [27]	0.8
elitecount	a positive integer specifying the number of individuals in the current generation guaranteed to survive in the next generation [27]	0.05 × *PopulationSize*
mutationfcn	a function that yields mutation children [27]	mutationgaussian

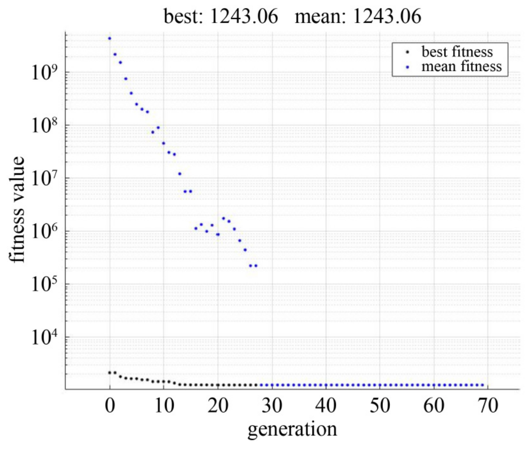


Fig. 12 Convergence diagram (semi-logarithmic) of first stage (OANN + GA) for 10-member truss.

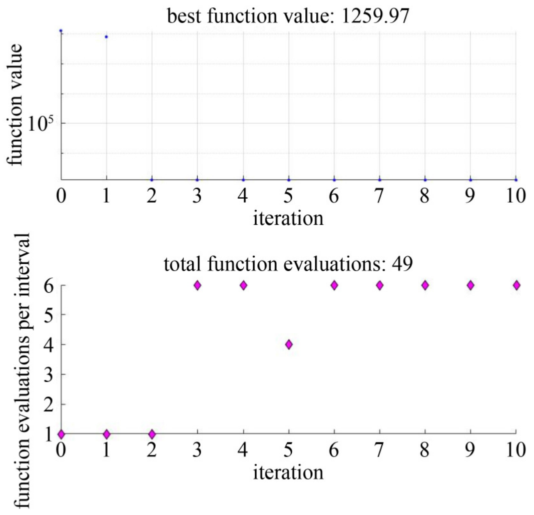


Fig. 13 Convergence diagram (semi-logarithmic) of second stage (OANN + GA + PS) for 10-member truss.

8.1.3 Convergence and runtime

The first example was solved using an Intel® Core (TM) i5-6200U CPU@2.3 GHz processor with two physical and four logic cores.

The elapsed times for the different stages of the proposed methodology are listed in Table 9. As shown, the elapsed time for modeling and calculating the outputs to create a dataset of 300 records is 1110 s, and the elapsed times to identify the initial search point by the AOA + PS and GA + PS algorithms are 374 and 186 s, respectively. Notably, using the GA algorithm reduced the elapsed time by approximately 50% as compared with using the AOA.

The total elapsed times for solving the problem using the OANN + GA + PS and OANN + AOA + PS algorithms were 1371 and 1784 s, respectively. To demonstrate the efficiency of the ANN as a surrogate, the problem was solved without using the optimized ANN as a surrogate model, i.e., using the final two components of the proposed method for the main time-consuming function. In this case, the elapsed times were 60360 and

Table 6 Optimal parameters obtained using OANN + GA + PS algorithm

item	remark	value
d_0	starting point	[83,2,64,2]
d_{opt}	final point (optimal design)	[86,2,63,2]
$W_{\text{t,opt}}\,(N)$	optimal weight of the structure	1.26×10^3 (N)

Table 7 Parameters used in AOA

items	remark	value
solution no.	number of search solutions	20
m_iter	maximum number of iterations	1000

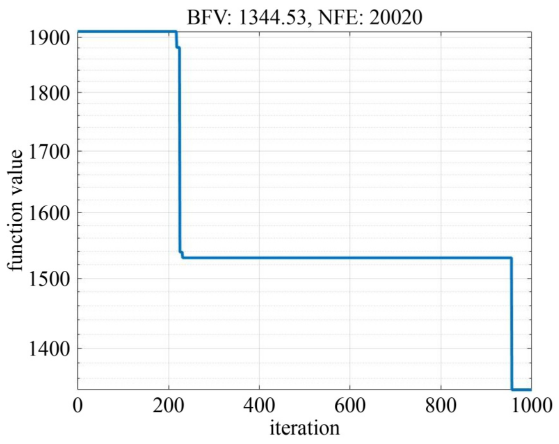


Fig. 14 Convergence diagram (semi-logarithmic) of first step (OANN + AOA).

9975 s for the AOA + PS and GA + PS algorithms, respectively. The optimized algorithm proved to be extremely effective as it reduced the elapsed times by up to 34 and 7 times for the AOA + PS and GA + PS algorithms, respectively.

The algorithms were executed 10 times to reduce the probability of premature convergence, and the best results are reported. Based on the results, both the OANN + GA + PS and OANN + AOA + PS algorithms performed well in solving the optimal design problem. Notably, both algorithms achieved the same optimal design. However, in the final stage, the NFES for the OANN + GA + PS and OANN + AOA + PS algorithms was 50 and 72, respectively. Notably, the OANN + GA + PS algorithm required a significantly lower NFES to achieve the same accuracy as the OANN + AOA + PS algorithm.

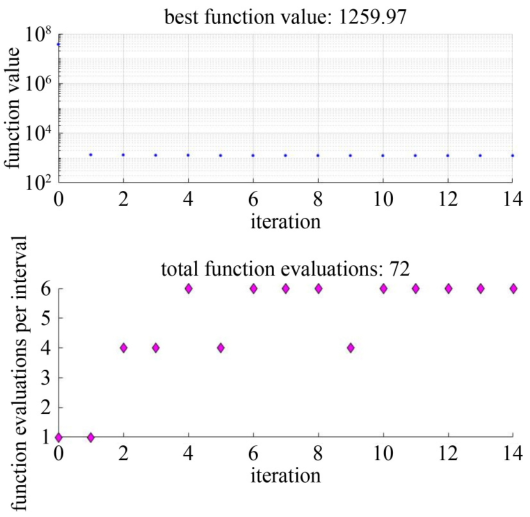


Fig. 15 Convergence diagram (semi-logarithmic) of second stage (OANN + AOA + PS).

Table 8 Optimal parameters obtained using OANN + AOA + PS algorithm

item	remark	value
d_0	search starting point	[85,2,74,2]
d_{opt}	the optimal	[86,2,63,2]
$W_{\text{t,opt}}\,(N)$	optimal weight of the structure	1.26×10^3 (N)

In the first phase of the search, the convergence diagram of the GA (Fig. 12) of first stage (OANN + GA) for 10-member truss.) shows that the GA achieved the optimal design in generation 24, and that the curves remained the same up to generation 69, thereby allowing the algorithm to obtain the optimal point. In the second phase of the search using the PS algorithm, the convergence plot is completely flat from the iteration 2 which indicates the optimal point is obtained with a slight change in the initial point. Because the GA and PS algorithms use different principles to obtain the optimal point and because both algorithms achieve the same optimal design, one can assume that premature convergence was avoided, and that a global optimum or nearly optimum design was achieved.

Table 9 Comparison of elapsed times for different stages of proposed methodology

method	component(s)			
	preparation of dataset and OANN training	identify the initial search point	identify the optimal point	total elapsed time
OANN + AOA + PS	1110	374	300	1784
AOA + PS (includes only time-consuming process)	–	60060	300	60360
GA + AOA + PS	1110	186	75	1371
GA + PS (includes only time-consuming process)	–	9900	75	9975

8.2 Example 2: 832-member, double-layer barrel vaults

In this example, a parametric code is developed to create a finite element model using the SM toolbox. The finite element model was created automatically by determining the input parameters, whereas the weight of the structure and considered constraints were calculated. The main parameters include the structural dimensions, number of spans in the *Y*-direction, number of panels in the *X*- and *Y*-directions, thickness between the top and bottom layers, dead and snow loads, specifications of steels used in building sections, and dimensions of the sections. As mentioned previously, this program uses several subroutines in modeling. Table 10 summarizes the performance of this subroutine.

Table 10 Subroutines created to form the finite element model

subprogram	task
B.L.M	to model the bottom layer of the roof
T.L.M	to model the upper layer of the roof
B.M	to model diagonal members between two roof layers
R.M	to create roof elements

Figures 16–18 show the submodels developed after the implementation of these subroutines. Figure 19 shows the final structure of a numerical example as a result of the abovementioned subroutines.

The objective function is assumed to be equal to the truss weight and is calculated using the following equation:

$$W_t \approx \gamma\pi \sum_{i=1}^{N_{el}} D_i t_i L_i, \tag{23}$$

where W_t is the structural weight, γ is the specific weight of steel, D_i is the cross-sectional diameter of each element, t_i is the thickness of each element, L_i is the length of each element, and N_{el} is the number of structural elements.

The code generated by the SM Toolbox automatically categorized the members of the roof layers into side and middle groups. The categorization was performed such that the number of members in the side and middle groups was almost equal (Fig. 20). The design constants and variables used in the optimization problem are shown in Table 11.

The change ranges of the design variables associated with the neural network are as follows:

$$X_{1,\mathrm{ANN}} \in \{1,2,3\}, \tag{24}$$

$$X_{2,\mathrm{ANN}} \in \{1,2,\ldots,+\infty\}, \tag{25}$$

$$X_{3,\mathrm{ANN}} \in \{1,2,3\}, \tag{26}$$

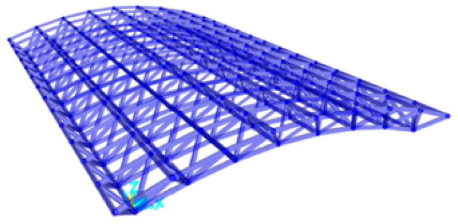


Fig. 16 Bracing elements.

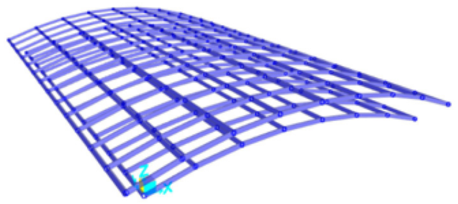


Fig. 17 Upper and lower layers.

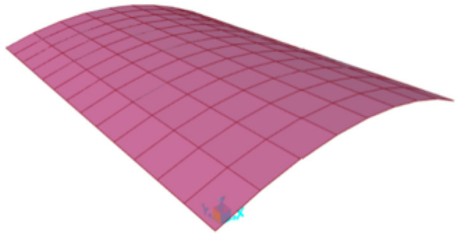


Fig. 18 Roof elements.

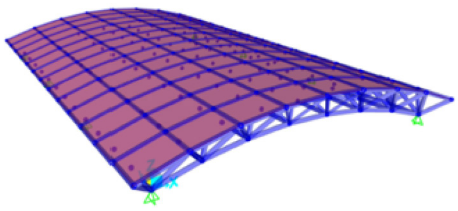


Fig. 19 Final structure.

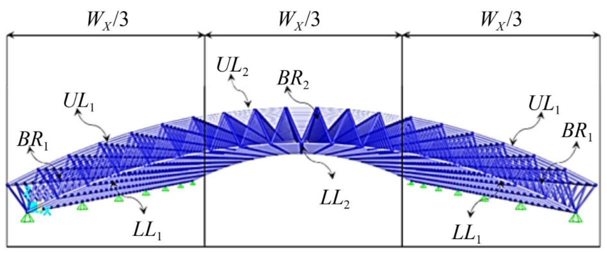


Fig. 20 Categorization of the structural elements.

where $C_{\max} = 0.4$.

The constraints of this problem, based on the outputs of the SAP2000 software, are formulated as follows:

$$g_1(X) = R_1(X) - 1 \leqslant 0, \tag{27}$$

$$g_2(X) = R_2(X) - 1 \leqslant 0, \tag{28}$$

Table 11 Design constants and variables used for solving a double-layer barrel vault

items	remarks	type	values
E	modulus of elasticity of steel	constant	210×10^3 N/mm^2
F_y	steel yield stress	constant	335 N/mm^2
γ	specific gravity of steel	constant	7850 kg·f/m^3
SL	amount of snow load	constant	1000 N/mm^2
DL	amount of dead load	constant	2500 N/mm^2
W_X	area dimension in the direction of the x-axis	constant	40000 mm
W_Y	area dimension in the direction of the y-axis	constant	60000 mm
N_{DX}	number of panels in x-direction	constant	7
N_{DY}	number of panels in y-direction, in one span	constant	4
N_{BY}	number of spans in y-direction	constant	3
K_R	effective buckling length coefficient in the elements used in the roof of the structure	constant	1
$d_1\,(H_{\mathrm{Roof}})$	roof thickness	design variable	1000 : 50 : 4000 mm
$d_2\,(f)$	height of the highest point of the parabolic roof	design variable	3000 : 50 : 6000 mm
$d_3\,(D_{\mathrm{UL1}})$	diameter of the sections used in the upper layer of the roof, group 1	design variable	50 : 5 : 250 mm
$d_4\,(t_{\mathrm{UL1}})$	thickness of the sections used in the upper layer of the roof, group 1	design variable	3 : 1 : 5 mm
$d_5\,(D_{\mathrm{UL2}})$	diameter of the sections used in the upper layer of the roof, group 2	design variable	50 : 5 : 250 mm
$d_6\,(t_{\mathrm{UL2}})$	thickness of the sections used in the upper layer of the roof, group 2	design variable	3 : 1 : 5 mm
$d_7\,(D_{\mathrm{BR1}})$	diameter of the sections used in the braces, group 1	design variable	50 : 5 : 250 mm
$d_8\,(t_{\mathrm{BR1}})$	thickness of the sections used in the braces, group 1	design variable	3 : 1 : 5 mm
$d_9\,(D_{\mathrm{BR2}})$	diameter of the sections used in the braces, group 2	design variable	50 : 5 : 250 mm
$d_{10}\,(t_{\mathrm{BR2}})$	thickness of the sections used in the braces, group 2	design variable	3 : 1 : 5 mm
$d_{11}\,(D_{\mathrm{LL1}})$	diameter of the sections used in the lower layer of the roof, group 1	design variable	50 : 5 : 250 mm
$d_{12}\,(t_{\mathrm{LL1}})$	thickness of the sections used in the lower layer of the roof, group 1	design variable	3 : 1 : 5 mm
$d_{13}\,(D_{\mathrm{LL2}})$	diameter of the sections used in the lower layer of the roof, group 2	design variable	50 : 5 : 250 mm
$d_{14}\,(t_{\mathrm{LL2}})$	thickness of the sections used in the lower layer of the roof, group 2	design variable	3 : 1 : 5 mm

$$g_3(X) = R_3(X) - 1 \leqslant 0, \quad (29)$$

$$g_4(X) = R_4(X) - 1 \leqslant 0, \quad (30)$$

$$g_5(X) = R_5(X) - 1 \leqslant 0, \quad (31)$$

$$g_6(X) = R_6(X) - 1 \leqslant 0, \quad (32)$$

where $R_1(X)$ and $R_2(X)$ represent the maximum stress control coefficients or capacity for the frame elements in the side and middle members of the lower layer of the barrel vault, respectively (Fig. 20); $R_3(X)$ and $R_4(X)$ represent the maximum stress control coefficients or capacity for the frame elements in the side and middle members of the brace members, respectively (Fig. 20); $R_5(X)$ and $R_6(X)$ represent the maximum stress control coefficients or capacity for the frame elements in the side and middle members of the upper layer of the barrel vault, respectively (Fig. 20). The parameters defining the geometry of the structure are shown in Figs. 21 and 22.

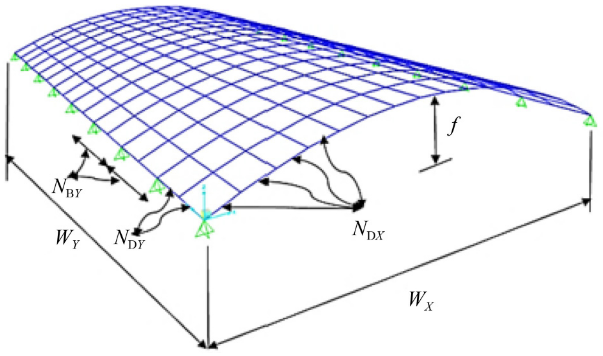


Fig. 21 Parameters $N_{DX}, N_{DY}, W_X,$ and W_Y.

AISC360-05 in the SAP2000 software was used to design the steel members. The parameters used in the neural network are presented in Table 12. Figures 23–27 show the performance and regression of the optimized neural network. The architecture of optimal neural network is depicted in Fig. 28.

As shown in Table 12, the surrogate model achieved a

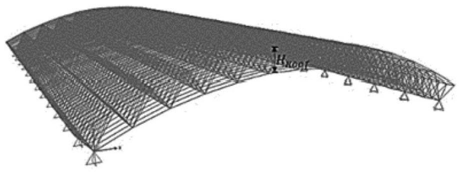


Fig. 22 Parameter H_{Roof}.

Table 12 Optimal parameters of neural network for double-layer barrel vault

item	remarks	value
N_{st}	number of records required to create data	1850
X_{OANN}	optimal variables related to the neural network	[3,11,1]
C	surrogate model quality index	0.394

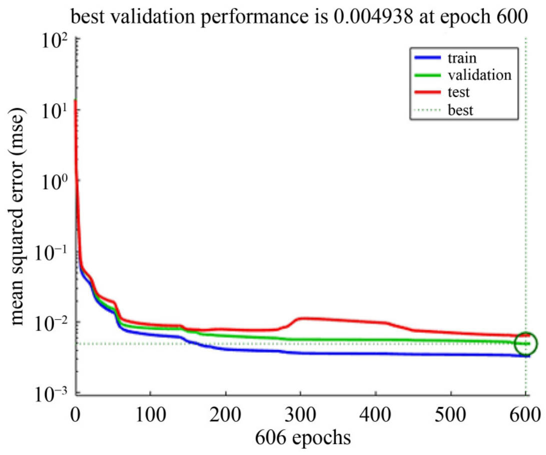


Fig. 23 Error vs. epoch for the training, validation, and test performances of the optimized neural network for double layer barrel vault.

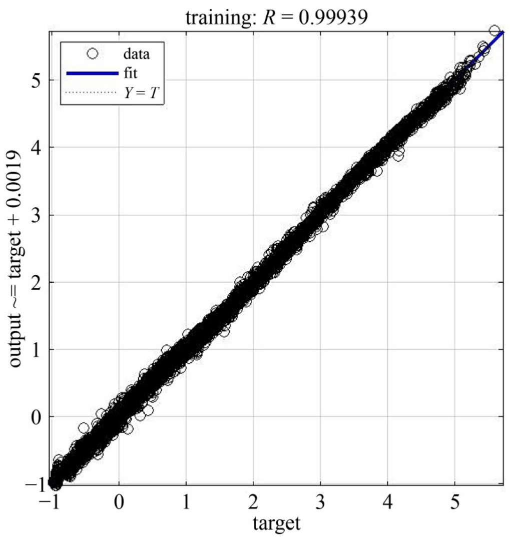


Fig. 24 Regression of optimized neural network learning data for double-layer barrel vault.

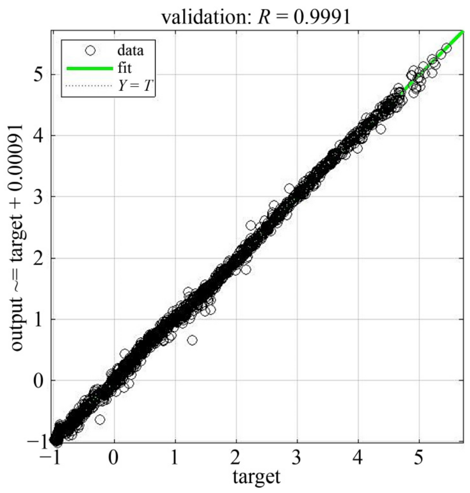


Fig. 25 Regression of optimized neural network validation data for double-layer barrel vault.

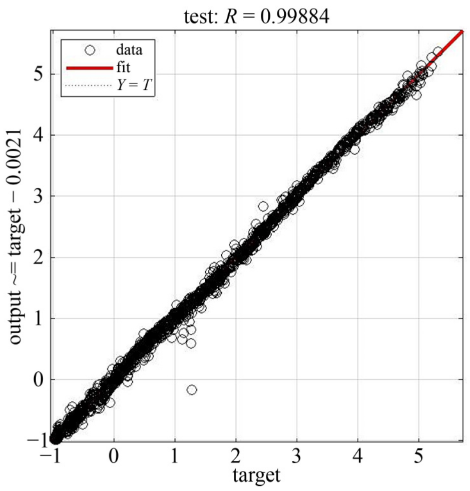


Fig. 26 Regression of optimized neural network test data for double-layer barrel vault.

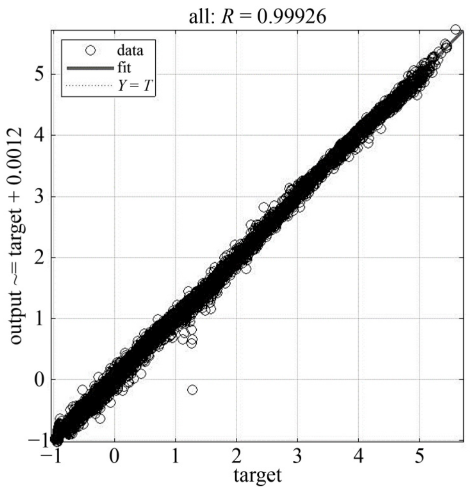


Fig. 27 Regression of all optimized neural network data for double-layer barrel vault.

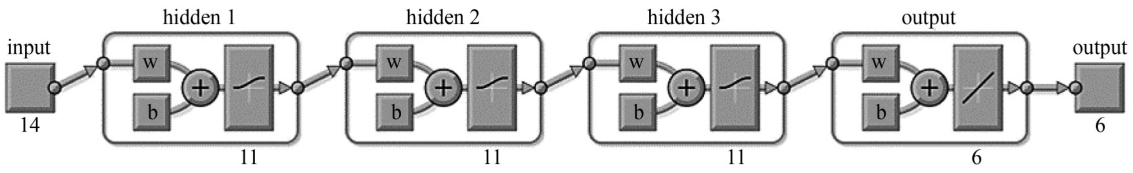


Fig. 28 Optimized neural network for double-layer barrel vault.

quality index of 0.394 using 1850 function calls, three hidden layers including 11 neurons, and a logsig transmission function.

Figures 29 and 30 show the convergence diagrams (semi-logarithmic) for a double-layer barrel vault in the first (OANN + GA) and second (OANN + GA + PS) stages, respectively.

The optimal parameters obtained using the OANN + GA + PS algorithm are shown in Table 13.

The results show that the proposed algorithm can achieve the optimal response 2352 times (1850 + 502) by calling the time consuming function.

Table 13 Optimal parameters obtained using OANN + GA + PS algorithm

item	remark	value
d_0	search starting point	[2350, 5900, 220, 4, 245, …, 4, 220, 4, 115, 3, 235, 5, 150, 4]
d_{opt}	the optimal	[2050, 5500, 205, 5, 230, …, 4, 225, 4, 60, 5, 235, 5, 145, 4]
$W_{\mathrm{t,opt}}$ (N)	optimal weight of the structure	8.5602×10^5

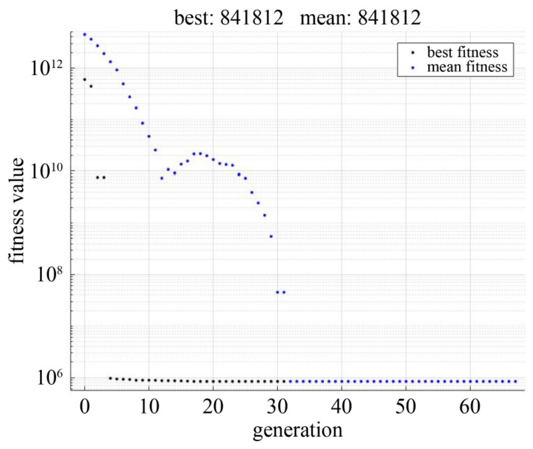


Fig. 29 Convergence diagram (semi-logarithmic) of double-layer barrel vault in first stage (OANN + GA).

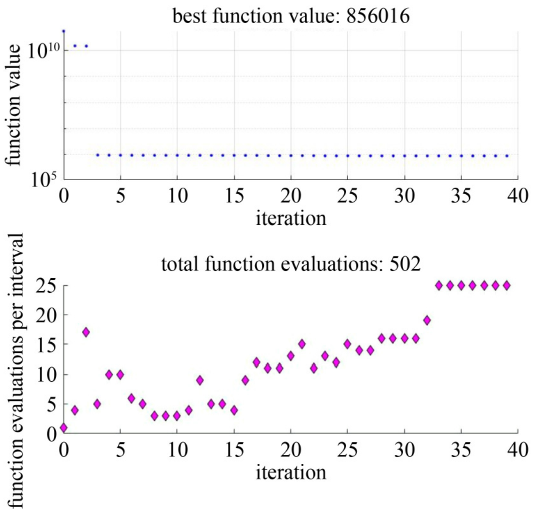


Fig. 30 Convergence diagram (semi-logarithmic) of double-layer barrel vault in second stage (OANN + GA + PS).

9 Conclusions

Herein, a combined method based on optimized neural networks and optimization algorithms was presented to solve optimization problems involving time-consuming functions. The main structure of this method comprises three steps: In the first step, an optimized neural network is calculated. Subsequently, the main optimization problem is solved using the surrogate model and a population-based algorithm. The algorithms considered in this step are the AOA and GA. In the final step, the abovementioned problem is solved using the optimal point obtained from the previous step and the PS algorithm. In this step, the optimal point obtained from the previous step is used as the initial point. As such, the main time-consuming function need not be called throughout the optimization process to perform the optimization, and the problem is solved using fewer calls. A surrogate function is introduced for the main problem constraints to improve the efficiency of the optimized neural network. Using the GA reduced the elapsed time by almost 50% compared with using the AOA.

Furthermore, the optimized algorithm proved to be extremely effective as it reduced the elapsed times by up to 34 and 7 times for the AOA + PS and GA + PS algorithms, respectively. The results showed that both the OANN + GA + PS and OANN + AOA + PS algorithms performed well in solving the optimal design problem, and that both algorithms achieved the same optimal design. However, in the final stage, the NFES for the OANN + GA + PS and OANN + AOA + PS algorithms was 50 and 72, respectively. Notably, the OANN + GA + PS algorithm required significantly fewer function evaluations to achieve the same accuracy as the OANN + AOA + PS algorithm.

References

1. Haftka R T, Gürdal Z. Elements of Structural Optimization. 3rd ed. Berlin: Springer Science & Business Media, 1992
2. Shobeiri V, Ahmadi-Nedushan B. Bi-directional evolutionary structural optimization for strut-and-tie modelling of three-dimensional structural concrete. Engineering Optimization, 2017, 49(12): 2055−2078
3. Fathnejat H, Ahmadi-Nedushan B. An efficient two-stage approach for structural damage detection using meta-heuristic algorithms and group method of data handling surrogate model. Frontiers of Structural and Civil Engineering, 2020, 14(4): 907−929
4. Bekdaş G, Yücel M, Nigdeli S M. Estimation of optimum design of structural systems via machine learning. Frontiers of Structural and Civil Engineering, 2021, 15(6): 1441−1452
5. Nguyen-Thoi T, Tran-Viet A, Nguyen-Minh N, Vo-Duy T, Ho-Huu V. A combination of damage locating vector method (DLV) and differential evolution algorithm (DE) for structural damage assessment. Frontiers of Structural and Civil Engineering, 2018, 12(1): 92−108
6. Es-Haghi MS, Shishegaran A, Rabczuk T. Evaluation of a novel Asymmetric Genetic Algorithm to optimize the structural design of 3D regular and irregular steel frames. Frontiers of Structural and Civil Engineering, 2020, 14(5):1110–1130
7. Ghasemi H, Park H S, Rabczuk T. A multi-material level set-based topology optimization of flexoelectric composites. Computer Methods in Applied Mechanics and Engineering, 2018, 332: 47−62
8. Ahmadi-Nedushan B, Varaee H. Optimal design of reinforced concrete retaining walls using a swarm intelligence technique. In: The First International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering. Funchal: Civil-Comp Press, 2009
9. Shakiba M, Ahmadi-Nedushan B. Engineering optimization using opposition based differential evolution. In: The First International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering. Funchal: Civil-Comp Press, 2009
10. Ahmadi-Nedushan B, Varaee H. Minimum cost design of concrete slabs using particle swarm optimization with time varying acceleration coefficients. World Applied Sciences Journal, 2011, 13(12): 2484−2494
11. Ahmadi-Nedushan B, Fathnejat H. A modified teaching–learning optimization algorithm for structural damage detection using a novel damage index based on modal flexibility and strain energy under environmental variations. Engineering with Computers, 2022, 38: 847−874
12. Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, Gandomi A H. The arithmetic optimization algorithm. Computer Methods in Applied Mechanics and Engineering, 2021, 376: 113609
13. Goldberg D E. Genetic Algorithms in Search and Optimization. Optimization. 13th ed. Boston: Addison-Wesley Professional, 1989
14. Booker A J, Dennis J E, Frank P D, Serafini D B, Torczon V, Trosset M W. A rigorous framework for optimization of expensive functions by surrogates. Structural Optimization, 1999, 17(1): 1−13
15. Latif M A, Saka M P. Optimum design of tied-arch bridges under code requirements using enhanced artificial bee colony algorithm. Advances in Engineering Software, 2019, 135: 102685
16. Nguyen T H, Vu A T. Using neural networks as surrogate models in differential evolution optimization of truss structures. In: International Conference on Computational Collective Intelligence. Da Nang: Springer, 2020
17. Vellido A, Lisboa P J G, Vaughan J. Neural networks in business: A survey of applications (1992−1998). Expert Systems with Applications, 1999, 17(1): 51−70
18. Papadrakakis M, Lagaros N D, Tsompanakis Y. Optimization of large-scale 3-D trusses using evolution strategies and neural networks. International Journal of Space Structures, 1999, 14(3): 211−223
19. Kaveh A, Gholipour Y, Rahami H. Optimal design of transmission towers using genetic algorithm and neural networks. International Journal of Space Structures, 2008, 23(1): 1−19
20. Krempser E, Bernardino H S, Barbosa H J C, Lemonge A C C. Differential evolution assisted by surrogate models for structural optimization problems. Civil-Comp Proc., 2012, 100
21. Penadés-Plà V, García-Segura T, Yepes V. Accelerated optimization method for low-embodied energy concrete box-girder bridge design. Engineering Structures, 2019, 179: 556−565
22. Wang J H, Jiang J H, Yu R Q. Robust back propagation algorithm as a chemometric tool to prevent the overfitting to outliers. Chemometrics and Intelligent Laboratory Systems, 1996, 34(1): 109−115
23. Panchal G, Ganatra A, Shah P, Panchal D. Determination of over-learning and over-fitting problem in back propagation neural network. International Journal of Soft Computing, 2011, 2(2): 40−51
24. Dietterich T. Overfitting and undercomputing in machine learning. ACM Computing Surveys, 1995, 27(3): 326−327
25. Lawrence S, Giles C L, Tsoi A C. Lessons in neural network training: overfitting may be harder than expected. In: Proceedings of the 14th National Conference on Artificial Intelligence. Menlo Park: AAAI, 1997
26. Sarle W S. Stopped Training and Other Remedies for Overfitting. In: Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics. Pittsburgh: Interface Foundation of North America, 1995
27. MathWorks. MATLAB Documantation. Natick: MathWorks, 2019
28. Barton R R, Meckesheimer M. Handbooks in Operations Research and Management Science 13. 2006
29. Simpson T W, Peplinski J D, Koch P N, Allen J K. Metamodels for computer-based engineering design: Survey and recommendations. Engineering with Computers, 2001, 17(2): 129−150
30. Jin R, Chen W, Simpson T W. Comparative studies of metamodelling techniques under multiple modelling criteria. Structural and Multidisciplinary Optimization, 2001, 23(1): 1−13
31. Li Y F, Ng S H, Xie M, Goh T N. A systematic comparison of metamodeling techniques for simulation optimization in Decision Support Systems. Applied Soft Computing, 2010, 10(4): 1257−1273
32. Kim B S, Bin Lee Y, Choi D H. Comparison study on the accuracy of metamodeling technique for non-convex functions. Journal of Mechanical Science and Technology, 2009, 23(4): 1175−1181

33. Moustapha M, Bourinet J M, Guillaume B, Sudret B. Comparative study of Kriging and support vector regression for structural engineering applications. ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems. Part A: Civil Engineering, 2018, 4(2): 0000950
34. Jiang C, Cai X, Qiu H, Gao L, Li P. A two-stage support vector regression assisted sequential sampling approach for global metamodeling. Structural and Multidisciplinary Optimization, 2018, 58(4): 1657−1672
35. Ahmadi Nedushan B, Chouinard L E. Statistical analysis in real time of monitoring data for Idukki Arch Dam. In: Annual Conference of the Canadian Society for Civil Engineering. Moncton: Canadian Society for Civil Engineering, 2003
36. Ghanem R, Spanos P D. Stochastic Finite Elements: A Spectral Approach. New York: Springer, 2003
37. Deng L Y. The design and analysis of computer experiments. Technometrics, 2004, 46(4): 488−489
38. Smola A J, Schölkopf B. A tutorial on support vector regression. Statistics and Computing, 2004, 14(3): 199−222
39. Ahmadi Nedushan B, Chouinard L E. Use of artificial neural networks for real time analysis of dam monitoring data. In: Annual Conference of the Canadian Society for Civil Engineering. Moncton: Canadian Society for Civil Engineering, 2003
40. Deng J, Gu D, Li X, Yue Z Q. Structural reliability analysis for implicit performance functions using artificial neural network. Structural Safety, 2005, 27(1): 25−48
41. Mukhopadhyay A, Iqbal A. Prediction of mechanical properties of hot rolled, low-carbon steel strips using artificial neural network. Materials and Manufacturing Processes, 2005, 20(5): 793−812
42. Mukhopadhyay A, Iqbal A. Comparison of ANN and MARS in prediction of property of steel strips. Applied Soft Computing Technologies, 2006, 34: 329−341
43. Shahani A R, Setayeshi S, Nodamaie S A, Asadi M A, Rezaie S. Prediction of influence parameters on the hot rolling process using finite element method and neural network. Journal of Materials Processing Technology, 2009, 209(4): 1920−1935
44. CSI. SAP2000 Integrated Software for Structural Analysis. Berkeley, CA: Computers & Structures Inc., 2011
45. Javanmardi R, Ahmadi-Nedushan B. Cost optimization of steel-concrete composite I- girder bridges with skew angle and longitudinal slope, using the Sm toolbox and the parallel pattern search algorithm. International Journal of Optimization in Civil Engineering, 2021, 11(3): 357−382
46. ShahBahrami J S. Parallel Processing and Programming by GPU (In Persian). Tehran: Nas, 2016
47. Mirjalili S. Genetic algorithm. Nature Computational Science, 2015, 28: 21−42
48. Shapiro, J. Genetic algorithms in machine learning. In Advanced Course on Artificial Intelligence, pp. 146-168. Springer, Berlin, Heidelberg, 1999
49. Holland J H. Genetic algorithms. Scientific American, 1992, 267(1): 66−72
50. Green W A. Vibrations of space structures. In: Hodnett, F. In: Proceedings of the Sixth European Conference on Mathematics in Industry. Verlag: European Consortium for Mathematics in Industry, 1991
51. Shimoda M, Liu Y, Wakasa M. Free-form optimization method for frame structures aiming at controlling time-dependent responses. Structural and Multidisciplinary Optimization, 2021, 63(1): 479−497
52. Makowski Z S. Book review: analysis, design and construction of steel space frames. International Journal of Space Structures, 2002, 17(2–3): 243–243
53. Kaveh A, Moradveisi M. Size and geometry optimization of double-layer grids using CBO and ECBO algorithms. Iranian Journal of Science and Technology. Transaction of Civil Engineering, 2017, 41(2): 101−112
54. Makowski Z S. Analysis, Design and Construction of Braced Barrel Vaults. Boca Raton: CRC Press, 2016
55. Kaveh A, Eskandari A, Eskandary A. Analysis of double-layer barrel vaults using different neural networks; a comparative study. International Journal of Optimization in Civil Engineering, 2021, 11(1): 113−141
56. Nowak A S, Collins K R. Reliability of Structures. 2nd ed. Boca Raton: CRC Press, 2012
57. CSI. Open Application Programming Interface (OAPI). Berkeley: Computers and Structures Inc., California, 2011
58. Hu Z, Du X. A random field approach to reliability analysis with random and interval variables. ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering, 2015, 1(4): 041005
59. Jiang C, Li W X, Han X, Liu L X, Le P H. Structural reliability analysis based on random distributions with interval parameters. Computers & Structures, 2011, 89(23−24): 2292−2302
60. Wang L, Wang X, Wang R, Chen X. Reliability-based design optimization under mixture of random, interval and convex uncertainties. Archive of Applied Mechanics, 2016, 86(7): 1341−1367