frontiers in Built Environment

ORIGINAL RESEARCH
published: 09 July 2020
doi: 10.3389/fbuil.2020.00102

A Comparative Study of Differential Evolution Variants in Constrained Structural Optimization

Manolis Georgioudakis[1] *and Vagelis Plevris*[2]*

[1] Institute of Structural Analysis and Antiseismic Research, School of Civil Engineering, National Technical University of Athens (NTUA), Athens, Greece, [2] Department of Civil Engineering and Energy Technology, OsloMet Oslo Metropolitan University, Oslo, Norway

Differential evolution (DE) is a population-based metaheuristic search algorithm that optimizes a problem by iteratively improving a candidate solution based on an evolutionary process. Such algorithms make few or no assumptions about the underlying optimization problem and can quickly explore very large design spaces. DE is arguably one of the most versatile and stable population-based search algorithms that exhibits robustness to multi-modal problems. In the field of structural engineering, most practical optimization problems are associated with one or several behavioral constraints. Constrained optimization problems are quite challenging to solve due to their complexity and high nonlinearity. In this work we examine the performance of several DE variants, namely the standard DE, the composite DE (CODE), the adaptive DE with optional external archive (JADE) and the self-adaptive DE (JDE and SADE), for handling constrained structural optimization problems associated with truss structures. The performance of each DE variant is evaluated by using five well-known benchmark structures in 2D and 3D. The evaluation is done on the basis of final optimum result and the rate of convergence. Valuable conclusions are obtained from the statistical analysis which can help a structural engineer in practice to choose the suitable algorithm for such kind of problems.

Keywords: structural optimization, differential evolution, DE, CODE, JADE, JDE, SADE

OPEN ACCESS

Edited by:
Marios C. Phocas,
University of Cyprus, Cyprus

Reviewed by:
Francesco Tornabene,
University of Salento, Italy
Aristotelis E. Charalampakis,
University of West Attica, Greece

***Correspondence:**
Vagelis Plevris
vageli@oslomet.no

Specialty section:
This article was submitted to Computational Methods in Structural Engineering, a section of the journal Frontiers in Built Environment

Received: *16 April 2020*
Accepted: *02 June 2020*
Published: *09 July 2020*

Citation:
Georgioudakis M and Plevris V (2020) A Comparative Study of Differential Evolution Variants in Constrained Structural Optimization. Front. Built Environ. 6:102. doi: 10.3389/fbuil.2020.00102

1. INTRODUCTION

The optimization of structures has been a topic of great interest for both scientists and engineering professionals, especially in recent years. Metaheuristic search algorithms are widely accepted as efficient approaches for handling difficult optimization problems. Such algorithms are designed for solving a wide range of optimization problems in an approximate way, without having to adapt explicitly to every single problem. Moreover, they can be generally applied to problems for which there exists no satisfactory problem-specific algorithm.

In recent years differential evolution (DE) (Storn and Price, 1997) has gained increasing interest for solving optimization problems in many scientific and engineering fields. Today, it is considered one of the most popular optimization algorithms for continuous optimization problems. The method was originally proposed by Storn and Price (1995) for minimizing non-differentiable and possibly nonlinear continuous functions. It is worth noting that although DE is an evolutionary algorithm, it bears no natural paradigm and it is not biologically inspired like most

other evolutionary algorithms. DE has exhibited very good performance in a variety of optimization problems from various scientific fields. It belongs to stochastic population-based evolutionary methods and like other evolutionary algorithms, it uses a population of candidate solutions and the search is done in a stochastic way by applying mutation, crossover, and selection operators to drive the population toward better solutions in the design space.

The main advantage of standard DE is the fact that it has only three control parameters that one needs to adjust. The performance of DE in a specific optimization problem depends largely on both the trial vector generation scheme and the choice of the control parameters. First, one needs to choose the trial vector generation scheme and then to adjust the control parameters for the optimization problem in order to achieve good optimization results. Finding the right values of the control parameters is not always easy and can become time consuming and difficult especially for specific hard problems. This has led the researchers to study and develop new advanced DE variants that exhibit adaptive and self-adaptive control parameters. In the case of adaptive parameter control (Eiben et al., 1999), the parameters are adapted based on feedback received during the search process.

Shukla et al. (2017) presented a modified mutation vector generation scheme for the basic DE for solving the stagnation problem. A new variant of DE was proposed and its performance was tested on 24 benchmark functions. Abbas et al. (2015) proposed a tournament-based parent selection variant of the DE algorithm in an effort to enhance the searching capability and improve convergence speed of DE. The paper also describes a statistical comparison of existing DE mutation variants, categorizing these variants based on their overall performance. Charalampakis and Tsiatas (2019) compare variants of Genetic Algorithms, Particle Swarm Optimization (Plevris and Papadrakakis, 2011), Artificial Bee Colony, Differential Evolution, and Simulated Annealing in truss sizing structural optimization problems. The authors claim that for the examined problems, DE is the most reliable algorithm, showing robustness, excellent performance and scalability. Mezura-Montes et al. (2006) present an empirical comparison of several DE variants in solving global optimization problems where 13 benchmark problems from the literature were examined and eight different variants were implemented.

In the present study, we investigate the performance of five popular DE variants in dealing with constrained structural optimization problems. More specific the following five problems are considered:

- The standard differential evolution (DE) (Storn and Price, 1997)
- The composite differential evolution (CODE) (Wang et al., 2011)
- The self-adaptive control parameters differential evolution (JDE) (Brest et al., 2006)
- The adaptive differential evolution with optional external archive (JADE) (Zhang and Sanderson, 2009)
- The self-adaptive differential evolution (SADE) (Qin et al., 2009).

We examine the performance of each algorithm in five structural optimization problems, three plane and two space truss benchmark structures where the objective is to minimize the structural weight subject to constraints on stresses and displacements.

The remainder of the paper is organized as follows. The second section contains the problem definition and the constraint handling scheme. Section 3 describes the standard DE, the most frequently used mutation schemes and the other four DE variants examined in the study. The numerical examples, the relevant results and a discussion on them are presented in section 4. Section 5 discusses the conclusions of the work.

2. PROBLEM DEFINITION

Many problems in structural engineering involve often dealing with a large number of design parameters which can affect the performance of the system. The design and testing of civil engineering structures requires often an iterative process with proper adjustment of the parameters, that can be hard and time consuming. Design optimization offers solutions to this problem by changing the design parameters in an organized and automated manner in an effort to reach an optimal solution. The target of the optimization is usually the minimization of a cost function.

In sizing structural optimization problems, the objective is associated with the minimization of the weight of the structure under some behavioral constraints that have usually to do with displacements and stresses. The design parameters have to do with the dimensions of the cross sections of the structural members. The present study is focused on 2D and 3D truss structures and the design variables are continuous representing the cross-sectional areas of the members of the structure. For such problems the objective function is usually the weight (or mass) of the structure and the problem is formulated as follows:

$$\begin{aligned} \min_{x_i} \quad & W(\mathbf{x}) = \sum_{i=1}^{N_e} L_i x_i \rho_i \\ \text{s.t.} \quad & g_k(\mathbf{x}) \leq 0, \quad k = \{1, \ldots, K\} \end{aligned} \tag{1}$$

where $W(\mathbf{x})$ is the total structural weight, N_e is the number of structural elements, $\mathbf{x} = \{x_1, \ldots, x_{N_e}\}$ is the vector which contains the cross-section areas x_i of all elements, L_i is the length and ρ_i is the material density of element i. In addition, $g(\mathbf{x})$ are the behavioral constraints, K in total. The behavioral constraints are relationships involving usually stresses and deflections of the various elements and nodes of the structure, for example the maximum stress, the maximum deflection, or the minimum load capacity to satisfy norms requirements. Many times, for practical purposes and for uniformity, the

areas of particular members are grouped together so that the number of design variables can become significantly smaller than the total number of elements of the structure. This is very convenient especially for big structures with a large number of elements. Grouping in optimization is also in line with actual grouping that is performed in practice due to symmetry and simplicity.

2.1. Constraint Handling

A common practice for dealing with an optimization problem that includes inequality constraints is the use of a penalty function. Such functions can be used to transform the original constrained problem to an unconstrained one. This has been a popular approach for dealing with such problems because it is simple and rather easy to implement. In the present study, the following penalty formulation is used to handle the optimization problem constraints:

$$F(\mathbf{x}) = f(\mathbf{x}) + P(\mathbf{x}) = f(\mathbf{x}) + \mu \sum_{k=1}^{N} H_k(\mathbf{x}) g_k^2(\mathbf{x}) \tag{2}$$

where $f(\mathbf{x})$ is the objective function to be minimized, $P(\mathbf{x})$ is the penalty term, $g_k(\mathbf{x})$ is the k-th constraint function in the form $g_k(\mathbf{x}) \leq 0$, $\mu \geq 0$ is a penalty factor that should be large enough (e.g., 10^6) and the $H_k(\mathbf{x})$ function is defined as follows:

$$H_i(\mathbf{x}) = \begin{cases} 1, \text{ if } g_i(\mathbf{x}) > 0 \\ 0, \text{ otherwise} \end{cases} \tag{3}$$

3. OPTIMIZATION ALGORITHMS

3.1. Standard DE

DE is a popular optimization method used for multidimensional real-valued functions which uses a population of individual solutions. The method does not require gradient information, which means that the optimization problem does not need to be differentiable. The algorithm searches the design space by maintaining a population of candidate solutions (individuals) and creating new solutions by combining existing ones according to a specific process. The candidates with the best objective values are kept in the next iteration of the algorithm in a manner that the new objective value of an individual is improved forming consequently part of the population, otherwise the new objective value is discarded. The process repeats itself until a given termination criterion is satisfied.

Let $\mathbf{x} \in \mathbb{R}^D$ designate a candidate solution in the current population, where D is the dimensionality of the problem being optimized and $f : \mathbb{R}^D \longrightarrow \mathbb{R}$ is the objective function to be minimized. The basic DE algorithm, following the "DE/rand/1" scheme, can be described schematically as follows:

TABLE 1 | Characteristics of the five test problems.

Description	Type	Elements	Nodes	DOFs	Design variables
25-bar space truss	3D	25	10	30	8
10-bar plane truss	2D	10	6	12	10
72-bar space truss	3D	72	20	60	16
17-bar plane truss	2D	17	9	18	17
200-bar plane truss	2D	200	77	154	29

Algorithm 1: Pseudocode of DE.

Data:
NP: population size, *F*: mutation factor, *CR*: crossover probability, *MAXFES*: maximum number of functions evaluations
INITIALIZATION $G = 0$; Initialize all *NP* individuals with random positions in the search space;
while $FES < MAXFES$ **do**
  for $i \leftarrow 1$ **to** *NP* **do**
    GENERATE three individuals x_{r1}, x_{r1}, x_{r1} from the current population randomly. These must be distinct from each other and also from individual x_i, i.e. $r_1 \neq r_2 \neq r_3 \neq i$
    MUTATION Form the donor vector using the formula: $\mathbf{v}_i = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3})$
    CROSSOVER The trial vector $\mathbf{u}_i$ is developed either from the elements of the target vector $\mathbf{x}_i$ or the elements of the donor vector $\mathbf{v}_i$ as follows:

$$u_{i,j} = \begin{cases} \mathrm{v}_{i,j}, \text{ if } r_{i,j} \leq CR \text{ or } j = j_{rand} \\ x_{i,j}, \text{ otherwise} \end{cases}$$

    where $i = \{1, \ldots, NP\}$, $j = \{1, \ldots, D\}$, $r_{i,j} \sim \cup(0, 1)$ is a uniformly distributed random number which is generated for each j and $j_{rand} \in \{1, \ldots, D\}$ is a random integer used to ensure that $\mathbf{u}_i \neq \mathbf{x}_i$ in all cases
    EVALUATE If $f(\mathbf{u}_i) \leq f(\mathbf{x}_i)$ then replace the individual $\mathbf{x}_i$ in the population with the trial vector $\mathbf{u}_i$
    $FES = FES + NP$
  end
  $G = G + 1$;
end

In every generation (iteration) G, Differential Evolution uses the mutation operator for producing the donor vector $\mathbf{v}_i$ for each individual $\mathbf{x}_i$ in the current population. For each target vector $\mathbf{x}_i = \{x_{i,1}, \ldots, x_{i,D}\}$ at generation G, the associated mutant vector $\mathbf{v}_i = \{\mathrm{v}_{i,1}, \ldots, \mathrm{v}_{i,D}\}$ can be produced using a specific mutation scheme. The six most widely used mutation schemes in differential evolution are described below:

"DE/rand/1" $\mathbf{v}_i = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3})$
"DE/best/1" $\mathbf{v}_i = \mathbf{x}_{best} + F(\mathbf{x}_{r_1} - \mathbf{x}_{r_2})$
"DE/rand/2" $\mathbf{v}_i = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) + F(\mathbf{x}_{r_4} - \mathbf{x}_{r_5})$
"DE/best/2" $\mathbf{v}_i = \mathbf{x}_{best} + F(\mathbf{x}_{r_1} - \mathbf{x}_{r_2}) + F(\mathbf{x}_{r_3} - \mathbf{x}_{r_4})$
"DE/current-to-best/1" $\mathbf{v}_i = \mathbf{x}_i + F(\mathbf{x}_{best} - \mathbf{x}_i) + F(\mathbf{x}_{r_1} - \mathbf{x}_{r_2})$

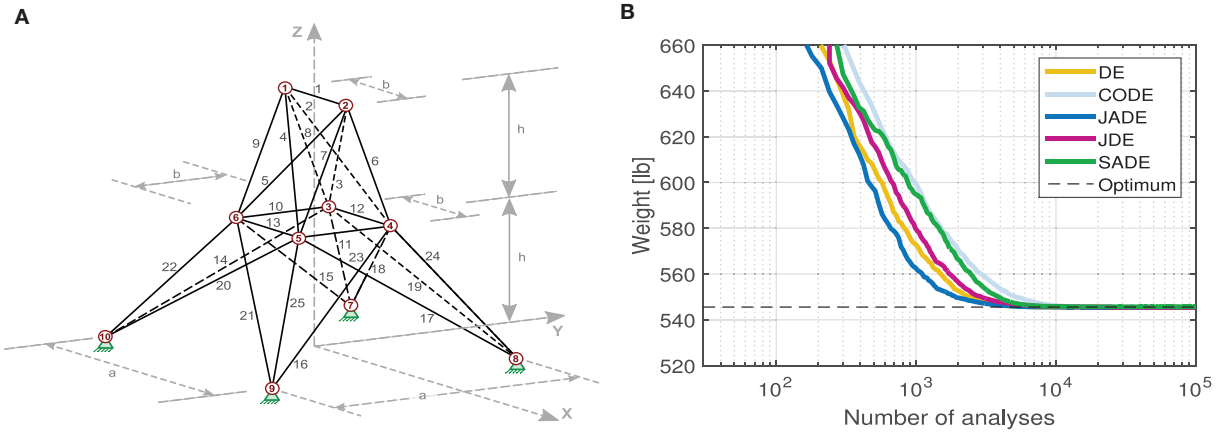

FIGURE 1 | 25-bar space truss **(A)** geometry; **(B)** algorithms' convergence histories (average of 30 runs).

"DE/current-to-rand/1" $\mathbf{v}_i = \mathbf{x}_i + rand(\mathbf{x}_{r_1} - \mathbf{x}_i) + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3})$

where the indices r_1, r_2, r_3, r_4, r_5 are random integers which are mutually exclusive, within the range $[1, NP]$ and are also different than index i ($r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq i$). These indices are generated once for each mutant vector. F is the scaling factor, a positive control parameter for scaling the difference vector, while x_{best} is the best individual vector, i.e., the individual with the best objective value in the population at the current generation G.

The main advantage of DE is the fact that it has only three control parameters that the user of the algorithm needs to adjust. These include the population size NP, where $NP \geq 4$, the mutation factor (or differential weight, or scaling factor) $F \in [0, 2]$ and the crossover probability (or crossover control parameter) $CR \in [0, 1]$. In the standard DE these control parameters were kept fixed for all the optimization process. The population size has a significant influence on the ability of the algorithm to explore. In case of problems with a large number of dimensions, the population size needs also to be large to make the algorithm capable of searching in the multi-dimensional design space. A population size of 30–50 is usually sufficient in most problems of engineering interest. The mutation factor F is a positive control parameter for scaling and controlling the amplification of the difference vector. Small values of F will lead to smaller mutation step sizes and as a result it will take longer for the algorithm to converge. Large values of F facilitate exploration, but can lead to the algorithm overshooting good optima. Thus, the value has to be small enough to enhance local exploration but also large enough to maintain diversity. The crossover probability CR has an influence on the diversity of DE, as it controls the number of elements that will change. Larger values of CR will lead to introducing more variation in the new population, therefore increasing it increases exploration. But again a compromise value has to be found to ensure both local and global search capabilities.

3.2. CODE

Studies have shown that both the control parameters and the schemes for the trial vector generation can have a significant influence on the algorithm's performance. Different trial vector generation schemes and control parameters can be therefore combined to improve the performance of the optimization process for different kinds of problems. Mallipeddi et al. (2011) was the first to attempt, trying to combine various schemes for trial vector generation with different control parameter settings.

Composite DE (CODE) (Wang et al., 2011) is based on the idea of randomly combining a number of trial vector generation schemes with several control parameter settings at each generation, for the creation of new trial vectors. These combination schemes are based on experimental results from the literature. In particular, CODE uses three different trial vector generation schemes and three control parameter settings, combining them randomly for the generation of trial vectors. The structure of CODE is rather simple, and the algorithm is easy

Algorithm 2: Pseudocode of CODE.

Data: NP, $MAXFES$
INITIALIZATION $G = 0$; $FES = NP$;
while $FES < MAXFES$ **do**
  $P_{G+1} = 0$
  for $i \leftarrow 1$ **to** NP **do**
    GENERATE three trial vectors $u_{i_1,G}, u_{i_2,G}$ and $u_{i_3,G}$ for the target vector $x_{i,G}$ using the three generation schemes "rand/1/bin," "rand/2/bin," and "current-to-rand/1," each with control parameter setting randomly selected from the parameter candidate pool: $[F = 1.0, CR = 0.1]$, $[F = 1.0, CR = 0.9]$ and $[F = 0.8, CR = 0.2]$.
    EVALUATE the objective function value of the three trial vectors $u_{i_1,G}, u_{i_2,G}$, and $u_{i_3,G}$
    CHOOSE the best trial vector $u^*_{i,G}$ from the three trial vectors
    $P_{G+1} = P_{G+1} Uselect(x^*_{i,G}, u^*_{i,G})$
    $FES = FES + 3$
  end
  $G = G + 1$;
end

to implement. The three trial vector generation schemes of the method are the following: (i) "rand/1/bin"; (ii) "rand/2/bin"; (iii) "current-to-rand/1." It has to be noted that in the case of the "current-to-rand/1"scheme, the binominal crossover operator is not applied. The three control parameter settings used are the following: (1) $F = 1.0$ and $CR = 0.1$; (2) $F = 1.0$ and $CR = 0.9$; (3) $F = 0.8$ and $CR = 0.2$.

TABLE 2 | 25-bar space truss: Load cases (loads in kips).

	Load case 1			Load case 2		
Node	P_x	P_y	P_z	P_x	P_y	P_z
1	1.0	10.0	−5.0	0.0	20.0	−5.0
2	0.0	10.0	−5.0	0.0	−20.0	−5.0
3	0.5	0.0	0.0	0.0	0.0	0.0
6	0.5	0.0	0.0	0.0	0.0	0.0

TABLE 3 | 25-bar space truss: allowable stresses for each design group.

Design variable	Members	Allowable tension (ksi)	Allowable compression (ksi)
1	1	40	−35.092
2	2–5	40	−11.590
3	6–9	40	−17.305
4	10,11	40	−35.092
5	12,13	40	−35.092
6	14–17	40	−6.759
7	18–21	40	−6.759
8	22–25	40	−11.082

In each generation, three trial vectors are generated for each target vector as follows: each of the three trial vector generation schemes is combined with a control parameter setting from the relevant pool, in a random manner. If the best one of the three is better than its target vector, then it enters the next generation. The pseudocode of the method is presented in Algorithm 2.

3.3. JDE

The standard DE algorithm includes a set of parameters that are kept fixed throughout the optimization process. These parameters would need to be adjusted for every single optimization problem in order to ensure optimal performance. Some researchers have claimed that the DE parameters are not so difficult to set manually (Storn and Price, 1997). Yet, others (Gamperle et al., 2002) claim that the process can be quite demanding especially for particular optimization problems. According to Liu and Lampinen (2002), the effectiveness, efficiency, and robustness of the DE algorithm are very sensitive to the values of the control parameters, while certain parameters may work well in a problem but not so well with other problems, which makes the optimal selection of the parameters a problem-specific question.

JDE features self-adaptive control parameter settings and has shown acceptable performance on benchmark problems (Brest et al., 2006). It uses the idea of the *evolution of the evolution*, i.e., it uses an evolution process to fine tune the optimization algorithm parameters. Although the idea sounds promising, the proof of convergence of self-adaptation algorithms is a difficult task in general. In JDE the parameter control technique is based on the self-adaptation of the parameters F and CR of the DE evolutionary process, producing a flexible DE which adjusts itself in order to achieve the best optimization outcome. According to Storn and Price (1997), DE behavior is more sensitive to the

TABLE 4 | Algorithms' statistical parameters for the 25-bar space truss problem.

	Algorithms				
	DE	**CODE**	**JDE**	**JADE**	**SADE**
Area [in^2]	**Best design**	**Best design**	**Best design**	**Best design**	**Best design**
1	0.0100	0.0100	0.0100	0.0100	0.0100
2	1.9324	1.9324	1.9325	1.9324	1.9325
3	2.9853	2.9853	2.9850	2.9853	2.9850
4	0.0100	0.0100	0.0100	0.0100	0.0100
5	0.0100	0.0100	0.0100	0.0100	0.0100
6	0.6842	0.6842	0.6842	0.6842	0.6843
7	1.7343	1.7343	1.7343	1.7343	1.7343
8	2.6513	2.6513	2.6513	2.6513	2.6512
Weight [lb]	**Best** **Worst**	**Best** **Worst**	**Best** **Worst**	**Best** **Worst**	**Best** **Worst**
	545.555 545.555	545.555 545.555	545.555 545.606	545.555 545.555	545.555 546.143
	Mean ± Std	**Mean ± Std**	**Mean ± Std**	**Mean ± Std**	**Mean ± Std**
	545.555 ± 0.000	545.555 ± 0.000	545.559 ± 0.010	545.555 ± 0.000	545.613 ± 0.122
	Median **COV**	**Median** **COV**	**Median** **COV**	**Median** **COV**	**Median** **COV**
	545.555 6.67E-11	545.555 2.12E-16	545.555 1.92E-05	545.555 2.12E-16	545.566 2.24E-04

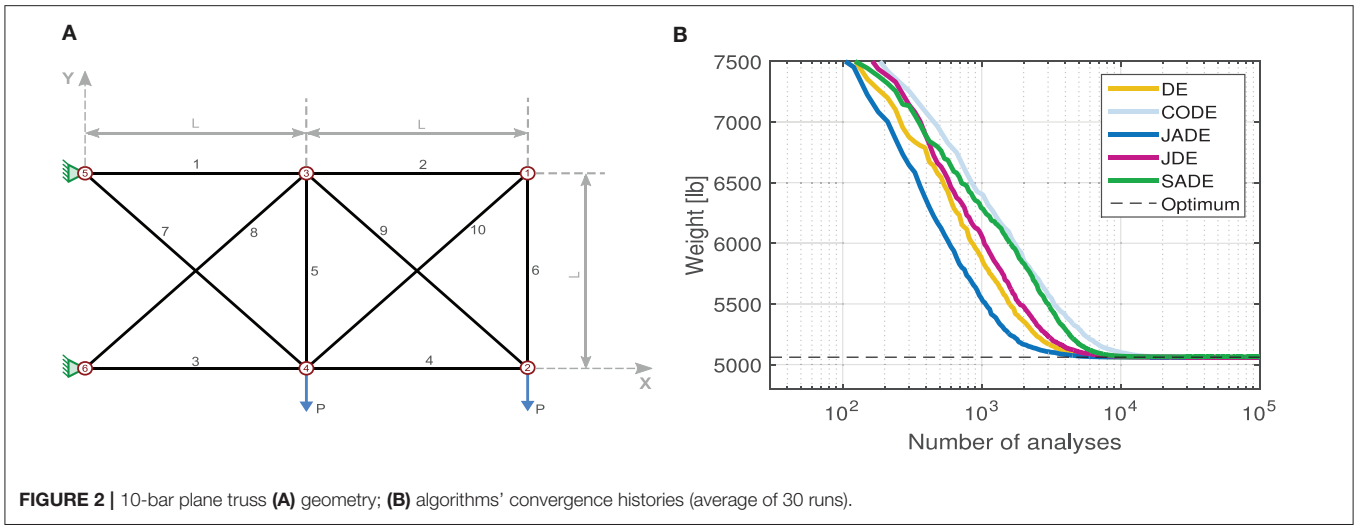

FIGURE 2 | 10-bar plane truss **(A)** geometry; **(B)** algorithms' convergence histories (average of 30 runs).

TABLE 5 | Algorithms' statistical parameters for the 10-bar plane truss problem.

	Algorithms									
	DE		**CODE**		**JDE**		**JADE**		**SADE**	
Area [in^2]	**Best design**		**Best design**		**Best design**		**Best design**		**Best design**	
1	30.5217		30.5218		30.5092		30.5218		30.6380	
2	0.1000		0.1000		0.1000		0.1000		0.1000	
3	23.1985		23.1999		23.1746		23.1999		23.2223	
4	15.2235		15.2229		15.2324		15.2229		15.1815	
5	0.1000		0.1000		0.1000		0.1000		0.1000	
6	0.5511		0.5514		0.5494		0.5514		0.5500	
7	7.4574		7.4572		7.4623		7.4572		7.4511	
8	21.0370		21.0364		21.0647		21.0364		20.9647	
9	21.5285		21.5284		21.5166		21.5284		21.5392	
10	0.1000		0.1000		0.1000		0.1000		0.1000	
Weight [lb]	**Best**	**Worst**	**Best**	**Worst**	**Best**	**Worst**	**Best**	**Worst**	**Best**	**Worst**
	5060.854	5060.898	5060.854	5060.854	5060.858	5076.674	5060.854	5061.372	5060.887	5079.279
	Mean ± Std		**Mean ± Std**		**Mean ± Std**		**Mean ± Std**		**Mean ± Std**	
	5060.862 ± 0.012		5060.854 ± 0.000		5061.444 ± 2.877		5060.886 ± 0.102		5064.092 ± 6.151	
	Median	**COV**	**Median**	**COV**	**Median**	**COV**	**Median**	**COV**	**Median**	**COV**
	5060.856	2.47E-06	5060.854	2.12E-11	5060.902	5.68E-04	5060.854	2.02E-05	5061.292	1.21E-03

choice of F than it is to the choice of CR. The study suggests the values of [0.5, 1] for F, [0.8, 1] for CR and $10D$ for NP, where D is the number of dimensions of the problem. The control parameters of the JDE algorithm are described as:

$$F_{i,G+1} = \begin{cases} F_l + rand_1 F_u, & \text{if } rand_2 < \tau_1 \\ F_{i,G}, & \text{otherwise} \end{cases} \quad (4)$$

$$CR_{i,G+1} = \begin{cases} rand_3, & \text{if } rand_4 < \tau_2 \\ CR_{i,G}, & \text{otherwise} \end{cases} \quad (5)$$

where $rand_j$ with $j \in \{1, 2, 3, 4\}$ are uniform random values in [0, 1] and τ_1, τ_2 represent probabilities to adjust factors F and CR with suggested values $\tau_1 = \tau_2 = 0.10$, while $F_l = 0.1$ and $F_u = 0.9$ and as a result the new F takes values in the range [0.1, 1]. The new CR takes values in [0, 1]. The new control parameter values take effect before the mutation and as a result they influence the mutation, crossover and selection operations of the new trial vector. The idea of JDE is that one does not need to guess good values of F and CR any more. The rules for self-adaptation of the two parameters are quite simple and easy to implement and use and as a result JDE does not increase significantly the complexity or the computational effort of the standard DE method. Experimental results have shown that JDE outperforms both the classic "DE/rand/1" and other schemes (Liu and Lampinen, 2005). The main contribution of JDE is that the

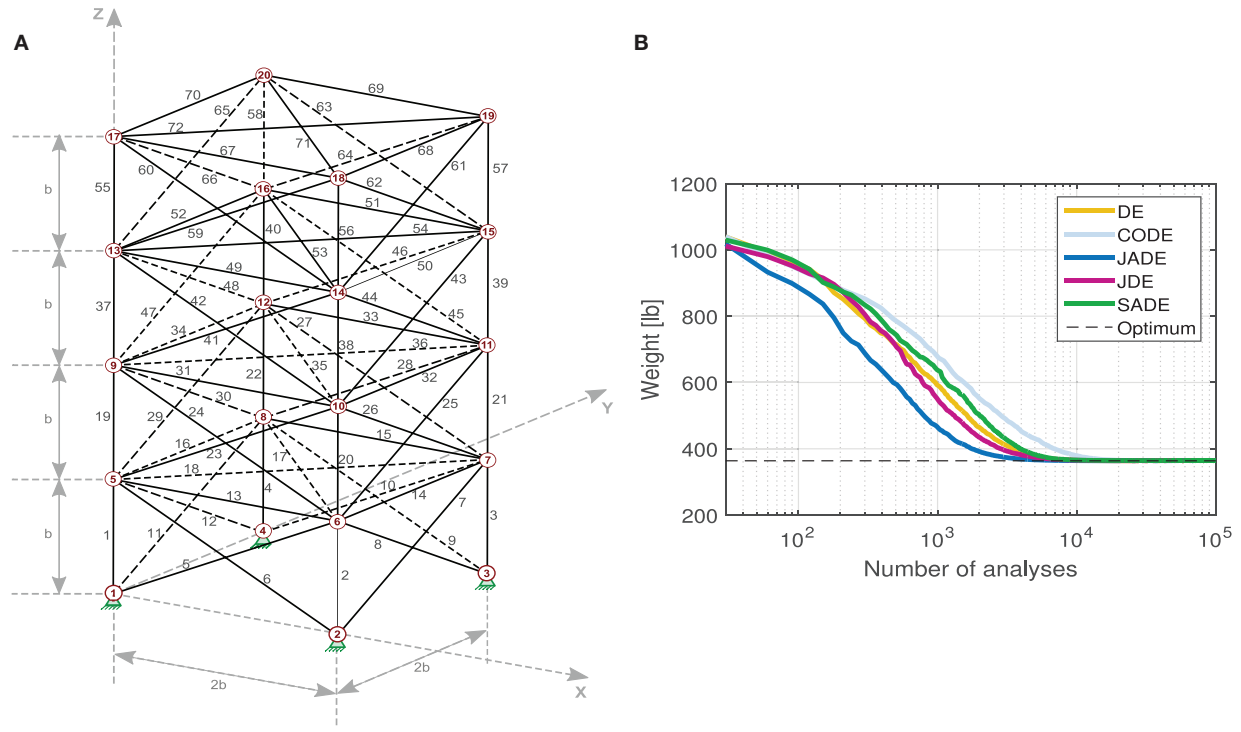

FIGURE 3 | 72-bar space truss **(A)** geometry; **(B)** algorithms' convergence histories (average of 30 runs).

TABLE 6 | Algorithms' statistical parameters for the 72-bar space truss problem.

	Algorithms				
	DE	**CODE**	**JDE**	**JADE**	**SADE**
Area [in^2]	**Best design**	**Best design**	**Best design**	**Best design**	**Best design**
1	1.8836	1.8873	1.8920	1.8868	1.8671
2	0.5172	0.5169	0.5172	0.5168	0.5192
3	0.0100	0.0100	0.0100	0.0100	0.0100
4	0.0100	0.0100	0.0100	0.0100	0.0100
5	1.2870	1.2901	1.2867	1.2943	1.2917
6	0.5177	0.5166	0.5174	0.5171	0.5150
7	0.0100	0.0100	0.0100	0.0100	0.0100
8	0.0100	0.0100	0.0100	0.0100	0.0100
9	0.5224	0.5214	0.5236	0.5248	0.5141
10	0.5176	0.5184	0.5176	0.5191	0.5196
11	0.0100	0.0100	0.0100	0.0100	0.0100
12	0.1153	0.1141	0.1174	0.1111	0.0999
13	0.1663	0.1665	0.1661	0.1668	0.1679
14	0.5359	0.5363	0.5355	0.5348	0.5417
15	0.4464	0.4455	0.4452	0.4464	0.4487
16	0.5773	0.5759	0.5724	0.5739	0.5809
Weight [lb]	**Best** / **Worst**	**Best** / **Worst**	**Best** / **Worst**	**Best** / **Worst**	**Best** / **Worst**
	363.825 / 363.893	363.824 / 363.835	363.826 / 363.908	363.826 / 364.026	363.862 / 365.665
	Mean ± Std	**Mean ± Std**	**Mean ± Std**	**Mean ± Std**	**Mean ± Std**
	363.839 ± 0.013	363.826 ± 0.003	363.845 ± 0.019	363.859 ± 0.051	364.145 ± 0.378
	Median / **COV**	**Median** / **COV**	**Median** / **COV**	**Median** / **COV**	**Median** / **COV**
	363.836 / 3.62E-05	363.825 / 6.89E-06	363.840 / 5.24E-05	363.839 / 1.40E-04	363.987 / 1.04E-03

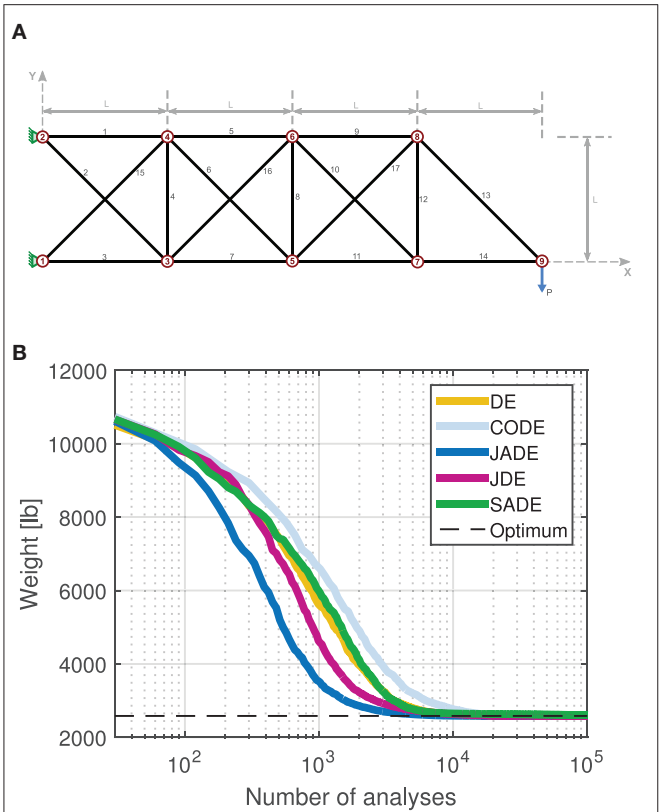

FIGURE 4 | 17-bar plane truss **(A)** geometry; **(B)** algorithms' convergence histories (average of 30 runs).

user no longer needs to guess good values of F and CR, which are problem-dependent.

3.4. JADE

The selection of the values of the mutation factor (F) and the crossover probability (CR) can significantly affect the performance of DE. Trial-and-error attempts for fine-tuning these control parameters can be successful but they require much time and effort and the result is always problem-specific. Researchers have suggested various self-adaptive mechanisms for dealing with this problem (Qin and Suganthan, 2005; Brest et al., 2006, 2007; Huang et al., 2006; Teo, 2006) in an effort to dynamically update the control parameters without having any prior knowledge of the characteristics of the problem.

JADE was proposed to improve the performance of the standard DE by implementing a new mutation scheme which updates control parameters in an adaptive way. The method introduces a new mutation scheme denoted as "DE/current-to-pbest" with an optional external archive. The algorithm, as described in Zhang and Sanderson (2009) utilizes not only the best solution, but a number of $100p\%, p \in (0, 1]$ good solutions. Compared to "DE/rand/k," greedy schemes such as the "DE/current-to-best/k" and the "DE/best/k" can benefit from their fast convergence by incorporating best solution information in the search procedure. However, this best solution information can also cause premature convergence problems due to reduced diversity in the population.

In the "DE/current-to-pbest/1" scheme (without archive), a mutation vector is generated as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_i(\mathbf{x}^p_{best,g} - \mathbf{x}_{i,g}) + F_i(\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \quad (6)$$

Where $\mathbf{x}^p_{best}$ is chosen randomly as one of the top 100p% individuals of the current population with p $\in$ (0, 1], and F_i is the mutation factor associated with x_i. In JADE, any of the top 100p% solutions can be randomly chosen to play the role of the single best solution in the standard "DE/current-to-best" scheme. This is because recently explored solutions that are not the best can still provide additional information about the desired search direction. In the "DE/current-to-pbest/1" scheme with archive, a mutation vector is generated as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_i(\mathbf{x}^p_{best,g} - \mathbf{x}_{i,g}) + F_i(\mathbf{x}_{r1,g} - \widetilde{\mathbf{x}}_{r2,g}) \quad (7)$$

Where $x_{i,g}$, $x_{r1,g}$ and $x^p_{best,g}$ are selected from the current population P in the usual way, while $\widetilde{x}_{r2,g}$ is randomly chosen individual (distinct from $x_{i,g}$ and $x_{r1,g}$) from the union $P \cup A$ of the current population and the set of archived inferior solutions A. The archive is initiated as empty. Consequently, after each iteration, the parent solutions that fail in the selection process are added to the archive. If the size of the archive exceeds a predefined value, for example NP, then some solutions are removed from the archive randomly. The "DE/current-to-pbest/1" scheme (without archive) is a special case of the "DE/current-to-pbest/1" scheme with archive, if we set the archive size equal to zero (i.e., empty archive). Numerical results have shown that JADE exhibits superior or at least comparable performance to other standard or adaptive DE algorithms (Zhang and Sanderson, 2009).

3.5. SADE

In SADE, both the trial vector generation schemes and their associated control parameter values can be gradually self-adapted according to their previous experiences of generating promising solutions (Qin et al., 2009). The method consists a self-adaptive DE variant, in which one trial vector generation scheme is selected from the candidate pool according to the probability learned from its success rate in generating improved solutions within a certain number of previous generations LP (learning period). The selected scheme is then applied to the corresponding target vector in order to obtain the trial vector. At each generation, the sum of the probabilities of choosing a scheme from the candidate pool are equal to 1.

In the beginning, all schemes have an equal probability to be selected, i.e., the probabilities with respect to each scheme are initialized as $1/K$, where K is the total number of schemes. The population size NP remains a user-specified parameter because it has to do with the complexity of the given optimization problem. The parameter F is approximated by a normal distribution with mean value 0.5 and a standard deviation of 0.3 which makes F fall within the range $[-0.4, 1.4]$ with a probability of 0.997. The control parameter K in the "DE/current-to-rand/1" scheme is randomly generated within [0,1] while CR is assumed to

TABLE 7 | Algorithms' statistical parameters for the 17-bar plane truss problem.

	Algorithms									
	DE		**CODE**		**JDE**		**JADE**		**SADE**	
Area [in^2]	**Best design**		**Best design**		**Best design**		**Best design**		**Best design**	
1	15.9247		15.9220		15.9334		15.9444		16.0150	
2	0.1000		0.1000		0.1000		0.1000		0.1029	
3	12.0678		12.0751		12.0650		12.0607		12.1435	
4	0.1000		0.1000		0.1000		0.1000		0.1000	
5	8.0542		8.0728		8.0616		8.0598		8.2137	
6	5.5646		5.5607		5.5598		5.5613		5.2921	
7	11.9416		11.9277		11.9116		11.9377		12.0136	
8	0.1000		0.1000		0.1000		0.1000		0.1000	
9	7.9393		7.9419		7.9789		7.9439		7.9410	
10	0.1000		0.1000		0.1000		0.1000		0.1000	
11	4.0570		4.0535		4.0585		4.0515		4.0535	
12	0.1000		0.1000		0.1000		0.1000		0.1000	
13	5.6488		5.6603		5.6558		5.6535		5.6138	
14	3.9925		4.0017		3.9987		4.0024		4.0490	
15	5.5652		5.5600		5.5523		5.5600		5.5988	
16	0.1000		0.1000		0.1000		0.1000		0.1104	
17	5.5928		5.5779		5.5819		5.5800		5.5517	
Weight [lb]	**Best**	**Worst**	**Best**	**Worst**	**Best**	**Worst**	**Best**	**Worst**	**Best**	**Worst**
	2581.893	2581.949	2581.890	2581.898	2581.895	2582.033	2581.890	2582.048	2582.578	2670.458
	Mean ± Std		**Mean ± Std**		**Mean ± Std**		**Mean ± Std**		**Mean ± Std**	
	2581.909 ± 0.013		2581.893 ± 0.002		2581.925 ± 0.030		2581.909 ± 0.030		2605.089 ± 21.226	
	Median	**COV**	**Median**	**COV**	**Median**	**COV**	**Median**	**COV**	**Median**	**COV**
	2581.907	5.06E-06	2581.893	8.12E-07	2581.915	1.16E-05	2581.899	1.17E-05	2600.607	8.15E-03

follow a normal distribution with mean value CR_m and standard deviation Std, where initially it is set $CR_m = 0.5$ and $Std = 0.1$. A set of random CR values is generated following the normal distribution and then applied to those target vectors to which the k-th scheme is assigned.

To adapt the CR values, memories $CRMemory_k$ are established to store those CR values with respect to the k-th scheme of successfully trial vectors generation, entering the next generation within the previous LP generations that keep the success and failure memories by storing CR values. During the first LP generations, CR values with respect to the k-th scheme are drawn from the normal distribution. At each generation after LP generations, the median value that stored in $CRMemory_k$ will be calculated to overwrite CR_{mk}. After evaluating the newly generated trial vectors, CR values in $CRMemory_k$ that correspond to earlier generations will be replaced by promising CR values obtained at the current generation with respect to the k-th scheme. The method is described in detail in Qin et al. (2009).

4. NUMERICAL EXAMPLES

The performance of each of the five optimization algorithms (the standard DE and the four DE variants) is examined in five well-known benchmark structural engineering test examples. The characteristics of each test example is given in **Table 1**. All test examples have been previously optimized by many researchers (Lee and Geem, 2004; Farshi and Alinia-ziazi, 2010; Gandomi et al., 2013; Talatahari et al., 2013) using various optimization algorithms.

For all test examples the objective is the minimization of the structural weight under constraints on stresses and displacements. The parameters of the optimization algorithms are the following: population size $NP = 30$, mutation factor $F = 0.6$, crossover probability $CR = 0.9$. More specifically, for the JDE variant, τ_1, and τ_2 probabilities are both taken equal to 0.1. JADE is used with the optional archive, with archive size equal to $NP = 30$, while $p = 0.05$. The maximum number of objective function evaluations is used as the termination criterion for all cases, with a value of 10^5. The value of μ used in Equation (2) is equal to 10^{10}. Furthermore, 30 independent runs have been conducted for each method examined and the convergence history results presented are the average results of the different runs. Other statistical quantities and measures are also presented, such as best, worst, mean, median, standard deviation, and coefficient of variation (COV).

4.1. 25-Bar Space Truss

The first test example is a 25-member space truss. The geometry of the structure is shown in **Figure 1A** where the dimensions a, b

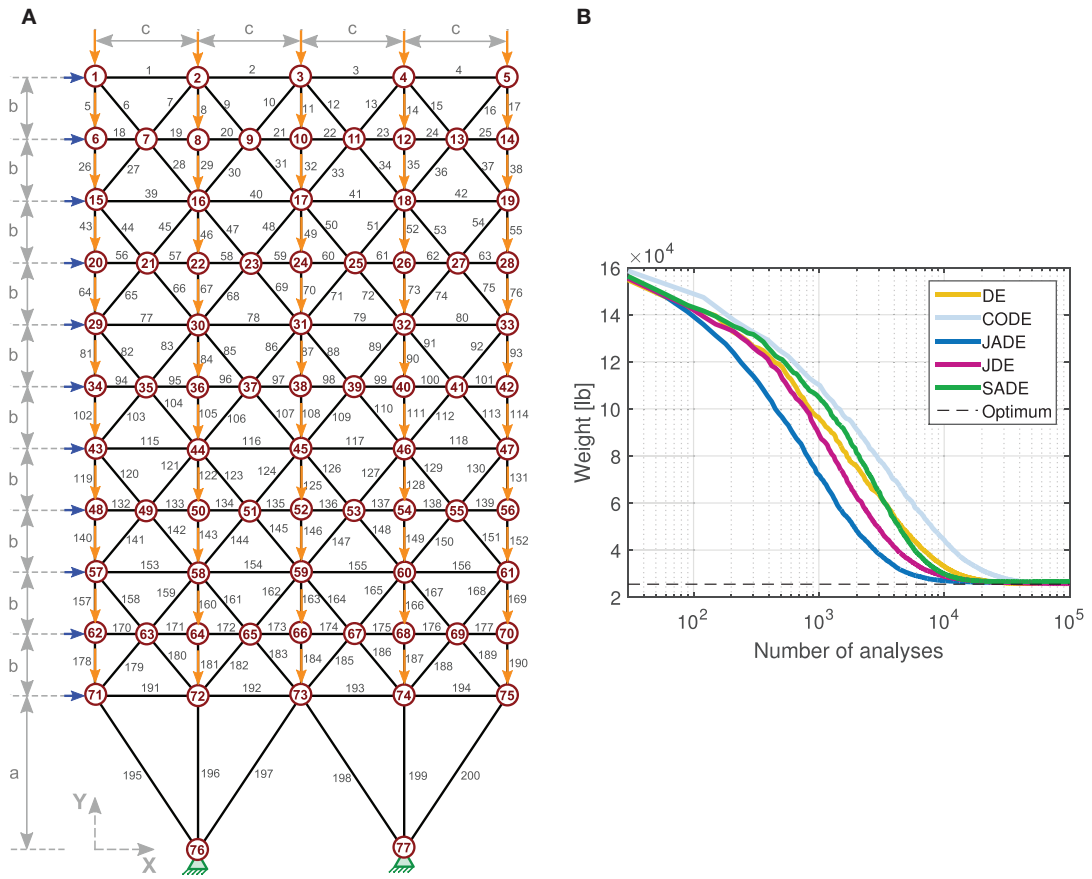

FIGURE 5 | 200-bar plane truss **(A)** geometry. Load Cases 1 and 2 are depicted with blue and orange arrows, respectively; **(B)** algorithms' convergence histories (average of 30 runs).

and h are equal to 200, 75, and 100 in, respectively. The density of the material is $\rho = 0.1$ lb/in^3 and the Young's modulus is $E =$ 10,000 ksi. There are two load cases, as shown in **Table 2**. There are eight groups where the members of the structure belong to. The cross section areas of each design group, in the range [0.01, 5] (in^2), are the design variables of the problem. The design variable groups and the stress constraints for each group are shown in **Table 3**. The maximum allowable displacement for each node is $d_{max} = \pm 0.35$ in every direction. The convergence history of the various DE methods is shown in **Figure 1B** as the average of 30 independent runs for each optimization algorithm.

Figure 1B shows the convergence history of the five algorithms. The horizontal dashed line represents the best solution ever found in the literature for this specific problem, for comparison purposes. The convergence histories reveal that all the five algorithms eventually manage to converge to approximately the same final result. However, the rate of convergence is different. JADE is the fastest, converging to the optimum earlier than the other algorithms. JADE, DE, and JDE appear to be ranked 1st, 2nd, and 3rd as far as the convergence speed is concerned. Then come SADE and CODE with similar convergence performance. **Table 4** reveals that all algorithms managed to find the same best solution (545.555). It is worth noting that three of them managed to find the same best solution in all 30 runs, as the best value and the worst value are the very same. Only JDE and SADE failed to deliver the best solution in all 30 runs, but again the difference is very small and the coefficient of variation has very small values equal to 1.92E-5 and 2.24E-4 for the two methods, respectively.

4.2. 10-Bar Plane Truss

The second test example is the standard benchmark 10-bar plane truss problem. The geometry of the structure is shown in **Figure 2A**. The characteristics of the structure are: Young's modulus $E = 10{,}000$ ksi, density of the material $\rho = 0.1$ lb/in^3, $L = 360$ in, $P = 100$ kip. There are 10 group members, i.e., each member of the structure belongs to its own group. The design variables represent the cross-section areas of each structural element in the interval [0.1, 35] (in^2). The maximum allowable stress (absolute value) is $\sigma_{allow} = 25$ ksi in tension or compression while the maximum allowable displacement in the $\pm x$ and $\pm y$ directions for each node is $d_{max} = 2$ in. The convergence history of the various DE methods is shown in **Figure 2B** as the average of 30 independent runs for each optimization algorithm.

Figure 2B shows that all methods converge to the optimum, but with different convergence rates. Again JADE shows the best convergence performance. The algorithms are ranked as JADE–DE–JDE–SADE–CODE from the fastest to the slowest in reaching the optimum. **Table 5** shows that all algorithms managed to find the optimum solution as their "best" run. DE and CODE were very successful in achieving the optimum result even in their "worst" runs. DE and CODE show very consistent performance in all runs and JADE is also good in that. JDE and SADE have the most variation in their results with values of *COV* equal to 5.68E-4 and 1.21E-3, respectively. It is worth noting that CODE exhibits the slowest convergence but simultaneously the most consistent behavior in reaching the same optimum value in all runs.

4.3. 72-Bar Space Truss

Figure 3A shows the third test example, a space truss with 72 members. The basis of the structure is a rectangle with $2b = 120$ in, while the total height is $4b = 240$ in. Nodes 1, 2, 3, and 4 are pinned on the ground while all the other nodes are free to move in all three directions. Each member has the following material characteristics: Young's modulus is $E = 10{,}000$ ksi and material density $\rho = 0.1$ lb/in^3. Two load cases have been considered. There are 16 groups of structural members. The design variables represent the cross-section areas of each group. There is no upper limit, while the lower limit is 0.01 in^2 for each design variable. The constraints are imposed on displacements and stresses. The maximum allowable stress (as an absolute value) is $\sigma_{allow} = 25$ ksi, in tension or compression, while the maximum allowable displacement in the $\pm x$ and $\pm y$ directions is $d_{max} = 0.25$ in, for each node. The convergence history of the various DE variants is shown in **Figure 3B** as the average of 30 independent runs for each method.

Again, **Figure 3B** shows that all methods converge to the same result with JADE being the fastest. The ranking from fastest to slowest convergence is now JADE–JDE–DE–SADE–CODE. Once again, CODE is very successful in obtaining almost the same optimum value in all runs ($COV = 6.89$E-6) as shown in **Table 6**. SADE exhibits the worst performance in terms of variation of the results ($COV = 1.04$E-3). In any case, all algorithms are again quite successful in almost all their 30 runs.

4.4. 17-Bar Plane Truss

The fourth test example is a 17-bar plane truss. The geometry of the structure is shown in **Figure 4A** where dimension L is equal to 100 in. The density of the material is $\rho = 0.268$ lb/in^3 and the Young's modulus is $E = 30{,}000$ ksi. The structure is subjected to the concentrated load P at node 9 equal to 100 kip. There is no grouping of members in this example, i.e., each member belongs to its own group and there are 17 groups in total. The design parameters, in the interval [0.1, 35] (in^2), are the cross-section areas of each member group. The maximum allowable stress (absolute value) is $\sigma_{allow} = 50$ ksi in tension or compression, while the maximum allowable displacement in the $\pm x$ and $\pm y$ directions is d_{max}=2 in, for each node. The convergence history of the various DE methods is shown in **Figure 4B** as the average of 30 independent runs for each optimization algorithm.

TABLE 8 | 200-bar space truss: Design variable groups.

Design variable	Members
1	1, 2, 3, 4
2	5, 8, 11, 14, 17
3	19, 20, 21, 22, 23, 24
4	18, 25, 56, 63, 94, 101, 132, 139, 170, 177
5	26, 29, 32, 35, 38
6	6, 7, 9, 10, 12, 13, 15, 16, 27, 28, 30, 31, 33, 34, 36, 37
7	39, 40, 41, 42
8	43, 46, 49, 52, 55
9	57, 58, 59, 60, 61, 62
10	64, 67, 70, 73, 76
11	44, 45, 47, 48, 50, 51, 53, 54, 65, 66, 68, 69, 71, 72, 74, 75
12	77, 78, 79, 80
13	81, 84, 87, 90, 93
14	95, 96, 97, 98, 99, 100
15	102, 105, 108, 111, 114
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

JADE is the fastest algorithm to converge, as shown in **Figure 4B**, while all algorithms reach the same solution eventually. **Table 7** shows that CODE exhibits the most stable performance with $COV = 8.12$E-7 as far as the final optimum is concerned. SADE has the most variation of the results with $COV = 8.15$E-3.

4.5. 200-Bar Plane Truss

The fifth example is a space truss consisting of 200 members. The geometry of the structure is shown in **Figure 5A** where dimensions a, b, and c are equal to 360, 144, and 240 in, respectively. The density of the material is $\rho = 0.283$ lb/in^3 and the Young's modulus is $E = 30{,}000$ ksi. There are three independent load cases (LC) as shown in **Figure 5A**: (i) LC1: 1.0 kip acting in the positive x-direction (blue arrows); (ii) LC2: 10 kips acting in the negative y-direction (orange arrows) and (iii) LC3: the first two load cases acting together.

TABLE 9 | Algorithms' statistical parameters for the 200-bar plane truss problem.

	Algorithms				
	DE	**CODE**	**JDE**	**JADE**	**SADE**
Area [in^2]	**Best design**	**Best design**	**Best design**	**Best design**	**Best design**
1	0.1441	0.1521	0.1340	0.1075	0.1306
2	0.9385	0.9443	0.9417	1.0709	0.9241
3	0.1033	0.1003	0.1021	0.100	0.1474
4	0.1001	0.1001	0.1003	0.1000	0.1000
5	1.9385	1.9420	1.9434	1.9700	1.9241
6	0.3006	0.3019	0.2948	0.2712	0.3086
7	0.1272	0.1043	0.1365	0.1275	0.1095
8	3.1152	3.1276	3.1028	3.0277	3.1119
9	0.1033	0.1052	0.1255	0.2896	0.1003
10	4.1154	4.1262	4.1021	4.0299	4.1119
11	0.4231	0.4144	0.4273	0.4731	0.4123
12	0.1442	0.1283	0.1034	0.1014	0.1772
13	5.4695	5.4412	5.4106	5.3787	5.3972
14	0.1055	0.1718	0.1527	0.1992	0.3181
15	6.4696	6.441	6.4171	6.3758	6.3971
16	0.5638	0.5894	0.5622	0.5847	0.6827
17	0.1486	0.1786	0.3149	0.1832	0.2080
18	7.9994	8.018	7.9494	7.9057	8.1121
19	0.1052	0.1419	0.2886	0.1489	0.1481
20	8.9995	9.0167	8.9541	8.9067	9.1119
21	0.7336	0.7937	0.9325	0.7762	0.8681
22	0.8221	0.5624	0.2133	0.4346	0.7341
23	11.1663	11.1755	11.1696	10.8659	11.5626
24	0.1109	0.1405	0.1037	0.1000	0.3027
25	12.1669	12.1750	12.1457	11.8743	12.5632
26	1.3310	1.2286	1.0143	1.0750	1.5131
27	5.5193	5.6929	6.2548	6.2908	4.6757
28	10.1646	10.2696	10.5478	11.2671	9.5501
29	14.4981	14.3823	14.1036	13.8505	15.1318
Weight [lb]	**Best** **Worst**	**Best** **Worst**	**Best** **Worst**	**Best** **Worst**	**Best** **Worst**
	25505.765 26160.407	25523.011 25919.182	25579.000 26541.229	25659.736 26813.984	25734.695 27636.608
	Mean ± Std	**Mean ± Std**	**Mean ± Std**	**Mean ± Std**	**Mean ± Std**
	25715.962 ± 185.129	25716.826 ± 120.334	25867.595 ± 191.068	26175.928 ± 301.066	26556.821 ± 518.116
	Median **COV**	**Median** **COV**	**Median** **COV**	**Median** **COV**	**Median** **COV**
	25688.199 7.20E-03	25728.346 4.68E-03	25847.327 7.39E-03	26165.435 1.15E-02	26522.826 1.95E-02

The members of the structure are divided into 29 groups in total, as shown in **Table 8**. The cross-section areas of each member group, in the interval [0.1, 35] (in^2), are the design variables of the problem. The stress of each member, in absolute terms, needs to be less than $\sigma_{allow} = 10$ ksi (in tension or compression). No displacement limit is set for this problem.

The convergence history of the various DE methods is shown in **Figure 5B** as the average of 30 independent runs for each optimization algorithm.

In this problem, JADE is again the fastest algorithm to converge to the final result, followed by JDE. **Table 9** shows that CODE exhibits the most stable performance with $COV = 4.68\text{E-}3$ as far as the final optimum is concerned. SADE has the most variation of the results (worst performance) with $COV = 1.95\text{E-}2$. The difference between the best and the worst solutions are 396.171 in CADE and 1901.913 in SADE.

5. DISCUSSION

The convergence history plots show that in all problems, JADE converges faster to the final result, followed by JDE and DE which exhibit similar performance with each other in terms of convergence speed. It appears that in the JADE

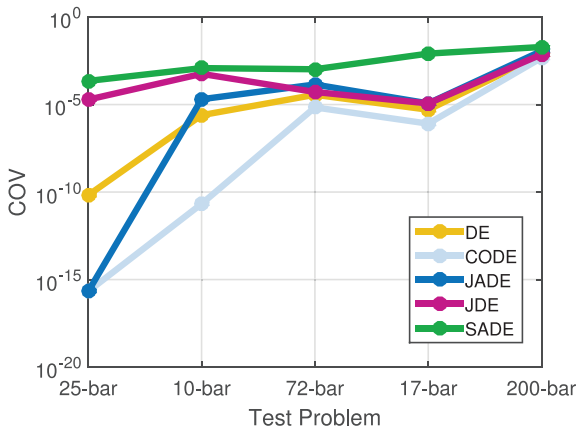

FIGURE 6 | Coefficient of variation (COV) values for each problem and algorithm.

algorithm, the parameter adaptation is beneficial in accelerating the convergence performance of DE by automatically updating the control parameters during the optimization search. This has effect even in the very first iterations of the algorithm, making JADE clearly stand out from the other variants, from the beginning of the search process. CODE exhibits the slowest convergence performance among the five algorithms. All algorithms manage to reach the final result with efficient accuracy in most of the optimization runs. CODE shows the most robust performance among all algorithms as it manages to find the same solution in almost all the optimization runs, with very small variation of the results among the 30 runs. CODE has the lowest values of the coefficient of variation (COV) metric in all five test problems, as shown in **Figure 6** (y-values in logarithmic scale). SADE shows the most variation of the results (highest values of COV). CODE guarantees very robust final results but at the expense of slower convergence speed, while JADE manages to provide satisfactory convergence speed and a good quality of the final optimum result. The performance of the standard DE algorithm is also rather good as DE manages to outperform some of the other adaptive schemes.

6. CONCLUSIONS

The present study investigates the performance of five DE variants, DE, CODE, JDE, JADE, and SADE, in dealing with constrained structural optimization problems. The structures examined are 2D and 3D trusses under single or multiple loading conditions. The constraints are imposed on stresses and displacements while the objective is to minimize the weight of each structure. Five well-known benchmark problems of truss structures have been considered in the comparison. In order to ensure neutrality and to compare the results of the various algorithms, 30 runs have been conducted for each test problem and each algorithm and various statistical metrics have been calculated.

From the statistical analysis that has been conducted, it can be concluded that the CODE algorithm guarantees robustness but with slower convergence speed, while JADE manages to provide a satisfactory compromise between the convergence speed and the quality of the final optimum result, compared to the other four competitors. Surprisingly, even in the case of the standard DE algorithm the performance is rather good as DE manages to outperform some other adaptive schemes, such as SADE, in terms of the quality of the final result and the convergence speed achieved. The same finding has also been reported by Charalampakis and Tsiatas (2019).

DATA AVAILABILITY STATEMENT

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

AUTHOR CONTRIBUTIONS

MG did the data collection, contributed in the numerical analysis of the test examples, analyzed and interpreted the data, and wrote parts of the paper. VP contributed in the conception and design of the work, analyzed and interpreted the data, wrote parts of the paper, and supervised the overall research work. All authors have made a substantial, direct and intellectual contribution to the work, and approved it for publication.

REFERENCES

Abbas, Q., Ahmad, J., and Jabeen, H. (2015). A novel tournament selection based differential evolution variant for continuous optimization problems. *Math. Probl. Eng.* 2015:205709. doi: 10.1155/2015/205709

Brest, J., Bokovic, B., Greiner, S., Aumer, V., and Maucec, M. S. (2007). Performance comparison of self-adaptive and adaptive differential evolution algorithms. *Soft Comput.* 11, 617–629. doi: 10.1007/s00500-006-0124-0

Brest, J., Greiner, S., Boskovic, B., Mernik, M., and Zumer, V. (2006). Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Trans. Evol. Comput.* 10, 646–657. doi: 10.1109/TEVC.2006.872133

Charalampakis, A. E., and Tsiatas, G. C. (2019). Critical evaluation of metaheuristic algorithms for weight minimization of truss structures. *Front. Built Environ.* 5:113. doi: 10.3389/fbuil.2019.00113

Eiben, A., Hinterding, R., and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Trans. Evol. Comput.* 3, 124–141. doi: 10.1109/4235.771166

Farshi, B., and Alinia-ziazi, A. (2010). Sizing optimization of truss structures by method of centers and force formulation. *Int. J. Solids Struct.* 47, 2508–2524. doi: 10.1016/j.ijsolstr.2010.05.009

Gamperle, R., Mller, S. D., and Koumoutsakos, P. (2002). *A Parameter Study for Differential Evolution*. Technical Report WSEAS NNA-FSFS-EC 2002, Interlaken.

Gandomi, A. H., Talatahari, S., Yang, X.-S., and Deb, S. (2013). Design optimization of truss structures using cuckoo search algorithm. *Struct. Design Tall Spec. Build.* 22, 1330–1349. doi: 10.1002/tal.1033

Huang, V. L., Qin, A. K., and Suganthan, P.N. (2006). "Self-adaptive differential evolution algorithm for constrained real- parameter optimization," in *2006 IEEE International Conference on Evolutionary Computation* (Vancouver, BC), 17–24. doi: 10.1109/CEC.2006.1688285

Lee, K. S., and Geem, Z. W. (2004). A new structural optimization method based on the harmony search algorithm. *Comput. Struct.* 82, 781–798. doi: 10.1016/j.compstruc.2004.01.002

Liu, J., and Lampinen, J. (2002). "On setting the control parameter of the differential evolution method," in *Proceedings of MENDEL 2002, 8th International Conference Soft Computing* (Brno), 11–18.

Liu, J., and Lampinen, J. (2005). A fuzzy adaptive differential evolution algorithm. *Soft Comput.* 6, 448–462. doi: 10.1007/s00500-004-0363-x

Mallipeddi, R., Suganthan, P. N., Pan, Q. K., and Tasgetiren, M. F. (2011). Differential evolution algorithm with ensemble of parameters and mutation strategies. *Appl. Soft Comput.* 11, 1679–1696. doi: 10.1016/j.asoc.2010.04.024

Mezura-Montes, E., Velzquez-Reyes, J., and Coello, C. A. C. (2006). "A comparative study of differential evolution variants for global optimization," in *GECCO '06: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* (Seattle, WA), 485–492. doi: 10.1145/1143997.1144086

Plevris, V. and Papadrakakis, M. (2011). A hybrid particle swarm–gradient algorithm for global structural optimization. *Comput. Aided Civil Infrastruct. Eng.* 26, 48–68. doi: 10.1111/j.1467-8667.2010.00664.x

Qin, A. K., Huang, V. L., and Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Trans. Evol. Comput.* 13, 398–417. doi: 10.1109/TEVC.2008.927706

Qin, A. K., and Suganthan, P. N. (2005). "Self-adaptive differential evolution algorithm for numerical optimization," in *2005 IEEE Congress on Evolutionary Computation* (Edinburgh), 1785–1791. doi: 10.1109/CEC.2005.1554904

Shukla, R., Hazela, B., Shukla, S., Prakash, R., and Mishra K. K. (2017). "Variant of differential evolution algorithm," in *Advances in Computer and Computational Sciences. Advances in Intelligent Systems and Computing*, Vol. 553, eds S. Bhatia, K. Mishra, S. Tiwari, and V. Singh (Singapore: Springer), 601–608. doi: 10.1007/978-981-10-3770-2_56

Storn, R., and Price, K. (1995). *Differential Evolution—A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces*. Technical Report, Berkeley, CA. Available online at: https://www.icsi.berkeley.edu/ftp/global/global/pub/techreports/1995/tr-95-012.pdf

Storn, R., and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* 11, 341–359. doi: 10.1023/A:1008202821328

Talatahari, S., Kheirollahi, M., Farahmandpour, C., and Gandomi, A. H. (2013). A multi-stage particle swarm for optimum design of truss structures. *Neural Comput. Appl.* 23, 1297–1309. doi: 10.1007/s00521-012-1072-5

Teo, J. (2006). Exploring dynamic self-adaptive populations in differential evolution. *Soft Comput.* 10, 673–686. doi: 10.1007/s00500-005-0537-1

Wang, Y., Cai, Z., and Zhang, Q. (2011). Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Trans. Evol. Comput.* 15, 55–66. doi: 10.1109/TEVC.2010.2087271

Zhang, J., and Sanderson, A. C. (2009). Jade: Adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* 13, 945–958. doi: 10.1109/TEVC.2009.2014613

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.