World Applied Sciences Journal 12 (1): 94-105, 2011
ISSN 1818-4952

Imperialist Competitive Ant Colony Algorithm for Truss Structures

[1]M.H. Sabour, [2]H. Eskandar and [2]P. Salehi

[1]University of Tehran, Iran
[2]M.S. in Mechanical Engineering, University of Semnan, Iran

Abstract: In this paper, an imperialist competitive algorithm (ICA) and ant colony optimization (ACO) are combined to reach to an efficient algorithm, called imperialist competitive ant colony optimization (ICACO). The ICACO is tested on several truss structures with discrete variables and is compared with the ICA method and other optimization methods such as heuristic particle swarm optimizer (HPSO). The results show that the ICACO is able to accelerate the convergence rate effectively and has the fastest convergence rate among these methods. The research shows the proposed ICACO can be effectively used to solve optimization problems for steel structures with discrete variables.

Key words: Optimization · Truss structures · Constraints · Imperialistic competition · Ant colony optimization · Discrete variables

INTRODUCTION

In recent decades, different optimizing algorithms for truss optimization are widely used and this makes the truss structure optimizers to be attractive for researchers in the optimization field. There are three main categories in structural optimization:

- Sizing Optimization (the cross-sectional areas of the members are considered as design variables [1, 2]).
- Shape Optimization (The nodal coordinates are considered as the design variables [2]).
- Topology Optimization (The location of links in which connect the nodes to each other, are considered as design variables [3]).

In optimizing a problem, two or three types may be considered at the same time.

Recently, new methods such as heuristic particle swarm optimization (HPSO) [4], genetic algorithm [5], simulated annealing (SA) [6], particle swarm optimization (PSO) [7] and other stochastic searching methods are used in optimizing the trusses.

This paper presents an imperialist competitive ant colony optimization (ICACO) algorithm, which is based on the standard imperialist competitive algorithm (ICA) that is one of the newest algorithms in optimization field [8-9] and the Ant Colony Optimization (ACO) scheme. The ICA method is inspired from a social-political phenomenon and has two great characteristics; a) high ability of this algorithm to search the optimum point even when facing with nonlinear optimization problems and b) fast convergence speed [9]. In this paper, the ICACO method is applied to the structural optimization problems. The ICACO algorithm has all the advantages that belong to the ICA algorithm. Furthermore, it has faster convergence rate than the ICA and other methods.

In present paper, in Section 2, the ICA and ACO algorithms are described. In Section 3, the new method is presented. The formula for discrete optimizing problems is driven in Section 4. Various examples are studied in Section 5 and the advantages of the ICACO are discussed. Conclusions are derived in Section 6.

Introduction to ICA and ACO Algorithm

Imperialist Competitive Algorithm: Imperialist competitive algorithm is inspired from the social-political process of imperialism and imperialistic competition. This algorithm (like many optimization algorithms) starts with an initial population. Each individual of the population is called a 'country'. Some of the best countries with the minimum cost are considered as the imperialist states and the rest will be the colonies of those imperialist states. All the colonies are distributed among the imperialist countries based on their power.

Corresponding Author: Pouya Salehi, Address: Unit 2, No 33, East 32 St (Shahid Gheysari St), South Allameh Tabatabayi Ave, Saadat Abad, Tehran, Iran.
Tel: +98- 912- 383 67 88, Email: pouya.salehi@gmail.com.

The ICA begins by defining a country or an array of variable values that are going to be optimized. When solving a N_{var} dimensional optimization problem, we assume that a country is a $1 \times N_{var}$ array (Equation (1)).

$$\text{Country} = [p_1, p_2, p_3, \ldots, p_{Nvar}] \quad (1)$$

A set of p_is are considered as the variables that should be optimized. By evaluating the cost function, f, for variables (p_1, p_2, p_3,…,p_{Nvar}), the cost of a country will be found (Equation (2)):

$$Cost_i = f(conuntry_i) = f(p_1, p_2, p_3, ..., p_{N_{var}}) \quad (2)$$

To define the algorithm, first of all, initial countries of size $N_{Country}$ are produced. Then, some of the best countries (with the size of N_{imp}) in the population are selected to be the imperialist states. Therefore the rest with the size N_{col} will form the colonies that belong to imperialists. Then, the colonies are divided among imperialists according to their power [8]. In such a way that the initial number of each empire's colonies has to be proportional to its power. So, the initial number of colonies of the nth empire will be [9]:

$$N.C_n = round\{\left|\frac{Cost_n}{\sum_{i=1}^{N_{imp}} Cost_i}\right| \times N_{col}\} \quad , \quad n = 1,2,...,N_{imp} \quad (3)$$

In the above equation, $N.C._n$ is the initial number of colonies of the nth empire and N_{col} is the total number of initial colonies. To divide the colonies, $N.C._n$ of the colonies are randomly chosen and given to the nth imperialist. These colonies along with the nth imperialist form the nth empire [9].

After dividing all colonies among imperialists and creating the initial empires, these colonies start moving toward their relevant imperialist country. This movement is a simple model of assimilation policy. This policy is shown in Figure (1). In this movement θ and X are random numbers with uniform distribution and d is the distance between the imperialist and the colony [8].

$$X \sim U(0, \beta \times d), \beta > 1 \quad (4)$$

$$\theta \sim U(-\gamma, \gamma) \quad (5)$$

In the above equation, β and γ are arbitrary numbers that modify the area that colonies randomly search around the imperialist. During any movement, if a colony reaches a better point than an imperialist, they will be replaced by each other.

Also, the total power of an empire is defined by the sum of the cost of the imperialist and some percentage of the mean cost of its colonies (Equation (6)) [9].

$$T.C._n = \text{Cost}(\text{imperialist}_n) + \xi\{\text{mean}(\text{Cost}(\text{colonies of empire}_n))\} \quad (6)$$

In the above equation, $T.C._n$ is the total power of the nth empire and ξ is a positive small number. For estimating the total power of an empire, a small amount should be chosen for ξ to make the cost of an imperialist more important than the cost of colonies.

After computing the total power of empires, usually the weakest colony (or colonies) of the weakest empire is chosen by other empires and the competition is started on possessing this colony. Each imperialist participating in this competition, according to its power, has a probable chance of possessing the cited colony.

To start the competition, at first, the weakest empire is chosen and then the possession probability of each empire is estimated. The possession probability P_p is related to the total power of the empire. In order to evaluate the normalized total cost of an empire, the following equation is used [9]:

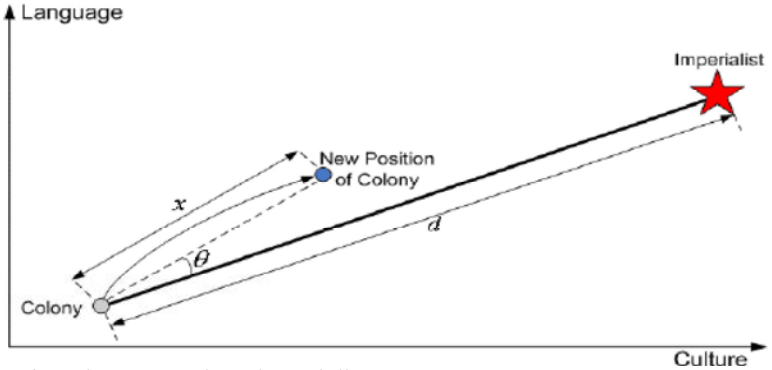

Fig. 1: The movement of a colony towards an imperialist.

$$N.T.C_n = \max_i \{T.C_i\} - T.C_n \quad (7)$$

In the above equation, $T.C_n$ is the total power of the n^{th} empire and $N.T.C_n$ is the normalized total power of n^{th} empire. When the normalized total power is obtained, the following equation is used to estimate the possession probability of each empire:

$$P_{P_n} = \left| \frac{N.T.C_n}{\sum_{i=1}^{N_{imp}} N.T.C_i} \right| \quad ,n = 1,2,...,N_{imp} \quad (8)$$

Note that in this way the powerful empires have more chance in possessing the weakest colony of the weakest empire.

In order to divide the given colonies among the empires, vector P is formed as follows:

$$P = [P_{P_1}, P_{P_2}, P_{P_3}, ..., P_{P_{N_{imp}}}] \quad (9)$$

After that, the vector R should be defined with the same size of vector P. The elements of vector R are random numbers between 0 and 1.

$$R = [r_1, r_2, r_3, ..., r_{N_{imp}}] \quad , r_1, r_2, r_3, ..., r_{N_{imp}} \sim U(0,1) \quad (10)$$

Then, vector D is constructed by subtracting R from P.

$$D = P - R = [D_1, D_2, D_3, ..., D_{N_{imp}}] = [P_{P_1} - r_1, P_{P2} - r_2, P_{P3} - r_3, ..., P_{P_{N_{imp}}} - r_{N_{imp}}] \quad (11)$$

When an empire achieves the maximum related index in D, it can take control of the given colony.

During the imperialistic competition, the weak empires will slowly lose their power and getting weak by the time. At the end of process, just one empire will remain that governs the whole colonies [9].

In most of the optimization methods, including the method which has been explained in this work, the criterion of stopping the algorithm can be given maximum iteration number, or in some continuous generations, the time in which the amount of the objective function has no improvement, etc. In this method, remaining one empire is also considered as the stopping condition.

In this algorithm, two methods are suggested for applying as constraints:

- Converting the constrained problem to an unconstrained problem, by using the penalty function which is the most common method. By using this method the objective function and the constraints, are transformed into the fallowing form:

$$\Phi(X, r_p) = F(X) + r_p P(X) \quad (12)$$

$$P(X) = \sum_{j=1}^{m} \{\max[0, g_j(X)]\}^2 \quad (13)$$

In the above equations, $\Phi(X, r_p)$ is the new objective function, $F(X)$ is the initial objective function, r_p is a positive penalty parameter, $g_j(X)$ are the constraints applied to the problem and m is the number of the constraints.

- A modified feasible-based mechanism. In this method, there are four rules [10]:

Rule 1: Any feasible solution is preferred to any infeasible solution.

Rule 2: Infeasible solutions including slight violation of the constraints (from 0.01 in the first iteration to 0.001 in the last iteration) are considered as feasible solutions.

Rule 3: Between two feasible solutions, the one that have the lower objective function value is preferred.

Rule 4: Between two infeasible solutions, the one that have the lower sum of constraint violation is preferred.

Since in recent works the second approach is used widely (such as HPSACO) and capability of this approach to find the global optimum is better than the first one, in this work also the second approach is employed.

The flowchart of Imperialist Competitive Algorithm is illustrated in figure (2).

Ant Colony Optimization: Ant colony optimization (ACO) was first proposed by Dorigo [10] as a multi-agent approach to solve difficult combinatorial optimization problems. Ants can find the shortest path to food by laying a pheromone (chemical) trail as they walk. Other ants follow the pheromone trail to food. Ants that happen to pick the shorter path will create a strong trail of pheromone faster than the ones choosing a longer path.

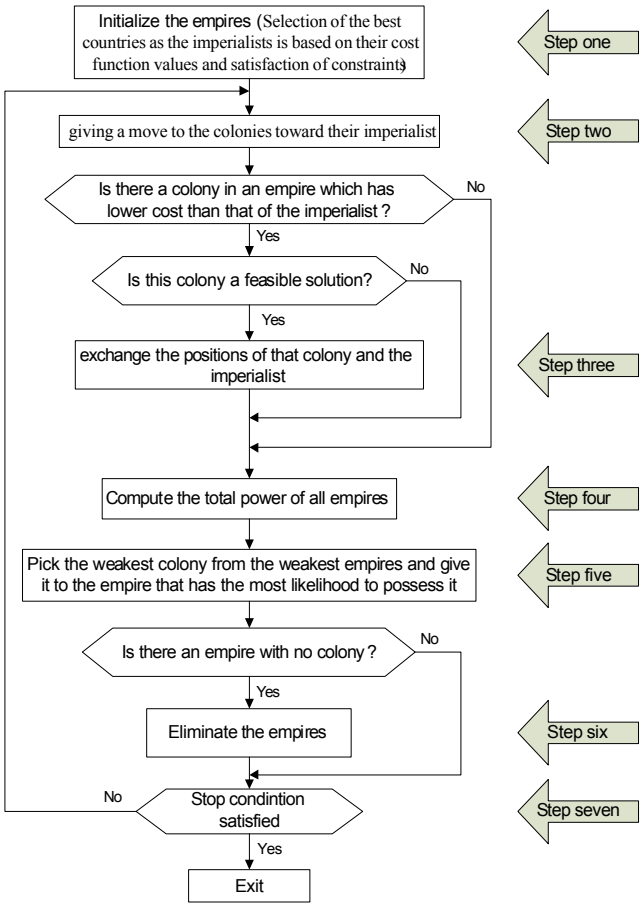

Fig. 2: Flowchart of the ICA.

Since stronger pheromone attracts ants better, more and more ants choose the shorter path until eventually all ants have found the shortest path. Consider the case of three possible paths to the food source with one longer than the others. Ants choose each path with equal probability. Ants that went and returned on the shortest path will cause it to have the most pheromone soonest. Consequently new ants will select that path first and further reinforce the pheromone level on that path. Eventually all the ants will follow the shortest path to the food [11]. One problem is premature convergence to a less than optimal solution because too much virtual pheromone was laid quickly. To avoid this stagnation, the pheromone associated with a solution disappears after a period of time. The ACO procedure is illustrated in figure (3) [12].

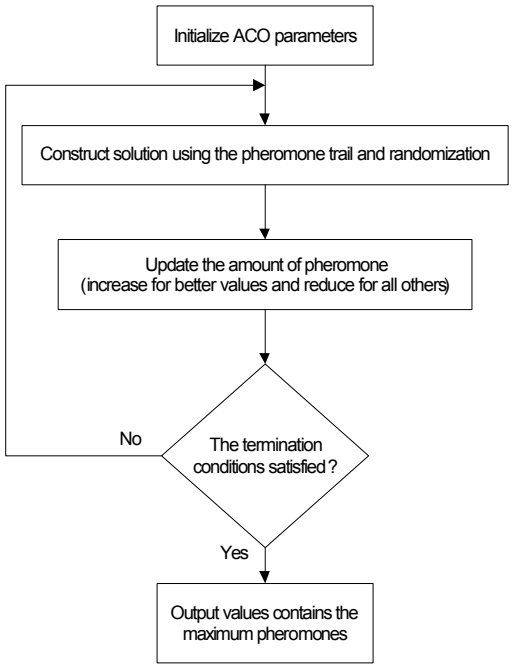

Fig. 3: The flow chart for ACO [12].

Imperialist Competitive Ant Colony Optimization: The Imperialist Competitive Ant Colony Optimization (ICACO) algorithm applies the ICA for searching global optimization, while ACO works as a local search,

wherein ants apply a pheromone-guided mechanism to refine the positions found by countries in the ICA.

In ACO stage, first of all, initial ants of size N_{col} is produced. These ants generate solutions around their relevant imperialist country which can be expressed as:

$$Ant_{j,n}^{k} = N(imperialist_n, s), j = 1,2,...,N.C_n,\ n = 1,2,...,N_{imp} \quad (14)$$

In the above equation, $N.C_n$ is the number of colonies of the nth empire. So:

$$Ant^{k} = \begin{bmatrix} Ant_{1,1} \\ \vdots \\ Ant_{N.C_1,1} \\ Ant_{1,2} \\ \vdots \\ Ant_{N.C_2,2} \\ \vdots \\ Ant_{N.C_{Nimp},Nimp} \end{bmatrix} \quad ,N.C_1 + ... + N.C_{N_{imp}} = N_{col} \quad (15)$$

Therefore, $Ant_{j,n}^{k}$ is the solution constructed by ant jth in empire nth in the stage k; $N(imperialist_n, \sigma)$ denotes a random number normally distributed with mean value imperialist nth and variance σ, where:

$$\sigma = (\text{Upper Bound} - \text{Lower Bound}) \times \eta \quad (16)$$

In the above equation, amount of upper bound and lower bound are selected from set D (section 5). Also, η is used to control the step size which in first trial is equal to 1 and by approaching to optimal point, reduces gradually and at the end tends to zero.

The ACO stage in the ICACO algorithm works as a helping factor to guide the exploration and to increase the control in the exploitation.

After generating Ants, the value of the objective function for each ant ($f(Ant_{j,n}^{k})$) is computed and the current position of ant jth in empire nth ($Ant_{j,n}^{k}$) is replaced with the position $Colony_{j,n}^{k}$ (the current position of colony jth in empire nth), if $f(Colony_{j,n}^{k})$ is bigger than $f(Ant_{j,n}^{k})$ and current ant is in the feasible space.

The Flowchart of Imperialist Competitive Ant Colony Optimization (ICACO) algorithm is illustrated in figure (4).

Mathematical Statement of Optimizing Discrete Structural Problems: Structural optimization problem with discrete variables can be formulated as a nonlinear programming problem. In the category of sizing optimization of a truss structure, the cross-section areas of the members are considered as the design variables. Each of the design variables is chosen from a list of discrete cross-sections based on production standard. In that case, the objective function would be the structure weight. The design cross-sections must also satisfy some inequality constraints equations, which restrict the discrete variables. Any structural optimization with discrete variables can be presented as follow [4]:

$$\begin{aligned} &\min && f(x^1, x^2, ..., x^d) \\ &\text{subject to :} \\ &&& g_q(x^1, x^2, ..., x^d) \le 0, \quad d = 1,2,..., N_{var} \\ &&& \qquad\qquad q = 1,2,..., M \\ &&& x^d \in S_d = \{X_1, X_2, ..., Xp\} \end{aligned} \quad (17)$$

$f(x^1, x^2,...,x^d)$ is the objective function which describe the weight of the truss, where $x^1, x^2,...,x^d$ are a set of design variables. S_d consists of all permissive discrete variables $\{X_1, X_2, ...Xp\}$ and x^d belongs to it. The inequality constraints are represented by $g_q(x^1, x^2,...,x^d) \le 0$. The numbers of design variables and inequality constraints are shown by N_{var} and M, respectively. The number of available variables is represented by p [4].

Numerical Examples: In this section, some truss optimization examples are optimized with the proposed method. Following examples show that the imperialist competitive ant colony algorithm, in comparison with algorithms like particle swarm optimization (PSO) and heuristic particle swarm optimization (HPSO) gives better results and faster convergence.

This algorithm is coded in MATLAB and is run with a Pentium 4, 2GH computer. In all of examples, the population is equal to 50. In equations (4) and (5), although we can choose any values for β and γ, but we give 2 to β and $\frac{\pi}{4}$ (Rad) to γ, we will have a good convergence of countries to the global minimum [9]. Also the number of the imperialist countries is considered 4 and ξ=0.05. These parameters are from published papers and are the same for both algorithm (ICA & ICACO).

In all the following examples, the finite element method [FEM] is used for analysis.

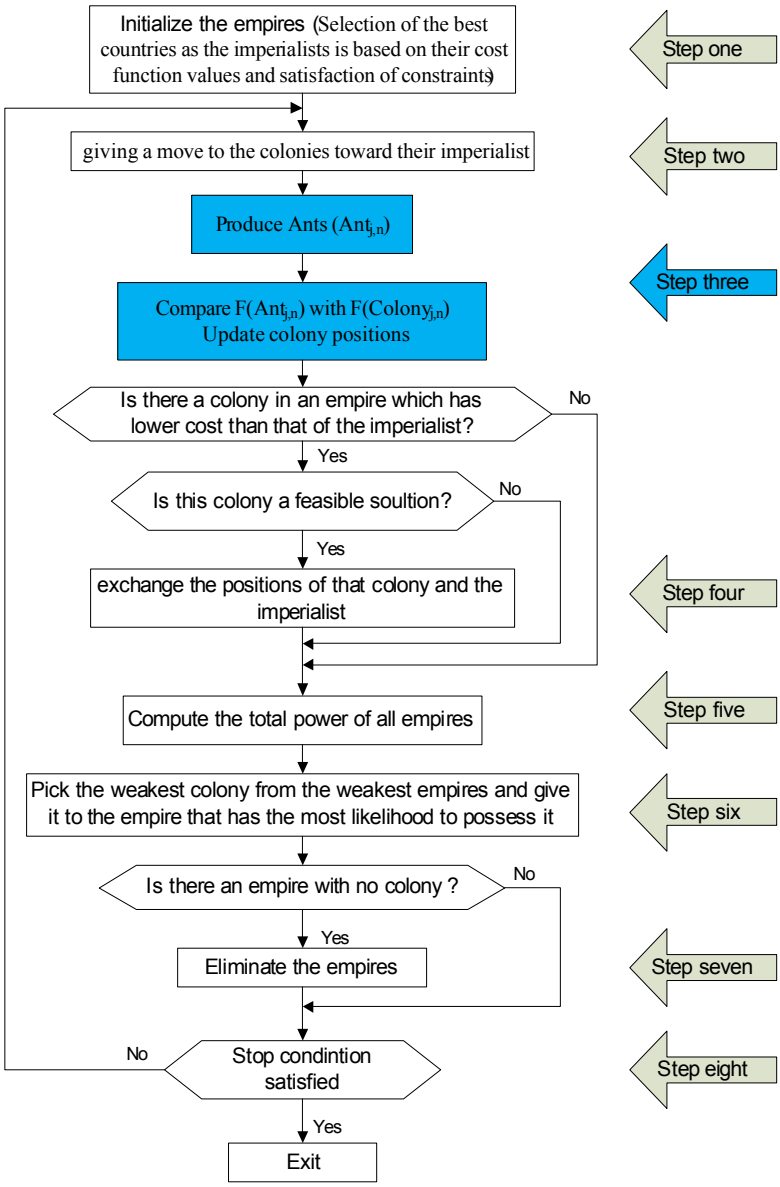

Fig. 4: Flowchart of the ICACO

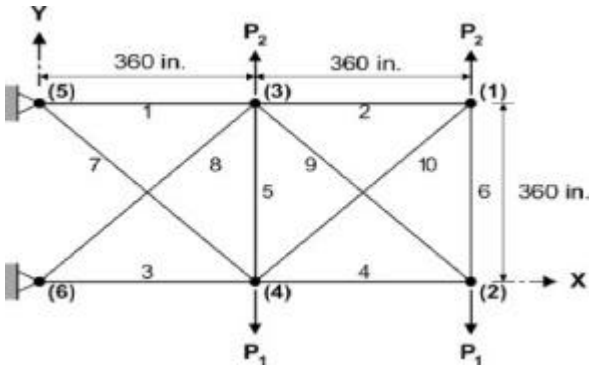

Fig. 5: A 10-bar planar truss structure

Six Node Truss: A 10-bar truss structure, shown in Figure (5), has previously been analyzed by many researchers, such as Wu [5], Rajeev [14].

The material density and the modulus of elasticity are $0.1\ lb/in^3$ (0.0272 N/cm^3) and $E = 10^4 ksi$ (68947.57Mpa), respectively. The stress limitation for each member of this structure is equal to 25ksi (±172.37Mpa) for compression and tension stresses. The allowable displacement for each node in both directions is ±2*in* (±0.0508m). The vertical load in nodes number 2 and 4 is equal to $P_1=10^5$ lbs and in nodes number 1 and 3 is equal to P_2= 0 lbs. In this problem the number of design variables is equal to 10 and two

Table 1: The results of the 10-bar truss optimization (case 1)

			Li *et al* [4]			
Truss area	ICACO	ICA	HPSO	PSO	PSOPC	Rajeev [14]
1	33.50	33.50	30.00	30.00	30.00	33.50
2	1.62	1.62	1.62	1.62	1.80	1.62
3	22.90	22.90	22.90	30.00	26.50	22.00
4	14.20	15.50	13.50	13.50	15.50	15.50
5	1.62	1.62	1.62	1.62	1.62	1.62
6	1.62	1.62	1.62	1.80	1.62	1.62
7	7.97	7.97	7.97	11.50	11.50	14.20
8	22.90	22.00	26.50	18.80	18.80	19.90
9	22.00	22.00	22.00	22.00	22.00	19.90
10	1.62	1.62	1.80	1.80	3.09	2.62
Weight	5490.74	5491.72	5531.98	5581.76	5593.44	5613.84

Table 2: Statistical results (case1)

	The results of the ICACO and ICA based on 26 independent calculation	
	ICACO	ICA
Best	5490.737	5491.717
Average	5490.737	5491.717
Worst	5490.737	5491.717
Std Dev	0.00	0.00

Table 3: The results of the 10-bar truss optimization (case 2)

			Li *et al* [4]			
Truss area	ICACO	ICA	HPSO	PSO	PSOPC	Rajeev [14]
1	31.00	31.50	31.50	24.50	25.50	30.50
2	0.10	0.10	0.10	0.10	0.10	0.10
3	24.00	23.50	24.50	22.50	23.50	23.00
4	15.00	15.00	15.50	15.50	18.50	15.50
5	0.10	0.10	0.10	0.10	0.10	0.10
6	0.50	0.50	0.50	1.50	0.50	0.50
7	7.50	7.50	7.50	8.50	7.50	7.50
8	21.00	20.50	20.50	21.50	21.50	21.0
9	21.00	21.50	20.50	27.50	23.50	21.5
10	0.10	0.10	0.10	0.10	0.10	0.10
Weight	5067.33	5070.42	5073.51	5243.71	5133.16	5059.9

Table 4: Statistical results (case 2)

	The results of the ICACO and ICA based on 26 independent calculation	
	ICACO	ICA
Best	5067.331	5070.419
Average	5067.925	5074.444
Worst	5070.419	5085.843
Std Dev	1.241	5.715

cases of discrete design variables are studied. In the first case, discrete variables are selected from the set D=[1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97, 11.50, 13.50, 13.90, 14.20, 15.50, 16.00, 16.90, 18.80, 19.90, 22.00, 22.90, 26.50, 30.00, 33.50](in^2) and in the second case, they are selected from the set D=[0.1, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.0, 9.5, 10.0, 10.5, 11.0, 11.5, 12.0, 12.5, 13.0, 13.5, 14.0, 14.5, 15.0, 15.5, 16.0, 16.5, 17.0, 17.5, 18.0, 18.5, 19.0, 19.5, 20.0, 20.5, 21.0, 21.5, 22.0, 22.5, 23.0, 23.5, 24.0, 24.5, 25.0, 25.5, 26.0, 26.5, 27.0, 27.5, 28.0, 28.5, 29.0, 29.5, 30.0, 30.5, 31.0, 31.5](in^2). A maximum iteration of 1000 steps is imposed.

Tables (1) and (3) give the comparison of optimal design results for the 10-bar planar truss structure for both cases, respectively. As it can be seen from the mentioned tables, the imperialist competitive ant colony method has better results. Tables (2) and (4) show 26 independent calculation results. In Figures (6) and (7), the convergence rate for the 10 bar structure is shown. It can be seen that the ICACO method has better convergence rate. The ICA algorithm finds the best solution in 117 iterations (5850 analyses) for first case and in 319 iterations (15950 analyses) for second case. Also, for this truss structure, HPSO needs more than 400 and 500 iterations to reach a good solution for first case and second case, respectively [4]. However, the ICACO algorithm finds the best solution in 54 iterations (2700 analyses) for first case and 118 iterations (5900 analyses) for second case.

15-bar Planar Truss: The 15-bar spatial truss structure shown in Figure (8) has been studied by Zhang [15]. The material density and the modulus of elasticity are $7800\,Kg/m^3$ and $E = 200Mpa$, respectively. The stress limitation for each member of this structure is equal to $\pm 120Mpa$. The allowable displacement for each node in both directions is $\pm 10mm$. In this example, there are 15 design variables. The design variables are selected from the set D= [113.2, 143.2, 145.9, 174.9, 185.9, 235.9, 265.9, 297.1, 308.6, 334.3, 338.2, 497.8, 507.6, 736.7, 791.2, 1063.7] (mm^2). The vertical loads used in this example are P_1=35KN, P_2=35KN and P_3=35KN. The maximum of iteration is considered as 500 steps.

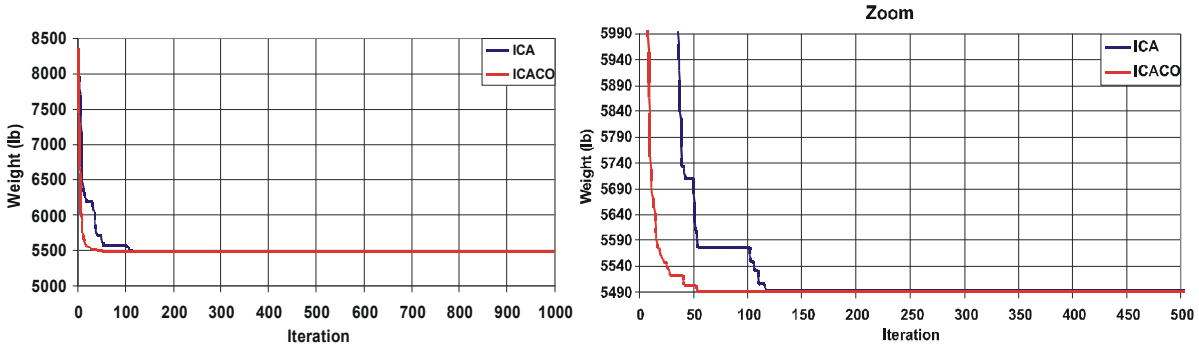

Fig. 6: Comparison of the convergence rates of the ICA and ICACO algorithms for the 10-bar planar truss structure (Case1)

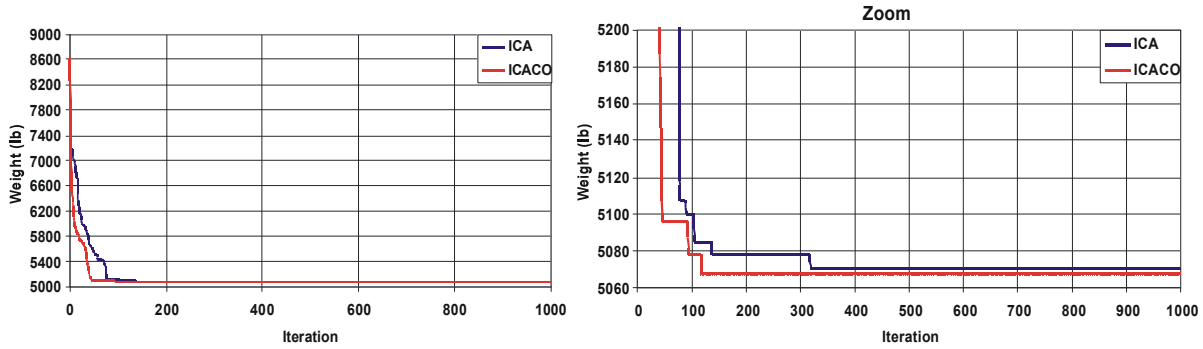

Fig. 7: Comparison of the convergence rates of the ICA and ICACO algorithms for the 10-bar planar truss structure (Case2)

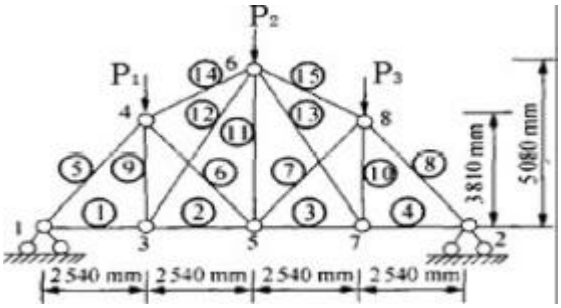

Fig. 8: A 15-bar planar truss structure

Table (5) gives the comparison of optimal design results for the 15-bar planar truss structure. As it can be seen from the results, the ICA and ICACO algorithms have same optimum results. Table (6) shows 26 independent calculation results. There is only one difference that, regarding the Figure (9), it can be seen that the ICACO method has better convergence rate. For this truss structure, it takes more than 100 and 300 iterations for the HPSO and the PSOPC algorithms to converge, respectively [4]. However, the ICA algorithm finds the best solution in 66 (3300 analyses) iterations and the ICACO algorithm takes 26 iterations (1300 analyses) to converge.

Table 5: The results of the 15-bar truss optimization

			Li *et al* [4]			
Truss area	ICACO	ICA	HPSO	PSO	PSOPC	Zhang [15]
1	113.2	113.2	113.2	185.9	113.2	308.6
2	113.2	113.2	113.2	113.2	113.2	174.9
3	113.2	113.2	113.2	143.2	113.2	338.2
4	113.2	113.2	113.2	113.2	113.2	143.2
5	736.7	736.7	736.7	736.7	736.7	736.7
6	113.2	113.2	113.2	143.2	113.2	185.9
7	113.2	113.2	113.2	113.2	113.2	265.9
8	736.7	736.7	736.7	736.7	736.7	507.6
9	113.2	113.2	113.2	113.2	113.2	143.2
10	113.2	113.2	113.2	113.2	113.2	507.6
11	113.2	113.2	113.2	113.2	113.2	279.1
12	113.2	113.2	113.2	113.2	113.2	174.9
13	113.2	113.2	113.2	113.2	185.9	297.1
14	334.3	334.3	334.3	334.3	334.3	235.9
15	334.3	334.3	334.3	334.3	334.3	265.9
Weight	105.735	105.735	105.735	108.84	108.96	142.117

Table 6: Statistical results

	The results of the ICACO and ICA based on 26 independent calculation	
	ICACO	ICA
Best	105.735	105.735
Average	105.735	105.735
Worst	105.735	105.735
Std Dev	0.00	0.00

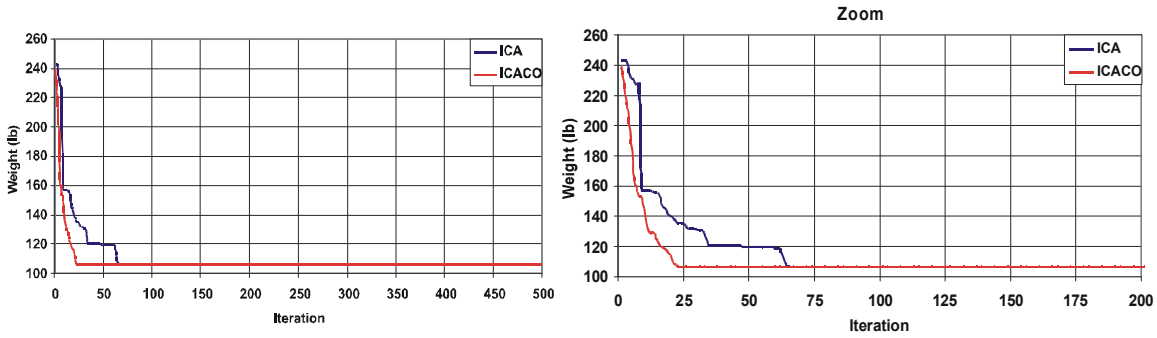

Fig. 9: Comparison of the convergence rates of the ICA and ICACO algorithms for the 15-bar planar truss structure

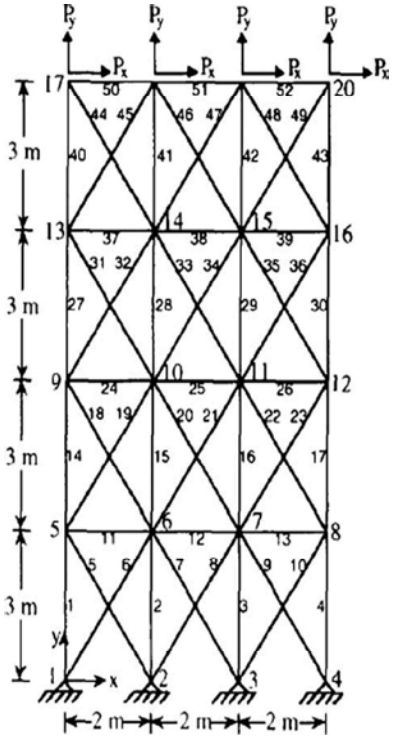

Fig. 10: A 52-bar planar truss structure

No.	in.²	mm²	No.	in.²	mm²
1	0.111	71.613	33	3.840	2477.414
2	0.141	90.968	34	3.870	2496.769
3	0.196	126.451	35	3.880	2503.221
4	0.250	161.290	36	4.180	2696.769
5	0.307	198.064	37	4.220	2722.575
6	0.391	252.258	38	4.490	2896.768
7	0.442	285.161	39	4.590	2961.284
8	0.563	363.225	40	4.800	3096.768
9	0.602	388.386	41	4.970	3206.445
10	0.766	494.193	42	5.120	3303.219
11	0.785	506.451	43	5.740	3703.218
12	0.994	641.289	44	7.220	4658.055
13	1.000	645.160	45	7.970	5141.925
14	1.228	792.256	46	8.530	5503.215
15	1.266	816.773	47	9.300	5999.988
16	1.457	939.998	48	10.850	6999.986
17	1.563	1008.385	49	11.500	7419.340
18	1.620	1045.159	50	13.500	8709.660
19	1.800	1161.288	51	13.900	8967.724
20	1.990	1283.868	52	14.200	9161.272
21	2.130	1374.191	53	15.500	9999.980
22	2.380	1535.481	54	16.000	10322.560
23	2.620	1690.319	55	16.900	10903.204
24	2.630	1696.771	56	18.800	12129.008
25	2.880	1858.061	57	19.900	12838.684
26	2.930	1890.319	58	22.000	14193.520
27	3.090	1993.544	59	22.900	14774.164
28	1.130	729.031	60	24.500	15806.420
29	3.380	2180.641	61	26.500	17096.740
30	3.470	2238.705	62	28.000	18064.480
31	3.550	2290.318	63	30.000	19354.800
32	3.630	2341.931	64	33.500	21612.860

Fig. 11: The available cross-section areas of the ASIC code [4]

52-bar Planar Truss: In the Figure (10), a 52-bar planar truss is shown which has been analyzed by Wu [5] and Lee [16]. The material density and the modulus of elasticity are $7800^{Kg}/_{m^3}$ and $E = 2.07\times10^5$, respectively. The stress limitation for each member of this structure is equal to $\pm180Mpa$. In this example, there are 12 design variables. The members of this structure are divided into 12 groups: (1)A_1-A_4, (2) A_5-A_{10}, (3) A_{11}-A_{13}, (4) A_{14}-A_{17}, (5) A_{18}-A_{23}, (6) A_{24}-A_{26},(7) A_{27}-A_{30}, (8) A_{31}-A_{36}, (9) A_{37}-A_{39}, (10) A_{40}-A_{43}, (11) A_{44}-A_{49},and (12) A_{50}-A_{52}. The design variables are selected from Figure (11). The vertical loads used in this example are P_x=100KN, P_y=200KN. The maximum of iteration is considered as 500 steps.

In Table (7), the results obtained from other methods of optimizing of 52-bar planar truss have been compared with the results of ICACO method. As it can be seen from the mentioned table, the ICACO method has better results. Table (8) shows 26 independent calculation results. Figure (12) gives the comparison of convergence rates of 52-bar planar truss structure. It can be seen that the ICACO method has better convergence rate. The ICA algorithm finds the best solution in 222 iterations (11100 analyses). However the ICACO algorithm takes 112 iterations (5600 analyses) to converge. Furthermore, for this planar truss structure, it can be observed that PSO and the PSOPC cannot find a good result, while the HPSO and HPSACO algorithms achieve good optimal results.

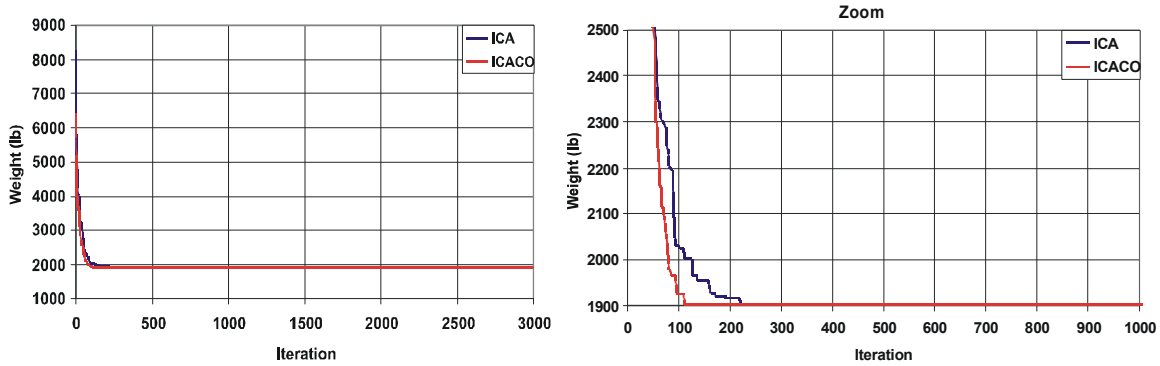

Fig. 12: Comparison of the convergence rates of the ICA and ICACO algorithms for the 52-bar planar truss structure

Table 7: The results of the 52-bar truss optimization

			Li *et al* [4]			Kaveh *et al* [13]	WU[5]	Lee[16]
Area group	ICACO	ICA	HPSO	PSO	PSOPC	HPSACO	GA	HS
1	4658.055	4658.055	4658.055	4658.055	5999.988	4658.055	4658.055	4658.055
2	1161.288	1161.288	1161.288	1374.190	1008.380	1161.288	1161.288	1161.288
3	494.193	363.225	363.225	1858.060	2696.770	494.193	645.160	506.451
4	3303.219	3303.219	3303.219	3206.440	3206.440	3303.219	3303.219	3303.219
5	940.000	940.000	940.000	1283.870	1161.290	1008.385	1045.159	940.000
6	494.193	494.193	494.193	252.260	729.030	285.161	494.193	494.193
7	2238.705	2238.705	2238.705	3303.220	2238.710	2290.318	2477.414	2290.318
8	1008.385	1008.385	1008.385	1045.160	1008.380	1008.385	1045.159	1008.385
9	494.193	641.289	388.386	126.450	494.190	388.386	285.161	2290.318
10	1283.868	1283.868	1283.868	2341.93	1283.870	1283.868	1696.771	1535.481
11	1161.288	1161.288	1161.288	1008.38	1161.290	1161.288	1045.159	1045.159
12	494.193	494.193	792.256	1045.16	494.190	506.451	641.289	506.451
Weight	1902.605	1903.366	1905.495	2230.16	2146.63	1904.83	1970.142	1906.76

Table 8: Statistical results for 52-bar truss

	The results of the ICACO and ICA based on 26 independent calculation	
	ICACO	ICA
Best	1902.605	1903.366
Average	1905.915	1908.884
Worst	1916.260	1915.283
Std Dev	4.498	5.582

However, HPSO needs more than 2000 iterations to reach a good solution [4] while HPSACO finds the optimum result in 212 iterations. So, the ICACO method has high convergence rate compared to these algorithms.

25-bar Spatial Truss: The next example considers the weight minimization of a 25-bar transmission tower as described by Wu [5], Rajeev [14] (Figure (13)).

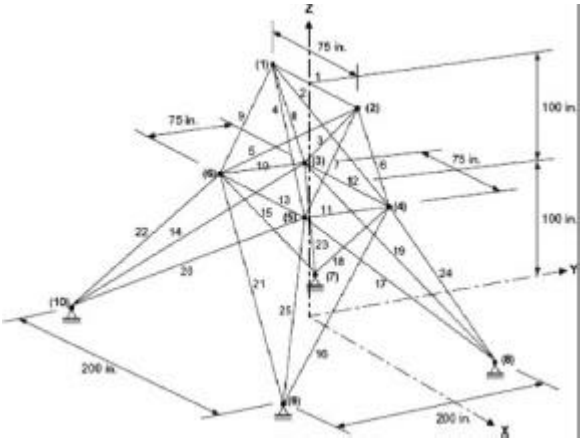

Fig. 13: A 25-bar spatial truss structure

The material density and the modulus of elasticity are $0.1\ lb/in^3$ (0.0272 N/cm^3) and $E = 10^4 ksi$(68947.57Mpa), respectively. The stress limitation for each member of this structure is equal to ±40000*psi*(±275.79 Mpa).

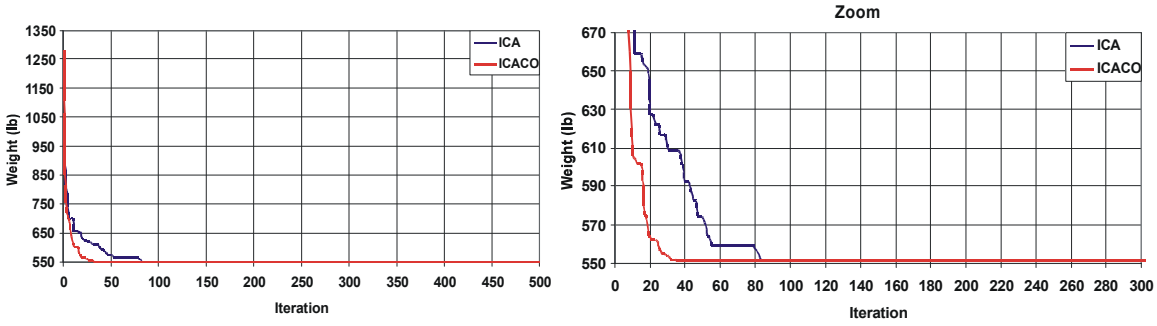

Fig. 14: Comparison of the convergence rates of the ICA and ICACO algorithms for the 25-bar spatial truss structure

Table 9: The loads applied to the nodes of 25-bar truss

	Load case 1			Load case2		
Node	F_x kips (kN)	F_y kips (kN)	F_z kips (kN)	$F_x$kips (kN)	$F_y$kips (kN)	F_z kips (kN)
1	1.0	10.0 (44.5)	-5.0 (22.25)	0.0	20.0 (89)	-5.0 (22.25)
2	0.0	10.0 (44.5)	-5.0 (22.25)	0.0	-20.0 (89)	-5.0 (22.25)
3	0.5	0.0	0.0	0.0	0.0	0.0
6	0.5	0.0	0.0	0.0	0.0	0.0

Table 10: The results of the 25-bar truss optimization

			Kaveh *et al* [13]			Li *et al* [4]	Wu [5]
Truss area	ICACO	ICA	HPSACO	HPSO	PSOPC	PSO	GA
1	0.111	0.111	0.111	0.111	0.111	1.0	0.307
2	2.130	2.130	2.130	2.130	1.563	2.62	1.990
3	2.880	2.880	2.880	2.880	3.380	2.62	3.130
4	0.111	0.111	0.111	0.111	0.111	0.25	0.111
5	0.111	0.111	0.111	0.111	0.111	0.307	0.141
6	0.766	0.766	0.766	0.766	0.766	0.602	0.766
7	1.620	1.620	1.620	1.620	1.990	1.457	1.620
8	2.620	2.620	2.620	2.620	2.380	2.880	2.620
Weight	551.14	551.14	551.14	551.14	556.90	567.49	556.43

Table 11: Statistical results for 52-bar truss

	The results of the ICACO and ICA based on 26 independent calculation	
	ICACO	ICA
Best	551.137	551.137
Average	552.016	552.056
Worst	554.743	554.743
Std Dev	1.347	1.374

The allowable displacement for each node in three directions is ±0.35*in*(±0.00889m). This structure consists of 25 members, the cross-sectional areas which are divided into 8 groups: (1) A_1, (2) A_2-A_5, (3) A_6-A_9, (4) A_{10}-A_{11}, (5) A_{12}-A_{13}, (6) A_{14}-A_{17}, (7) A_{18}-A_{21} and (8) A_{22}-A_{25}. The loads applied to this structure are described in Table (9). The design variables are selected from figure (11). The maximum of iteration is considered as 500 steps.

Table (10) gives the comparison of optimal design results for the 25-bar spatial truss structure. As it can be seen from the results, the ICA and ICACO method have same optimum results. Table (11) shows 26 independent calculation results. There is only one difference that, regarding the Figure (14), it can be seen that the ICACO method has the fastest convergence rate. The ICA algorithm finds the best solution in 84 iterations (4200 analyses). However the ICACO algorithm takes 35 iterations (1750 analyses) to converge. Furthermore, for this spatial truss structure, it takes about 200 and 400 iterations for the PSOPC and the PSO algorithms to converge, respectively [4]. So, the ICACO method has high convergence rate in the first iterations compared to these algorithms.

CONCLUSION

In this paper ICACO is developed for optimal design of trusses. ICACO is based on ICA and ACO. In this method, ACO helps ICA process not only to efficiently perform the global exploration for rapidly attaining the feasible solution space but also effectively helps to reach optimal or near optimal solution.

The efficiency of the ICACO algorithm presented in this paper is tested for optimum design of four planar and spatial structures. The results show that the ICACO algorithm converges more quickly than the ICA and other methods.

REFERENCES

1. Kaveh, A. and S. Talatahari, 2009. Size optimization of space trusses using Big Bang-Big Crunch algorithm. Computers and Structures, 87: 1129-1140.
2. Rahami, H., A. Kaveh and Y. Gholipour, 2008. Sizing, geometry and topology optimization of trusses via force method and genetic algorithm. Engineering Structures, 30: 2360-2369.
3. Rasmussen, M.H. and M. Stolpe, 2008. Global optimization of discrete truss topology design problems using a parallel cut-and-branch method. Computers and Structures, 86: 1527-1538.
4. Li, L.J., Z.B. Huang, and F. Liu, 2009. A heuristic particle swarm optimization method for truss structures with discrete variables. Computers and Structures, 87: 435-443.
5. Wu, S.J. and P.T. Chow, 1995. Steady-state genetic algorithms for discrete optimization of trusses. Comput Struct, 56: 979-991.
6. Kirkpatrick, S., C. Gelatt and M. Vecchi, 1983. Optimization by Simulated Annealing. Sci., 220: 671-680.
7. Perez, R.E. and K. Behdinan, 2007. Particle swarm approach for structural design optimization. Comput Struct, 85: 1579-1588.
8. Atashpaz-Gargari, E. and C. Lucas, 2007. Imperialist Competitive Algorithm: An Algorithm for Optimization Inspires by Imperialistic Competition. IEEE Congress on Evolutionary Computation, Singapore.
9. Khabbazi, A., E. Atashpaz-Gargari and Lucas, C., 2009. Imperialist competitive algorithm for minimum bit error rate beamforming. Int. J. Bio-Inspired Computation, 125-133.
10. Dorigo, M., 1992. Optimization, learning and natural algorithms (in Italian): PhD diss. Dipartimento di Elettronica. Politecnico di Milano, IT.
11. Haupt, R.L. and S.E. Haupt, 2004. Practical genetic algorithms: 2nd ed, Wiley and Sons. New Jersey.
12. Kaveh., A. and S. Talatahari, 2009. Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Computers and Structures, 87: 267-283.
13. Kaveh., A. and S. Talatahari, 2009. A particle swarm ant colony optimization for truss structures with discrete variables. Journal of Constructional Steel Res., 65: 1558-1568.
14. Rajeev, S. and C.S. Krishnamoorthy, 1992. Discrete optimization of structures using genetic algorithm. J. Struct Eng, ASCE, 118: 1123-250.
15. Zhang, Y.N., J.P. Liu, B. Liu, C.Y. Zhu and Y. Li, 2003. Application of improved hybrid genetic algorithm to optimize. J. South China. Univ. Technol, 33: 69-72.
16. Lee, K.S., Z.W. Geem, S.H. Lee and K.W. Bae, 2005. The harmony search heuristic algorithm for discrete structural optimization. Eng. Optim., 37: 663-84.