Vietnam Journal of Mechanics, VAST, Vol. 38, No. 4 (2016), pp. 307 – 317
DOI:10.15625/0866-7136/7476

AN EVOLUTIONARY-BASED OPTIMIZATION ALGORITHM FOR TRUSS SIZING DESIGN

Pham Hoang Anh
National University of Civil Engineering, Hanoi, Vietnam
E-mail: anhpham.nuce@gmail.com
Received November 28, 2015

Abstract. In this paper, the optimal sizing of truss structures is solved using a novel evolutionary-based optimization algorithm. The efficiency of the proposed method lies in the combination of global search and local search, in which the global move is applied for a set of random solutions whereas the local move is performed on the other solutions in the search population. Three truss sizing benchmark problems with discrete variables are used to examine the performance of the proposed algorithm. Objective functions of the optimization problems are minimum weights of the whole truss structures and constraints are stress in members and displacement at nodes. Here, the constraints and objective function are treated separately so that both function and constraint evaluations can be saved. The results show that the new algorithm can find optimal solution effectively and it is competitive with some recent metaheuristic algorithms in terms of number of structural analyses required.

Keywords: Structural optimization, evolutionary-based optimization, metaheuristics, truss structure, sizing optimization.

1. INTRODUCTION

Truss optimization is one of the most popular design problems and has been an extensive research area both in modeling and development of optimization methods. Often the weight of truss structure is to be minimized subject to stress, displacement, and/or natural frequency constraints. This optimization task is in general difficult to solve because of non-linear constraints and non-convex feasible region. This means that the convergence of traditional gradient-based optimization methods cannot be ensured [1].

Metaheuristics such as genetic algorithms, particle swarm algorithms and evolution algorithms have been increasingly proposed as alternative techniques for optimization of truss structures [2]. Recently developed metaheuristics have shown good performance, for example, are: harmony search algorithm (SAHS) [3], teaching-learning-based optimization (TLBO) algorithm [4, 5], chaotic swarming of particle (CSP) algorithm [6], colliding bodies optimization (CBO) [7, 8], flower pollination algorithm (FPA) [9], water

cycle algorithm (WCA) [10] and mine blast algorithm (MBA) [10, 11], adaptive dimensional search (ADS) [12], and some new variants of the well-known differential evolution including adaptive differential evolution algorithm (ADEA) [1], improved constrained differential evolution using discrete variables (D-ICDE) [13], adaptive elitist differential evolution (aeDE) [14] and reliability-based improved constrained differential evolution (SORA-ICDE) [15].

The contribution of this paper is a new evolutionary-based algorithm which is applied to optimal sizing of truss structures. The formulation of the optimization problem and the constraint handling rules are presented in Section 2. In Section 3, the new algorithm is described in details. Validation and efficiency of the proposed algorithm in finding optimal structural optimization solutions are given in Section 5, in which three well-known truss structures have been examined and the results have been compared with some state-of-the-art metaheuristics. Conclusions are presented in Section 6.

2. TRUSS SIZING OPTIMIZATION

2.1. Problem formulation

In sizing design optimization of truss structures the goal is to find a minimum weight by selecting the cross-sectional areas of structural members such that the final design satisfies strength and serviceability requirements determined by standard design codes. The cross-section areas can be continuous values between the lower bound l_i and upper bound u_i or discrete values form a set S of P given values (often are the cross-sections according to production standards).

For a given truss structure, the objective is to find vector of cross-section areas for N_m members of the structure $\mathbf{A} = [A_1, A_2, \ldots, A_{N_m}]$ which minimizes the following weight objective function

$$W = \sum_{i=1}^{N_m} \rho_i L_i A_i \,, \tag{1}$$

subjected to N_c design constraints $g_j \leq g_{0,j},\ j = 1, \ldots, N_c$, where ρ_i and L_i are the material density and length of the i-th member, respectively; g_j are N_c constraint functions (displacement or stress in this paper) and $g_{0,j}$ are allowable values of g_j.

In this paper, the constraints are modified from the conventional form of constrained optimization as Eq. (2) and constraint violation is determined by Eq. (3)

$$c_j = \frac{g_j}{g_{0,j}} - 1 \leq 0, \tag{2}$$

$$C = \sum_{j}^{N_c} \max\{0, c_j\}. \tag{3}$$

2.2. Constraint handling

To solve the above constrained optimization problem, the constraints and the objective function are treated separately. The following rules are applied:

1. A feasible solution is better than any infeasible one.

2. Between two feasible solutions or two solutions with equal constraint violation, the one having smaller objective function value is better.

3. Between two infeasible solutions, the one having smaller constraint violation is better.

These rules were originally suggested by Deb [16] to handle the constrained problem for genetic algorithm, which has been shown successful for other metaheuristics. For handling bound constraints, cutting-off technique [17] is adopted, i.e. the generated violating value is substituted by the bound value, since in many cases the optimum solution is located at one of the bound of a given design variable.

3. PROPOSED EVOLUTIONARY-BASED OPTIMIZATION ALGORITHM

Like many other algorithms, the proposed method also uses a population P of NP vectors (individuals) of N_d design variables $\mathbf{x}_k = [x_{k,1}, x_{k,2}, \ldots, x_{k,N_d}]$ $(k = 1, 2, \ldots, NP)$. The population is then restructured by survival individuals evolutionally. The initial population is generated as

$$x_{k,i} = l_i + rand[0,1].(u_i - l_i), \quad i = 1, \ldots, N_d, \tag{4}$$

where $rand[0,1]$ is a uniformly distributed random real value in the range [0,1].

At the current generation, each individual $\mathbf{x}_k^{current}$ of the population is updated using either a global move or a local move. For $\mathbf{x}_k^{current}$ among $p \times NP$ random solutions $(1/NP < p < 1)$, the updating Eq. (5) (global step) is applied,

$$\mathbf{x}_k^{new} = \mathbf{x}_k^{current} + \begin{cases} \mathbf{r}(\mathbf{x}_{best} - \mathbf{x}_k^{current}), & \text{if } \mathbf{x}_k^{current} \neq \mathbf{x}_{best} \\ (\mathbf{r}_1 - \mathbf{r}_2)\mathbf{x}_{best}, & \text{if } \mathbf{x}_k^{current} \equiv \mathbf{x}_{best} \end{cases} \tag{5}$$

otherwise the updating Eq. (6) (local step) is used.

$$\mathbf{x}_k^{new} = \mathbf{x}_k^{current} + \begin{cases} \mathbf{r}(\mathbf{x}_l - \mathbf{x}_m), & \text{if } \mathbf{x}_l \text{ better than } \mathbf{x}_m \\ \mathbf{r}(\mathbf{x}_m - \mathbf{x}_l), & \text{otherwise} \end{cases} \tag{6}$$

In Eq. (5) and Eq. (6), $\mathbf{x}_{best}$ is the best found solution in the current population, $\mathbf{x}_l$ and $\mathbf{x}_m$ are two different individuals randomly chosen in the population $(1 \leq l \neq m \leq NP)$, $\mathbf{r}$, $\mathbf{r}_1$, and $\mathbf{r}_2$ are vectors of N_d uniformly distributed random numbers within the range [0, 1]. The global move allows a solution to take a random step toward the current best solution, whereas the local move allows a solution move around it current location.

If the updated solution $\mathbf{x}_k^{new}$ is better than $\mathbf{x}_k^{current}$, then $\mathbf{x}_k^{new}$ becomes the k-th member in P of the next generation; otherwise, the old individual $\mathbf{x}_k^{current}$ is retained. The search proceeds until either a convergence criterion is met or the maximum number of iteration/function evaluations is reached. The pseudo code of the proposed algorithm is given in Fig. 1.

In the comparison between the two solutions, based on the constraint handling rules in Section 2.2, the following are taken place:

1. If $\mathbf{x}_k^{current}$ is infeasible: $\mathbf{x}_k^{new}$ is better than $\mathbf{x}_k^{current}$ when $\mathbf{x}_k^{new}$ has smaller constraint violation. Therefore, evaluation of objective function is not necessary.

```
Initialize a population of NP random solutions
Evaluate objective function and constraint violation of each solution
Set the number of top-ranked solution (or set p)
while termination condition not satisfied
    Select pNP random solutions in the population
    for k = 1 to NP do
        if x_k^current is among pNP selected solutions
            Generate new solution x_k^new by global move via Eq. (5)
        else
            Generate new solution x_k^new by local move via Eq. (6)
        end if
        If the new solution is better, update it in the next generation
    end for
end while
```

Fig. 1. Pseudo code of the proposed algorithm

2. If $\mathbf{x}_k^{current}$ is feasible: $\mathbf{x}_k^{new}$ is better than $\mathbf{x}_k^{current}$ when $\mathbf{x}_k^{new}$ is feasible and has smaller objective function value. Thus, in case $\mathbf{x}_k^{new}$ has equal or larger objective function value, the evaluation of constraint violation of $\mathbf{x}_k^{new}$ is not necessary.

To adapt the algorithm for problems with discrete variables, the rounding technique can be applied, i.e. each value of a design variable vector, right after created either at initial state or later in the updating, is rounded to the nearest value in the discrete value list.

4. TEST PROBLEMS

Three well-known test problems of truss sizing optimization are employed in this paper. The design problems were assigned to minimize the weight of truss subjected to stress and displacement constraints. The design variables are cross-section areas of the truss members which are chosen from a given list of discrete values.

4.1. Ten-bar planar truss

The truss layout is illustrated in Fig. 2. The structure is subjected to load $P = -100$ kips at node 2 and node 4. The design variables are the bar element cross-section areas. The cross-section areas (in^2) are chosen from the list: 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, 11.5, 13.5, 13.9, 14.2, 15.5, 16.0, 16.9, 18.8, 19.9, 22.0, 22.9, 26.5, 30.0, 33.5. The modulus of elasticity and material density are 10^4 ksi and 0.1 lb/in^3. The allowable stress is 25 ksi for both tension and compression stress in each member. The allowable displacement of nodes in x and y directions is 2 in.

4.2. Twenty-five-bar space truss

The truss layout is illustrated in Fig. 3. The material density equals to 0.1 lb/in^3 and modulus of elasticity equals to 10^4 ksi. The cross-section areas divided into eight

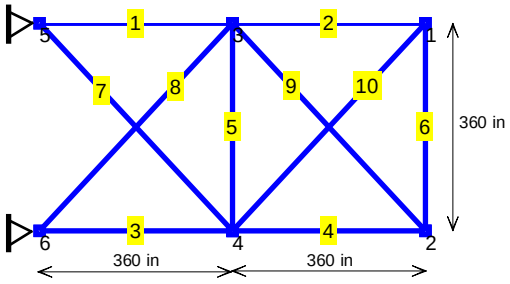


Fig. 2. Ten-bar truss layout

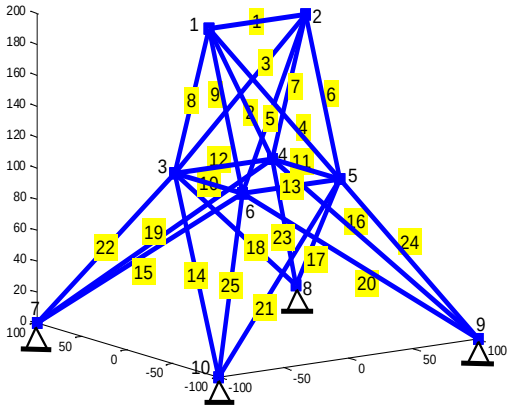


Fig. 3. Twenty-five-bar truss layout

member groups. The displacement constraints require that the maximum displacements at nodes 1 and 2 be limited to 0.35 in, in both the x and y directions. The constraints for stress are 40 ksi for both tension and compression stress in each member. The loading data is listed in Tab. 1. The cross-section areas (in^2) are chosen from the list: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2, 3.4.

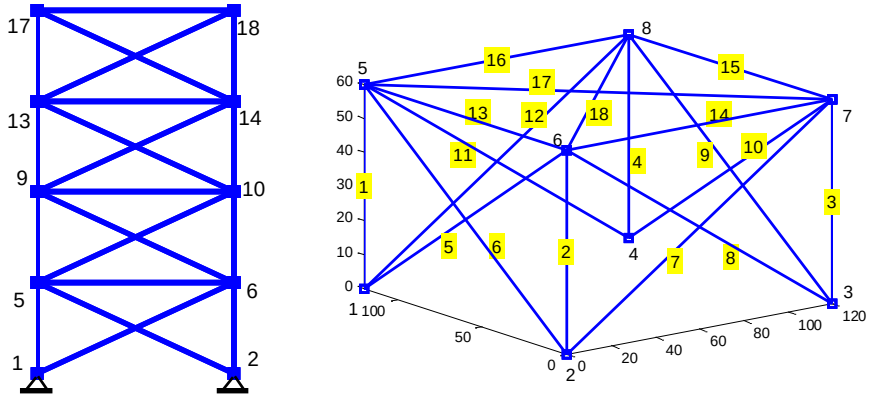
Table 1. Loading conditions for 25-bar truss

Condition	Node	F_x (kips)	F_y (kips)	F_z (kips)
1	1	1	-10	-10
	2	0	-10	-10
	3	0.5		
	6	0.6		

4.3. Seventy-two-bar space truss

The truss layout is depicted in Fig. 4. The cross-section areas are divided into sixteen member groups and are chosen from the list of 25 options: $0.1, 0.2, \ldots, 2.5$ in^2. The constraints involve a maximum allowable displacement of 0.25 in at the nodes 1 to 16 along the x and y directions, and a maximum allowable stress in each member of 25 ksi. The density of the material is 0.1 lb/in^{3}and the modulus of elasticity is equal to 10^4ksi. Two load cases are given in Tab. 2.

The proposed algorithm was used to solve each problem with 20 optimization runs. Finite-element method was applied to calculate stress in truss members and displacements at nodes. The algorithm and analysis program are implement in MATLAB 7 R2012a environment. The parameter setting in the considered optimization problems is: the population size $NP = 30$, the rate of global search $p = 0.2$, i.e. using 6 solutions



(a) (b)

Fig. 4. 72-bar truss layout: (a) numbering scheme; (b) element numbering for first story

Table 2. Loading conditions for 72-bar truss

Load case	Node	F_x (kips)	F_y (kips)	F_z (kips)
1	17	5	5	-5
2	17	0	0	-5
	18	0	0	-5
	19	0	0	-5
	20	0	0	-5

for global search. The termination condition is when the number of iterations exceeds a predefined value, which is 100 for the ten-bar truss and the 25-bar truss problems, and 200 for the 72-bar truss problem.

5. RESULTS

5.1. Ten-bar truss

Optimization results obtained by the proposed algorithm are compared with some most recent metaheuristics, including MBA [11], TLBO [5], ADS [12] and aeDE [14]. The statistical results of optimal weight are given in Tab. 3, including the minimum, the median, the mean, the maximum values and the standard deviation. The proposed algorithm results in a minimum design weight of 5490.74 lb for the truss, which is the same as results by aeDE, ADS and TLBO and lighter than that of MBA. On average of 20 runs, the mean weight of the proposed algorithm is 5504.0206 lb, which is better than those of ADS and MBA and as good as that of TLBO and aeDE. More importantly, the proposed algorithm exhibits improved computational efficiency when compared to aeDE, TLBO and MBA. The average number of structural analyses required by the proposed

algorithm to converge is 1669. In the best run, the proposed algorithm can find the minimum designed weight (5490.7379 lb) within 1377 truss analyses. In 100 iterations, there are on average approximately 1893 actual calls for constraint evaluation, i.e. about 1107 structural analyses are skipped.

Table 3. Comparison on optimal weights (lb) of 10-bar truss

Size of members (in^2)	This paper	aeDE [14]	ADS [12]	TLBO [5]	MBA [11]
1	33.5	33.5	33.5	33.5	30.00
2	1.62	1.62	1.62	1.62	1.62
3	22.9	22.9	22.9	22.9	22.90
4	14.2	14.2	14.2	14.2	16.90
5	1.62	1.62	1.62	1.62	1.62
6	1.62	1.62	1.62	1.62	1.62
7	7.97	7.97	7.97	7.97	7.97
8	22.9	22.9	22.9	22.9	22.90
9	22	22	22	22	22.90
10	1.62	1.62	1.62	1.62	1.62
Min. weight (lb)	5490.7379	5490.7379	5490.74	5490.74	5507.758
Median weight (lb)	5494.4676	-	-	-	-
Mean weight (lb)	5504.0206	5502.623	5539.97	5503.21	5527.296
Max. weight (lb)	5538.3529	5549.204	5591.43	-	5536.965
Standard deviation	16.0528	20.780	35.86	20.33	11.38
Number of analyses	1377	2380	1000	5183	3600

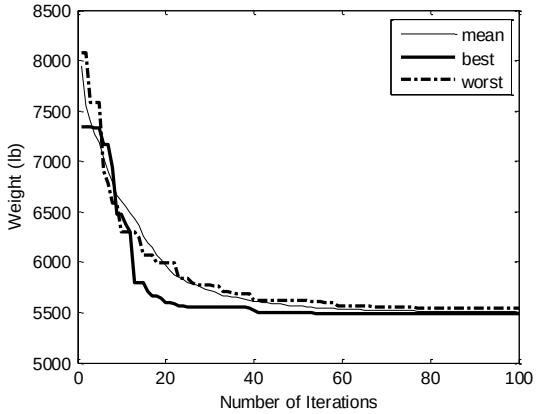


Fig. 5. Typical convergence history of ten-bar truss

Fig. 5 shows the typical convergence history of the proposed algorithm for this truss design problem, including the mean of all runs, the best run and the worst run.

5.2. Twenty-five-bar truss

Tab. 4 gives the statistical results of optimum weight of the 25-bar truss by the proposed algorithm over 20 runs. The best truss design developed by the proposed algorithm weighs 484.3286 lb, which is lighter than the design by aeDE [14], MBA [11] and TLBO [5]. The proposed algorithm also uses a significantly smaller number of structural analyses than MBA and TLBO. The proposed algorithm requires an average of approximately 1469 structural analyses to converge to a design, while values reported by TLBO and MBA are 4910 and 2150, respectively. In the best run, the proposed algorithm can find the minimum designed weight within 998 truss analyses. In 100 iterations, the algorithm skips about 1159 constraint evaluations.

Table 4. Comparison on optimal weights (lb) of 25-bar truss

Size of members (in^2)	This paper	aeDE [14]	TLBO [5]	MBA [11]
1	0.1	0.1	0.1	0.1
2-5	0.4	0.3	0.3	0.3
6-9	3.4	3.4	3.4	3.4
10-11	0.1	0.1	0.1	0.1
12-13	2.2	2.1	2.1	2.1
14-17	1	1	1	1
18-21	0.4	0.5	0.5	0.5
2-25	3.4	3.4	3.4	3.4
Min. weight (lb)	484.3286	484.854	484.854	484.854
Median weight (lb)	484.3286	-	-	-
Mean weight (lb)	484.4075	485.014	484.91	484.885
Max. weight (lb)	485.3797	486.100	-	485.048
Standard deviation	0.2572	0.273	0.17	7.2E-02
Number of analyses	998	1440	4910	2150

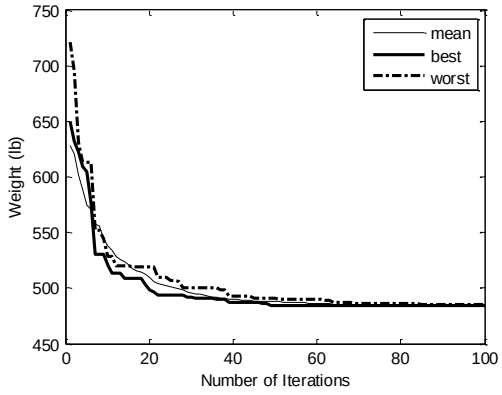


Fig. 6. Typical convergence history of twenty-five-bar truss

Fig. 6 shows the typical convergence history of the proposed algorithm for this truss design problem, including the mean of all runs, the best run and the worst run.

5.3. Seventy-two-bar truss

The statistical optimization results of the proposed algorithm for 72-bar truss are given in Tab. 5, together with results reported by CBO [8], improved MBA [10], WCA [10] and improved magnetic charged system search (IMCSS) [18]. The best design obtained by the proposed algorithm is 385.5427 lb, which is identical to the results given by the other methods. However, the proposed algorithm has mean optimum weight of 386.5024 lb, which is a little bit higher than that of WCA and IMBA. The proposed algorithm is also more computationally efficient than the other algorithms. On average, approximately 2613 structural analyses are required by the proposed algorithm to converge and the best run requires 2158 structural analyses. The average number of structural analyses skipped in 200 iterations is 2767.

Table 5. Comparison on optimal weights (lb) of 72-bar truss

Size of members (in^2)	This paper	CBO [8]	IMCSS [18]	IMBA [10]	WCA [10]
1-4	1.9	1.9	2	1.9	1.9
5-12	0.5	0.5	0.5	0.5	0.5
13-16	0.1	0.1	0.1	0.1	0.1
17-18	0.1	0.1	0.1	0.1	0.1
19-22	1.4	1.4	1.3	1.4	1.4
23-30	0.5	0.5	0.5	0.5	0.5
31-34	0.1	0.1	0.1	0.1	0.1
35-36	0.1	0.1	0.1	0.1	0.1
37-40	0.5	0.5	0.5	0.5	0.5
41-48	0.5	0.5	0.5	0.5	0.5
49-52	0.1	0.1	0.1	0.1	0.1
53-54	0.1	0.1	0.1	0.1	0.1
55-58	0.2	0.2	0.2	0.2	0.2
59- 66	0.6	0.6	0.6	0.6	0.6
67-70	0.4	0.4	0.4	0.4	0.4
71-72	0.6	0.6	0.6	0.6	0.6
Min. weight (lb)	385.5427	385.54	385.54	385.542	385.542
Median weight (lb)	386.5368	-	-	-	-
Mean weight (lb)	386.5024	401	-	385.765	385.842
Max. weight (lb)	387.9427	460.98	-	387.942	386.800
Standard deviation	0.9965	16.99	-	0.41	0.55
Number of analyses	2158	4500	3625	5750	3200

Fig. 7 shows the typical convergence history of the proposed algorithm for this truss design problem, including the mean of all runs, the best run and the worst run.

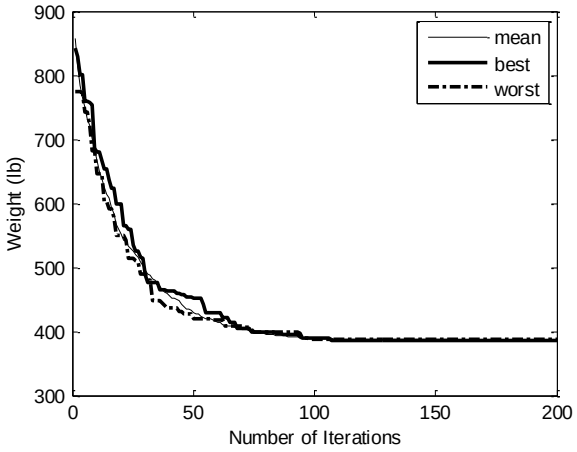


Fig. 7. Typical convergence history of seventy-two-bar truss

6. CONCLUSION

A new evolutionary-based optimization is introduced and applied for optimal sizing of truss structures. Computational results obtained from some benchmark truss optimization problems illustrate the efficiency of the proposed algorithm. The convergence rate and computational cost (in terms of structural analyses) are also very competitive with some state-of-the-art metaheuristics for truss sizing found in literature. The potential of the proposed algorithm in solving more complex and large-scale optimization problems will be explored in further study.

REFERENCES

[1] S. Bureerat and N. Pholdee. Optimal truss sizing using an adaptive differential evolution algorithm. *Journal of Computing in Civil Engineering*, **30**, (2), (2015). doi:10.1061/(ASCE)CP.1943-5487.0000487.

[2] M. Stolpe. Truss optimization with discrete design variables: A critical review. *Structural and Multidisciplinary Optimization*, **53**, (2), (2016), pp. 349–374. doi:10.1007/s00158-015-1333-x.

[3] S. O. Degertekin. Improved harmony search algorithms for sizing optimization of truss structures. *Computers & Structures*, **92**, (2012), pp. 229–241. doi:10.1016/j.compstruc.2011.10.022.

[4] S. O. Degertekin and M. S. Hayalioglu. Sizing truss structures using teaching-learning-based optimization. *Computers & Structures*, **119**, (2013), pp. 177–188. doi:10.1016/j.compstruc.2012.12.011.

[5] C. V. Camp and M. Farshchin. Design of space trusses using modified teaching-learning based optimization. *Engineering Structures*, **62**, (2014), pp. 87–97. doi:10.1016/j.engstruct.2014.01.020.

[6] A. Kaveh, R. Sheikholeslami, S. Talatahari, and M. Keshvari-Ilkhichi. Chaotic swarming of particles: A new method for size optimization of truss structures. *Advances in Engineering Software*, **67**, (2014), pp. 136–147. doi:10.1016/j.advengsoft.2013.09.006.

[7] A. Kaveh and V. R. Mahdavi. Colliding bodies optimization method for optimum design of truss structures with continuous variables. *Advances in Engineering Software*, **70**, (2014), pp. 1–12. doi:10.1016/j.advengsoft.2014.01.002.

[8] A. Kaveh and V. R. Mahdavi. Colliding bodies optimization method for optimum discrete design of truss structures. *Computers & Structures*, **139**, (2014), pp. 43–53. doi:10.1016/j.compstruc.2014.04.006.

[9] G. Bekdaş, S. M. Nigdeli, and X.-S. Yang. Sizing optimization of truss structures using flower pollination algorithm. *Applied Soft Computing*, **37**, (2015), pp. 322–331. doi:10.1016/j.asoc.2015.08.037.

[10] A. Sadollah, H. Eskandar, A. Bahreininejad, and J. H. Kim. Water cycle, mine blast and improved mine blast algorithms for discrete sizing optimization of truss structures. *Computers & Structures*, **149**, (2015), pp. 1–16. doi:10.1016/j.compstruc.2014.12.003.

[11] A. Sadollah, A. Bahreininejad, H. Eskandar, and M. Hamdi. Mine blast algorithm for optimization of truss structures with discrete variables. *Computers & Structures*, **102**, (2012), pp. 49–63. doi:10.1016/j.compstruc.2012.03.013.

[12] O. Hasançebi and S. K. Azad. Adaptive dimensional search: a new metaheuristic algorithm for discrete truss sizing optimization. *Computers & Structures*, **154**, (2015), pp. 1–16. doi:10.1016/j.compstruc.2015.03.014.

[13] V. Ho-Huu, T. Nguyen-Thoi, M. Nguyen-Thoi, and L. Le-Anh. An improved constrained differential evolution using discrete variables (D-ICDE) for layout optimization of truss structures. *Expert Systems with Applications*, **42**, (20), (2015), pp. 7057–7069. doi:10.1016/j.eswa.2015.04.072.

[14] V. Ho-Huu, T. Nguyen-Thoi, T. Vo-Duy, and T. Nguyen-Trang. An adaptive elitist differential evolution for optimization of truss structures with discrete design variables. *Computers & Structures*, **165**, (2016), pp. 59–75. doi:10.1016/j.compstruc.2015.11.014.

[15] V. Ho-Huu, T. Nguyen-Thoi, L. Le-Anh, and T. Nguyen-Trang. An effective reliability-based improved constrained differential evolution for reliability-based design optimization of truss structures. *Advances in Engineering Software*, **92**, (2016), pp. 48–56. doi:10.1016/j.advengsoft.2015.11.001.

[16] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, **186**, (2), (2000), pp. 311–338. doi:10.1016/S0045-7825(99)00389-8.

[17] G. C. Onwubolu. Differential evolution for the flow shop scheduling problem. In *New Optimization Techniques in Engineering*, pp. 585–611. Springer, (2004). doi:10.1007/978-3-540-39930-8_24.

[18] A. Kaveh, B. Mirzaei, and A. Jafarvand. An improved magnetic charged system search for optimization of truss structures with continuous and discrete variables. *Applied Soft Computing*, **28**, (2015), pp. 400–410. doi:10.1016/j.asoc.2014.11.056.