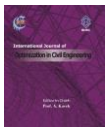
INTERNATIONAL JOURNAL OF OPTIMIZATION IN CIVIL ENGINEERING
Int. J. Optim. Civil Eng., 2017; 7(3)413-431



NEW META-HEURISTIC OPTIMIZATION ALGORITHM USING NEURONAL COMMUNICATION

S. Asil Gharebaghi[*, †] and M. Ardalan Asl
Department of Civil Engineering, K. N. Toosi University of Technology, Tehran, Iran

ABSTRACT

A new meta-heuristic method, based on Neuronal Communication (NC), is introduced in this article. The neuronal communication illustrates how data is exchanged between neurons in neural system. Actually, this pattern works efficiently in the nature. The present paper shows it is the same to find the global minimum. In addition, since few numbers of neurons participate in each step of the method, the cost of calculation is less than the other comparable meta-heuristic methods. Besides, gradient calculation and a continuous domain are not necessary for the process of the algorithm. In this article, some new weighting functions are introduced to improve the convergence of the algorithm. In the end, various benchmark functions and engineering problems are examined and the results are illustrated to show the capability, efficiency of the method. It is valuable to note that the average number of iterations for fifty independent runs of functions have been decreased by using Neuronal Communication algorithm in comparison to a majority of methods.

Keywords: meta-heuristic algorithm; global minimum; neuronal communication.

Received: 10 October 2016; Accepted: 6 February 2017

1. INTRODUCTION

For decades, extensive researches have been implemented to find the optimum solution of engineering problems. The gradient based classical methods are not capable of solving new problems, especially when the derivatives of a fitness function do not exist. Therefore, the meta-heuristic methods which use natural concepts have been introduced as the new efficient solutions. In these methods, Nature is assumed as a guideline. Similar to the radar, which has been invented based on the behaviour of a bat, new optimization methods are defined using such natural events. Dorigo et al. [1] used ant colony behaviour and Eberhart and Kennedy [2] utilized birds' immigration to find the best solution in the search domain

[*]Corresponding author: Department of Civil Engineering, K. N. Toosi University of Technology, Tehran, Iran
[†]E-mail address: asil@kntu.ac.ir (S. Asil Gharebaghi)

for mathematical problems.

The modeling of natural phenomenon in conjunction with stochastic laws is a common criterion of meta-heuristic methods [3]. These methods use a natural event as an idea to provide a new optimization algorithm. For example, Holland [4] and Goldberg [5] proposed Genetic Algorithm (GA), which is based on evolutionary biological process. Kennedy and Eberhart [6] introduced Particle Swarm Optimization (PSO) according to the birds' migration. As another example, Kirkpatrick et al. [7] introduced Simulated Annealing (SA) upon natural material modeling. Some other meta-heuristic methods have been proposed by Fogel et al. [8], De Jong [9], Koza [10]. Glover [11] proposed Tabu search algorithm and Rashedi [12] introduced the gravity search algorithm. Kaveh and Talatahari [13] put forward the charged system search.

This article is established based on the concept of Neuronal Communication (NA). The neuronal communication includes receiving excitation, data analysis and making a suitable response. This method models the behaviour of neurons to find the global minimum. This model has been proved to be efficient in nature, so it could be the same in engineering problems.

1.1 Definition of the neuronal communication

Neuronal communication is a set of neurons with physical contact whose input and output signal create a recognizable circuit. The connection of neurons is an electrochemical process. The interface between neurons is called dendrite which uses synapse (Fig. 1) to connect the neurons. Another connector between neurons is the axon (Fig. 2) which works as an output connection. If the intensity of input signal exceeds a determined value (threshold), the neuron will send an action potential, as an output signal, from the axon hillock to neighbors. In this way, a permanent circuit is established among all neurons. Hence, the characteristics of a neuron may affect its neighbors and consequently all of the neural system. In response to an excitation, all neurons participate and behave as an integrated system. In simple words, a neuron receives data by some connections. After analyzing, if the result is acceptable, the neuron will send the response to other neurons. Mostly the limited numbers of neurons are in contact with each neuron. This fact decreases the data transmission and helps to improve the performance of the system. Because of neuronal communication, the neurons affect each other simultaneously and the final response depends on the characteristics of all neurons.

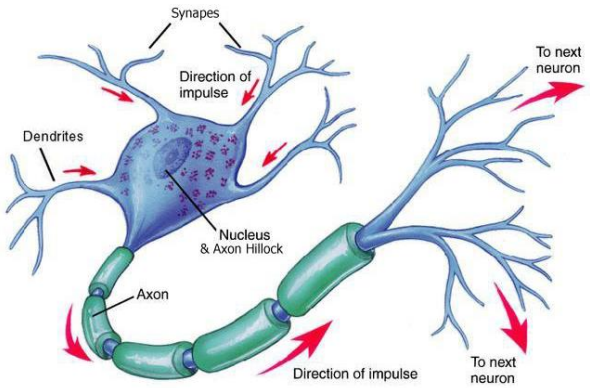


Figure 1. Components of a neuron

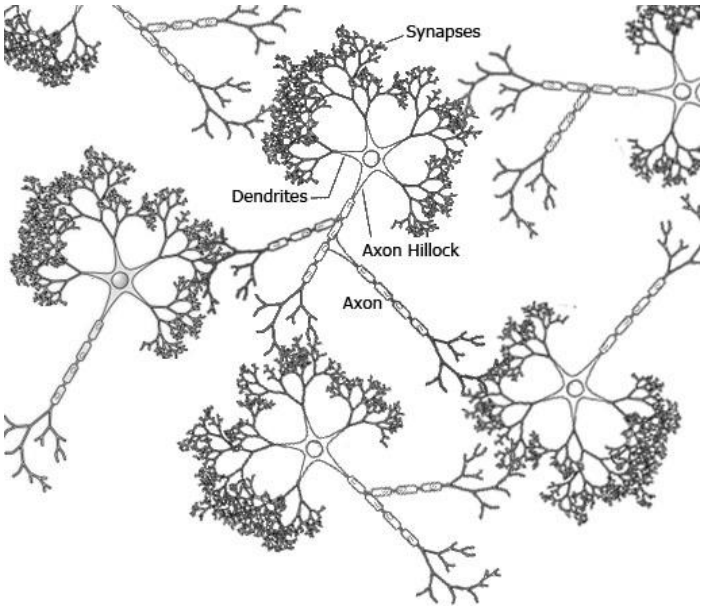


Figure 2. The pattern of neuronal connections

In a population-based optimization method, there are independent agents whose inputs including fitness function values, are specified. If some efficient connections are provided among agents, the convergence of the algorithm to the global minimum can be improved. As a result, each agent can be assumed as a neuron which has neuronal communication with its neighbors. As aforementioned, the neuron gets inputs from the neighbors. After analyzing, if the result is acceptable, the final result will be sent to the directly connected neurons (neighbors). The method is going to find the global minimum, so neurons should gradually move to find the new better location. Unlike the natural neurons, the neurons in this method do not have fixed location and they approach to the global minimum. Because of permanent movement, the neighbors must frequently change their location.

As mentioned, when the intensity of inputs reaches a specified level, the action potential (response) is sent to the neighbors. Consequently, the acceptance criterion of the new solution, in this method, can be the improvement of the fitness function value.

Similar to natural neurons, a small numbers of neighbors are connected to each neuron. This concept is considered in the process of the algorithm. In such a way that, the data transmission rate decreases and all agents are connected directly or indirectly. Each neuron may affect the convergence path of others although they may not be connected directly. In the next sections, the components of the algorithm are illustrated and then some numerical examples are examined to show the efficiency and performance of the algorithm.

2. NEURONAL COMMUNICATION ALGORITHM

At first stage of the algorithm, it is assumed that some neurons are randomly distributed in the search space and each one is connected to some neighbors (Fig. 3). Next, the fitness function value of all agents is determined. Actually, in each step of the iteration, if the

neuronal communication method could find the new suitable location for neurons, the algorithm would probably converge to the global minimum. Finding a new suitable location for the i^{th} neuron is the purpose of the method. Besides, in the algorithm, each neuron can inform neighbors about its location and its fitness function value. This information is the only guidance of neurons movement. The interaction between each neuron and the neuronal population can only be transmitted by the neighbors. This interaction is in accordance with the following procedure:

- The neighbors send their information, including the location and the fitness function value, to the i^{th} neuron.
- If the fitness function value of the j^{th} neighbor is less than its counterpart of the i^{th} neuron, the i^{th} neuron will tend to approach the j^{th} neighbor.
- Conversely, if the fitness function value of the j^{th} neighbor is greater than the i^{th} neuron's, the i^{th} neuron will tend to get away from the j^{th} neighbor.

In this way, the i^{th} neuron moves from a location with a higher fitness function value to a location with the lower one.

If the fitness function value of the j^{th} neighbor is considerably less than the other neighbors, the i^{th} neuron should tend towards it with the corresponding magnitude. Therefore, the method is capable of conducting the neuron to the better location using the weighted function in accordance with the fitness function value of its neighbors.

As a result, the j^{th} neighbor conducts i^{th} neuron to a weighted path. The path can be presented with a vector which connects the i^{th} neuron to the j^{th} neighbor (the red arrows in Fig. 3). Clearly, if the i^{th} neuron approaches the j^{th} neighbor, the arrow should be drawn from i to j and vice versa. Ultimately, the normalized resultant vector of weighted paths (the bold black arrow in Fig. 3) shows the final movement path of the i^{th} neuron.

In this method, the movement of a neuron in a new iteration depends on the information of the previous one. It means the data of the current iteration participates in determining the new location of the neuron. It decreases the sensitivity of the method to the variation of neighbors. Hence, the information of each iteration should be saved for the next iteration.

Because of constant movement of neurons, the neighbors may change frequently. Therefore, the new neighbors of a certain neuron should be selected in each iteration of the algorithm. The current version of the algorithm uses a weighted random function to select the new neighbors. In fact, the random selection approach decreases the probability of getting trapped in the local minima. In other words, if the i^{th} neuron is going to be trapped in local minima, selecting new neighbors and getting new information help to change the path and converge to the global minimum.

In Fig. 3, the neuronal communication of i^{th} neuron and its neighbors are shown. The bold arrow indicates the movement direction of i^{th} neuron in the new iteration. In this figure, f_i and f are the fitness function value of i^{th} neuron and its neighbors respectively.

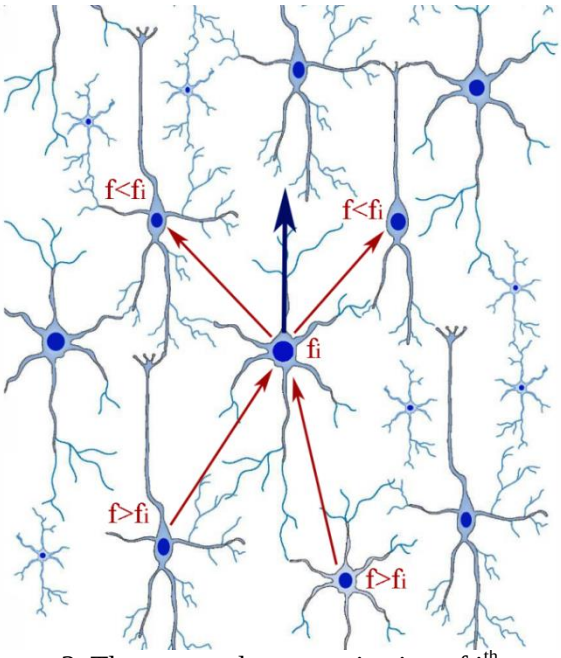


Figure 3. The neuronal communication of i^{th} neuron

In brief, the presented algorithm consists of four subroutines or functions, which create an adaptive structure to conduct the neurons to the global minimum efficiently. More precisely, in the first step of each iteration, the new neighbors of the i^{th} neuron are selected randomly. Then, the new location of the i^{th} neuron is obtained using fitness function value of the neighbors as aforementioned. Consequently, if the new location is better than the previous one, it will be considered as the new location of the i^{th} neuron.

The flowchart of Neuronal Communication algorithm is presented in Fig. 4.

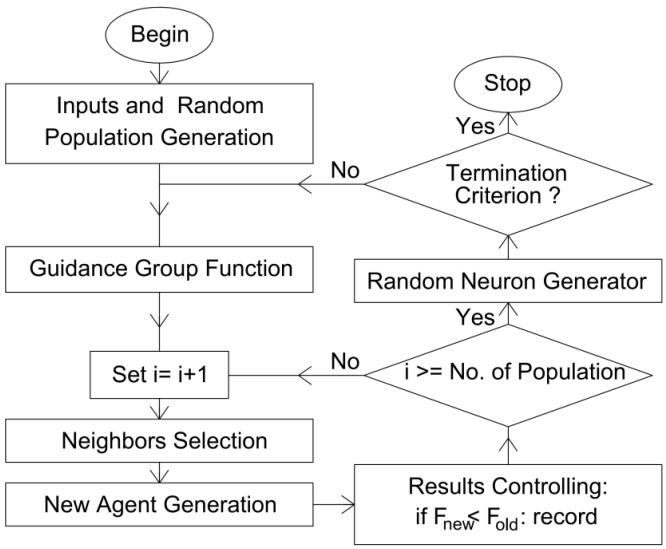


Figure 4. The flowchart of neuronal communication algorithm

In the next section, the components of the NC algorithm and its procedure are illustrated step by step.

2.1 Details of neuronal communication algorithm

The procedure of the NC algorithm, in accordance with the flowchart of Fig. 4, can be presented as follows.

Step 1: Inputs and Random Population Generation. In the first step, the parameters of the algorithm are defined. They include the followings:

Boundaries of search space ($a \leq X \leq b$), the maximum number of iterations (N_i), the initial and ultimate number of population (N_{ns}, N_{ne}), the initial and ultimate number of neighbors (N_{bs}, N_{be}), the number of guidance group members (m), the number of randomly generated neurons in each iteration (q) and constant coefficients (a_1, a_2, a_3, a_4).

The number of neuronal population and neighbors could be changed in each step of the algorithm. A dynamic number of population is required in the algorithm process. Actually, in some first iterations NC needs the maximum number of population; however, the number of population begins to decrease when the iteration number increases, or when the algorithm begins to converge to the final result. The dynamic size of population decreases the cost of calculation while the effects on the convergence of the algorithm are negligible. At its simplest form, a linear function is utilized to determine the number of population (N_n) and neighbors (N_b), based on the number of iterations. The suggested values of parameters are as Eqs. (1a)-(1c).

$$0.1N_{ns} \leq N_{ne} \leq 0.8N_{ns} \tag{1a}$$

$$5 \leq N_{bs} \leq 0.3N_{ns} \quad , \quad 3 \leq N_{be} \leq 0.1N_{ns} \tag{1b}$$

$$3 \leq m \leq 30 \quad , \quad 1 \leq q \leq 20 \tag{1c}$$

The first generation is distributed using Eq. (2).

$$\overrightarrow{X^1} = \vec{a} + (\vec{b} - \vec{a}) \times rand \tag{2}$$

where, $\vec{a}$ and $\vec{b}$ are vectors containing boundaries of variables, and *rand* is a uniform random number generator, which generate real numbers in the range of [0, 1]. $\overrightarrow{X^1}$ is the vector of neuron location in the first iteration of the algorithm. The superscript shows the iteration number.

After first population generation, its corresponding fitness function should be saved in F.

Step 2: Guidance Group Function. The aim of this step is to utilize the pattern of neurons with the best results. This set of neurons is called guidance group.

During the process of the algorithm, some neurons have no acceptable results and could not converge to a better solution. They increase the cost of computation, while have no positive effect on the convergence. A new function is introduced herein, which moves these neurons to some suitable locations. This process improves the performance of the algorithm.

New locations are determined using the mean and standard deviation of the location of the guidance group. Although in the first step of the algorithm neurons were randomly distributed using the uniform random distribution, in the second step the neurons distribute according to the normal distribution data, which is based on the guidance group data. The guidance group (i_m) consists of m neurons with the best results. The new locations are used in order to relocate the neurons with the worst results (i_{ma}).

In simple words, in each iteration, the algorithm estimates the location of the neurons with the best results to move the neurons with the worst results to some better locations. The applied functions are shown in Eq. (3).

$$\left.\begin{aligned} \mu &= Mean[\overrightarrow{X^n}(I_m)] \\ \sigma &= SD[\overrightarrow{X^n}(I_m)] \end{aligned}\right\} \rightarrow \overrightarrow{X^n}(i_{ma}) = a_1 \times (2 \times rand - 1) \times \sigma + \mu \tag{3}$$

where, $\overrightarrow{X^n}(I_m)$ shows the location of the guidance group. The *Mean* and *SD* are the mean and standard deviation functions respectively. i_{ma} indicates the set of neurons with the worst results. The results of the present study have shown that it is better to choose i_{ma} so that $i_{ma} < 0.05N_{ns}$. The coefficient a_1 controls the distribution range of the new neurons. In the present version of the algorithm, a_1 is in the range of [1, 5].

Step 3: Neighbors Selection. In the third step, the neighbors of the i[th] neuron are selected using a random selector with cumulative weighted function (G). The neighbors are selected from the all neuronal population (R) except the neuron with the best result (i_m), the neurons provided by guidance group (i_{ma}) and i[th] neuron. The neighbors' selection is done using Eqs. (4)-(5b). Besides, the weight of j[th] neuron is within [0, 1] as Eq. (4).

$$w_j = \frac{F_{\max} - F_j}{F_{\max} - F_{\min}} \quad , \quad \alpha_j = \frac{w_j}{\sum_{t=1}^{Nn} w_t} \quad for \;\; j,\; t \in R - \{i, i_m, i_{ma}\} \tag{4}$$

where, F_{max} and F_{min} are the maximum and minimum of the fitness function of the neurons respectively.

The cumulative weighted function G is an array. Each element of this array contains the summation of the weight of j[th] neuron and all its previous neurons in the global numbering of neurons. The array can be produced using Eqs. (5a) and (5b):

$$G_1 = \alpha_1 \tag{5a}$$

$$G_j = G_{j-1} + \alpha_j \qquad for \;\; j = 2,3,... \in R - \{i, i_m, i_{ma}\} \tag{5b}$$

For selecting each neighbor of the i[th] neuron, the minimum value of j should be determined in such a way that $rand \leq G_j$. Here, $j = 1,2,3,... \in R - \{i, i_m, i_{ma}\}$ and *rand* is a uniform random number generator in the range of [0, 1]. Indeed, for obtaining the neighbors of the i[th] neuron (I_b), this procedure should be repeated N_b times.

The random selection of the neighbors decreases the probability of being trapped in a local minimum. In fact, when the ith neuron is going to be trapped in local minima, changing the neighbors and getting new data help the neuron to escape from the situation.

Step 4: New Agent Generation. In the fourth step, a new location is proposed for the ith neuron using the neuronal communication method as aforementioned. If the fitness function value of the proposed location is less than the previous one, this location will be the new location of the ith neuron.

This step consists of two subroutines or functions. The exponential weighting function H increases the efficacy of the neurons with better fitness function value to enhance the convergence of the algorithm. This function guides the local search of the algorithm to find the global minimum.

Moreover, the vector $\overrightarrow{X_c}$ represents the steady movement of the algorithm procedure. This vector is provided using the location of neighbors in the previous iteration. It helps to increase the search capability of the method. Thus, this vector plays the role of the global search of the algorithm.

The exponential weighting function H is obtained by Eq. (6).

$$H_j = \frac{a_2^{W_j}}{\sum_{t=1}^{N_b} a_2^{W_t}} \qquad for \quad j,\ t \in I_b \tag{6}$$

where, a_2 is the base of the exponential function and is assumed to be in the range of [1, 5].

The attraction or repulsion caused by neighbors is applied along the line between each neighbor and the ith neuron. This direction can be represented by the unit vector $\overrightarrow{U}$ in accordance with Eq. (7).

$$\overrightarrow{U_j} = \frac{\overrightarrow{X_j^n} - \overrightarrow{X_i^n}}{\left\| \overrightarrow{X_j^n} - \overrightarrow{X_i^n} \right\|} \times Sign(F_i - F_j) \qquad for \quad j \in I_b \tag{7}$$

where, *Sign*(x) indicates the sign of x. $\overrightarrow{X_i^n}$ shows the location of the ith neuron in the nth iteration and $\left\| \overrightarrow{X_i^n} \right\|$ is the norm of the location vector $\overrightarrow{X_i^n}$. I_b shows the neighbors of the ith neuron. The vector $\overrightarrow{X_c}$ which represents the continuous movement in the algorithm is defined as Eq. (8).

$$\overrightarrow{X_c} = \sum_{j=1}^{N_b} H_j \,.\, \left\| \overrightarrow{X_i^{n-1}} - \overrightarrow{X_j^{n-1}} \right\| \,.\, \overrightarrow{U_j} \qquad , \qquad j \in I_b \tag{8}$$

Here, $\overrightarrow{X_i^{n-1}}$ is the location of the ith neuron in the n-1th iteration and n shows the iteration

number of the method. N_b indicates the number of the neighbors of the i[th] neuron.

The direction vector $\overrightarrow{T_1}$ and step length T_2 are defined according to Eqs. (9) and (10).

$$\overrightarrow{T_1} = C_1 \times \sum_{j=1}^{N_b} H_j . \overrightarrow{U_j} + C_2 \times \frac{\overrightarrow{X_c}}{\left\| \overrightarrow{X_c} \right\|} \qquad for \quad j \in I_b \tag{9}$$

$$T_2 = C_1 \times \sum_{j=1}^{N_b} H_j . \left\| \overrightarrow{X_i^n} - \overrightarrow{X_j^n} \right\| + C_2 \times \left\| \overrightarrow{X_c} \right\| \qquad for \quad j \in I_b \tag{10}$$

The coefficients C_1 and C_2 are suggested as Eq. (11a) and (11b).

$$C_1 = a_2^{\frac{a_3 \times n}{N_i}} \tag{11a}$$

$$C_2 = a_2^{\frac{-a_4 \times n}{N_i}} \tag{11b}$$

where, N_i is the maximum number of iterations. a_3 and a_4 are some constant coefficients, which are assumed to be in the range of [1, 5] and [1, 10] respectively.

In these equations, $\overrightarrow{T_1}$ represents the direction vector and T_2 is the step length of i[th] neuron movement. It is obtained using the neighbors of the i[th] neuron in the n[th] and n-1[th] iterations. The coefficients C_1 and C_2 help to convert the global search of the method to the local search. Hence, these values have a profound impact on the convergence of the algorithm. More accurately, C_1 is an incremental factor that amplifies the local search of the algorithm. On the other hand, C_2 is a decreasing factor and inspires the global search. In the algorithm procedure, the global search is gradually converted to the local search. Consequently, a new exponential function is suggested in order to improve the process.

Finally, the new suggested location of the i[th] neuron is obtained as Eq. (12).

$$\overrightarrow{X_i^{n+1}} = \overrightarrow{X_i^n} + rand \times \frac{\overrightarrow{T_1}}{\left\| \overrightarrow{T_1} \right\|} \times T_2 \tag{12}$$

where, $\overrightarrow{X_i^{n+1}}$ and $\overrightarrow{X_i^n}$ are the suggested and present location of the neuron, respectively.

Notably, the location $\overrightarrow{X_i^{n+1}}$ will be the new location of the i[th] neuron only if the fitness function value of $\overrightarrow{X_i^{n+1}}$ is less than the value of the present location $\overrightarrow{X_i^n}$. In each iteration, the process of neighbors selection and finding the new location of each neuron should be repeated.

Additionally, the process of the algorithm starts with the global search and tends to the local search. The global search needs the larger number of agents than the local one. Hence, the number of population is dynamic and may change during the algorithm. Therefore, the

cost of calculation decreases without noticeable reduction in the convergence capability.

Step 5: Random Neuron Generator. To prevent a neuron from being trapped in local minima, a stochastic function is utilized to generate new random neurons. This is what exactly happens in the fifth step of the NC algorithm. This function replaces the neurons with the worst fitness function (I_x) with the new random neurons according to Eq. (13).

$$\overrightarrow{X^n}(I_x) = \vec{a} + (\vec{b} - \vec{a}) \times rand \tag{13}$$

Here, $\vec{a}$ and $\vec{b}$ are vectors containing boundaries of variables in the search space.

In this way, the method utilizes a set of functions based on the neuronal communication to converge into the global minimum. Finally, the algorithm continues until one of the following termination criteria is true:

- The current iteration number of the algorithm exceeds the maximum allowed number of iterations.
- The sequence of the best fitness function values converge so that the difference between the best values of two immediate iterations falls into a predefined tolerance value.

3. ENGINEERING BENCHMARK PROBLEMS

In this section, the efficiency of the neuronal communication algorithm is evaluated and the results of some different methods are compared with the present method.

3.1 Unconstrained problems

For unconstrained optimization problems, the algorithm is compared with different versions of GA [14] and the methods CSS [13] and RO [15] in various mathematical problems [13]. The benchmark problems are introduced in Table 1 and the compared results are shown in Table 2.

Table 1: The Specifications of benchmark problems [13]

Name	F(x)	Domain	F_{min}
Aluffi-Pentiny (AP)	$\frac{1}{4}x_1^4 - \frac{1}{2}x_1^2 + \frac{1}{10}x_1 + \frac{1}{2}x_2^2$	$[-10,10]^2$	-0.352386
Bohachevsky-1 (Bf1)	$x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$	$[-100,100]^2$	0.000000
Bohachevsky-2 (Bf2)	$x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$	$[-50,50]^2$	0.000000
Becker and Lago	$(\lvert x_1\rvert - 5)^2 + (\lvert x_2\rvert - 5)^2$	$[-10,10]^2$	0.000000
Branin	$(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$	$x_1 \in [-5,10]$ $x_2 \in [0,15]$	0.397887

Camel	$4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5,5]^2$	-1.031600
Cosine Mixture (CM)	$\sum_{i=1}^{n} x_i^2 - \frac{1}{10}\sum_{i=1}^{n}\cos(5\pi x_i)$	$[-1,1]^4$	-0.400000
DeJoung	$x_1^2 + x_2^2 + x_3^2$	$[-5.12,5.12]^3$	0.000000
Exponential (EXP2, EXP4, EXP8)	$-\exp(-0.5\sum_{i=1}^{n} x_i^2)$	[-1,1] n= 2, 4, 8	-1.000000
Goldstein and price	$[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times$ $[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	$[-2,2]^2$	3.000000
Griewank-2	$1+\frac{1}{200}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2}\frac{\cos(x_i)}{\sqrt{i}}$	$[-100,100]^2$	0.000000
Hartman-3	$-\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$ $a=\begin{bmatrix}3&10&30\\0.1&10&35\\3&10&30\\0.1&10&35\end{bmatrix}; c=\begin{bmatrix}1\\1.2\\3\\33.2\end{bmatrix}; p=\begin{bmatrix}0.3689&0.117&0.2673\\0.4699&0.4387&0.747\\0.1091&0.8732&0.5547\\0.03815&0.5743&0.8828\end{bmatrix}$	$[0,1]^3$	-3.862782
Hartman-6	$-\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$ $a=\begin{bmatrix}10&3&17&3.5&1.7&8\\0.05&10&17&0.1&8&14\\3&3.5&1.7&10&17&8\\17&8&0.05&10&0.1&14\end{bmatrix}; c=\begin{bmatrix}1\\1.2\\3\\3.2\end{bmatrix}$ $p=\begin{bmatrix}0.1312&0.1696&0.5569&0.0124&0.8283&0.5886\\0.2329&0.4135&0.8307&0.3736&0.1004&0.9991\\0.2348&0.1451&0.3522&0.2883&0.3047&0.6650\\0.4047&0.8828&0.8732&0.5743&0.1091&0.0381\end{bmatrix}$	$[0,1]^6$	-3.322368
Rastrigin	$x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2)$	$[-1,1]^2$	-2.000000
Rosenbrock	$\sum_{i=1}^{n-1}\left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$	$[-30,30]^2$	0.000000

It is valuable to note that Tsoulos [14] studied some different versions of Genetic Algorithm which are introduced as GEN, GEN-S, GEN-S-M and GEN-S-M-LS. The method CSS [13] utilized the charged system principles to find global minimum. Similarly, the method RO [15] introduced a new meta-heuristic algorithm using the Snell's light refraction law.

The perspective view and related contour lines of some benchmark functions are shown in Fig. 5.

Table 2: Performance comparison for the benchmark problems

Function	GEN	GEN-S	GEN-S-M	GEN-S-M-LS	CSS	RO	NC
AP	1360 (0.99)	1360	1277	1253	804	331	**304**
Bf1	3992	3356	1640	1615	1187	677	**394**
Bf2	20234	3373	1676	1636	742	582	**353**
BL	19596	2412	2439	1436	423	**303**	314
Branin	1442	1418	1404	1257	852	463	**355**
Camel	1358	1358	1336	1300	575	332	**250**
CM	2105	2105	1743	1539	1563	802	**678**
Dejoung	9900	3040	1462	1281	630	452	**236**
Exp2	938	936	817	807	132	136	**75**
Exp4	3237	3237	2054	1496	867	382	**252**
Exp8	3237	3237	2054	1496	1426	1287	**820**
Goldstein and Price	1478	1478	1408	1325	682	451	**272**
Griewank	18838 (0.91)	3111 (0.91)	1764	1652 (0.99)	1551	1091 (0.98)	**1078**
Hartman3	1350	1350	1332	1274	860	-	**472**
Hartman6	2562 (0.54)	2562 (0.54)	2530 (0.67)	1865 (0.68)	1783	-	**1459**
Rastrigin	1533 (0.97)	1523 (0.97)	1392	1381	1402	**1013** (0.98)	1289
Rosenbrock	9380	3739	1675	1462	1452	-	**1132**

The values in Table 2 indicate the average numbers of function evaluation in 50 independent runs. The number in parenthesis represents the ratio of successful runs in which the method has found the global minimum. The predefined accuracy of the method is taken as $\varepsilon = |f_{\min} - f_{final}| = 10^{-4}$. The absence of the parentheses shows that the algorithm has been successful in all independent runs. It can be seen that only CSS and NC algorithm have been unconditionally successful in all fifty runs of all benchmark problems.

3.1.1 Comparing results

The method GEN-S-M-LS has better results than the other methods, which are based on GA. This method utilizes some auxiliary mechanisms such as an improved stopping law, the new mutation mechanism and an iterative approach in the local search. On the other hand, the methods CSS and RO improve the results more effectively than GA based methods. Totally, the Neuronal Communication algorithm (NC) converged to the global minimum faster than RO, CSS and GA based methods.

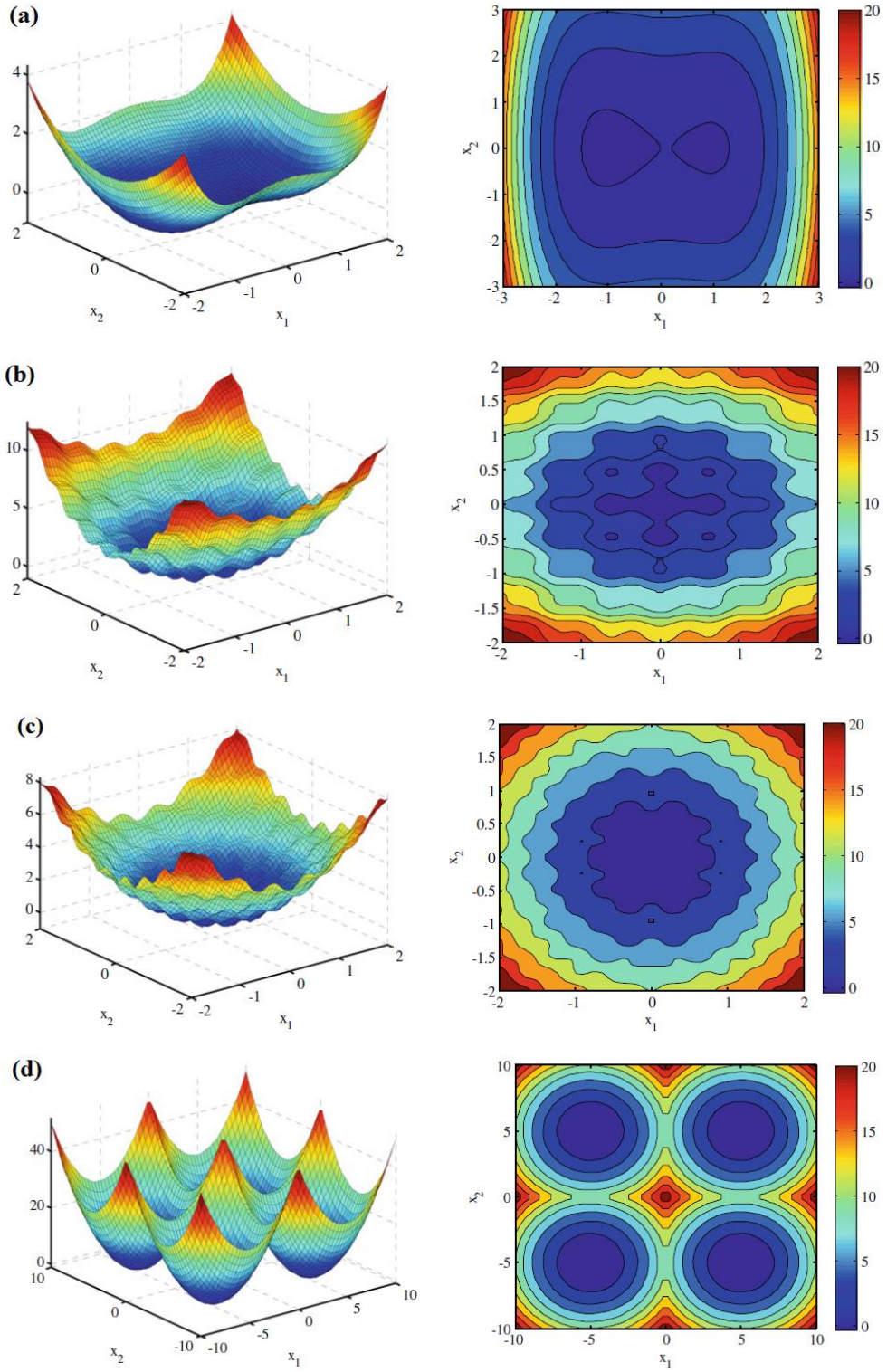


Figure 5. A perspective view and related contour lines for some of function in two dimensional forms. (a) Aluffi-Pentiny. (b) Bohachevsky-1. (c) Bohachevsky-2. (d) Becker and Lago

3.2 Constrained problems

3.2.1 A pressure vessel design problem

In this section, the optimal design of the cylindrical vessel, displayed in Fig. 6, is considered as a constrained optimization problem. The objective is to minimize the total cost including the cost of material, forming and welding [16]. This function is shown in Eq. (14).

$$f_{\text{cost}}(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \tag{14}$$

where, x_1 is the thickness of the shell (T_s), x_2 is the thickness of the head (T_h), x_3 the inner radius (R) and x_4 is the length of cylindrical section of the vessel (L). T_s and T_h are integer multiples of 0.0625 inch and R and L are real numbers.

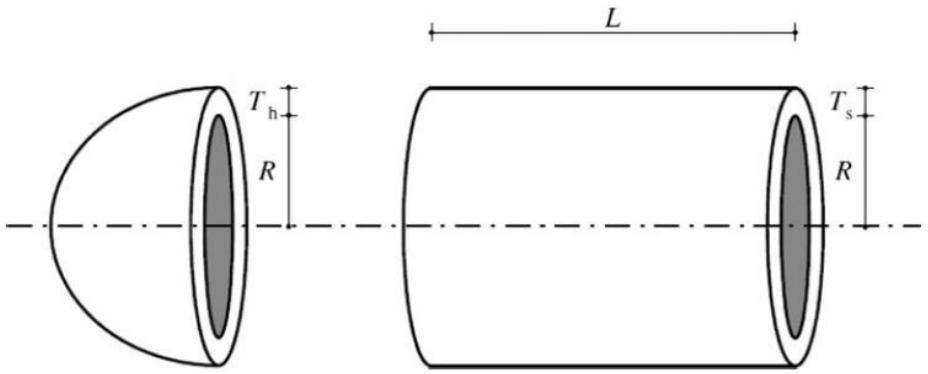


Figure 6. A Schematic shape of the pressure vessel

The constraints and the design space can be stated as Eq. (15).

$$\begin{aligned} g_1(X) &= -x_1 + 0.0193x_3 \le 0 \\ g_2(X) &= -x_2 + 0.00954x_3 \le 0 \\ g_3(X) &= -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0 \\ g_4(X) &= x_4 - 240 \le 0 \end{aligned} \qquad \begin{aligned} 0 &\le x_1 \le 99 \\ 0 &\le x_2 \le 99 \\ 10 &\le x_3 \le 200 \\ 10 &\le x_4 \le 200 \end{aligned} \tag{15}$$

The constraints are applied to the algorithm using the penalty function method. The best results of various developed methods and corresponding statistical simulation results are shown in Table 3 and Table 4, respectively. The results are obtained from ten independent runs of the methods. Although some methods such as Montes and Coello [22] and Kaveh and Talatahari [23] have better results than the others, the Neuronal Communication method (NC) provides the best results. The standard deviation value of the NC algorithm is not the best, for instance in comparison to Coello [19]. However, the mean value of the NC algorithm has an error of about 0.6%, according to the minimum value of cost function, while [19] estimates the cost function with 3.9% of error. Besides, the values of mean and standard deviation of the NC algorithm are comparable to the other methods.

Table 3: Optimum results for the pressure vessel

Methods	Optimal Design Variable				f_{cost}
	x_1 (T_s)	x_2 (T_h)	x_3 (R)	x_4 (L)	
Sandgren [16]	1.125000	0.625000	47.700000	117.701000	8129.1036
Kannan and Karamer [17]	1.125000	0.625000	58.291000	43.690000	7198.0428
Deb and Gene [18]	0.937500	0.500000	48.329000	112.679000	6410.3811
Coello [19]	0.812500	0.437500	40.323900	200.000000	6288.7445
Coello and Montes [20]	0.812500	0.437500	42.097398	176.654050	6059.9463
He and Wang [21]	0.812500	0.437500	42.091266	176.746500	6061.0777
Montes and Coello [22]	0.812500	0.437500	42.098087	176.640518	6059.7456
Kaveh and Talatahari [23]	0.812500	0.437500	42.098353	176.637751	6059.7258
NC (the current study)	0.812500	0.437500	42.098446	176.636596	**6059.1313**

Table 4: Statistical results of different methods for the pressure vessel

Methods	Best	Mean	Worst	Standard Deviation
Sandgren [16]	8129.1036	N/A	N/A	N/A
Kannan and Karamer [17]	7198.0428	N/A	N/A	N/A
Deb and Gene [18]	6410.3811	N/A	N/A	N/A
Coello [19]	6288.7445	6293.8432	6308.1497	**7.4133**
Coello and Montes [20]	6059.9463	6177.2533	6469.3220	130.9297
He and Wang [21]	6061.0777	6147.1332	6363.8041	86.4545
Montes and Coello [22]	6059.7456	6850.0049	7332.8798	426.0000
Kaveh and Talatahari [23]	6059.7258	6081.7812	6150.1289	67.2418
NC (the current study)	6059.1313	6093.2716	6203.7628	40.9574

N/A: Not available.

3.2.2 A 10-bar planar truss

The optimal design of the 10-bar truss, shown in Fig. 7, is considered as another example of constrained optimization problem. More accurately, the weight of the truss is considered as the objective function. In this problem, the stress limit of the members is σ_0= ±172.37 MPa (25 ksi). The nodal displacements in the vertical direction are limited to ±5.08 cm (2.0 in) and the density of the material is ρ= 2767.99 kg/m^3 (0.1 lb/in^3). The minimum cross section and the modulus of elasticity are A_0= 0.6451 cm^2 (0.1 in^2) and E= 6.89×10^4 MPa (10^4 ksi), respectively.

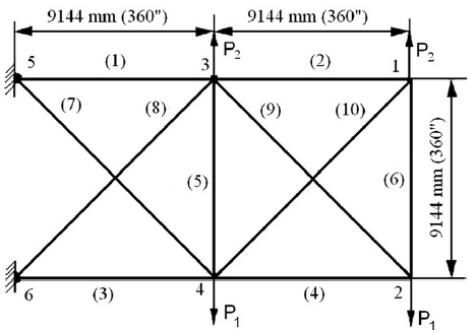


Figure 7. A 10-bar planar truss [24]

Two different load cases are considered herein. In the first case, P_1= 444.82 kN (100 kips) and P_2= 0. Additionally, in the second case, P_1= 667.233 kN (150 kips) and P_2= 222.411 kN (50 kips). In Tables 5 and 6, the results of the suggested method are compared with various methods and the improvement of the results is shown. The results are obtained from twenty independent runs of the methods. Although, some methods such as Lamberti and Pappalettere [27] and Sedaghati [28], in case-1, and Rizzi [32] and John et al. [33], in case-2, approached the minimum weight of the truss, the present study results the best value among the other methods.

Table 5: Comparison of optimum designs of 10-bar truss (case-1)

Member Number	Schmit and Farshi [24]	Schmit and Miura [25]	Venkayya [26]	Lamberti and Pappalettere [27]	Sedaghati [28]	Kaveh and Rahami [29]	Li et al. [30]	Farshi and Ziazi [31]	NC (the present study)
1	33.430	30.670	30.420		30.521	30.667	30.569	30.520	30.528
2	0.100	0.100	0.128		0.100	0.100	0.100	0.100	0.100
3	24.260	23.760	23.410		23.199	22.872	22.974	23.204	23.205
4	14.260	14.590	14.910		15.222	15.344	15.148	15.223	15.218
5	0.100	0.100	0.101	*	0.100	0.100	0.100	0.100	0.100
6	0.100	0.100	0.101		0.551	0.463	0.547	0.551	0.551
7	8.388	8.578	8.696		7.457	7.479	7.493	7.466	7.457
8	20.74	21.070	21.080		21.036	20.965	21.159	21.034	21.036
9	19.690	20.960	21.080		21.528	21.702	21.556	21.529	21.522
10	0.100	0.100	0.186		0.100	0.100	0.100	0.100	0.100
Weight (lb)	5089.00	5076.85	5084.90	5060.88	5060.85	5061.90	5061.03	5061.40	5060.85

* Not mentioned.

Table 6: Comparison of optimum designs of 10-bar truss (case-2)

Member Number	Schmit and Farshi [24]	Schmit and Miura [25]	Venkayya [26]	Rizzi [32]	John et al. [33]	Li et al. [30]	Farshi and Ziazi [31]	NC (the present study)
1	24.290	23.550	25.190	23.530	23.590	23.743	23.527	23.530
2	0.100	0.100	0.363	0.100	0.1000	0.101	0.100	0.100
3	23.350	25.290	25.420	25.290	25.250	25.287	25.294	25.290
4	13.660	14.360	14.330	14.370	14.370	14.413	14.376	14.368
5	0.100	0.100	0.417	0.100	0.100	0.100	0.100	0.100
6	1.969	1.970	3.144	1.970	1.970	1.969	1.969	1.969
7	12.670	12.390	12.080	12.390	12.390	12.362	12.404	12.398
8	12.540	12.810	14.610	12.830	12.800	12.694	12.824	12.852
9	21.970	20.340	20.260	20.330	20.370	20.323	20.330	20.296
10	0.100	0.100	0.513	0.100	0.100	0.103	0.100	0.100
Weight (lb)	4691.84	4676.96	4895.60	4676.92	4676.93	4677.70	4677.80	4676.89

4. CONCLUSIONS

A new meta-heuristic algorithm has been introduced in this article based on the Neuronal Communication. The Neuronal Communication represents the data transmission pattern between neurons in dealing with an external excitation to provide the suitable response. In this method, the neurons are considered as agents and the pattern of data exchange between neurons is utilized to find the global minimum. In this pattern, each neuron is connected to its neighbors directly and to the others indirectly. In fact, not only the number of neighbors for each neuron is not equal to the total number of population, but the number of neighbors for each neuron changes dynamically in each iteration. The results have showed that the cost of calculation in each iteration is less than the other comparable methods. Additionally, a dynamic selection of neighbors reduces the probability of being trapped in local minima. Also, some new weighting functions have been defined, which improve the capability and performance of the algorithm. The weights are defined using the fitness function value of the neurons. This algorithm is examined for various constrained and unconstrained benchmark functions and the advantages of the presented method are compared with the other meta-heuristic counterparts. In constrained benchmark problems, it has been shown that NC has been unconditionally successful for fifty independent runs, while some other methods failed in specific benchmark problems. Moreover, in constrained problems, it has been showed that the NC algorithm not only is capable of finding the global minimum, but its standard deviation, in a statistical analysis, is comparable to some other methods in the literature.

REFERENCES

1. Dorigo M, Maniezzo V, Colorni A. The ant system: optimization by a colony of cooperating agents, *IEEE Transact on Syst, Man, Cybernet, Part B*, **41**(1996) 26-9.
2. Eberhart RC, Kennedy J. A new optimizer using particle swarm theory, *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, 1995.
3. Lee KS, Geem ZW. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, *Compu Meth Appl Mech Eng* 2005; **194**: 3902-33.
4. Holland J. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, 1975.
5. Goldberg D. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
6. Kennedy J, Eberhart R. Particle swarm optimization, *Proceedings of the IEEE International Conference on Neural Networks*, Perth, Australia, 1995, pp. 1942-8.
7. Kirkpatrick S, Gelatt C, Vecchi M. Optimization by simulated annealing, *Sci* 1983; **220**: 671–80.
8. Fogel LJ, Owens AJ, Walsh MJ. *Artificial Intelligence through Simulated Evolution*, Chichester, Wiley, 1996.

9. De Jong K. Analysis of the behavior of a class of genetic adaptive systems, PhD thesis, University of Michigan, Ann Arbor, MI, 1975.
10. Koza JR. Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems, Report No. STAN-CS- 90-1314, Stanford University, Stanford, CA, 1990.
11. Glover F. Heuristic for integer programming using surrogate constraints, *Decis Sci* 1971, **8**: 156-66.
12. Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA: a gravitational search algorithm, *Inform Sci* 2009; **48**: 179-232.
13. Kaveh A, Talatahari S. A novel heuristic optimization method: charged system search, *Acta Mech* 2010, **289**: 213-67.
14. Tsoulos IG. Modifications of real code genetic algorithm for global optimization, *Appl Mathemat Comput* 2008; **203**: 598–607.
15. Kaveh A, Khayatazad M. A new meta-heuristic method: Ray Optimization, *Comput Struct* 2012; **112-113**: 283-94.
16. Sandgren E. Nonlinear integer and discrete programming in mechanical design, *Proceedings of the ASME Design Technology Conference*, Kissimine, FL, 1988, pp. 95-105.
17. Kannan BK, Kramer SN. An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design, *J Mech Des: ASME DC* 1994; **116**: 318–20.
18. Deb K, Gene AS. *A Robust Optimal Design Technique for Mechanical Component Design*, *Evolutionary Algorithms in Engineering Applications*, In Dasgupta D, Michalewicz Z, Eds, Springer, Berlin, 1997, pp. 497–514.
19. Coello CAC. Use of a self-adaptive penalty approach for engineering optimization problems, *Comput Indust* 2000; **41**: 113–27.
20. Coello CAC, Montes EM. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, *Adv Eng Inform* 2002, **16**: 193-203.
21. He Q, Wang L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems, *Eng Applicat Artificial Intell* 2007, **20**: 89–99.
22. Montes EM, Coello CAC. An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, *Int J General Syst*, **37**(4) 443–73.
23. Kaveh A, Talatahari S. An improved ant colony optimization for constrained engineering design problems, *Eng Comput* 2010, **27**(1): 155-82.
24. Schmit LA, Farshi B. Some approximation concepts for structural synthesis, *AIAA J* 1974; **12**: 692–9.
25. Schmit LA, Miura H. Approximation concepts for efficient structural synthesis, NASA-CR-2552, NASA, Washington, DC, 1976.
26. Venkayya VB. Design of optimum structures, *Comput Struct* 1971, **1**: 265-309.
27. Lamberti L, Pappalettere C. Move limits definition in structural optimization with sequential linear programming (Parts I & II), *Comput Struct* 2003; **81**: 197-238.
28. Sedaghati R. Benchmark case studies in structural design optimization using the force method, *Int J Solids Struct* 2005, **42**: 5848-71.

29. Kaveh A, Rahami H. Analysis, design and optimization of structures using force method and genetic algorithm, *Int J Numer Meth Eng* 2006, **65**: 1570–84.
30. Li LJ, Huang ZB, Liu F, Wu QH. A heuristic particle swarm optimizer for optimization, *Comput Struct* 2007, **85:** 340–9.
31. Farshi B, Ziazi AA. Sizing optimization of truss structures by method of centers and force formulation, *Int J Solids Struct* 2010, **47**: 2508-24.
32. Rizzi P. Optimization of multi-constrained structures based on optimality criteria, *In AIAA/ASME/SAE 17th Structures, Structural Dynamics, and Materials Conference*, King of Prussia, PA, 1976.
33. John KV, Ramakrishnan CV, Sharma KG. Minimum weight design of trusses using improved move limit method of sequential linear programming, *Comput Struct*, 1987; **27**: 583-91.