Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 00 (2019) 000–000

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

1st International Conference on Optimization-Driven Architectural Design (OPTARCH 2019)

On the Performance of Differential Evolution Variants in Constrained Structural Optimization

Manolis Georgioudakis[a] and Vagelis Plevris[b]*

[a]*School of Civil Engineering, National Technical University of Athens (NTUA), Zografou Campus, 15780 Athens, Greece*
[b]*Department of Civil Engineering and Energy Technology, OsloMet—Oslo Metropolitan University, Pilestredet 35, Oslo 0166, Norway*

Abstract

Constrained optimization is a highly important field of engineering as most real-world optimization problems are associated with one or several constraints. Such problems are often challenging to solve due to their complexity and high nonlinearity. Differential evolution (DE) is arguably one of the most versatile and stable population-based search algorithms that exhibits robustness to multi-modal problems and has shown to be very efficient when solving constrained global optimization problems. In this paper we investigate the performance of several DE variants existing in the literature such as the traditional 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. The performance of each DE variant is quantified by using three well-known benchmark structures in 2D and 3D. It is shown that JADE, which updates control parameters in an adaptive way, truly exhibits superior performance and outperforms the other DE variants in all the cases examined.

Keywords: structural optimization; differential evolution; de; code; jade; jde; sade

* Corresponding author. Tel.: +47 67238853
E-mail address: vageli@oslomet.no

1. Introduction

Differential Evolution (DE), originally proposed by Storn and Price [1] is a popular optimization metaheuristic which exhibits very good performance in a wide variety of problems from various scientific fields. DE, like other evolutionary algorithms is population-based using a stochastic search technique employing mutation, crossover and selection operators to move the population towards the global optimum. In recent years, DE has gained increasing interest for solving optimization problems over continuous space in many scientific and engineering fields [2-7].

DE has only a few parameters to adjust. These are called control parameters and include the *population size NP*, ($NP \geq 4$), the *mutation factor* (or *differential weight*) $F \in [0, 2]$ and the *crossover probability* (or *crossover control parameter*) $CR \in [0, 1]$. The mutation factor F is a positive control parameter for scaling the difference vector. The performance of DE in a specific problem depends largely on both the trial vector generation strategy and the choice of the control parameters. First, one needs to choose the trial vector generation strategy and then one has to adjust the control parameters for the optimization problem at hand. Finding the right control parameters is not a trivial task and it can become difficult and time consuming for specific problems. This has drawn the attention of researchers who have been studying and developing new DE variants with adaptive and self-adaptive control parameters. In adaptive parameter control [8], the parameters are adapted based on feedback received during the search process while in self-adaptive control, the parameters themselves are encoded into the chromosome and evolved from generation to generation.

In the present paper we investigate the performance of several popular DE variants existing in the literature, namely the traditional DE [1], the composite differential evolution (CoDE) [5], the self-adaptive control parameters differential evolution (jDE) [2], the adaptive differential evolution with optional external archive (JADE) [4] and the self-adaptive differential evolution (SaDE) [3], for handling constrained structural optimization problems. In particular, we examine various 2D and 3D truss benchmark structures where the weight of the structure needs to be minimized subject to constraints on stresses and displacements.

The remainder of the paper is organized as follows. The standard DE and the most frequently used mutation strategies are presented in Section 2. The different DE variants examined in the study are presented in Section 3. Section 4 describes the numerical examples and the relevant results while in Section 5 there is a discussion of the results and the conclusions of the work.

2. Standard DE and various mutation strategies

DE optimizes a problem by maintaining a population of candidate solutions and creating new candidate solutions (individuals) by combining existing ones according to some simple formulas, and then keeping whichever candidate solution has the best objective value, i.e. if the new position of an individual is improved then it is accepted and forms part of the population, otherwise the new position is discarded. The process is repeated until a termination criterion is satisfied.

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

- Initialize all *NP* individuals with random positions in the search space
- Until a termination criterion is satisfied, repeat:
  - For each individual $\boldsymbol{x}_i$ in the population, do:
    - Pick three individuals $\boldsymbol{x}_{r1}$, $\boldsymbol{x}_{r2}$, $\boldsymbol{x}_{r3}$ from the population at random. They must be distinct from each other as well as from individual $\boldsymbol{x}_i$, i.e. $r_1 \neq r_2 \neq r_3 \neq i$
    - Form the donor vector using the formula (mutation operation)

$$\boldsymbol{v}_i = \boldsymbol{x}_{r1} + F \cdot \left(\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3} \right) \tag{1}$$

    - The trial vector $\boldsymbol{u}_i$ is developed either from the elements of the target vector $\boldsymbol{x}_i$ or the elements of the donor vector $\boldsymbol{v}_i$ as follows

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } r_{i,j} \le CR \text{ or } j = j_{rand} \\ x_{i,j} & \text{otherwise} \end{cases} \quad (2)$$

where $i = \{1, \ldots, NP\}$, $j = \{1, \ldots, D\}$, $r_{i,j} \sim U(0,1)$ is a uniformly distributed random number which is generated for each j and $j_{rand} \in \{1, \ldots, D\}$ is a random integer. j_{rand} is used to ensure that $\boldsymbol{u}_i \neq \boldsymbol{x}_i$ in all cases

- In the trial vector $\boldsymbol{u}_i$, if any element $u_{i,j}$ (corresponding to dimension j) is out of bounds, i.e. either above the upper limit or below the lower limit for the specific dimension, then set this element to the upper limit or the lower limit, respectively
- If $f(\boldsymbol{u}_i) \le f(\boldsymbol{x}_i)$ then replace the individual $\boldsymbol{x}_i$ in the population with the trial vector $\boldsymbol{u}_i$

At each generation G, DE employs the mutation operator to produce the donor vector $\boldsymbol{v}_i$ for each individual $\boldsymbol{x}_i$ in the current population. For each target vector $\boldsymbol{x}_i = \{x_{i,1}, \ldots, x_{i,D}\}$ at generation G, its associated mutant vector $\boldsymbol{v}_i = \{v_{i,1}, \ldots, v_{i,D}\}$ can be generated with a specific mutation strategy. The five most frequently used mutation strategies in DE are described in Table 1.

Table 1. Most frequently used mutation strategies in DE.

Strategy	Equation used
"DE/rand/1"	$\boldsymbol{v}_i = \boldsymbol{x}_{r1} + F\left(\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}\right)$
"DE/best/1"	$\boldsymbol{v}_i = \boldsymbol{x}_{best} + F\left(\boldsymbol{x}_{r1} - \boldsymbol{x}_{r2}\right)$
"DE/current-to-best/1"	$\boldsymbol{v}_i = \boldsymbol{x}_i + F\left(\boldsymbol{x}_{best} - \boldsymbol{x}_i\right) + F\left(\boldsymbol{x}_{r1} - \boldsymbol{x}_{r2}\right)$
"DE/rand/2"	$\boldsymbol{v}_i = \boldsymbol{x}_{r1} + F\left(\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}\right) + F\left(\boldsymbol{x}_{r4} - \boldsymbol{x}_{r5}\right)$
"DE/best/2"	$\boldsymbol{v}_i = \boldsymbol{x}_{best} + F\left(\boldsymbol{x}_{r1} - \boldsymbol{x}_{r2}\right) + F\left(\boldsymbol{x}_{r3} - \boldsymbol{x}_{r4}\right)$

In the above formulas, the indices r_1, r_2, r_3, r_4 and r_5 are mutually exclusive random integers generated within the range [1, NP] which 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 randomly generated once for each mutant vector. $\boldsymbol{x}_{best}$ is the best individual vector at the generation G, i.e. the individual of the current population associated with the best objective value.

3. DE variations examined in the study

3.1. CODE

It has been observed that the trial vector generation strategies and control parameters have a significant influence on the performance of DE. The various trial vector generation strategies and control parameter settings can show distinct advantages and, therefore, they can be effectively combined to solve different kinds of problems. Composite DE (CoDE) [5] is based on the idea of randomly combining several trial vector generation strategies with a number of control parameter settings at each generation, to create new trial vectors. In particular, the method uses three trial vector generation strategies and three control parameter settings, randomly combining them to generate trial vectors in an effort to improve performance [5]. The structure of CoDE is simple, and the algorithm is rather easy to implement.

The three trial vector generation strategies of the method are: (1) "rand/1/bin"; (2) "rand/2/bin"; (3) "current-to-rand/1". It has to be noted that in the "current-to-rand/1" strategy, the binominal crossover operator is not applied. The three control parameter settings are: (1) [F = 1.0, C_r = 0.1]; (2) [F = 1.0, C_r = 0.9]; (3) [F = 0.8, C_r = 0.2]. At each generation, each trial vector generation strategy is used to create a new trial vector with a control parameter setting

randomly chosen from the parameter candidate pool. As a result, three trial vectors are generated for each target vector. Then the best one enters the next generation if it is better than its target vector [5].

3.2. JDE

The standard DE includes a set of parameters which are kept fixed throughout the entire evolutionary process, that need to be adjusted for every single optimization problem. The process can be quite demanding and more difficult than expected [9]. According to [10], the effectiveness, efficiency, and robustness of the DE algorithm are very sensitive to the values of these control parameters, while some parameters may work well in a problem but not so well on other problems, which makes the selection of the parameters a problem-specific question. According to [1], DE behavior is more sensitive to the choice of F than it is to the choice of CR. The study suggests values in the region [0.5, 1] for F, [0.8, 1] for CR and the value of $10 \cdot D$ for NP, where D is the number of dimensions of the problem.

JDE is a DE variant which features self-adaptive control parameter settings and has showed good performance on numerical benchmark problems [2]. JDE uses the idea of the *evolution of the evolution*, i.e. using an evolution process to fine tune the optimization algorithm parameters. Although the idea sounds promising, the proof of convergence of EAs with self-adaptation is a difficult topic. 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. The control parameters of the JDE algorithm are described as:

$$F_{i,G+1} = \begin{cases} F_l + rand_1 \cdot F_u, & \text{if } rand_2 < \tau_1 \\ F_{i,G}, & \text{otherwise} \end{cases} \tag{3}$$

$$CR_{i,G+1} = \begin{cases} rand_3, & \text{if } rand_4 < \tau_2 \\ CR_{i,G}, & \text{otherwise} \end{cases} \tag{4}$$

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 $F_u = 0.9$. As a result, the updated F takes values in the range [0.1, 1] while the updated CR takes values in [0, 1]. The new 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 behind JDE is that one does not need to guess good values of F and CR anymore. The rules for self-adaptation of the two parameters are quite simple and easy to program and use and as a result JDE does not increase significantly the complexity or the computational effort in comparison to the standard DE. Experimental results have shown that JDE outperforms the classic "DE/rand/1/bin" scheme [11], among other schemes also.

3.3. JADE

The performance of DE is in many cases dependent on the control parameters such as the mutation factor and the crossover probability. Trial-and-error attempts for fine-tuning the control parameters can be successful but they require much time and effort. Researchers have suggested various self-adaptive mechanisms for dealing with this problem [2, 12-16] in an effort to dynamically update the control parameters without prior knowledge of the characteristics of the optimization problem. JADE was proposed to improve the performance of the standard DE by implementing a new mutation strategy denoted as "DE/current-to-pbest" which updates control parameters in an adaptive way, with an optional external archive. This method can be considered as a generalization of the classic "DE/current-to-best" technique.

The algorithm, as described in [4] utilizes not only the best solution, but a number of good solutions. Any of the top $100 \cdot p\%$, $p \in (0, 1]$, solutions can be selected (randomly) in "DE/current-to-pbest" to play the role of the single best solution in "DE/current-to-best". This can be beneficial because recently explored solutions that are not the very best can still provide additional information about the desired search direction. Compared to "DE/rand/k", greedy strategies 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. However, this best solution information can also cause premature convergence problems due to the resultant reduced population diversity. The "DE/current-to-pbest/1" strategy (without archive) is a special case of the "DE/current-to-pbest/1" strategy with archive, if we set the archive size equal to zero (empty archive). Details on the method and how exactly it is implemented can be found in [4] where it was also shown that JADE is superior or at least comparable to other classic or adaptive DE algorithms.

3.4. SADE

SaDE is a self-adaptive DE where both trial vector generation strategies and their associated control parameter values are gradually self-adapted by learning from their previous experiences in generating promising solutions [3]. In the SaDE algorithm, with respect to each target vector in the current population, one trial vector generation strategy 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. Then the selected strategy is applied to the corresponding target vector to obtain a trial vector. At each generation, the probabilities of choosing each strategy in the candidate pool are summed to 1.

In the beginning, the probabilities with respect to each strategy are initialized as $1/K$, where K is the number of strategies, i.e., all strategies have an equal probability to be chosen. The parameter NP remains a user-specified parameter because it highly relies on the complexity of a given problem. On the other hand, F is approximated by a normal distribution with mean value 0.5 and a standard deviation of 0.3 denoted by N(0.5, 0.3) [3]. As a result, F falls in the range [-0.4, 1.4] with probability equal to 0.997. The control parameter K in the "DE/current-to-rand/1" strategy is randomly generated within [0, 1]. The CR is assumed to follow a normal distribution with mean CR_m and standard deviation Std, where CR_m is initialized as 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 strategy is assigned. To adapt the CR, memories $CRMemory_\mathrm{k}$ are established to store those CR values with respect to the k-th strategy that have generated trial vectors successfully entering the next generation within the previous LP generations.

During the first LP generations, CR values with respect to the k-th strategy are generated by the normal distribution. At each generation after LP generations, the median value stored in $CRMemory_\mathrm{k}$ will be calculated to overwrite CRm_k. Then CR values can be generated according to the normal distribution when applying the k-th strategy. After evaluating the newly generated trial vectors, CR values in $CRMemory_\mathrm{k}$ that correspond to earlier generations will be replaced by promising CR values obtained at the current generation with respect to the k-th strategy. Details on the method and its implementation can be found in [3].

4. Numerical examples

The performance of each of the five optimization algorithms (the standard DE and the four DE variants) is examined in three well-known benchmark structural engineering test examples:

1. 10-bar plane truss with 10 design variables;
2. 25-member space truss with 8 design variables;
3. 72-member space truss with 16 design variables.

For all test examples the tuning parameters of the optimization algorithms are selected as follows: the population size NP, the mutation factor F and the crossover probability CR are taken equal to 30, 0.6 and 0.9, respectively. 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 the size of the archive equal to NP = 30, while p = 0.05. The maximum number of function evaluations is used as the stopping criterion for all cases, with a value of 10^5. Furthermore, because of the stochastic nature of the algorithms, for each method, 30 independent runs have been done and the convergence history results presented are the average results of the different runs.

For a nonlinear optimization problem with inequality constraints, a common practice of incorporating constraints is to define a penalty function so that the constrained problem is transformed into an unconstrained problem. The following penalty formulation is used to handle the constraints of optimization problems in the present study:

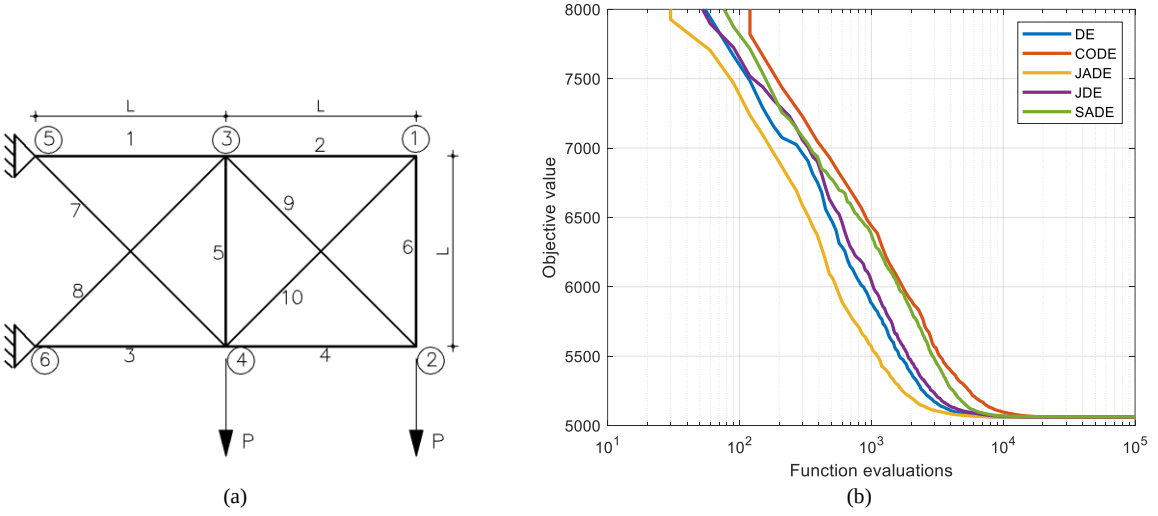
$$F(x) = f(x) + P(x) = f(x) + \mu \sum_{k=1}^{N} H_k(x) g_k^2(x) \tag{5}$$

where $f(x)$ is the value of the objective function to be optimized, $P(x)$ is the penalty term, $g_k(x)$ is the k-th constraint function in the form $g_k(x) \leq 0$, N is the total number of constraints, $\mu \geq 0$ is a penalty factor that should be large enough (e.g. 10^5) and $H_k(x)$ function is defined as follows:

$$H_i(x) = \begin{cases} 1 & \text{if } g_i(x) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

4.1. Example 1. The 10-bar plane truss

This is a standard benchmark 10-bar plane truss problem [17]. The structure is shown in Fig. 1(a) with the following structural characteristics: modulus of elasticity E=10000 ksi, material weight ρ=0.1 lb/in^3, length L=360 in, load P=100 kip. The structural members are divided into 10 groups. The design variables are the cross-section areas of each member group in the interval [0.1, 35] (in^2). The maximum allowable displacement in the $\pm x$ and $\pm y$ directions for each node is d_{max} = 2 in, while the maximum allowable stress (absolute value) is σ_{allow} = 25 ksi in tension or compression and the objective is to minimize the weight of the structure under the specified constraints. For this specific benchmark problem, various results from the literature can be found in [18]. The convergence history of the various DE methods is shown in Fig. 1(b) as the average from 30 independent runs for each method.

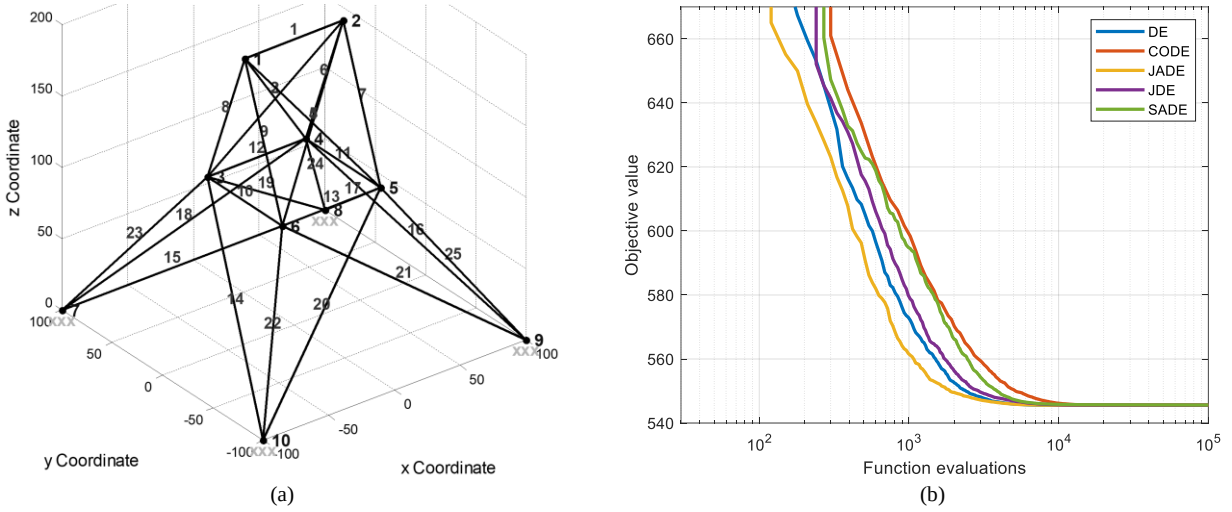


(a) (b)

Fig. 1. (a) The 10-bar plane truss (coordinates in inches), (b) Convergence history for the 10-bar plane truss example.

4.2. Example 2. 25-member space truss

The second test example is a 25-member space truss [17]. The structure is depicted in Fig. 2(a). The "xxx" sign for nodes 7, 8, 9, 10 of the model denotes a restriction for all three DOFs of the node (pinned constraint). Variations of this test example can be found in the literature [18, 19]. The problem described below is the one described in [19], as in [18] the load cases and the stress constraints are different, leading to different, non-comparable results. Details about the geometry, the constraints and the load cases of the problem can be found in [17]. The convergence history of the various DE methods is shown in Fig. 2(b) as the average from 30 independent runs for each method.

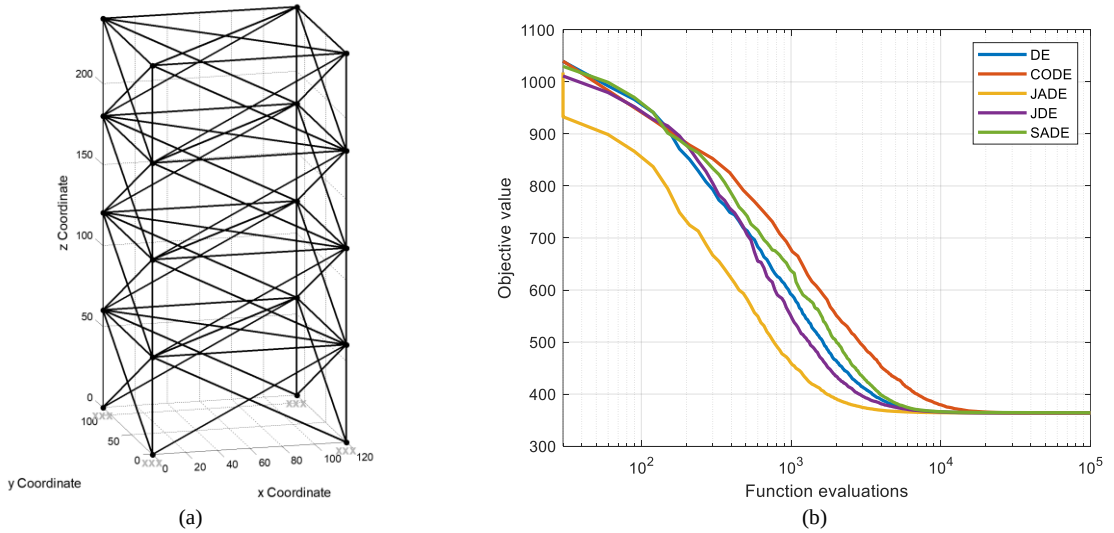


(a) (b)

Fig. 2. (a) The 25-member space truss (coordinates in inches), (b) Convergence history for the 25-member space truss example.

4.3. Example 3. 72-member space truss

The third test example is a space truss with 72 members [17], shown in Fig. 3(a). The example can be found in [20-22], among others. The modulus of Elasticity is $E=10000\,\text{ksi}$ and the material weight $\rho=0.1\,\text{lb/in}^3$. The basis of the structure is a rectangle with a side of 120 in, while the total height is $4\times60\,\text{in}=240\,\text{in}$. The "xxx" sign for the basis nodes of the model denotes a restriction for all three DOFs of the node (pinned constraint). The basis nodes, numbered 1, 2, 3, 4 are fixed on the ground. The structural members are divided into 16 groups and two load cases are considered. The design variables are the cross-section areas of each member group with a lower limit of $0.01\,\text{in}^2$ and no upper limit. The constraints are imposed on stresses and displacements. The maximum allowable displacement in the $\pm x$ and $\pm y$ directions for each node is $d_{\max}=0.25\,\text{in}$, while the maximum allowable stress (as an absolute value) is $\sigma_{\text{allow}}=25\,\text{ksi}$ in tension or compression and the objective is to minimize the weight of the structure under the specified constraints. Details about the geometry, loading and constraints of the problem can be found in [17]. The convergence history of the various DE methods is shown in Fig. 3(b) as the average from 30 independent runs for each method.



(a) (b)

Fig. 3. (a) The 72-member space truss (coordinates in inches), (b) Convergence history for the 72-member space truss example.

5. Discussion and conclusions

We investigated the performance of five DE variants, namely the standard DE, CoDE, JDE, JADE and SaDE, for handling constrained 2D and 3D truss structural optimization problems where the weight of the structure needs to be minimized subject to constraints on stresses and displacements. Three well-known benchmark problems have been considered. For each problem and each method, we ran each optimization algorithm 30 times and the average results are presented in terms of the convergence history, i.e. a diagram of the objective value vs the function evaluations. It is shown that in all three example problems, the JADE variant exhibits superior performance and outperforms the other DE variants. It appears that parameter adaptation is beneficial to improve the optimization performance of DE by automatically updating control parameters to appropriate values during the search. It is worth noting that the convergence of the standard DE algorithm is faster compared to the three other DE variants (CoDE, JDE and SaDE) for the first and second test examples (10-bar 2D truss and 25-member 3D truss).

To end up to more reliable conclusions about the performance and efficiency of each DE variant, a more detailed investigation needs to be performed, adding more structural analysis examples and also checking various metrics relevant to the performance of the different methods.

References

[1] Storn R and Price K. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization* (1997); 11: 341-359. journal article. DOI: 10.1023/a:1008202821328.

[2] Brest J, Greiner S, Boskovic B, et al. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Trans Evol Comput* (2006); 10: 646-657. DOI: 10.1109/TEVC.2006.872133.

[3] Qin AK, Huang VL and Suganthan PN. Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization. *IEEE Trans Evol Comput* (2009); 13: 398-417. DOI: 10.1109/TEVC.2008.927706.

[4] Zhang J and Sanderson AC. JADE: Adaptive Differential Evolution With Optional External Archive. *IEEE Trans Evol Comput* (2009); 13: 945-958. DOI: 10.1109/TEVC.2009.2014613.

[5] Wang Y, Cai Z and Zhang Q. Differential Evolution With Composite Trial Vector Generation Strategies and Control Parameters. *IEEE Trans Evol Comput* (2011); 15: 55-66. DOI: 10.1109/TEVC.2010.2087271.

[6] Yang M, Cai Z, Li C, et al. An improved adaptive differential evolution algorithm with population adaptation. *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. Amsterdam, The Netherlands: ACM, (2013), p. 145-152.

[7] Pham HA. Truss optimization with frequency constraints using enhanced differential evolution based on adaptive directional mutation and nearest neighbor comparison. *Adv Eng Soft* (2016); 102: 142-154. DOI: https://doi.org/10.1016/j.advengsoft.2016.10.004.

[8] Eiben AE, Hinterding R and Michalewicz Z. Parameter control in evolutionary algorithms. *IEEE Trans Evol Comput* (1999); 3: 124-141. DOI: 10.1109/4235.771166.

[9] Gamperle R, Muller S and Koumoutsakos A. A Parameter Study for Differential Evolution. *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation* (2002); 10: 293-298.

[10] Liu J and Lampinen J. On Setting the Control Parameter of the Differential Evolution Method. In: Matoušek R and Ošmera P, (eds.). *8th International Conference on Soft Computing (MENDEL 2002)*. Brno(2002), p. 11-18.

[11] Liu J and Lampinen J. A Fuzzy Adaptive Differential Evolution Algorithm. *Soft Comput* (2005); 9: 448-462. DOI: 10.1007/s00500-004-0363-x.

[12] Brest J, Zumer V and Maucec MS. Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization. In: *2006 IEEE International Conference on Evolutionary Computation* 16-21 July 2006 2006, pp.215-222.

[13] Brest J, Bošković B, Greiner S, et al. Performance comparison of self-adaptive and adaptive differential evolution algorithms. *Soft Computing* (2007); 11: 617-629. DOI: 10.1007/s00500-006-0124-0.

[14] Huang VL, Qin AK and Suganthan PN. Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization. In: *2006 IEEE International Conference on Evolutionary Computation* 16-21 July 2006 2006, pp.17-24.

[15] Qin AK and Suganthan PN. Self-adaptive differential evolution algorithm for numerical optimization. In: *2005 IEEE Congress on Evolutionary Computation* 2-5 Sept. 2005 2005, pp.1785-1791 Vol. 1782.

[16] Teo J. Exploring dynamic self-adaptive populations in differential evolution. *Soft Computing* (2006); 10: 673-686. DOI: 10.1007/s00500-005-0537-1.

[17] Plevris V and Papadrakakis M. A Hybrid Particle Swarm - Gradient Algorithm for Global Structural Optimization. *Comput-Aided Civ Infrastruct Eng* (2011); 26: 48-68. DOI: 10.1111/j.1467-8667.2010.00664.x.

[18] Perez RE and Behdinan K. Particle swarm approach for structural design optimization. *Comput Struct* (2007); 85: 1579-1588.

[19] Zhou M and Rozvany GIN. DCOC: An optimality criteria method for large systems. Part II: Algorithm. *Struct Optimization* (1993); 6: 250-262.

[20] Adeli H and Kamal O. Efficient optimization of space trusses. *Comput Struct* (1986); 24: 501-511.

[21] Adeli H and Park HS. *Neurocomputing for Design Automation*. Boca Raton, FL, USA: CRC Press, Inc., (1998).

[22] Sarma KC and Adeli H. Fuzzy Genetic Algorithm for Optimization of Steel Structures. *J Struct Eng (ASCE)* (2000); 126: 596-604.