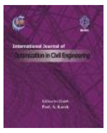
INTERNATIONAL JOURNAL OF OPTIMIZATION IN CIVIL ENGINEERING
Int. J. Optim. Civil Eng., 2015; 5(2):241-254



IMPROVED BAT ALGORITHM FOR OPTIMUM DESIGN OF LARGE-SCALE TRUSS STRUCTURES

S. Talatahari[*, †] and A. Kaveh
[1]*Department of Civil Engineering, University of Tabriz, Tabriz, Iran*
[2]*Department of Civil Engineering, Iran University of Science and Technology, Tehran-16, Iran*

ABSTRACT

Deterring the optimum design of large-scale structures is a difficult task. Great number of design variables, largeness of the search space and controlling great number of design constraints are major preventive factors in performing optimum design of large-scale truss structures in a reasonable time. Meta-heuristic algorithms are known as one of the useful tools to deal with these problems. This paper presents an improved bat algorithm for optimizing large-scale structures. The capability of the algorithm is examined by comparing the resulting design parameters and structural weight with those of other methods from literature.

Received: 12 January 2015; Accepted: 20 March 2015

KEY WORDS: modified bat algorithm; large-scale truss structures; optimum design.

1. INTRODUCTION

In the field of optimization, finding optimum design of structures are known one of difficult and complex problems. Engineers can often perform suitable designs by some trial and error approaches for small structures. However for large-scale ones, great number of design variables, largeness of the search space, and controlling great number of design constraints are major preventive factors in performing optimum design in a reasonable time [1]. Therefore, using optimization algorithms to solve such difficult problems is inevitable. In general, the optimization methods are divided into two groups, known as mathematical

[*]Corresponding author: Department of Civil Engineering, Tabriz University.
[†]E-mail address: talatahari@tabrizu.ac.ir (S. Talatahari)

methods and met-heuristic ones. The first group often uses gradient values of functions (or its approximated values) and can find local optimum points with a small effort. However, in contrast to meta-heuristic algorithms, they cannot guarantee reaching the global point (or even near to it) for large-scale problems with con-convex and non-smooth search spaces.

The lack of dependency on gradient information, inherent capability to deal with both discrete and continuous design variables and automated global search features to produce near-optimum solutions (if not the global optimum) for complicated problems directed the researchers toward using meta-heuristic algorithms [2]. Therefore, meta-heuristic algorithms are known to be robust tools for dealing with today's large-scale engineering problems of increased complexity. These approaches are derivative-free methods and make use of the ideas inspired from the nature or social phenomenon, such as the biological evolutionary process (e.g., genetic algorithm (GA) [3], differential evolution (DE) [4] and biogeography-based optimization (BBO) [5]), physical phenomena (e.g. simulated annealing (SA) [6], charged system search (CSS) [7,8], Colliding Bodies Algorithm (CBO) [9]) or animal behavior (e.g., particle swarm optimization (PSO) [10], ant colony optimization (ACO) [11], artificial bee colony (ABC) [12], ant cuckoo search (CS) [13], firefly algorithm (FA) [14], krill herd (KH) [15] and bat algorithm (BA) [16]), etc. A detailed review of these algorithms as well as their applications in engineering optimization problems can be found in Yang et al. [17] and Kaveh [18].

In this paper, the BA is slightly improved and applied to optimal design of large-scale truss structures. Bat Algorithm (BA) has been developed based on the echolocation behavior of microbats. An extensive review of BA and its new variants can be found in the work of Yang and He [19]. They reviewed a wide range of diverse applications and case studies on this algorithm. However, because of the newness of BA compared to other techniques, few articles have been published concerning its application in structural optimization problems. Gandomi et al. [20] utilized BA to solve several constraint optimization problems. Optimum design of truss structures for minimizing the weight subject to stress, stability and displacement constraints according to American Institute of Steel Construction-Allowable Stress Design (AISC-ASD) specification is performed in [2]. Kaveh and Zakian [21] applied this algorithm to optimize some benchmark truss and frame structures. Also, BA is applied to discrete size optimization of steel frames by Hasançebi and Carbas [22].

The mechanism proposed in this paper is tested on two large-scale benchmark truss structures. Numerical results are compared to those of other methods available in literature. The performance study demonstrates the efficiency of the proposed method.

2. INTRODUCTION TO BAT ALGORITHM

In order to make the paper self-explanatory, the characteristics of BA is briefly explained in the following sub-sections.

2.1. Standard bat algorithm (BA)

BA is a multi-agent approach that simulates the behavior of microbats [23–24]. In echolocation, each pulse only lasts a few thousandths of a second (up to about 8–10 ms).

Nevertheless, it has a constant frequency which is usually in the range of 25–150 kHz corresponding to the wavelengths of 2–14 mm. In BA, the echolocation properties of microbats can be idealized as the following rules [16]:

1. All bats use echolocation to sense distance, and they also ‘‘know’’ the difference between food/prey and background barriers in some magical way;

2. Bats randomly fly with velocity v_i at position x_i with a fixed frequency f_{min}, varying wavelength k and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r\epsilon[0,1]$, depending on the proximity of their target;

3. Although the loudness can vary in many ways, it is assumed that the loudness varies from a large (positive) A_0 to a minimum constant value A_{min}.

The basic steps of BA can be summarized as the pseudo-code shown in Fig. 1, [20].

For each bat (i), its position x_i and velocity v_i in a d-dimensional search space should be defined. x_i and v_i should also be subsequently updated during the iterations. The new solutions x_i^t and velocities v_i^t at time step t can be calculated by:

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta \tag{1}$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x^*) f_i \tag{2}$$

$$x_i^t = x_i^{t-1} + v_i^{t-1} \tag{3}$$

where β in the range of [0,1] is a random vector drawn from a uniform distribution. Here, x^* is the current global best location (solution), which is located after comparing all the solutions among all the n bats. As the product $\lambda_i f_i$ is the velocity increment, either f_i (or λ_i) can be used to adjust the velocity change while fixing the other factor λ_i (or f_i), depending on the type of the problem of interest. For implementation, $f_{min} = 0$ and $f_{max} = 100$ are used, depending on the domain size of the problem of interest. Initially, each bat is randomly assigned a frequency that is drawn uniformly from $[f_{min}, f_{max}]$.

For the local search part, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using a local random walk, as

$$x_{new} = x_{old} + \varepsilon A^t \tag{4}$$

where the random number ε is drawn from [−1,1], while $A^t = \langle A_i^t \rangle$ is the average loudness of all the bats at this time step. The update of the velocities and positions of bats have some similarities to the procedure in the standard particle swarm optimization as f_i essentially controls the pace and range of the movement of the swarming particles. To a degree, BA can be considered as a balanced combination of the standard particle swarm optimization and the intensive local search controlled by the loudness and pulse rate. Once a bat found its prey, the loudness usually decreases and the rate of pulse emission increases. In this case, the loudness can be chosen as any value of convenience. For simplicity, $A_0 = 1$ and $A_{min} = 0$ can be used. Assuming $A_{min} = 0$ means that a bat has just found the prey and temporarily stop emitting any sound, we have:

$$A_i^{t+1} = \alpha A_i^t \tag{5}$$

$$r_i^{t+1} = r_i^0 (1 - e^{-\gamma t}) \tag{6}$$

where α and γ are constants. In the simplest case, $\alpha = \gamma$ can be used. For the simulations in this study, $\alpha = \gamma = 0.9$, Ref. [6].

```
Objective function $f$ (x), x = ($x1$, ...,$xd$)$^T$
Initialize the bat population x$i$ ($i$ = 1,2, ...,$n$) and v$i$
Define pulse frequency fi at x$i$
Initialize pulse rates ri and the loudness Ai
while (t <Max number of iterations)
    Generate new solutions by adjusting frequency,
    and updating velocities and locations/solutions
    if (rand > ri)
        Select a solution among the best solutions
        Generate a local solution around the selected best solution
    end if
    Generate a new solution by flying randomly
    if (rand < Ai & $f$ (x$i$) < $f$ (x*))
        Accept the new solutions
        Increase ri and reduce Ai
    end if
    Rank the bats and find the current best x$^*$
end while
Postprocess results and visualization
```

Figure 1. Pseudo-code of the bat algorithm (BA)

2.2. Improved BA

Exploration ability of the bat algorithm is acceptable, however its convergence speed needs to be improved for large-scale problems in order to reduce the required number of iterations. Here, the average loudness of bats, A^t is defined dynamically as [21]:

$$A_i^{t+1} = A_i^{\max} e^{c.t} \ , \ c = \frac{\ln(A_i^{\max} - A_i^{\min})}{t_{\max}} \tag{7}$$

where $A_i^{\max}$ and $A_i^{\min}$ are the minimum and maximum values for A_i , respectively. t_{max} stands for maximum number of iterations. Proper tuning of this parameter reduces the number of the iterations

Also in the improved BA, a microbat is allowed to update its echolocation parameters each time when it produces a solution that surpasses its individual best, not the global best necessarily, [2]. Also, the Eq. (6) is modified as follows for adaptation of pulse rate parameter, [2]:

$$r_i^{t+1} = 1-(1-r_i^0)\gamma^{t+1} \le r_{\max} \tag{8}$$

Eq. (6) facilitates a more gradual change of pulse rate parameter from its initial (minimum) value of r_i^0 towards $r_{\max}$, whereas in Eq. (6) the pulse rate immediately approaches $r_{\max}$ in a few iterations and remains stationary at this value thereafter, Ref. [2]. A graphical comparison of the Eq. (6) and Eq. (8) is presented in Fig. 2.

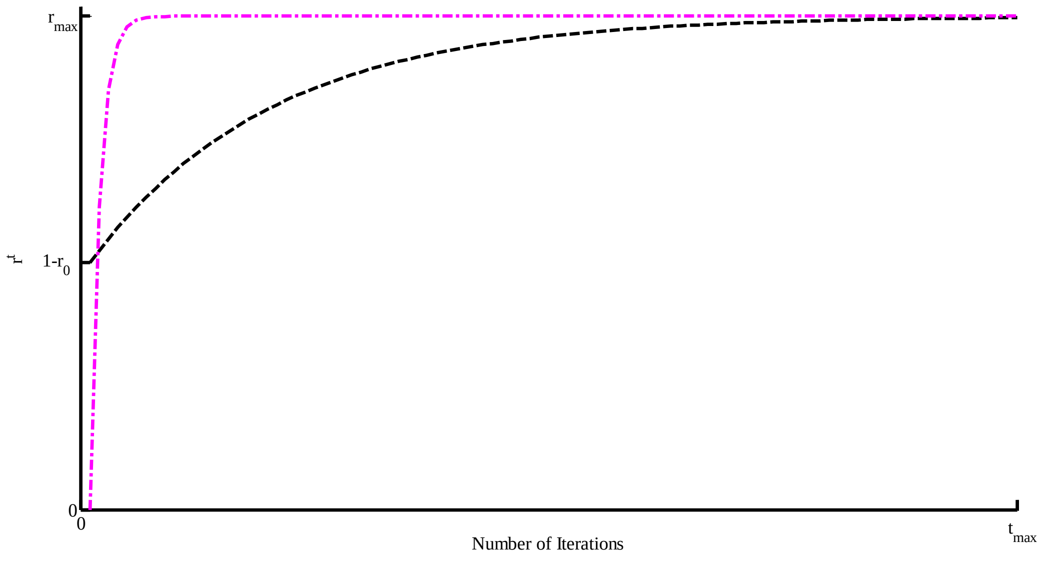


Figure 2. Comparison of pulse rate adaptation strategies

3. STRUCTURAL OPTIMIZATION PROBLEMS

3.1 Statement of the optimization design problem

The general formulation of the weight minimization problem for a truss structure is as follows:

$$\begin{aligned} &\text{minimize} && W(x)=\sum_{i=1}^{n}\rho_i\cdot x_i\cdot L_i \\ &\text{subject to :} && \delta_{\min}\le\delta_{\mathrm{i}}\le\delta_{\max} && i=1,2,....,m \\ &&& \sigma_{\min}\le\sigma_{\mathrm{i}}\le\sigma_{\max} && i=1,2,....,n \\ &&& \sigma_i^b\le\sigma_{\mathrm{i}}\le 0 && i=1,2,....,nc \\ &&& x_{\min}\le x_i\le x_{\max} && i=1,2,....,ng \end{aligned} \tag{9}$$

where $W(x)$ is the weight of the structure; n is the number of members making up the structure; m represents the number of nodes; nc denotes the number of compression elements; ng is the number of groups (number of design variables); ρ_i is the material density of member i; L_i denotes the length of member i; x_i represents the cross-sectional area of member *i* chosen from the set of areas between $x_{\min}$ and $x_{\max}$; *min* is the lower bound and *max* is the upper bound; σ_i and δ_i are the stress and nodal deflection respectively; σ_i^b denotes allowable buckling stress in member *i* when it is in compression.

3.2 Constraint handling

The penalty function is utilized to handle the constraints. After analyzing a structure, the deflection of each node and the stress in each member are obtained. These values are compared with allowable limits to calculate the penalty functions as:

$$\begin{cases} \sigma_i^{\min} < \sigma_i < \sigma_i^{\max} & \Rightarrow \Phi_\sigma^{(i)} = 0 \\ \sigma_i^{\min} > \sigma_i \text{ or } \sigma_i^{\max} < \sigma_i & \Rightarrow \Phi_\sigma^{(i)} = \dfrac{\sigma_i - \sigma_i^{\min/\max}}{\sigma_i^{\min/\max}} \end{cases} \qquad i = 1,2,\dots,n \tag{10}$$

$$\begin{cases} \sigma_b < \sigma_i < 0 & \Rightarrow \Phi_{\sigma b}^{(i)} = 0 \\ \sigma_k < 0 \ \wedge \ \sigma_i < \sigma_b & \Rightarrow \Phi_{\sigma b}^{(i)} = \dfrac{\sigma_i - \sigma_b}{\sigma_b} \end{cases} \qquad i = 1,2,\dots,nc \tag{11}$$

$$\begin{cases} \delta_i^{\min} < \delta_i < \delta_i^{\max} & \Rightarrow \Phi_\delta^{(i)} = 0 \\ \delta_i^{\min} > \delta_i \text{ or } \delta_i^{\max} < \delta_i & \Rightarrow \Phi_\delta^{(i)} = \dfrac{\delta_i - \delta_i^{\min/\max}}{\delta_i^{\min/\max}} \end{cases} \qquad i = 1,2,\dots,m \tag{12}$$

In optimizing structures, objective is to find the minimum amount of merit function. Merit function is defined as

$$Mer^k = \varepsilon_1 \cdot W^k + \varepsilon_2 \cdot (\Phi_\sigma^k + \Phi_\delta^k + \Phi_{\sigma b}^k)^{\varepsilon_3} \tag{13}$$

Here, Mer^k is the merit function for ant *k*; ε_1, ε_2 and ε_3 are the coefficients of merit function. Φ_σ^k, Φ_δ^k and $\Phi_{\sigma b}^k$ are the summation of stress penalties, summation of nodal deflection penalties and summation of buckling stress penalties for ant *k*, respectively.

For multiple loadings, after analyzing the structure and determining the penalty functions for each component of the load, the total penalty function is calculated through addition of penalty functions of stress, buckling stress for each member, and deflection for each node, as:

$$Mer^k = \varepsilon_1 \cdot W^k + \varepsilon_2 \cdot \sum_{i=1}^{np} (\Phi_{\sigma(i)}^k + \Phi_{\delta(i)}^k + \Phi_{\sigma b(i)}^k)^{\varepsilon_3} \tag{14}$$

where *np* is the number of multiple loadings. In this paper, for a better control on other parameters, ε_1 is set to 1 and the coefficient ε_2 is taken as the weight of the structure and the coefficient ε_3 is set in a way that the penalties decrease. In the first iterations of the search process, ε_3 is set to 1.5 but gradually it is increased to 3, Ref. [1].

4. NUMERICAL EXAMPLES

In this section, two large-scale truss optimization examples are optimized utilizing the presented method. Then the final results are compared to the solutions of other methods to demonstrate the efficiency of this approach. The proposed algorithm is coded in Matlab and structures are analyzed using the direct stiffness method.

4.1 A 244-bar transformation tower

The first example is a 244-bar transmission tower shown in Fig. 3. Members of the transmission tower are initially collected into 26 groups as given by Saka [25] but in this study all members of the transmission tower are linked into 32 groups to enlarge the problem [26]. The value of the modulus of elasticity is taken as 30,450 ksi (210,000 MPa) and the material density is 0.1 lb/in^3 (2767.990 kg/m^3). The allowable value of 20.30 ksi (140 MPa) is employed for tensile stresses and the formulation of buckling obeying AISC-ASD (1989) [27] code is considered for compressive stresses. The displacement limitations of ±1.77 in (4.5 cm) are imposed on nodes 1 and 2, and limitations of ±1.18 in (3.0 cm) on nodes 17, 24 and 25 in x-direction. These nodes are subjected to the displacement limits of ±0.59 in (1.5 cm) in y-direction. The load cases considered are shown in Table 1. The minimum cross-sectional area of all members is 0.775 in^2 (5.0 cm^2) and the maximum cross-sectional area is 20.0 in^2 (129.03 cm^2).

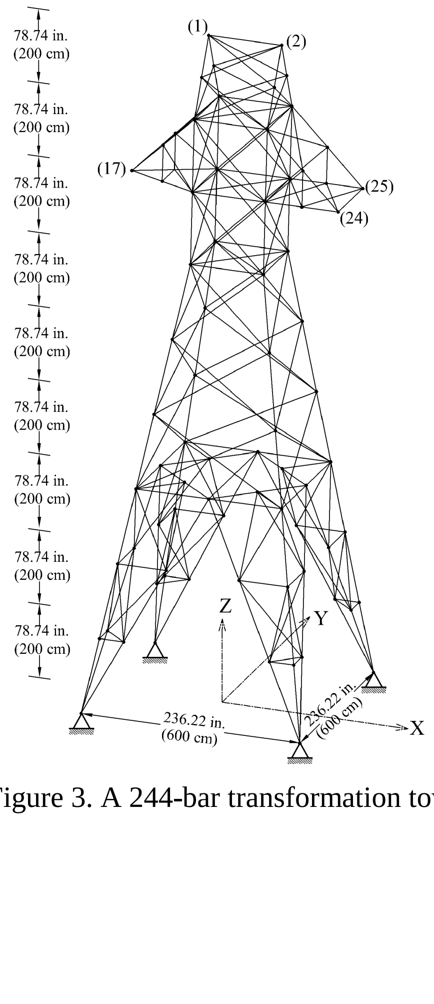


Figure 3. A 244-bar transformation tower

Table 1: Loading conditions for the 244-bar transformation tower

	Case 1			Case 2		
Node	P_X kips (kN)	P_Y kips (kN)	P_Z	P_X	P_Y kips (kN)	P_Z
1	-2.448 (10)	-6.744 (30)	0.0	0.0	-80.899 (360)	0.0
2	2.448 (10)	-6.744 (30)	0.0	0.0	-80.899 (360)	0.0
17	8.568 (35)	-20.224 (90)	0.0	0.0	-40.449 (180)	0.0
24	42.82 (175)	-10.112 (45)	0.0	0.0	-20.224 (90)	0.0
25	42.82 (175)	-10.112 (45)	0.0	0.0	-20.224 (90)	0.0

The maximum number of analyses is 15,000 for the BA. The BA achieves the best solution 2,374.05 kg while the different imperialist competitive algorithm (i.e. CICA, OICA and ICA algorithms [28]) achieves 2,478.95 kg, 2,517.29 kg and 2,562.09 kg, respectively. The HPSACO and PSOPC algorithms [26] achieved 2,415.02 kg and 2,652.56kg, respectively. Clearly, BA algorithm can find the best results comparing the other methods. Although, the differences between the results of HPSACO and BA are small, however it is worth to note that HPSACO utilizes the PSO with two auxiliary tools (ACO and HS) and if one adds these tools to this new algorithm, obviously the resultant method will be improved. Fig. 4 shows the convergence history for the minimum weight of 244-bar transformation tower solved by the BA-based method.

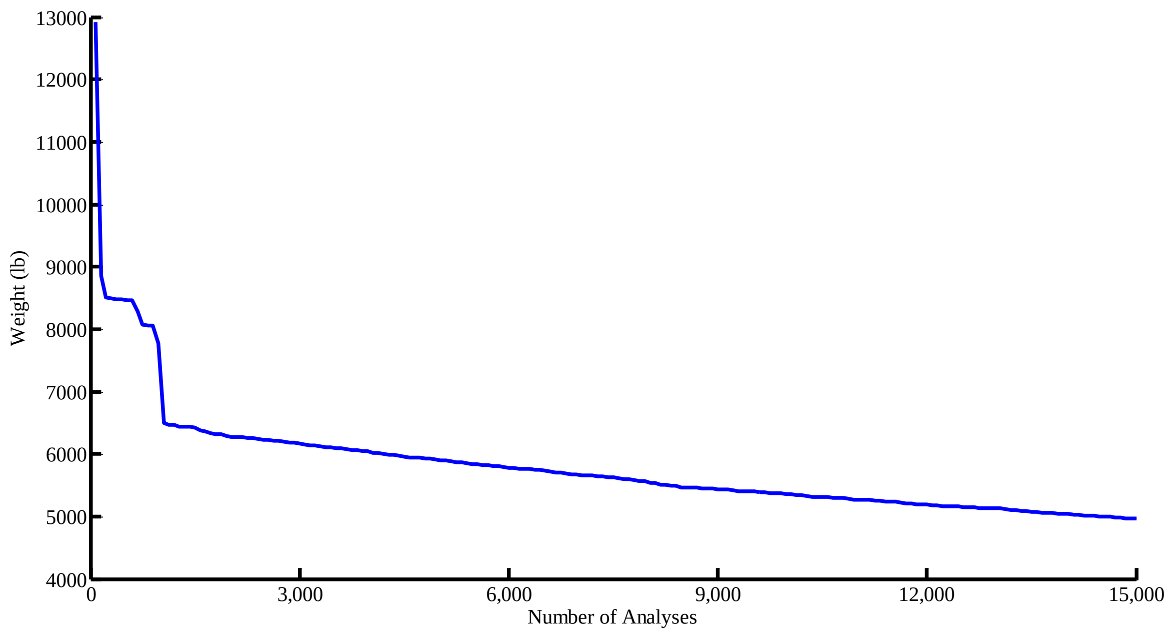


Figure 4. Convergence history of BA for the 244-bar transformation tower

4.2 A 942-bar spatial truss

A 26-story-tower space truss containing 942 elements and 244 nodes is considered as the second large-scale example. Fifty-nine design variables are used to represent the cross-sectional areas of 59 element groups in this structure, employing the symmetry of the structure. Fig. 5 shows the geometry and the 59 element groups. The material density is 0.1 lb/in^3 (2767.990 kg/m^3) and the modulus of elasticity is 10,000 ksi (68,950 MPa). The members are subjected to the stress limits of ±25 ksi (172.375 MPa) and the four nodes of the top level in the x, y, and z directions are subjected to the displacement limits of ±15.0 in (38.10 cm) (about 1/250 of the total height of the tower). The allowable cross-sectional areas in this example are selected from 0.1 to 20.0 in^2 (from 0.6452 cm^2 to 129.032 cm^2). The loading on the structure consists of:

1) The vertical load at each node in the first section is equal to −3 kips (−13.344 kN);

2) The vertical load at each node in the second section is equal to −6 kips (−26.688 kN);

3) The vertical load at each node in the third section is equal to −9 kips (−40.032 kN);

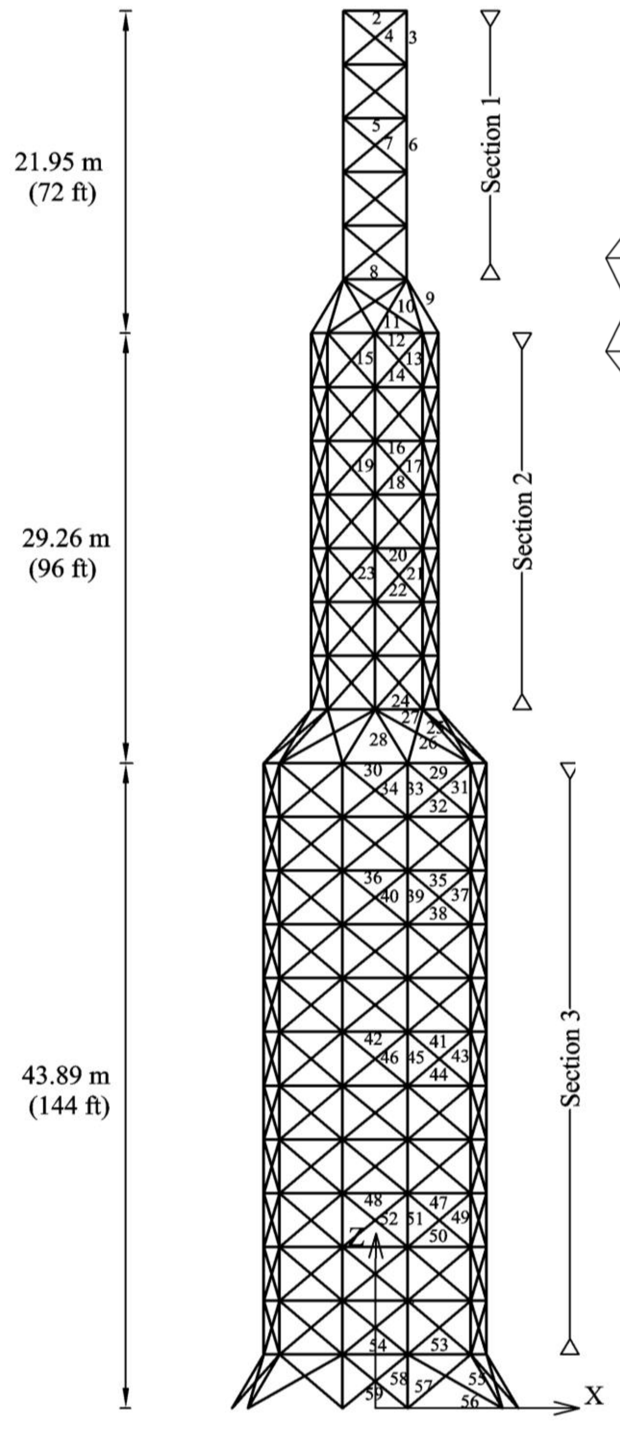
4) The horizontal load at each node on the right side in the x direction is equal to −1 kips (−4.448kN);

5) The horizontal load at each node on the left side in the x direction is equal to 1.5 kips (6.672kN);

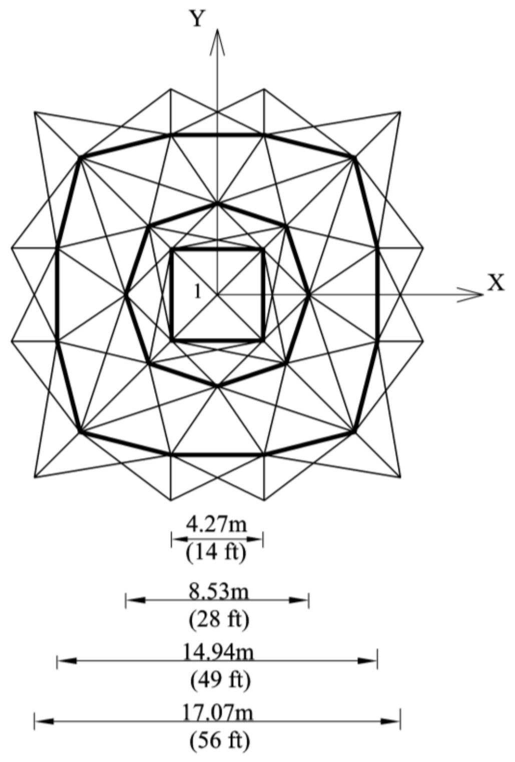
6) The horizontal load at each node on the front side in the y direction is equal to −1 kips (−4.448kN);

7) The horizontal load at each node on the back side in the x direction is equal to 1 kips (4.448kN).

This example has been optimized using 5 meta-heuristic algorithms, previously. The CSS method [8] achieved a good solution after 15,000 analyses and found an optimum weight of 47,371 lb (210,716 N). The best weights for the GA, PSO, BB–BC and HBB–BC were 56,343 lb (250,626 N), 60,385 lb (268,606 N), 53,201 lb (236,650 N) and 52,401 lb (233,091 N), respectively [1]. The new algorithm can find the best result among others as shown in Table 2. The best result of this algorithm is equal to 46,015lb (204,684 N). The new algorithm has better performance in terms of standard deviation and the average weight. It converges to a solution after 15,000 analyses of structures in average. Table 3 provides the statistic information for this example and the convergence history is shown in Fig. 6.



(a) Side view



(b) Top view

Figure 5. A 942-bar spatial truss

Table 2: The results of the optimum design of the BA algorithm for the second example

Optimal cross-sectional areas (cm^2)								
	Members	Area		Members	Area		Members	Area
1	A_1	1.037	21	A_{21}	2.575	41	A_{41}	0.516
2	A_2	2.078	22	A_{22}	0.360	42	A_{42}	0.698
3	A_3	1.472	23	A_{23}	3.195	43	A_{43}	20.445
4	A_4	0.511	24	A_{24}	5.087	44	A_{44}	0.5346
5	A_5	0.681	25	A_{25}	18.907	45	A_{45}	1.577
6	A_6	16.556	26	A_{26}	0.523	46	A_{46}	0.483
7	A_7	0.362	27	A_{27}	2.570	47	A_{47}	0.521
8	A_8	3.086	28	A_{28}	18.787	48	A_{48}	1.164
9	A_9	2.208	29	A_{29}	4.869	49	A_{49}	19.870
10	A_{10}	3.833	30	A_{30}	4.864	50	A_{50}	0.852
11	A_{11}	0.803	31	A_{31}	13.328	51	A_{51}	3.796
12	A_{12}	1.047	32	A_{32}	0.870	52	A_{52}	0.406
13	A_{13}	2.730	33	A_{33}	0.901	53	A_{53}	11.861
14	A_{14}	0.535	34	A_{34}	1.290	54	A_{54}	17.975
15	A_{15}	20.282	35	A_{35}	0.135	55	A_{55}	18.179
16	A_{16}	1.329	36	A_{36}	0.202	56	A_{56}	3.225
17	A_{17}	2.022	37	A_{37}	18.407	57	A_{57}	2.636
18	A_{18}	0.538	38	A_{38}	0.652	58	A_{58}	5.326
19	A_{19}	18.070	39	A_{39}	1.425	59	A_{59}	0.258
20	A_{20}	0.324	40	A_{40}	0.357			
Weight		204684 N						

Table 3: Performance comparison for the second example

	GA	PSO	BB–BC	HBB–BC	CSS	BA
Best weight (lb)	56343 (250,626 N)	60385 (268606 N)	53201 (236650 N)	52401 (233091 N)	47371 (210716 N)	46015 (204,684 N)
Average weight (lb)	63223 (281230 N)	75242 (334693 N)	55206 (245568 N)	53532 (238122 N)	48603 (216197 N)	47856 (212874 N)
Std Dev (lb)	6640.6 (29,539 N)	9906.6 (44,067N)	2621.3 (11660 N)	1420.5 (6318 N)	950.4 (4,227 N)	712.36 (3,168 N)
No. of analyses	50,000	50,000	50,000	30,000	15,000	15,000

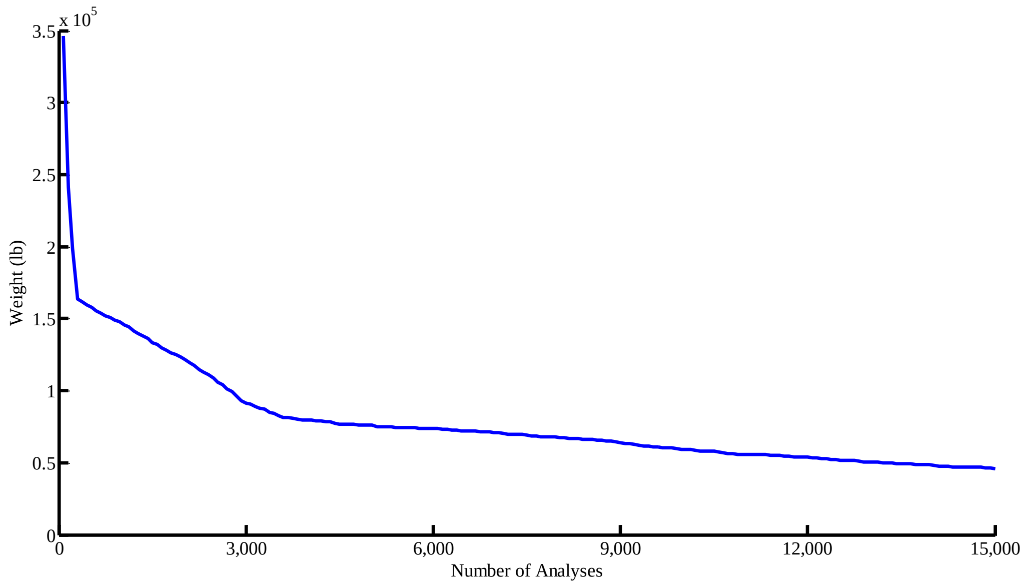


Fig. 6 Convergence history of BA for the 942-bar spatial truss

5. CONCLUDING REMARKS

Determining the optimum design of large-scale structures is known as one of difficult optimization problems. In this paper, the bat algorithm is improved and applied to solve these problems. Exploration ability of the bat algorithm is acceptable; however, its convergence speed needs to be improved for large-scale problems. In this paper, three improvements are suggested to solve this problem and improve the BA-based algorithm. The first improvement corresponds to the average loudness of bats which is defined dynamically. The second one belongs to updating echolocation parameters in which a microbat is allowed to update its echolocation parameters each time when it produces a solution. Finally, the pulse rate parameter is redefined to facilitate a more gradual change of it. These mechanisms reduce the required number of analyses. The robustness of the new algorithm is tested by optimum design of two large-scale trusses. The results illustrate the efficiency of the proposed algorithm.

REFERENCES

1. Kaveh A, Talatahari S. Size optimization of space trusses using big bang–big crunch algorithm, *Comput Struct* 2009; **87**(17-18): 1129-40.
2. Hasançebi O, Teke T, Pekcan O. A bat-inspired algorithm for structural optimization, *Comput Struct* 2013; **128**: 77-90.
3. Sivaraj R, Ravichandran T. A review of selection methods in genetic algorithm, *Inter J Eng Sci Technol* 2011; **3**: 3792-7.
4. Das S, Suganthan PN. Differential evolution: A survey of the state-of-the-art, *Trans*

Evol Comput. IEEE 2011; **15**: 4-31.
5. Simon D. Biogeography-based optimization, *Trans Evol Comput, IEEE* 2008; **12**: 702-13.
6. Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing, *Science* 1983; **220**(4598): 671-80.
7. Kaveh A, Talatahari S. Optimal design of skeletal structures via the charged system search algorithm, *Struct Multidiscip Optim* 2010b; **41**(6): 893-911.
8. Nouhi B, Talatahari S, Kheiri H, Cattani C. Chaotic charged system search with a feasible-based method for constraint optimization problems, *Math Prob Eng* 2013; Article ID 391765, 8 pages.
9. Kaveh A, Mahdavai VR. Colliding bodies optimization: A novel meta-heuristic method, *Comput Struct* **139**(2014): 18-27.
10. 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.
11. Geem ZW. Harmony Search Algorithms for Structural Design**,** Springer Verlag, 2009.
12. Karaboga D, Gorkemli B, Ozturk C, Karaboga N. A comprehensive survey: artificial bee colony (ABC) algorithm and applications, *Artific Intelligence Rev* 2014; **42**: 21-57.
13. Yang XS, Deb S. Cuckoo search: recent advances and applications, *Neural Comput Applic* 2014; **24**: 169-74.
14. Yang XS. Firefly algorithm, stochastic test functions and design optimization, *Inter J Bio-inspired Comput* 2010; **2**(2): 78-84.
15. Gandomi AH, Alavi AH. Krill herd: a new bio-inspired optimization algorithm, *Commun Nonlinear Sci Numer Simul* 2012; **17**(12): 4831-45.
16. Yang XS. A new metaheuristic bat-inspired algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NISCO 2010) (Eds JR Gonzalez et al), Studies in Computational Intelligence, Springer Berlin, 284, Springer, 2010, pp. 65-74.
17. Gandomi AH, Yang XS, Talatahari S, Alavi AH (Editors). Metaheursitic Applications in Structures and Infrastructures, Elsevier, 2013, ISBN: 9780123983640.
18. Kaveh A. Advances in Metaheuristic Algorithms for Optimal Design of Structures, Springer Verlag, Switzerland, 2015.
19. Yang XS, He X. Bat algorithm: literature review and applications, *Inter J Bio-Inspired Comput.* 2013; **5**(3): 141-9.
20. Gandomi AH, Yang XS, Alavi AH, Talatahari S. Bat algorithm for constrained optimization tasks, *Neural Comput Applic* 2013; **22**(6): 1239-55.
21. Kaveh A, Zakian P. Enhanced bat algorithm for optimal dsesign of skeletal structures, *Asian J Civil Eng* 2014; **15**(2): 179-212.
22. Hasancebi O, Carbas S. Bat inspired algorithm for discrete size optimization of steel frames, *Adv Eng Softw* 2014; **67**: 173-85.
23. Altringham JD. Bats: biology and behavior, Oxford University Press, Oxford, 1996.
24. Richardson P. Bats. Natural History Museum, London, 2008.
25. Saka MP. Optimum design of pin-jointed steel structures with practical applications, *J Struct Eng, ASCE* 1990; **116**(10): 2599-2620.
26. Kaveh A, Talatahari S. Particle swarm optimizer, ant colony strategy and harmony

search scheme hybridized for optimization of truss structures, *Comput Struct* 2009; **87**(5-6): 267-83

27. American Institute of Steel Construction (AISC). Manual of steel construction-allowable stress design. 9th ed. Chicago, IL, 1989.
28. Talatahari S, Kaveh A, Sheikholeslami R. chaotic imperialist competitive algorithm for optimum design of truss structures, *Struct Multidiscip Optim* 2012; **46**(3): 355-67.