Asia-Pacific Journal of Science and Technology

https://www.tci-thaijo.org/index.php/APST/index

Published by the Research and Technology Transfer Affairs Division,
Khon Kaen University, Thailand

Multi-mutation scheme adaptive differential evolution for solving truss sizing optimization

Natee Panagant[1], *, Sujin Bureerat[1]

[1] Sustainable and Infrastructure Research and Development Center, Department of Mechanical Engineering, Faculty of Engineering, Khon Kaen University, Khon Kaen, Thailand.
*Correspondent author: natee_panagant@kkumail.com

Abstract

A novel adaptive differential evolution algorithm called Multi-Mutation Scheme Adaptive Differential Evolution (MMADE) is developed in this article. Several truss sizing optimization problems have been posed for performance test. The proposed adaptive algorithm is integrated with adaptive scaling factor, crossover ratio and mutation schemes selection. Results obtained from the proposed algorithm are compared to recent adaptive algorithms from literature. The MMADE show very competitive performance compared to those state-of-the-art adaptive algorithms.

Keywords: Meta-heuristics, Adaptive algorithms, Truss sizing optimization, Differential evolution, Multi-mutation schemes.

1. Introduction

Currently, optimization techniques are some of the most important tools in engineering structural design. A variety of optimization techniques are employed to solve engineering optimization problems [1-5]. By performing optimum design, a structural designer can achieve lighter but stiffer structures which can lead to lower production or construction cost.

A truss structure is one of the most widely used structure types due to its capability of handling high load with light weight structural elements. Design variables in truss optimization problems can be categorized into several types, topology, shape, and sizing. Truss design problems may consisted of one type of the design variables or a combination of those variables simultaneously. There are many recent researchers attempting to improved optimization techniques for solving single-objective [6-16] and multiple-objective [17-18] truss optimization problems.

Differential Evolution (DE) is one of the simplest and most powerful stochastic optimizers. DE is first introduced by Storn and Price [19] and proved to be one of the best optimizers at that time in 1996 IEEE International Conference on Evolutionary Computation (CEC1996) [20]. However, like other general meta-heuristic optimizers, performance of DE can be fluctuated due to optimization problems encountered and some control parameters. So, after DE was first introduced, there have been many researchers contributing to development of adaptive DE to increase the search performance of DE and overcome its shortcomings.

Many variants of DE were developed in the past few decades. The main control parameters that affect the performance of DE in the literature are population size, mutation scheme, scaling factor (F), and crossover ratio (CR). In 2005, Liu and Lampinen invented a Fuzzy Adaptive Differential Evolution (FADE) [21] by employing fuzzy logic controllers to adapt F and CR in mutation and crossover operation of DE. In the same year, Qin and Suganthan presented Self-Adaptive Differential Evolution (SaDE) [22] in CEC2005. Similarly to FADE, the control parameters, F and CR are also adaptive control parameters, but SaDE is embedded with different adaptive strategy and both parameters are not required to be pre-specified. Then, in 2006, Teo presented Differential evolution with self-adapting populations (DESAP) [23]. This is the first attempt to included adaptive population size along with adaptive *F* and *CR*. In 2006, Brest et al introduced new DE variance with self-adaptive control parameters [24], *F* and *CR* are randomly selected with a new adaptive strategy. In 2009, Zhang et al introduced JADE [25], adaptive DE with an optional external archive. Parent vectors that generated better offspring are stored in the archive and have a chance to be selected for mutating target vectors in a subsequent iteration. Later, Tanabe and Fukunaga demonstrated the improving version of JADE, Success-History based Adaptive DE (SHADE) [26] in 2013. SHADE is integrated with adaptive parameter memory that stored multiple values of *F* and *CR*. With this improvement, SHADE is proved to be one of the best state-of-the-art optimization algorithms in CEC 2013. The continually improved version of SHADE with linear population size reduction called L-SHADE [27] was also presented by the same author in 2014, the winner of CEC 2014 algorithm competition.

In this article, Multi-Mutation Scheme Adaptive Differential Evolution (MMADE) is developed. The proposed algorithm is integrated with three adaptive mutation schemes to balance exploitation and exploration search abilities. The adaptive strategies for mutation schemes: selection, *F*, *CR* are introduced. Details of proposed algorithm and test problems are described in section 2. Then, results, discussion and conclusion demonstrated in Sections 3, 4 and 5 respectively.

2. Materials and methods

2.1 Test problems

There are 6 test cases of truss sizing optimization problems evaluated with MMADE, 2 test cases of a 10-bar truss, 1 test case of a 25-bar truss, 2 test cases of a 72-bar truss and 1 test case of a 200-bar truss. The 10-bar and 200-bar trusses are planar trusses while the 25-bar and 72-bar trusses are space trusses. All test cases are single-objective truss sizing optimization problems. The goal of the optimizer is to minimize mass of the structures under specific constraints i.e. allowable stress and displacement. All details of test cases including their configurations and finite element models are presented in [9]. Notice that only stress and displacement constraints are specified while buckling constraints are excluded in this study. The details of material properties, loadings, constraints and design variable groupings are provided in the following sub-sections.

2.1.1 CASE I: 10-bar truss with the first loading condition

The truss is subjected to P_1 (F_y = -100 *kips*) at node 2 and 4. The design variables are cross-section areas of all truss members, so there are 10 design variables equal to the number of truss members in this test case. The allowable stress (both compressive and tension) and displacement are 25 *ksi* and 2 *in* respectively. Minimum and maximum cross-section areas are 0.1 and 70 in^2 respectively. Material density and modulus of elasticity are 0.1 lb/in^3 and 10^4 *ksi* respectively.

2.1.2 CASE II: 10-bar truss with the second loading condition

All details of this test case are similar to *CASE I* except the loadings. In this case, the truss is subjected to P_1 (F_y = -150 *kips*) at node 2 and 4 and P_2 (F_y = -50 *kips*) at node 1 and 3.

Table 1 Details of design variables grouping and allowable stress of *CASE III*

Group index	Element index	Allowable compressive stress	Allowable tension stress
1	1	35.092	40
2	2-5	11.590	40
3	6-9	17.305	40
4	10-11	35.092	40
5	12-13	35.092	40
6	14-17	6.7590	40
7	18-21	6.9590	40
8	22-25	11.082	40

2.1.3 CASE III: 25-bar truss

The structure is subject to 2 load conditions detailed as follows:

- First load condition: P_1 (*Fy* = 20 *kips* and *Fz* = -5 *kips*) applied on node 1 and P_2 (*Fy* = -20 *kips* and *Fz* = -5 *kips*) applied on node 1.

- Second load condition: P_1 (*Fx* = 1 *kips*, *Fy* = 10 *kips* and *Fz* = -5 *kips*) applied on node 1, P_2 (*Fy* = 10 *kips* and *Fz* = -5 *kips*) applied on node 2 and P_3 (*Fx* = 0.5 *kips*) applied on node 3 and 6.

There are 8 groups of design variables where the element members of each group share the same size of cross-section area and allowable stress. The details of design variables grouping and allowable stress are provided in table 1. Minimum and maximum cross-section areas are 0.01 and 10 in^2 respectively. An allowable displacement is 0.35 *in*. Material density and modulus of elasticity are 0.1 lb/in^3 and 10^4 *ksi* respectively.

Table 2 Details of design variables grouping of *CASE IV-VI*

CASE IV and V		CASE VI	
Group index	Element index	Group index	Element index
1	1-4	1	1, 2, 3, 4
2	5-12	2	5, 8, 11, 14, 17
3	13-16	3	19, 20, 21, 22, 23, 24
4	17-18	4	18, 25, 56, 63, 94, 101, 132, 139, 170, 177
5	19-22	5	26, 29, 32, 35, 38
6	23-30	6	6, 7, 9, 10, 12, 13, 15, 16, 27, 28, 30, 31, 33, 34, 36, 37
7	31-34	7	39, 40, 41, 42
8	35-36	8	43, 46, 49, 52, 55
9	37-40	9	57, 58, 59, 60, 61, 62
10	41-48	10	64, 67, 70, 73, 76
11	49-52	11	44, 45, 47, 48, 50, 51, 53, 54, 65, 66, 68, 69, 71, 72, 74, 75
12	53-54	12	77, 78, 79, 80
13	55-58	13	81, 84, 87, 90, 93
14	59-66	14	95, 96, 97, 98, 99, 100
15	67-70	15	102, 105, 108, 111, 114
16	71-72	16	82, 83, 85, 86, 88, 89, 91, 92, 103, 104, 106, 107, 109, 110, 112, 113
		17	115, 116, 117, 118
		18	119, 122, 125, 128, 131
		19	133, 134, 135, 136, 137, 138
		20	140, 143, 146, 149, 152
		21	120, 121, 123, 124, 126, 127, 129, 130, 141, 142, 144, 145, 147, 148, 150, 151
		22	153, 154, 155, 156
		23	157, 160, 163, 166, 169
		24	171, 172, 173,174, 175, 176
		25	178, 181, 184, 187, 190
		26	158, 159, 161, 162, 164, 165, 167, 168, 179, 180, 182, 183, 185, 186, 188,189
		27	191, 192, 193, 194
		28	195, 197, 198, 200
		29	196, 199

2.1.4 CASE IV: 72-bar truss with the first loading condition

The structure is subject to 2 load conditions detailed as follows:

- First load case: P_1 (*Fx* = 5 *kips*, *Fy* = 5 *kips* and *Fz* = -5 *kips*) applied on node 17.
- Second load case: P_1 (*Fz* = -5 *kips*) applied on node 17,18,19 and 20.

There are 16 groups of design variables in this case. The details of design variables grouping are provided in table 2. Minimum and maximum cross-section areas are 0.1 and 35 *in*^{2} respectively. Allowable stress (both compressive and tension) and displacement are 25 *ksi* and 0.25 *in* respectively. Material density and modulus of elasticity are 0.1 *lb/in*^{3} and 10^4 *ksi* respectively.

2.2 Multi-Mutation schemes Adaptive Differential Evolution (MMADE)

DE variants commonly reproduce population in each iteration with 3 DE operators, mutation, crossover, and selection. From the literature, there are several factors, mutation schemes, mutation ratio, crossover ratio, and population size that mostly affect the performance of DE. In the proposed algorithm, MMADE, the adaptive strategies for mutation schemes, mutation ratio, and crossover ratio are integrated to improve search performance of the algorithm. First, NP population or solution vectors (size of a solution vector is $NV \times 1$ while NV is number of design variables or problem dimensions) are randomly generated. Then, all population are updated with mutation, crossover, and selection sequentially in each iteration. All adaptive control parameters are updated at the end of each iteration.

2.2.1 Mutation

In the proposed algorithm, there are 3 mutation schemes employed as details in equation (10-12).

Scheme-1: "DE/current/1"

$$\boldsymbol{v}_{i,g} = \boldsymbol{x}_{i,g} + F_i(\boldsymbol{x}_{r1,g} - \boldsymbol{x}_{r2,g}) \tag{10}$$

Scheme-2: "DE/current-to-pbest/1"

$$\boldsymbol{v}_{i,g} = \boldsymbol{x}_{i,g} + F_i(\boldsymbol{x}_{pbest,g} - \boldsymbol{x}_{i,g}) + F_i(\boldsymbol{x}_{r1,g} - \boldsymbol{x}_{r2,g}) \tag{11}$$

Scheme-3: "DE/pbest/1

$$\boldsymbol{v}_{i,g} = \boldsymbol{x}_{pbest,g} + F_i(\boldsymbol{x}_{r1,g} - \boldsymbol{x}_{r2,g}) \tag{12}$$

where, $\boldsymbol{x}$ is current population, $\boldsymbol{v}$ is mutation vectors, i is an individual index of the current population, $r1, r2$, and $r3$ are random indices of sampled from members of the current population, g is a generation number, $pbest$ is a random index of the top p percent best population of the current generation and $F_i \in [0,1]$ is a scaling factor of individual i.

In each iteration, the mutation schemes applied to each individual is selected based on selection probabilities while the scaling factor is randomly generated by normal distribution with mean (μ_F) and standard deviation (σ_F). The sum of probabilities for mutation schemes (α_1, α_2, α_3) to be selected are always equal to unity. Four adaptive parameters used in the mutation scheme, μ_F, α_1, α_2, and α_3 will be updated with adaptive strategies at the end of each iteration. The purpose of this multi-mutation scheme is to balance exploitation and exploration search ability of the algorithm. The first one (Equation (10)) is the least greedy scheme while the second scheme (Equation (11)) provides more exploitation ability and the third mutation in Equation (12) is the greediest.

2.2.2 Crossover

Crossover will be operated on a mutant vectors ($\boldsymbol{v}$) and its parent ($\boldsymbol{x}$) with the so-called binomial crossover as described in equation (13) to create a trial vector ($\boldsymbol{u}$).

$$u_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } \text{rand}(0,1) \le CR_i \text{ or } j = j_{rand} \\ x_{j,i,g} & \text{otherwise} \end{cases} \tag{13}$$

where, j is a design variable index, j_{rand} is a random design variable index, $\text{rand}(0,1) \in [0,1]$ is a uniformly distributed pseudo random number and $CR_i \in [0,1]$ is a crossover ratio of individual i. CR_i of each individual is also randomly generated based on normal distribution with mean (μ_{CR}) and standard deviation (σ_{CR}) in the same way as F_i. μ_{CR} is the only one adaptive parameter in crossover process that will be updated at the end of each iteration.

After each trial vector being generated, each design variable is truncated to specific lower and upper boundary as demonstrated in JADE [25]. The truncated rule is describe in equation (14).

$$u_{j,i,g} = \begin{cases} (u_{j,i,g} + x_{L\,j})/2 & \text{if } u_{j,i,g} < x_{L\,j} \\ (u_{j,i,g} + x_{U\,j})/2 & \text{if } u_{j,i,g} < x_{U\,j} \\ u_{j,i,g} & \text{otherwise} \end{cases} \tag{14}$$

where, x_{L_j} and x_{U_j} are lower and upper bound of j-th design variable respectively.

2.2.3 Selection

In selection process, the better individual between a trial vector $\boldsymbol{u}_{i,g}$ and its parent $\boldsymbol{x}_{i,g}$ will survive to next iteration as described in equation (15).

$$\boldsymbol{x}_{i,g+1} = \begin{cases} \boldsymbol{u}_{i,g} & \text{if } f(\boldsymbol{u}_{i,g}) < f(\boldsymbol{x}_{i,g}) \\ \boldsymbol{x}_{i,g} & \text{otherwise} \end{cases} \tag{15}$$

Where, $fp(\boldsymbol{u}_{i,g})$ and $fp(\boldsymbol{x}_{i,g})$ are fitness values of the trial vector and its parent respectively.

2.2.4 Adaptation strategies

All adaptive parameters are adjusted at the end of each iteration. There are 5 adaptive parameters, μ_F, μ_{CR}, α_1, α_2, and α_3 employed in the proposed algorithm. The adaptive parameters are updated with learning strategies related to parameters that produce better offspring in each iteration. Lehmer mean of successful updated μ_F and μ_{CR} are calculated with equation (16-17). Then, the new values of μ_F and μ_{CR} are updated with a learning rate (LR) if there is at least one successful updated individual and reset to initial values otherwise as described in equation (18-19).

$$Lmean_{\mu_F} = \frac{\sum_{k=1}^{Ns}({\mu_F}_{s,k}{}^2)}{\sum_{k=1}^{Ns}({\mu_F}_{s,k})} \tag{16}$$

$$Lmean_{\mu_{CR}} = \frac{\sum_{k=1}^{Ns}({\mu_{CR}}_{s,k}{}^2)}{\sum_{k=1}^{Ns}({\mu_{CR}}_{s,k})} \tag{17}$$

$$\mu_{F,g+1} = \begin{cases} \mu_{F,g} + LR \cdot Lmean_{\mu_F} & \text{if } Ns > 0 \\ \mu_{F,0} & \text{otherwise} \end{cases} \tag{18}$$

$$\mu_{CR,g+1} = \begin{cases} \mu_{CR,g} + LR \cdot Lmean_{\mu_{CR}} & \text{if } Ns > 0 \\ \mu_{CR,0} & \text{otherwise} \end{cases} \tag{19}$$

where, $Lmean_{\mu_F}$ and $Lmean_{\mu_{CR}}$ are the Lehmer means of μ_F and μ_{CR} respectively, Ns is the number of successful updated individuals in the current iteration, ${\mu_F}_{s,k}$ and ${\mu_{CR}}_{s,k}$ are the scaling factor and crossover ratio of each successful updated individual respectively.

If there is any successful updated individual, α_1, α_2, and α_3 are updated related to the number of successful updated individuals that are generated by their corresponding mutation schemes, otherwise, they are reset to the initial values. For the updating procedure, the probabilities are described in equation (20-21). It should be noted that the sum of the probabilities are always normalized to unity in equation (21).

$$\alpha'_{m,g+1} = \alpha_{m,g} + LR \cdot (Ns_{\alpha_m}/Ns)\ , m = 1,2,3 \tag{20}$$

$$\alpha_{m,g+1} = \begin{cases} \alpha'_{m,g+1}/(\alpha'_{1,g+1} + \alpha'_{2,g+1} + \alpha'_{3,g+1}) & \text{if } Ns > 0 \\ \alpha_{m,0} & \text{otherwise} \end{cases}, m = 1,2,3 \tag{21}$$

where, Ns_{α_m} is number of success updated individuals by mutation scheme-m.

2.3 Experimental setup

For ease of comparison, the maximum number of function evaluations for *CASE I-V* and *CASE VI* are specified as 10,000 and 20,000 which are equal to the literature [9]. The population size of *CASE I-V* and *CASE VI* are 50 and 100 respectively. The initial values of μ_F and μ_{CR} are 0.5 and 0.8 respectively while σ_F and σ_{CR} are both set to be 0.1. The initial values of α_1, α_2, and α_3 are all equal to 1/3. The learning rate (LR) for all adaptive strategies is equal to 0.1. The control parameters of the penalty function technique, c_1, c_2, and k are equal to 1e-5, 1e-9, and 10 respectively.

3. Results

In this numerical tests, 30 independent runs of the proposed algorithm are performed. The results obtained from using MMADE are evaluated and compared to the results of state-of-the-art algorithms from the literature

[9]. The mean and standard deviation of weight found by MMADE and best algorithms from the literature are compared in table 3.

Table 3 Comparison of mean weights and standard deviation found by MMADE and best algorithms from the literature.

Problem	MMADE (mean±std)	Literature (mean±std)(name of the best algorithm)
CASE I	5061.443±2.8768	5060.961 ± 0.061 (SHADE)
CASE II	4677.152±0.1124	4677.412 ± 0.3657 (SHADE)
CASE III	545.163±0.0003	545.163 ± 0.0006 (L-SHADE)
CASE IV	379.989±0.1985	379.985 ± 0.2285 (SHADE)
CASE V	364.210±0.1710	364.261 ± 0.2280 (SHADE)
CASE VI	25919.11±227.18	26109.67 ± 187.34 (SHADE)

The details of best results of *CASE I–VI* found by MMADE are provided in table 4. Details of optimum design variables (cross-section areas), weight and constrained violations are all included in the table.

Table 4 Details of the best trusses found by MMADE

Problem	*CASE I*	*CASE II*	*CASE III*	*CASE VI*	*CASE V*	*CASE IV*
	30.566	23.55759	0.01000375	1.901606	1.874741	0.1146819
	0.1000598	0.1000414	1.986214	0.5166726	0.5234692	0.9473768
	23.2117	25.21872	2.994766	0.1000834	0.01035941	0.1623782
	15.19771	14.32949	0.01000039	0.10035	0.01016275	0.1080223
	0.100006	0.1000034	0.01000239	1.243426	1.313161	1.953826
	0.5526223	1.969816	0.683598	0.512497	0.5101077	0.2413752
	7.459013	12.41554	1.677024	0.1000511	0.01022529	0.2683819
	21.04136	12.86294	2.662229	0.1007234	0.01049506	3.108701
	21.49924	20.32941		0.5216062	0.5299416	0.1205973
	0.1000016	0.1000485		0.5187426	0.5172217	4.133854
				0.1000433	0.01014704	0.356142
				0.1005843	0.1059103	0.1265806
				0.1564847	0.1673201	5.440595
				0.5462572	0.534087	0.220856
Set of optimum				0.4041166	0.4509461	6.421065
Design variables (in^2)				0.5637517	0.5744491	0.5311663
						0.3697953
						8.020133
						0.2341736
						9.102079
						0.843345
						0.6041449
						11.45215
						0.1458555
						12.58396
						1.174508
						5.288524
						10.20159
						14.7895
Weight	5060.867	4676.992	545.162	379.704	363.931	25607.53
Max stress constraint	0	0	0	0	0	0
Max displacement constraint	1.8553e-7	0	1.0784e-6	0	0	-

4. Discussion

After 30 independent runs of each test case being evaluated, MMADE shows very competitive results in table 3. For fair comparisons, it should be noted that all optimization runs are evaluated with equal maximum number of function evaluations being limited as demonstrated in the literature [9]. Instead of using the best weights for algorithm performance comparison which cannot measure consistency of each optimizer, mean weights from 30 independent runs of each test which indicates both performance and search consistency of the optimizers are preferred in this study. Compared to best adaptive optimizers from the literature, MMADE can provided better mean weights in *CASE II*, *V*, and *VI*. Mean weight found by MMADE in *CASE III* is equal to L-SHADE from the literature, but MMADE provided lower standard deviation. SHADE from the literature still perform better in *CASE I* and *IV*. Information of the best results found by MMADE for all test cases are provided in table 4. The constraint violations of the results are less than 10^{-5} in all cases.

5. Conclusions

While most of the optimizers in the literature focus on development of adaptive strategies for a scaling factor, a crossover ratio and a population size to increase search performance of DE. Although there are several optimizers, JADE, SHADE, and L-SHADE presented external archive to improve mutation operator, but there is only one mutation scheme employed in those algorithms. In the proposed optimizer, MMADE, the authors present an alternative way to improve the performance and search consistency of DE. The adaptive multi-mutation scheme is integrated together with the adaptive scaling factor and crossover ratio. MMADE is provided very competitive results compared to state-of-the-art adaptive optimizers by achieving better mean weights in 3 out of 6 test problems. To expand the capabilities of MMADE to handle more complex problems like simultaneous topology, shape and size optimization problems or large scale problems, additional test problems should be evaluated and some improvements may be required in the future work. The evaluated results should be compared to more state-of-the-art optimizers to further assure the performance of the proposed algorithm.

6. Acknowledgements

The authors are grateful for support from the Royal Golden Jubilee Ph.D. Program (Grant No. PHD/0130/2557) and the Thailand Research Fund (BRG5880014).

7. References

[1] Channarong T, Jindapetch N, Thongnoo K.,2017. An optimal hard disk head fly-height control algorithm for soft error rate reduction during the read-write process. Asia-Pacific Journal of Science and Technology 16, 240-51.

[2] Hassajan S, Lamom A, Cheerarot R.,2017. Effect of materials selection on the minimum cost for optimal design of reinforced concrete beams using hill climbing algorithm. Asia-Pacific Journal of Science and Technology 17, 385-400.

[3] Sriworamas K, Bureerat S, Vangpaisal T.,2017. Multi objective evolutionary algorithms for pipe network design and rehabilitation: comparative study on large and small scale problems. Asia-Pacific Journal of Science and Technology 17, 366-74.

[4] Kunakote T, Bureerat S.,2017. Multiobjective two-stage optimization of a plate structure using a population-based incremental learning method. Asia-Pacific Journal of Science and Technology 19, 233-44.

[5] Kaewploy S.,2015. Optimal parameters in precipitation hardening of 6061 aluminium alloy using box-behnken design. Asia-Pacific Journal of Science and Technology 20, 369-80.

[6] Rahami H, Kaveh A, Gholipour Y.,2008. Sizing, geometry and topology optimization of trusses via force method and genetic algorithm. Engineering Structures 30, 2360-9.

[7] Ahrari A, Atai AA, Deb K.,2015. Simultaneous topology, shape and size optimization of truss structures by fully stressed design based on evolution strategy. Engineering Optimization 47, 1063-84.

[8] Bureerat S, Pholdee N.,2015. Optimal truss sizing using an adaptive differential evolution algorithm. Journal of Computing in Civil Engineering 30, 04015019.

[9] Pholdee N, Bureerat S.,2017. A comparative study of eighteen self-adaptive metaheuristic algorithms for truss sizing optimisation. KSCE Journal of Civil Engineering, 1-2.

[10] Panagant N, Bureerat S.,2018. Truss topology, shape and sizing optimization by fully stressed design based on hybrid grey wolf optimization and adaptive differential evolution. Engineering Optimization 10, 1-7.

[11] Tejani GG, Savsani VJ, Patel VK.,2016. Adaptive symbiotic organisms search (SOS) algorithm for structural design optimization. Journal of Computational Design and Engineering 3, 226-49.

[12] Tejani GG, Savsani VJ, Patel VK., 2016. Modified sub-population teaching-learning-based optimization for design of truss structures with natural frequency constraints. Mechanics Based Design of Structures and Machines 44, 495-513.
[13] Savsani VJ, Tejani GG, Patel VK.,2016. Truss topology optimization with static and dynamic constraints using modified subpopulation teaching–learning-based optimization. Engineering Optimization 48, 1990-2006.
[14] Tejani GG, Savsani VJ, Patel VK, Mirjalili S.,2017 Truss optimization with natural frequency bounds using improved symbiotic organisms search. Knowledge-Based Systems, 11.
[15] Tejani GG, Savsani VJ, Bureerat S, Patel VK.,2017. Topology and Size Optimization of Trusses with Static and Dynamic Bounds by Modified Symbiotic Organisms Search. Journal of Computing in Civil Engineering 32, 04017085.
[16] Tejani GG, Saysani VJ, Patel VK, Bureerat S.,2017. Topology, shape, and size optimization of truss structures using modified teaching-learning based optimization. Advances In Computational Design 2, 313-31.
[17] Noilublao N, Bureerat S.,2011. Simultaneous topology, shape and sizing optimisation of a three-dimensional slender truss tower using multiobjective evolutionary algorithms. Computers & Structures 89, 2531-8.
[18] Pholdee N, Bureerat S.,2014. Hybrid real-code population-based incremental learning and approximate gradients for multi-objective truss design. Engineering Optimization 46, 1032-51.
[19] Storn R, Price K.,1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11, 341-59.
[20] Storn R, Price K.,1996. Minimizing the real functions of the ICEC'96 contest by differential evolution. In Evolutionary Computation, 1996., Proceedings of IEEE International Conference on May 20, 842-844.
[21] Liu J, Lampinen J.,2005. A fuzzy adaptive differential evolution algorithm. Soft Computing 9, 448-62.
[22] Qin AK, Suganthan PN. 2005. Self-adaptive differential evolution algorithm for numerical optimization. InEvolutionary Computation. The 2005 IEEE Congress on 2005 Sep 2 2, 1785-1791.
[23] Teo J.,2006. Exploring dynamic self-adaptive populations in differential evolution. Soft Computing 10, 673-86.
[24] Brest J, Greiner S, Boskovic B, Mernik M, Zumer V.,2006. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE transactions on evolutionary computation 10, 646-57.
[25] Zhang J, Sanderson AC.,2009. JADE: adaptive differential evolution with optional external archive. IEEE Transactions on evolutionary computation 13, 945-58.
[26] Tanabe R, Fukunaga A.,2013. Success-history based parameter adaptation for differential evolution. InEvolutionary Computation (CEC), IEEE Congress on 2013 Jun 20, 71-78.
[27] Tanabe R, Fukunaga AS.,2014. Improving the search performance of SHADE using linear population size reduction. InEvolutionary Computation (CEC), IEEE Congress on 2014 Jul 6, 1658-1665.