TRUSS SIZING OPTIMIZATION USING ENHANCED DIFFERENTIAL EVOLUTION WITH OPPOSITION-BASED MUTATION AND NEAREST NEIGHBOR COMPARISON

Pham Hoang Anh[1*]

***Summary**: An optimization algorithm based on differential evolution (DE) is presented for optimal truss sizing design. The algorithm applies a simple opposition-based mutation scheme and the so-called nearest neighbor comparison method to the classical DE. The opposition-based mutation can accelerate the convergence, while the nearest neighbor comparison, which uses neighborhood information to judge the order relation between two solution points, can omit an unfavorable solution without evaluating it. Four well-known truss sizing problems with continuous variables are used to examine the performance of the proposed algorithm. The results show that the new DE algorithm can effectively obtain high quality solutions and it is competitive to some modern metaheuristic algorithms in the literature.*

***Keywords**: Truss sizing optimization; differential evolution; opposition-based method; nearest neighbor comparison.*

Received: September 9th, 2016, revised: September 23th, 2016, accepted: October 13th, 2016

1. Introduction

Truss optimization is one of the most popular design problems and has been an extensive research area both in modeling and in development of optimization methods. Often the weight of truss structure is minimized while subjected to stress and/or displacement 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.

Metaheuristics such as genetic algorithms, particle swarm algorithms and evolution algorithms have been increasingly proposed as alternative techniques for optimization of truss structures [1]. Some recent methods which have shown good performance are harmony search algorithm [2], teaching-learning-based optimization algorithm [3, 4], chaotic swarming of particle algorithm [5], colliding bodies optimization [6, 7], flower pollination algorithm [8]. As a common issue in metaheuristics, most of these techniques, however, require a high number of structural analyses to reach a near optimum. It will be computationally demanding for large-scale problems.

In this paper, a metaheuristic algorithm based on differential evolution (DE) is presented for truss sizing problem. The motivation of using DE is that DE has simple structure, requires few control parameters and is shown very efficient for various kind of optimization problem [9]. DE has also been utilized successfully for truss optimization. Wang et al. [10] reported a very first study for optimization of truss with continuous and discrete variables by DE. Wu and Tseng [11] proposed a multi-population differential evolution with a penalty-based, self-adaptive strategy for optimization of the size, topology and shape of truss structures subjected to allowable stress, deflection and kinematic stability constraints. Silva et al. [12] used different DE variants in a dynamic manner, applied an adaptive constraint handling technique and proposed the DUVDE algorithm for engineering design, including application to optimal sizing of truss structures. Krempser et al. [13, 14] introduced the SMDE, which is the combination of surrogate models and DE for sizing optimization of truss. Recently, some new variants of DE have been presented for truss optimization, including discrete variable handling technique combined with the improved $(\mu+\lambda)$ - constrained differential evolution [15], adaptive differential evolution algorithm [16], adaptive

[1] *Dr, Faculty of Building and Industrial Construction. National University of Civil Engineering (NUCE).*
** Corresponding author. E-mail: anhpham.nuce@gmail.com.*

elitist differential evolution [17] and reliability-based improved constrained differential evolution [18]. Noticeably, modifications can enhance the performance of DE in terms of both optimum solution found and number of function evaluations.

This paper introduces simple techniques to improve the performance of DE, including an opposition-based mutation and a nearest neighbor comparison method and presents an opposition-based DE with nearest neighbor comparison (ODE-NNC). The strategy is to use the order relation between design variable vectors to guide DE toward more favorable region and to skip unnecessary function evaluations. The proposed algorithm is examined with four classical truss sizing design problems with continuous variables. The optimization results are compared with those of some modern metaheuristics.

The organization of the rest of this paper is as follows. In section 2, the optimization and the constraint handling rules are presented. The new ODE-NNC algorithm is described in section 3. Section 4 gives details of the test problems and numerical results are shown and discussed in section 5. Conclusions are given in section 6.

2. Truss optimization problem

A truss optimization problem is typically formulated as:

$$\begin{aligned} &\text{Minimize } f(\mathbf{x}), \quad \mathbf{x}=[x_1,x_2,\cdots,x_n] \\ &\text{subjected to } g_j(\mathbf{x}) \le g_{0,j},\ j=1,\cdots,m \\ &\qquad\qquad l_i \le x_i \le u_i \end{aligned} \tag{1}$$

where x is a n dimension vector of design variables, $f(\mathbf{x})$ is objective function (the weight of truss in this paper), $g_j(\mathbf{x})$ are m constraint functions (displacement and/or stress in this paper) and $g_{0,j}$ are allowable values of $g_j(\mathbf{x})$. Values l_i and u_i are the lower bound and upper bound of x_i, respectively.

In order to facilitate the new techniques proposed in this paper, the constraints are rewritten as in Eq. (2) and constraint violation is determined by Eq. (3):

$$c_j(\mathbf{x}) = g_j(\mathbf{x}) - g_{0,j} \le 0 \tag{2}$$

$$C(\mathbf{x}) = \sum_{j}^{m} \max\{0, c_j(\mathbf{x})\} \tag{3}$$

To solve the above constrained optimization problem, in this study the Deb's constraint handling rules [19] are applied. Of two solutions x_1 and x_2:

$$\mathbf{x}_1 \text{ is better than } \mathbf{x}_2 \text{ if } \begin{cases} f(\mathbf{x}_1) < f(\mathbf{x}_2), \text{for } C(\mathbf{x}_1) = C(\mathbf{x}_2) \\ C(\mathbf{x}_1) < C(\mathbf{x}_2), \text{for } C(\mathbf{x}_1) \ne C(\mathbf{x}_2) \end{cases} \tag{4}$$

where $f(\cdot)$ and $C(\cdot)$ are the objective function values and the constraint violation of a solution, respectively. Eq. (4) is equivalent to the lexicographic orders in which the constraint violation precedes the function value.

For handling bound constraints, cutting-off technique [20] 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 bounds of a given design variable.

3. Enhanced differential evolution

3.1 Differential evolution

Differential evolution (DE), which is introduced by Storn and Price in 1995, is a population-based optimizer. DE uses a population of NP vectors (individuals) $\mathbf{x}_k (k=1,2,...NP)$ of the design variables. The population is then restructured by survival individuals evolutionally. The initial population is generated as

$$x_{k,i} = l_i + rand[0,1].(u_i - l_i), \quad i = 1,...,n \tag{5}$$

where *rand* [0,1] is a uniformly distributed random real value in the range [0,1]. Each individual x_k of the current population is compared with a trial individual generated by the operations of “mutation” and “crossover”. The better one will be selected as the member for the next population. The basic procedure (DE/rand/1/bin) is as follows.

For each individual x_k of the current population,

Step 1: “Mutation” is performed and a mutant vector **y** is given by

$$\mathbf{y} = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) \tag{6}$$

with r_1, r_2, r_3 are randomly chosen integers and $1 \le r_1 \ne r_2 \ne r_3 \ne k \le NP$; F is a real and constant factor usually chosen in the interval $[0,1]$, which controls the amplification of the differential variation ($\mathbf{x}_{r_2}$ - $\mathbf{x}_{r_3}$). In Eq. (6), $\mathbf{x}_{r_1}$ is called the base vector, while the others are called the difference vectors.

Step 2: "Crossover" with $\mathbf{x}_k$ is then introduced to increase the diversity. The trial vector $\mathbf{z}$ are determined by:

$$z_i = \begin{cases} y_i & \text{if } (rand[0,1] \le Cr) \text{ or } (r = i) \\ x_{k,i} & \text{if } (rand\,[0,1] > Cr) \text{ and } (r \ne i) \end{cases} \tag{7}$$

Here, r is randomly chosen integer in the interval $[1,n]$; Cr is user-defined crossover constant in [0, 1].

Step 3: "Selection": the vector $\mathbf{z}$ is compared with $\mathbf{x}_k$. If $\mathbf{z}$ is better than $\mathbf{x}_k$, then $\mathbf{z}$ becomes a member in the population of the next generation; otherwise, $\mathbf{x}_k$ is retained.

Traditionally, the evolution in DE bases on random mutation, i.e. the base and difference vectors for mutation are randomly selected from the current population. Thus, the beneficial information of the population cannot be fully exploited to guide the search of DE through mutation [21]. Moreover, every new trial solution is evaluated and many of them do not survive in selection. The evaluation of such trial solutions is useless and should be avoided.

To enhance the performance of DE, the present paper introduces a simple opposition-based mutation scheme and the so-called nearest neighbor comparison, which can increase the convergence and reduce possibly useless function evaluations.

3.2 Opposition-based mutation

The scale difference is determined based on the order relation of the two difference vectors, which has the same concept of the well-known opposition based method presented for improving DE search performance in literatures [22, 23]. By that, the mutant vector is created as following:

$$\mathbf{y} = \mathbf{x}_{pbest} + \begin{cases} F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}), & \text{if } \mathbf{x}_{r_2} \text{better than } \mathbf{x}_{r_3}, \\ F(\mathbf{x}_{r_3} - \mathbf{x}_{r_2}), & \text{otherwise} \end{cases} \tag{8}$$

This mutation guarantees that the scaled difference vector directs toward the better vector. Thus, there is 50% chance of improving a solution. In the conventional DE, the probability to improve the solution cannot be figured depending on the location of the base vector.

In addition, the base vector is selected randomly from the whole population as traditional DE only in the early generations. At later generations when all the solutions in the population are feasible, a base vector, x_{pbest}, is selected randomly among $p \times NP$ ($p \in (0,1]$) top-ranked solutions, where the rank is based on the fitness value. The concept of using several top-ranked solutions in mutation was introduced in JADE by Zhang and Sanderson [24] and successfully implemented in some other DE variants, such as LMDE [25] and ADEA [16]. The difference of the proposed DE in this paper is the use of random base vector at the beginning to maintain the diversity of the population and prevent premature convergence.

3.3 Nearest neighbor comparison

The idea is to use the nearest neighbor solution in the search population to judge whether a trial solution is worth evaluating, so that useless evaluation can be avoided. A trial vector $\mathbf{z}$ obtained after crossover by Eq. (7) is skipped when its nearest neighbor $\mathbf{x}_{nei}$ is worse than the compared vector. This nearest neighbor vector of the trial vector is searched in the current population using normalized distance measure:

$$d(\mathbf{x}, \mathbf{z}) = \sqrt{\sum_{i=1}^{n} \left(\frac{x_i - z_i}{\max\limits_k x_{k,i} - \min\limits_k x_{k,i}} \right)^2} \tag{9}$$

where $d(\mathbf{x},\mathbf{z})$ is distance between two vectors $\mathbf{x}$ and $\mathbf{z}$. The search will move to the next parent vector in the population. Otherwise, $\mathbf{z}$ is evaluated and selection (as in step 3 in section 3.1) is carried out.

The nearest neighbor comparison has been recently introduced by Pham [26] for unconstrained optimization problems. Using this method, trial vectors which are likely worse than the target vector will be often omitted. Thus, it is expected that useless function evaluation will be reduced during the searching process.

4. Test problems

Four classical test problems of truss sizing optimization widely used in literature are employed in this paper. The design problems are to minimize the weight of truss subjected to stress and displacement constraints. The design variables are cross-section areas of the truss members. The descriptions of the problems are given in below.

Ten-bar planar truss

The truss layout is illustrated in Fig. 1. The structure is subjected to a vertical load P=-100 kips at node 2 and node 4. The design variables are the bar cross-section areas in the range [0.1, 40] in^2. The modulus of elasticity and material density are 10^4 ksi and 0.1 lb/in^3, respectively. The allowable stress is 25 ksi for both tension and compression in each member. The allowable displacement of nodes in x and y directions is 2 in.

Twenty-five-bar space truss

The truss layout is illustrated in Fig. 2. The material density equals to 0.1 lb/in^3 and the modulus of elasticity equals to 10^4 ksi. The cross-section areas are categorized into eight member as shown in Fig. 2. The displacement constraints require that the maximum displacements at nodes 1 and 2 be limited within 0.35 in, in both the x and y directions. The constraints for stress are listed in Table 1. The loading data are listed in Table 2. The cross-section areas are in the interval [0.01, 3.5] in^2.

Seventy-two-bar space truss

The truss layout is depicted in Fig. 3. The cross-section areas are categorized into sixteen groups as shown in Fig. 3 and have the minimum value of 0.1 in^2. The constraints involve a maximum allowable displacement of 0.25 in at nodes from 5 to 20 along the x and y directions, and a 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 10^4 ksi. Two load cases are given in Table 3.

Two hundred-bar plane truss

The truss structure layout is shown on the Fig. 4. The structure is subjected to three loading conditions: (1) 1.0 kip acting in the positive x-direction at nodes 1, 6, 15 20, 29, 34, 43, 48, 57, 62 and 71; (2) 10.0 kip acting in the negative y-direction at nodes 1-6, 8, 10, 12, 14, 16-20, 22, 24, 26, 28-34, 36, 38, 40, 42-48, 50, 52, 54, 56-62, 64, 66, 68, and 70-75; and (3) conditions 1 and 2 acting together. The design variables include all bar cross-section areas, which are categorized into 29 groups as showed in Fig. 4. Material density and modulus of elasticity are 0.283 lb/in^3 and 30000 ksi, respectively. The sizing design problem is to minimize structural mass subject to stress constraints (allowable stress of 10 lb/in^2). The cross-section areas are in the interval [0.1, 15] in^2.

Table 1. *Member groups for 25-bar truss and allowable stress values*

Group	Member	Allowable tensile stress	Allowable compressive stress
1	1	40.0	35.092
2	2, 3, 4, 5	40.0	11.590
3	6, 7, 8, 9	40.0	17.305
4	10, 11	40.0	35.092
5	12, 13	40.0	35.092
6	14, 15, 16, 17	40.0	6.759
7	18, 19, 20, 21	40.0	6.959
8	22, 23, 24, 25	40.0	11.082

Table 2. *Loading conditions for 25-bar truss*

Condition	Node	F_x (kips)	F_y (kips)	F_z (kips)
1	1	0	20	-5
	2	0	-20	-5
2	1	1	10	-5
	2	0	10	-5
	3	0.5		
	6	0.5		

Table 3. *Loading conditions for 72-bar truss*

Load case	Node	F_x (kips)	F_y (kips)	F_z (kips)
1	1	5	5	-5
2	1	0	0	-5
	2	0	0	-5
	3	0	0	-5
	4	0	0	-5

The proposed ODE-NNC was used to solve each problem with 20 random runs. The parameter setting for DE in all tests is: F=0.5, Cr=0.9, NP=50. The rate of top-ranked solutions for selecting the base vector is p=0.2 [24-25] (ADEA [16] used p=0.1). Finite element method using two-node tension/compression linear element is applied to calculate the stresses and displacements. All codes are implemented in MATLAB R2012a.

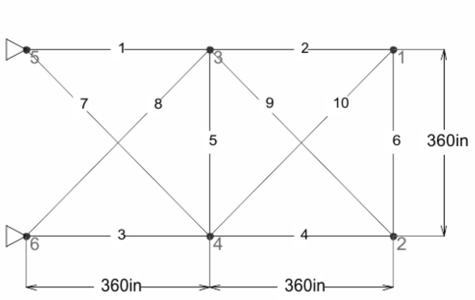

Figure 1. *Ten-bar truss layout*

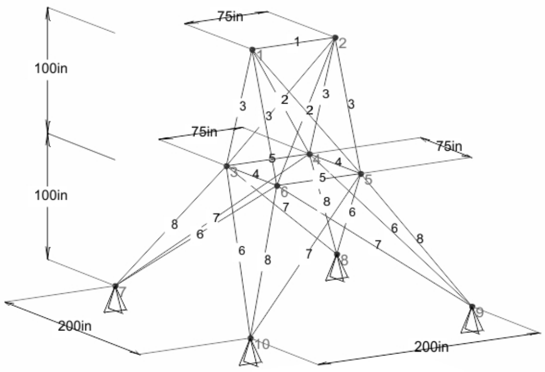

Figure 2. *Twenty-five-bar truss layout*

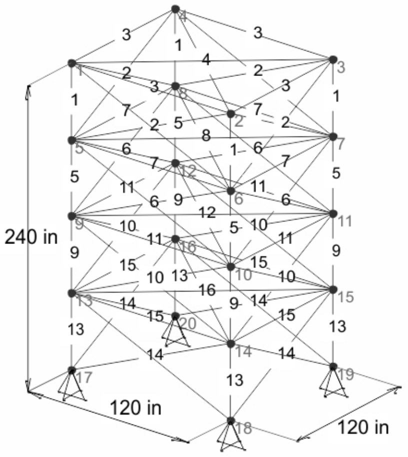

Figure 3. *Seventy-two-bar truss layout*

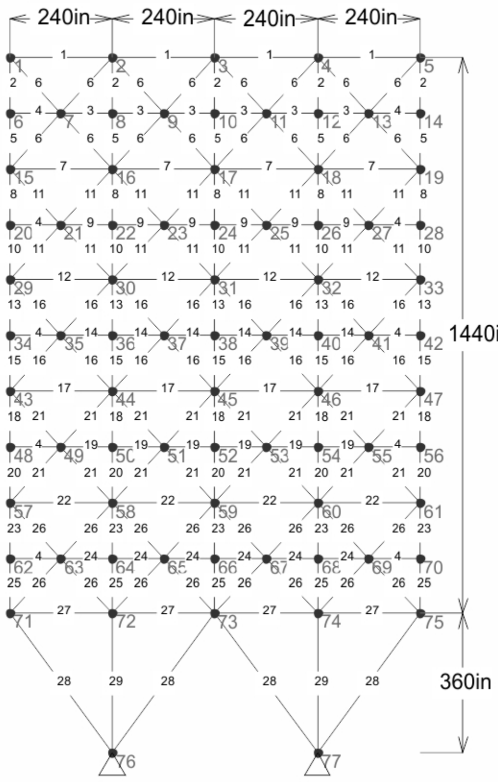

Figure 4. *Two-hundred-bar truss layout*

5. Results

10-bar truss

The proposed DE is compared with SAHS [2], TLBO [4], and ADEA [16]. The statistical results of optimal weight are given in Table 4, including the best weight, the average weight and the standard deviation of weight. In addition, the number of structural analyses is shown. It is seen that with lower number of function evaluations, the proposed DE obtained the best results.

Table 4. *Comparison on optimal weights (lb) of 10-bar truss*

Size of members (in^2)	ODE-NNC	SAHS [2]	TLBO [4]	ADEA [16]
1	30.53407525	30.394	30.6684	30.5139
2	0.1	0.100	0.1000	0.1000
3	23.21132872	23.098	23.1584	23.2052
4	15.22821542	15.491	15.2226	15.2084
5	0.1	0.100	0.1000	0.1000
6	0.552468879	0.529	0.5421	0.5318
7	7.456968561	7.488	7.4654	7.4585
8	21.03644835	21.189	21.0255	21.0512
9	21.50740940	21.342	21.4660	21.5391
10	0.1	0.100	0.1000	0.1000
Best weight (lb)	**5060.8568**	5061.42	5060.973	5060.8949
Average weight (lb)	**5060.8916**	5061.95	5064.808	5062.5965
Standard deviation	**0.03500**	0.71	6.3707	3.9401
Number of analyses	**7000**	7081	13767	10000

25-bar truss

Table 5 gives the statistical results after 20 runs. With respect to the average weight, ODE-NNC shows the best performance among the compared optimizers, which are CBO [6], CSP [5], TLBO [4], FPA [8] and ADEA [16]. The proposed algorithm also uses a lower number of structural analyses and it provides much smaller standard deviation of weight, i.e. it is more stable than the other algorithms. In addition, no constraint violation occurred for the best optimum result found (only TLBO and ADEA also produced feasible best optima, which are slightly higher than those obtained by ODE-NNC). The best solution by CBO, CSP and FPA violate the stress or displacement constraints.

Table 5. *Comparison on optimal weights (lb) of 25-bar truss*

Size of grouped members (in^2)	ODE-NNC	CBO [6]	CSP [5]	TLBO [4]	FPA [8]	ADEA [16]
1	0.01	0.0100	0.010	0.0100	0.0100	0.0100
2	1.9870825181	2.1297	1.910	1.9878	1.8308	5.6406
3	2.9934723860	2.8865	2.798	2.9914	3.1834	8.5941
4	0.01	0.0100	0.010	0.0102	0.0100	0.0100
5	0.01	0.0100	0.010	0.0100	0.0100	0.0100
6	0.6836859318	0.6792	0.708	0.6828	0.7017	1.9368
7	1.6768853783	1.6077	1.836	1.6775	1.7266	4.7857
8	2.6624969662	2.6927	2.645	2.6640	2.5713	7.5921
Best weight (lb)	545.16303235	**544.310**	545.09	545.175	545.159	545.1657
Average weight (lb)	**545.16487915**	545.256	545.20	545.483	545.730	545.2200
Standard deviation	**0.0025168864**	0.294	0.487	0.306	0.59	0.0730
Number of analyses	**5000**	9090	17500	12199	8149	10000

72-bar truss

The optimization results for 72-bar truss are given in Table 6. The proposed ODE-NNC shows better results and uses less function evaluations than 2D-CBO [7], CSP [5], TLBO [4], and ADEA [16]. Only FPA [8] has better optimal weight with smaller number of analyses, but its best optimal solution violates the displacement constraint about 0.2% (calculated from the given results). It is noted that the group order in this study is different from that in the other studies.

Table 6. *Comparison on optimal weights (lb) of 72-bar truss*

Size of grouped members (in^2)	ODE-NNC	2D-CBO [7]	CSP [5]	TLBO [4]	FPA [8]	ADEA [16]
1	0.156483260607825	1.892462	1.94459	1.8807	1.8758	1.8861
2	0.54481218991688	0.510027	0.50260	0.5142	0.5160	0.5231
3	0.40971397227306	0.100000	0.10000	0.1000	0.1000	0.1000
4	0.568839232361355	0.100000	0.10000	0.1000	0.1000	0.1000
5	0.525079174028263	1.266465	1.26757	1.2711	1.2993	1.2576
6	0.515172597743923	0.509992	0.50990	0.5151	0.5246	0.5043
7	0.100000127683359	0.100000	0.10000	0.1000	0.1001	0.1000
8	0.1	0.100000	0.10000	0.1000	0.1000	0.1000
9	1.26684996919847	0.524176	0.50674	0.5317	0.4971	0.5200
10	0.513608433102961	0.517540	0.51651	0.5134	0.5089	0.5235
11	0.1	0.100000	0.10752	0.1000	0.1000	0.1000
12	0.1	0.100000	0.10000	0.1000	0.1000	0.1000
13	1.88493087136548	0.156420	0.15618	0.1565	0.1575	0.1568
14	0.513976704396166	0.546158	0.54022	0.5429	0.5329	0.5394
15	0.100001034903325	0.414840	0.42229	0.4081	0.4089	0.4083
16	0.1	0.569529	0.57941	0.5733	0.5731	0.5734
Best weight (lb)	379.61746300	379.6217	379.97	379.632	**379.095**	379.6851
Average weight (lb)	379.64216812	379.6446	381.56	379.759	**379.534**	379.9205
Standard deviation	**0.0238141413**	0.251520	1.803	0.149	0.272	0.1842
Number of analyses	10000	12000	10500	21542	**9029**	10000

***Table 7.** Comparison on optimal weights (lb) of 200-bar truss*

Size of grouped members (in^2)	ODE-NNC	CSP [5]	TLBO [4]	FPA [8]	JADE [16]	ADEA [16]
1	0.1009407066	0.1480	0.146	0.1425	0.1093	0.1020
2	0.9713329039	0.9460	0.941	0.9637	1.2775	1.1193
3	0.1137715559	0.1010	0.100	0.1005	0.1344	0.1000
4	0.1048564579	0.1010	0.101	0.1000	0.1492	0.1223
5	1.9744803988	1.9461	1.941	1.9514	2.6144	1.9622
6	0.2279566799	0.2979	0.296	0.2957	0.3510	0.2693
7	0.1026704064	0.1010	0.100	0.1156	0.2125	0.1719
8	3.1194375266	3.1072	3.121	3.1133	4.1176	3.0690
9	0.1025627515	0.1010	0.100	0.1006	0.1012	0.1004
10	4.1244558843	4.1062	4.173	4.1100	5.4221	4.1509
11	0.3066377259	0.4049	0.401	0.4165	0.5540	0.4317
12	0.1	0.1944	0.181	0.1843	0.1159	0.2122
13	5.4036370804	5.4299	5.423	5.4567	7.2124	5.3974
14	0.1689822306	0.1010	0.100	0.1000	0.1067	0.1102
15	6.4082016517	6.4299	6.422	6.4559	8.5517	6.3959
16	0.4813379325	0.5755	0.571	0.5800	0.6902	0.6141
17	0.1493274843	0.1349	0.156	0.1547	0.6228	0.2666
18	7.9196312178	7.9747	7.958	8.0132	10.6379	7.9408
19	0.1526482562	0.1010	0.100	0.1000	0.1172	0.1471
20	8.9165133323	8.9747	8.958	9.0135	11.9860	8.9445
21	0.6528020015	0.70648	0.720	0.7391	1.1944	0.8141
22	0.1913866147	0.4225	0.478	0.7870	0.2338	1.1050
23	10.768132810	10.8685	10.897	11.1795	14.8917	11.2893
24	0.1014667543	0.1010	0.100	0.1462	0.1739	0.1004
25	11.776343351	11.8684	11.897	12.1799	16.2167	12.2891
26	0.8002239775	1.035999	1.080	1.3424	1.3311	1.4742
27	7.0077729842	6.6859	6.462	5.4844	8.1343	5.3417
28	11.359710091	10.8111	10.799	10.1372	14.0876	9.8931
29	13.537282086	13.84649	13.922	14.5262	18.8163	14.9127
Best weight (lb)	**25169.687529**	25467.9	25488.15	25521.81	25610.2086	25800.5708
Average weight (lb)	**25510.124193**	25547.6	25533.14	25543.51	25985.05665	26851.1460
Standard deviation	355.54955260	135.09	27.44	**18.13**	177.03358	1,038.1452
Number of analyses	20000	31700	28059	**10685**	20000	20000

200-bar truss

The statistical results of optimal weight are given in Table 7. The proposed algorithm is the best optimizer when compared with CSP [5], TLBO [3], FPA [8], JADE and ADEA [16]. The proposed algorithm also uses an equal or lower number of structural analyses (except for FPA).

6. Conclusions

An optimization algorithm based on differential evolution, ODE-NNC, is presented for truss sizing optimization. The opposition-based mutation and the nearest neighbor comparison are introduced to the conventional differential evolution. The opposition-based mutation biases the search direction while the nearest neighbor comparison omits likely worse solutions without evaluation. Numerical results show that ODE-NNC outperforms various modern metaheuristic algorithms. The optimizer in this study is based on DE with fixed parameters. There is still potential for further improvement, e.g. integrating the techniques with adaptive parameters.

Acknowledgments

This study was carried out within the project supported by National University of Civil Engineering, Vietnam (NUCE) under grant number 98-2015/KHXD.

References

1. Stolpe, M. (2016), "Truss optimization with discrete design variables: a critical review", *Structural and Multidisciplinary Optimization*, 53(2), 349-374.
2. Degertekin S.O. (2012), "Improved harmony search algorithms for sizing optimization of truss structures", *Computers & Structures*, 92, 229-241.
3. Degertekin S.O., Hayalioglu M.S. (2013), "Sizing truss structures using teaching-learning-based optimization", *Computers & Structures*, 119, 177-188.
4. Camp C.V., Farshchin M. (2014), "Design of space trusses using modified teaching–learning based optimization", *Engineering Structures*, 62, 87-97.
5. Kaveh A., Sheikholeslami R., Talatahari S., Keshvari-Ilkhichi M. (2014), "Chaotic swarming of particles: a new method for size optimization of truss structures", *Advances in Engineering Software*, 67, 136-147.
6. Kaveh A., Mahdavi V.R. (2014), "Colliding bodies optimization method for optimum design of truss structures with continuous variables", *Advances in Engineering Software*, 70, 1-12.
7. Kaveh A., Mahdavi V.R. (2015), "Two-dimensional colliding bodies algorithm for optimal design of truss structures", *Advances in Engineering Software*, 83, 70-79.
8. Bekdaş G., Nigdeli S.M., Yang X.S. (2015), "Sizing optimization of truss structures using flower pollination algorithm", *Applied Soft Computing*, 37, 322-331.
9. Das S., Suganthan P.N. (2011), "Differential evolution: A survey of the state-of-the-art", *IEEE Transactions on Evolutionary Computation*, 15(1), 4-31.
10. Wang Z., Tang H., Li P. (2009, December), "Optimum design of truss structures based on differential evolution strategy", In: *IEEE International Conference Information Engineering and Computer Science 2009*, pp 1-5.
11. Wu C.Y., Tseng K.Y. (2010), "Truss structure optimization using adaptive multi-population differential evolution", *Structural and Multidisciplinary Optimization*, 42(4), 575-590.
12. da Silva E.K., Barbosa H.J., Lemonge A.C. (2011), "An adaptive constraint handling technique for differential evolution with dynamic use of variants in engineering optimization", *Optimization and Engineering*, 12(1-2), 31-54.
13. Krempser E., Augusto D.A., Barbosa H.J.C. (2013, May), "Improved Surrogate Model Assisted Differential Evolution with an Infill Criterion", *10th World Congress on Structural and Multidisciplinary Optimization*, Orlando, Florida, USA
14. Krempser E., Bernardino H., Barbosa, H.J.C., Lemonge A. (2012), "Differential evolution assisted by surrogate models for structural optimization problems", In: *Proceedings of the international conference on computational structure technology (CST)*, Civil-Comp Press (Vol. 49).
15. Ho-Huu V., Nguyen-Thoi T., Nguyen-Thoi M.H., Le-Anh L. (2015), "An improved constrained differential evolution using discrete variables (D-ICDE) for layout optimization of truss structures", *Expert Systems with Applications*, 2(20), 7057-7069.
16. Bureerat S., Pholdee N. (2015), "Optimal Truss Sizing Using an Adaptive Differential Evolution Algorithm", *Journal of Computing in Civil Engineering*, 04015019.
17. Ho-Huu V., T. Nguyen-Thoi, T. Vo-Duy, and T. Nguyen-Trang (2016a), "An adaptive elitist differential evolution for optimization of truss structures with discrete design variables", *Computers & Structures*, 165, 59-75.
18. Ho-Huu V., T. Nguyen-Thoi, L. Le-Anh, and T. Nguyen-Trang (2016b), "An effective reliability-based improved constrained differential evolution for reliability-based design optimization of truss structures", *Advances in Engineering Software*, 92, 48-56.
19. Deb K. (2000), "An efficient constraint handling method for genetic algorithms", *Computer methods in applied mechanics and engineering*, 186(2), 311-338.
20. Onwubolu G.C. (2004), "Differential evolution for the flow shop scheduling problem", In: *New Optimization Techniques in Engineering*, Springer Berlin Heidelberg, pp 585-611.
21. Liao J., Cai Y., Wang T., Tian H., Chen Y. (2015), "Cellular direction information based differential evolution for numerical optimization: an empirical study", *Soft Computing*, 20(7), 2801-2827.
22. Pholdee N., Bureerat S., Park W.-W Kim D.-K, Im Y.-T., Kwon H.-C, et al. (2015), "Optimization of flatness of strip during coiling process based on evolutionary algorithms", *International Journal of Precision Engineering and Manufacturing*, 16(7), 1493-1499.
23. Rahnamayan S., Tizhoosh H.R., & Salama M.M.A. (2008), "Opposition-Based Differential Evolution", *Evolutionary Computation, IEEE Transactions on*, 12(1), 64-79.
24. Zhang J., Sanderson A.C. (2009), "JADE: adaptive differential evolution with optional external archive", *IEEE Transactions on Evolutionary Computation*, 13(5), 945-958.
25. Takahama T., Sakai S. (2012, June), "Differential evolution with dynamic strategy and parameter selection by detecting landscape modality", In *IEEE Congress on Evolutionary Computation (CEC) 2012*, pp 1-8.
26. Pham H.A. (2015), "Reduction of function evaluation in differential evolution using nearest neighbor comparison", *Vietnam Journal of Computer Science*, 2(2), 121-131.