COMPARISON OF METAHEURISTIC ALGORITHMS FOR SIZE OPTIMIZATION OF TRUSSES

Aristotelis E. Charalampakis[1]

[1]Department of Civil Engineering
Gediz University
Izmir, 35665, Turkey
e-mail: aristotelis.charalampakis@gediz.edu.tr, web page: www.charalampakis.com

Keywords: Truss size optimization, GA, PSO, DE, SA, ABC.

Abstract. *Metaheuristic algorithms have emerged as the best way of solving complex optimization problems. Consequently, the literature includes a large and growing number of applications of metaheuristics for the size optimization of trusses. Generally speaking, these studies do not focus in the comparison between algorithms. Motivated by this, we present a framework for an unbiased and meaningful comparison between different metaheuristic methods. Based on this framework, a critical evaluation of a number of metaheuristic algorithms is presented, which includes Genetic Algorithms, Particle Swarm Optimization, Artificial Bee Colony, Simulated Annealing and Differential Evolution variants. The differences in performance are highlighted and an explanation for their behavior is attempted. It is found that Differential Evolution is the best optimizer in terms of performance, robustness and scalability. We also demonstrate that, although the methods considered in this study are well-established, often better designs than the ones found in the literature are discovered.*

1 INTRODUCTION

Structural optimization has always been a topic of great interest among engineers. In most real-life optimization problems, featuring multimodality and non-convex feasible regions, the use of simple gradient methods is problematic. This led to mathematical programming (MP) and optimality criteria (OC) methods, which have been used extensively in the past [1]. In MP, direct minimization is attempted e.g. using decomposition into a sequence of linear programming problems. The use of approximations readily produces good designs which, however, are not necessarily globally optimal. In OC methods, assumptions on the conditions in the optimum state (e.g. "fully stressed design") provide simple recursion formulas for redesign. Obviously, these methods do not solve the problem in a proper mathematical manner and may not converge for highly redundant structures [1].

Nowadays, metaheuristic algorithms have emerged as the best way for solving complex optimization problems. These algorithms are usually inspired by evolution, swarm intelligence or physical phenomena principles and their widespread use is justified by a number of important advantages such as easy implementation, lack of dependency on gradient or other problem-specific information and good performance with global search characteristics [2]. Consequently, the literature includes a large and growing number of applications of metaheuristics for optimization of trusses. Examples include Genetic Algorithms (GAs) [3], [4], Simulated Annealing (SA) [5], Harmony Search (HS) [6]-[8], Artificial Bee Colony (ABC) [9], Particle Swarm Optimization (PSO) [10]-[14], Teaching-Learning Based Optimization (TLBO) [15]-[16], Big Bang – Big Crunch (BB-BC) [17]-[18], Colliding Bodies Optimization algorithm (CBO) [19], Enhanced Bat Algorithm (EBA) [20], Charged System Search (CSS) [21], Differential Evolution [22]-[23], Hybrid Particle Swarm - Swallow Swarm (HPSSO) [24], Hybrid Particle Swarm - Ant Colony Strategy - Harmony Search (HPSACO) [25], Chaotic Swarming of Particles (CSP) [26], to name a few.

Generally speaking, studies on truss size optimization do not focus in the comparison between algorithms. Usually, only the best final design is compared which may have been found using an excessive computational budget or, in certain cases, with constraint violations. Thus, a need for an overall and unbiased comparison is evident. Motivated by this, a framework for an unbiased and meaningful comparison between different metaheuristic methods is presented. Based on this framework, we perform a comparison of a number of metaheuristic algorithms which includes Genetic Algorithms, Particle Swarm Optimization, Artificial Bee Colony, Simulated Annealing and Differential Evolution variants. The comparison is based on several benchmark problems of varying complexity which have been studied by numerous researchers using a large variety of methods; this allows for absolute rather than relative comparison. The differences in performance are highlighted and an explanation for their behavior is attempted. It is also shown that, although the methods considered in this study are well-established, often better designs than the ones found in the literature are

produced.

2 STATEMENT OF THE OPTIMIZATION PROBLEM

In this study, the weight minimization problem of a truss structure with D sizing variables is formally stated as follows.

Minimize:

$$f(\mathbf{x}) = W(\mathbf{x}) + P(\mathbf{x}) \tag{1}$$

with $W(\mathbf{x}) = \sum_{i=1}^{D} (L_i x_i \rho_i) =$ *structural weight and* $P(\mathbf{x}) =$ *penalty function,*

subject to:
side constraints $\mathbf{x}_L \le \mathbf{x} \le \mathbf{x}_U$ *(vector inequalities apply element-wise);*
additional constraints on element stresses, buckling stresses and nodal displacements, depending on problem definition.

In the above formulation, $\mathbf{x} = \{x_1, x_2, ..., x_D\} =$ vector containing the cross-sectional area of each group of elements, $\mathbf{x}_L$ and $\mathbf{x}_U =$ vectors defining the minimum and maximum allowable areas, respectively, L_i and $\rho_i =$ total length and specific weight of the i-th group of bars.

Ideally, the penalty should be kept as low as possible, just above the limit below which infeasible solutions are optimal [27]. There are many types of penalty functions; for a survey on the state-of-the-art, see Ref. [27]. In this study we use a static penalty rule with a small constant term to keep all best solutions strictly within the feasible domain, i.e. we do not accept any violation at all. A normalized constraint violation function is defined as follows:

$$v_n^j = \frac{v_o^j}{v_a^j} - 1, \tag{2}$$

where, v_n^j is the normalized violation of the j-th optimization constraint (stress, displacement, buckling stress, etc), v_o^j is the corresponding value computed for a candidate solution, and v_a^j is the allowable constraint limit. The penalty function takes the form:

$$P(\mathbf{x}) = \sum_{j=1}^{N_c} \delta^j \left(A v_n^j + B \right), \tag{3}$$

where, $A = 10^6$, $B = 10^3$, $N_c =$ number of constraints and $\delta^j =$ activation key defined as:

$$\delta^j = \begin{cases} 1, & v_n^j > 0 \\ 0, & v_n^j \le 0 \end{cases}. \tag{4}$$

The constant term B is added when even the slightest violation occurs. This has proved to be very effective in keeping the best solutions free of penalties, without the need to compare both the feasibility of candidate designs and their objective value.

3 METAHEURISTIC ALGORITHMS

3.1 Basic features of the trade study

The following assumptions/rules have been used equally for all algorithms:

1. The number of structural analyses required in the optimization process was chosen as the best performance indicator to compare algorithms of different nature and configuration.
2. Thirty independent runs with different random seeds were carried out for each test case and each optimization algorithm in order to obtain statistically significant results.
3. A long period random number generator of L'Ecuyer with Bays-Durham shuffle and added safeguards [28] was used. Henceforth, $r =$ random variable with uniform distribution in the interval (0,1), sampled anew each time it is required, and $\mathbf{r} =$ corresponding vector.
4. For each test problem, a value-to-reach (VTR) defines the limit between success and failure. The VTR is 1% heavier than the best feasible design obtained in this study or reported in literature. Hence "successful" designs lie very close to the optimum region of design space. The final design is usually

more fine-tuned when the VTR threshold is reached early in the optimization process.

5. In this study the computational budget is set to $2500D$ structural analyses and the best design is achieved strictly within this limit. This budget balances two conflicting aspects: (a) it is high enough to achieve good designs (often better than those reported in literature); (b) it is not too excessive so as to conflict with limitations on computing resources.
6. Cross-sectional areas of elements are rounded to three decimal digits *before* performing structural analysis in double precision without any further rounding. Literature designs including more than three decimal digits are exactly reproduced in this paper; however, the total weight is re-evaluated and may be slightly different from the source value. The inversion of the stiffness matrix is performed using Gauss-Jordan elimination with full pivoting [28].

3.2 Standard Genetic Algorithm (SGA)

Genetic Algorithms (GAs) are evolutionary algorithms that originated from the work of Holland [29] and have been employed in virtually any problem imaginable. The so-called standard genetic algorithm (SGA) can be described by the following pseudocode:

1. Initialize the population of individuals (chromosomes);
2. Calculate the fitness of each individual in the population;
3. Select individuals to form a new population according to each one's fitness;
4. Perform crossover and mutation to form a new population;
5. Repeat steps (2–5) until some condition is satisfied.

The parameters of SGA are set as follows: gene length L_g = depending on the problem, so that the phenotypic step is less than 10^{-3}; population size P =50; single crossover with crossover probability 0.7; jump mutation probability $1/P$; creep mutation probability $L_c/D/P$ (L_c = chromosome length in bits); tournament selection with 2 individuals; and elitism with 1 individual.

3.3 Hybrid Genetic Algorithm (HGA)

The standard GA formulation is hybridized by combining the search space reduction method (SSRM) and a local optimization algorithm to form the Hybrid Genetic Algorithm (HGA). SSRM is a systematic method which gradually reduces the search space. It facilitates the optimization algorithm by focusing into the promising areas which leads to better solutions. The SSRM embedded in HGA was presented in [30] and later employed in several cases, e.g. [31]. It is based on statistical analysis of a population of P candidate designs, where each one is assigned a weight w depending on its quality. For minimization problems:

$$w_k = \frac{\max\left(f\right)}{f_k},\ k \in \left\{1,2,...,P\right\}, \tag{5}$$

where $\max\left(f\right) = \max\left\{f_1, f_2, ..., f_P\right\}$ is the objective value of the worst design in the population, which is assigned a weight of unity. The weighted mean value of design variable i is then calculated as:

$$m_i = \frac{\sum_{k=1}^{P}\left(w_k x_{ik}\right)}{\sum_{k=1}^{P}\left(w_k\right)},\ i = \left\{1,2,...,D\right\}, \tag{6}$$

where x_{ik} is the value of the i-th variable of the k-th candidate design. The descriptive weighted standard deviation of design variable i is given as:

$$s_i = \sqrt{\frac{\sum_{k=1}^{P}\left(w_k\left(x_{ik} - m_i\right)^2\right)}{\sum_{k=1}^{P}\left(w_k\right)}}. \tag{7}$$

The new trial lower and upper bounds of design variable i are formed symmetrically around m_i:

$$\begin{aligned} \bar{x}^i_{L,j} &= m_i - q\ s_i \\ \bar{x}^i_{U,j} &= m_i + q\ s_i \end{aligned}, \tag{8}$$

where j is an index for SSRM steps and q is a scalar parameter. Finally, the new range of values of design variable i is the intersection (common part) of the previous range and the trial range:

$$\left[x_{L,j+1}^{i}, x_{U,j+1}^{i}\right] = \left[x_{L,j}^{i}, x_{U,j}^{i}\right] \cap \left[\overline{x}_{L,j}^{i}, \overline{x}_{U,j}^{i}\right], \tag{9}$$

Based on Eq. (9), the search space is not allowed to expand. Instead, it is gradually reduced on a variable-by-variable basis, if justified by the statistical analysis. The scalar q controls the aggressiveness of SSRM. Small values may stochastically lead to exclusion of a promising area in subsequent SSRM steps; large values cancel the beneficial effect of SSRM. In this study, $q = 3$ is set which is a rather conservative choice [30]. The chromosomes of the existing population are substituted by their closest counterpart in the new mapping. If a variable falls out of a new bound, it is set equal to it.

Regarding variable discretization, the gene length is set to 10 bits irrespective of the problem at hand. This leads to a relatively small chromosome length, which greatly facilitates the GA. Refining of the solutions is achieved through SSRM, which is triggered automatically. For the test problems considered in this study, this happens every 10000 function evaluations.

A local optimization algorithm, namely the Greedy Descend Hill Climber (GDHC [2]), is also embedded in the HGA. The final 5000 function evaluations of each run are dedicated to GDHC. The best solution found so far becomes seed; this solution is continuously improved by flipping the bits of the chromosome from left to right, keeping the best result as a reference. When a full cycle of bit flips has been concluded without improvement the local optimum has been found [2] and the process is terminated. If the limit of 5000 function evaluations is reached prior to finding the local optimum, the reference solution becomes final.

3.4 Enhanced Particle Swarm Optimization (EPSO)

Particle Swarm Optimization (PSO) is based on the social sharing of information among members, which produces behavioral patterns that offer an evolutionary advantage (avoid predators, seek food and mates). The method, introduced by Kennedy and Eberhart [32], searches the design space by adjusting the trajectories of individuals called "particles". The particles are attracted towards the positions of both their personal best solution and the best solution of the whole population (the "swarm") in a stochastic manner. In the basic PSO algorithm, we assume that the population consists of P particles. Each particle is characterized by its position and velocity, determined at time instant k by the corresponding vectors $\mathbf{x}_k$ and $\mathbf{v}_k$. Initially, the particles are distributed randomly in the box-constrained design space, so that:

$$\mathbf{x}_L \le \mathbf{x}_0^m \le \mathbf{x}_U \;\; \forall m \in \{1,2,...,P\}. \tag{10}$$

The initial velocities of the particles are also chosen randomly:

$$-\mathbf{v}_0^{\max} \le \mathbf{v}_0^m \le \mathbf{v}_0^{\max} \;\; \forall m \in \{1,2,...,P\}, \tag{11}$$

where $\mathbf{v}_0^{\max} = \gamma\left(\mathbf{x}_U - \mathbf{x}_L\right)$ and $\gamma =$ a scalar parameter. The position vector of particle m at the next time instant $k+1$ is given as:

$$\mathbf{x}_{k+1}^m = \mathbf{x}_k^m + \mathbf{v}_{k+1}^m, \tag{12}$$

where the time step Δt between the distinct time instants is assumed to be equal to unity. The velocity vector $\mathbf{v}_{k+1}^m$ is given as:

$$\mathbf{v}_{k+1}^m = w_k \; \mathbf{v}_k^m + c_1 \; \mathbf{r}_1 \circ (\mathbf{p}_k^m - \mathbf{x}_k^m) + c_2 \; \mathbf{r}_2 \circ (\mathbf{p}_k^g - \mathbf{x}_k^m), \tag{13}$$

where, $w_k =$ inertia factor at time instant k; c_1, $c_2 =$ cognitive and social parameters, respectively; $\mathbf{p}_k^m =$ best ever position vector of particle m up to and including time instant k; $\mathbf{p}_k^g =$ best ever position vector amongst all particles up to and including time instant k; and the $\circ$ operator indicates element-by-element multiplication. The side constraints of Eqs. (10) and (11) are enforced after each time step.

It is known that standard PSO suffers from convergence rate problems, due to the delicate balance between exploration and exploitation that is required. This was evident in the problems considered in this study, so an enhanced variant is used instead. This variant (EPSO) is based on the work of Fourie and Groenwold [12] and has been successfully applied in a parameter identification problem [33]-[35]. The differences with respect to the basic PSO algorithm are the following:

- If the best solution found in the whole swarm is not improved over a period of h consecutive steps, then it is assumed that the velocities are large and the algorithm cannot locate better solutions due to overshooting. For this reason, both the inertia factor and the maximum velocity are reduced as follows:

$$\text{If } f(\mathbf{p}_k^g) = f(\mathbf{p}_{k-h}^g) \text{ then } \begin{cases} w_{k+1} = a \; w_k \\ \mathbf{v}_{k+1}^{\max} = \beta \; \mathbf{v}_k^{\max} \end{cases} \text{ else } \begin{cases} w_{k+1} = w_k \\ \mathbf{v}_{k+1}^{\max} = \mathbf{v}_k^{\max} \end{cases}. \tag{14}$$

- The craziness operator assigns a random velocity vector to a particle which hence moves away from the swarm and explores other regions of the search space. The operator is activated with a probability P_{cr} as follows:

$$\text{If } r < P_{cr} \text{ then randomly assign } \mathbf{v}_{k+1}^{m} \text{ with } -\mathbf{v}_{k+1}^{\max} \le \mathbf{v}_{k+1}^{m} \le \mathbf{v}_{k+1}^{\max} . \tag{15}$$

- The individual with the worst performance is moved to the best ever position of the swarm.
- If the velocity vector $\mathbf{v}_k^m$ resulted in an improvement of $\mathbf{p}_k^g$ then, instead of Eq. (12), the following rule ("elite velocity") is used for particle m:

$$\mathbf{x}_{k+1}^{m} = \mathbf{p}_k^g + c_3 \ \mathbf{r}_3 \circ \mathbf{v}_k^m , \tag{16}$$

where, $c_3 =$ a scalar parameter. This is a difference between the present implementation and Ref. [12], where a single random variable r_3 is used to multiply the whole velocity vector $\mathbf{v}_k^m$.

In this study, the parameters of EPSO are set as follows: $P = 20$, $c_1 = 0.5$, $c_2 = 1.6$, $\gamma = 0.4$, $w_0 = 1.40$, $h = 3$, $a = 0.99$, $\beta = 0.95$, $P_{cr} = 0.22$, and $c_3 = 1.30$.

3.5 Artificial Bee Colony (ABC)

In the category of Swarm Intelligence algorithms, Karaboga and Basturk [36] proposed the Artificial Bee Colony (ABC) algorithm, a stochastic algorithm inspired by the foraging behaviour of honey bees. In ABC, the colony of artificial bees is divided into two equal groups, i.e. the employed bees and the onlookers. The number of employed bees is equal to the number of food sources (possible solutions) around the hive. Initially, the ABC generates a random initial distribution of *SN*/2 food source positions, where *SN* = total colony size. Next, the food source positions are subjected to repeated cycles of improvement. The employed bee i produces a modification on the position vector of the associated food source i:

$$\overline{x}_{ij} = x_{ij} + (2r-1)\left(x_{ij} - x_{kj}\right), \tag{17}$$

where, $i,k \in \{1,2,...,SN/2\} =$ random indices with $i \neq k$ and $j \in \{1,2,...,D\} =$ a randomly chosen dimension of the D-dimensional vector $\mathbf{x}_i$ to be modified. A greedy selection is performed between the current and modified vector; the best either becomes, or remains, the food source i. Next, the employed bees share the nectar information with the onlooker bees on the dance area. An onlooker bee chooses each food source with a probability given by:

$$p_i = \frac{fit_i}{\sum_{k=1}^{SN/2} fit_k}, \tag{18}$$

where, $fit_i =$ fitness of food source i. For minimization problems, this can be evaluated by:

$$fit_i = \frac{1}{1+f_i}, \tag{19}$$

where, $f_i =$ objective value of food source i. The onlooker bee also produces a modification on the selected food source according to Eq. (17) and applies a greedy selection criterion between the current and modified vector. Finally, if a food source is not improved over a period of $LIMIT$ cycles, then the associated employed bee becomes scout. This means that the food source is re-initialized as:

$$\mathbf{x} = \mathbf{x}_L + \mathbf{r} \circ \left(\mathbf{x}_U - \mathbf{x}_L\right). \tag{20}$$

In this study, the parameters of ABC algorithm are set as follows: $SN = 50$ and $LIMIT = SN/2 \times D$.

3.6 Differential Evolution (DE)

Differential Evolution (DE) is a stochastic optimization method which was introduced by Storn and Price [37]. It has no natural paradigm and is usually applied to problems with continuous design variables. An early version of DE was initially conceived under the term "Genetic Annealing" and published in a programmer's magazine [38]. The DE algorithm is extremely simple; the uncondensed C-style pseudocode of the algorithm spans less than 25 lines [38]. In classic DE, a population of P individuals is randomly dispersed within the design space, as follows:

$$\begin{aligned} &\mathbf{x}_L \le \mathbf{x}_{i,0} \le \mathbf{x}_U \ \ \forall i \in \{1,2,...,P\} \\ &\mathbf{P}_{\mathbf{x},g} = \left(\mathbf{x}_{i,g}\right),\ i \in \{1,2,...,P\},\ g \in \{0,1,...,g_{\max}\}, \\ &\mathbf{x}_{i,g} = \left(x_{j,i,g}\right),\ j \in \{1,2,...,D\}. \end{aligned} \tag{21}$$

where $\mathbf{P}_{\mathbf{x},g}$ = array of P vectors (solutions); $\mathbf{x}_{i,g}$ = D-dimensional vector representing a candidate solution; $g_{\max}$ = maximum number of generations; i = index for vectors, g = index for generations, j = index for design variables; and the parentheses indicate an array. At each generation g, a mutated population $\mathbf{P}_{\mathbf{v},g} = \left(\mathbf{v}_{i,g}\right)$ is formed based on the current population $\mathbf{P}_{\mathbf{x},g}$, as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r0,g} + F\left(\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}\right), \tag{22}$$

where, r_0, r_1 and r_2 are mutually exclusive random integers in $\{1,2,...,P\}$, which are also different from index i; $\mathbf{x}_{r0,g}$ = base vector; and F = a scalar parameter. After using Eq. (22), design variables are reset to their respective bounds in case a mutated solution moves out of the initial design space. Next, a trial population $\mathbf{P}_{\mathbf{u},g} = \left(\mathbf{u}_{i,g}\right)$ is formed, consisting of individuals created from the parent and mutated populations, as follows:

$$\mathbf{u}_{i,g} = \left(u_{j,i,g}\right) = \begin{cases} v_{j,i,g}, & \text{if } \left(r \le C_r \text{ or } j = j_{rand}\right) \\ x_{j,i,g}, & \text{otherwise} \end{cases}, \tag{23}$$

where, j_{rand} = a random index in $\{1,2,...,P\}$ that ensures that at least one design variable will originate from the mutant vector $\mathbf{v}_{i,g}$; and C_r = a scalar parameter in the range [0,1]. The final step of the algorithm is a greedy selection criterion, which for minimization problems is expressed as:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g}, & \text{if } f\left(\mathbf{u}_{i,g}\right) \le f\left(\mathbf{x}_{i,g}\right) \\ \mathbf{x}_{i,g}, & \text{otherwise} \end{cases}. \tag{24}$$

The above-described classic DE implementation is denoted as rand/1/bin [38], or DE1 herein for short. DE1 usually demonstrates stronger exploration capability and thus is more suitable for solving multimodal problems [39]. Following recommendations in [38], $F = 0.5$ while a high value of $C_r = 0.9$ is expected to perform well with non-separable functions. The population size is set as $P = 50$ for all problems.
Another popular DE variant is denoted as best/1/bin [38] in which the currently best vector of the population is used as base vector. In addition, jitter is introduced to F and, thus, Eq. (22) becomes:

$$\begin{aligned} \mathbf{v}_{i,g} &= \mathbf{x}_{best,g} + F_j\left(\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}\right) \\ F_j &= F + d\left(r - 0.5\right) \end{aligned}, \tag{25}$$

where $d = 0.001 =$ magnitude of jitter [38]. This variant was tested in this study and found to be too greedy, as it showed great initial performance which was quickly followed by stagnation in the more difficult problems. Instead, another DE variant is included which is denoted as DE3 or "rand-best/1/bin" [34], [35]. Let us define r_b as the expected ratio of evaluations with a random base vector to the total number of evaluations. If $r<r_b$, then evolution proceeds according to rand/1/bin with jitter. Conversely, best/1/bin is used. Based on information presented in a separate section, a high value of $r_b = 0.90$ is chosen in order to promote exploration.

Since its inception, DE has proved to be very efficient for many optimization problems. However, its performance is dependent on the choice of both the strategy for the generation of trial vectors and the values of control parameters. Although there exist suggestions for parameter settings, there is no fixed parameter setting that is equally suitable for various problems or even at different evolution stages of a single problem. For this purpose, several adaptive algorithms have been proposed, such as SaDE [39], JADE [40] and SaNSDE [41]. In this study we consider SaDE [39] as the final DE variant, in which four competing strategies are utilized simultaneously, namely rand/1/bin, rand-to-best/2/bin, rand/2/bin and current-to-rand/1, with K = 4 = total number of strategies. For the implementation details, see Ref. [39].

3.7 Simulated Annealing (SA)

Simulated Annealing (SA) is inspired by the annealing process of physical systems which, being at a high-energy state, are gradually cooled down until their minimum energy level is reached. The idea that this process can be formulated into an optimization algorithm was first put forth by Kirkpatrick et al. [42]. The implementation of the SA algorithm used in this study, outlined below, is based on the work of Balling [43]. The algorithm begins with the creation of D random designs in the box-constrained design space $\mathbf{x}_L \le \mathbf{x} \le \mathbf{x}_U$. The best of these designs becomes the current design $\mathbf{x}_c$. This preliminary loop is done in order to avoid the possibility that the starting point for the optimization is too poor. Following the selection of an appropriate cooling schedule, the current design $\mathbf{x}_c$ is subjected to small perturbations to create a candidate design $\mathbf{x}_a$.

Whenever $\mathbf{x}_a$ is better than $\mathbf{x}_c$, it replaces it with probability of 1. For a minimization problem, this is expressed mathematically as:

$$\Delta f = f(\mathbf{x}_a) - f(\mathbf{x}_c) \le 0 \Rightarrow P(\mathbf{x}_a \to \mathbf{x}_c) = 1. \tag{26}$$

If $\mathbf{x}_a$ is worse than $\mathbf{x}_c$, it still replaces it with a probability given by:

$$\Delta f = f(\mathbf{x}_a) - f(\mathbf{x}_c) > 0 \Rightarrow P(\mathbf{x}_a \to \mathbf{x}_c) = e^{-\Delta f/(Kt_c)} \le 1, \tag{27}$$

where, $K =$ Boltzman parameter and $t_c =$ the current system temperature, corresponding to the cooling cycle c. The Boltzman parameter is not kept constant during optimization. Instead, it is updated prior to using Eq. (27) as follows:

$$K_{N_a+1} = \frac{K_{N_a} N_a + |\Delta f|}{N_a + 1}, \tag{28}$$

where, $N_a =$ number of accepted designs so far, and $K_{N_a} =$ previous Boltzman parameter. Initially, $N_a = 0$ and $K_0 = 1$. The starting and final system temperatures are given by:

$$t_s = -\frac{1}{\ln(P_s)},\ t_f = -\frac{1}{\ln(P_f)}, \tag{29}$$

where, P_s and $P_f =$ starting and final probability of acceptance, respectively. The temperature is reduced gradually in N_c cooling cycles, as follows:

$$t_{c+1} = cf \times t_c, \tag{30}$$

where, $cf \in (0,1) =$ cooling factor, given by:

$$cf = \left(\frac{t_f}{t_s}\right)^{\frac{1}{N_c - 1}}. \tag{31}$$

At each cooling cycle, a number of inner loops of perturbation/improvement of the current design $\mathbf{x}_c$ is executed. For each loop, an array of integers representing the design variables (1 to D) is shuffled. According to the sequence indicated in the shuffled array, each design variable i is in turn perturbed to form the candidate design $\mathbf{x}_a$ according to:

$$x_{ia} = x_{ic} + (2r - 1) d_i, \tag{32}$$

where, $d_i =$ magnitude of perturbation of variable i. Note that the side constraints $\mathbf{x}_L \le \mathbf{x}_a \le \mathbf{x}_U$ are re-enforced after using Eq. (32). It has been observed that inner loops are more important at low temperatures. For this reason, the repetitions I of the inner loop are not kept constant but instead they are determined as follows:

$$I = round\left(I_s + (I_f - I_s)\frac{t_c - t_s}{t_f - t_s}\right), \tag{33}$$

where, I_s and $I_f =$ starting and final number of repetitions of inner loops, respectively. Note that in SA the current design is occasionally replaced by a poorer design due to Eq. (27). This means that even if a better design replaces the current design due to (26), it may not be the best ever design. For our purposes, it is desirable to keep track of the best design found so far during the optimization process. This best design $\mathbf{x}_b$ is updated after each function evaluation. In this study, the parameters of SA algorithm are set as follows: $P_s = 0.5$, $P_f = 1E-07$, $N_c = 300$, $I_s = 1$, $I_f = 3$, $\mathbf{d} = 0.01(\mathbf{x}_U - \mathbf{x}_L)$.

4 TEST PROBLEMS AND OPTIMIZATION RESULTS

4.1 Spatial 25-bar tower (D=8)

The spatial 25-bar tower with 10 nodes shown in Fig. 1 has been studied extensively in the literature. All bars are made of the same material with E = 10000 ksi and ρ = 0.1 lb/in^3. Cross-sectional areas of bars can vary between 0.01 and 35 in^2. Bars are grouped in eight groups (D = 8) with different compressive stress limit but the same tensile stress limit (see Table 1). In addition, displacements of top nodes 1 and 2 must be less than 0.35 inches in all directions. The structure is subject to two independent loading conditions (see Table 2).

Member group	Members	Compressive stress limit [ksi]	Tensile stress limit [ksi]
1	1	35.092	40
2	2, 3, 4, 5	11.590	40
3	6, 7, 8, 9	17.305	40
4	10, 11	35.092	40
5	12, 13	35.092	40
6	14, 15, 16, 17	6.759	40
7	18, 19, 20, 21	6.959	40
8	22, 23, 24, 25	11.082	40

Table 1: Member grouping and stress limits for the spatial 25-bar tower.

Node	P_x [kips]	P_y [kips]	P_z [kips]
Load Case I			
1	0	20	-5
2	0	-20	-5
Load Case II			
1	1	10	-5
2	0	10	-5
3	0.5	0	0
6	0.5	0	0

Table 2: Load cases for the spatial 25-bar tower.

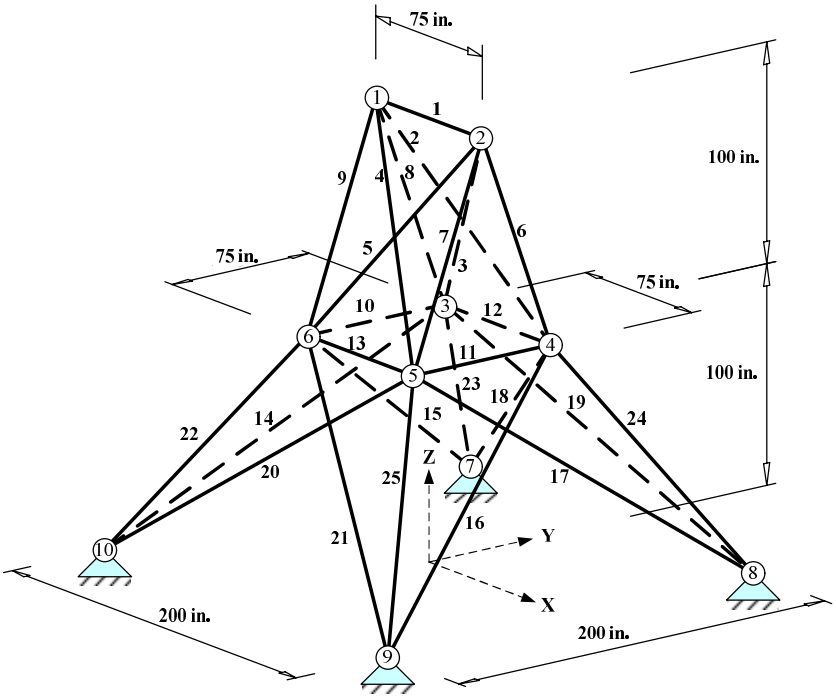

Fig. 1: Schematic of the spatial 25-bar tower.

Table 3 shows that DE1 and DE3 obtained the same best design, which is slightly better than the one found in literature in terms of weight (including MSPSO [13] by a very small margin). This design was hence used as reference (VTR = 545.172 × 1.01 = 550.623 lb). The statistical data shown in Table 4 indicate that EPSO ranked right after DE in terms of best weight but was less robust than ABC and HGA. The average progress of the best solution and the variation of success rate of each algorithm with respect to the number of structural analyses are compared in Fig. 2. It can be seen that DE1 and DE3 could reach 100% success after ~750D analyses, while SaDE follows at ~1250D analyses. SA, being the only method not employing a population of solutions but rather perturbing/improving a single solution, was the slowest to converge.

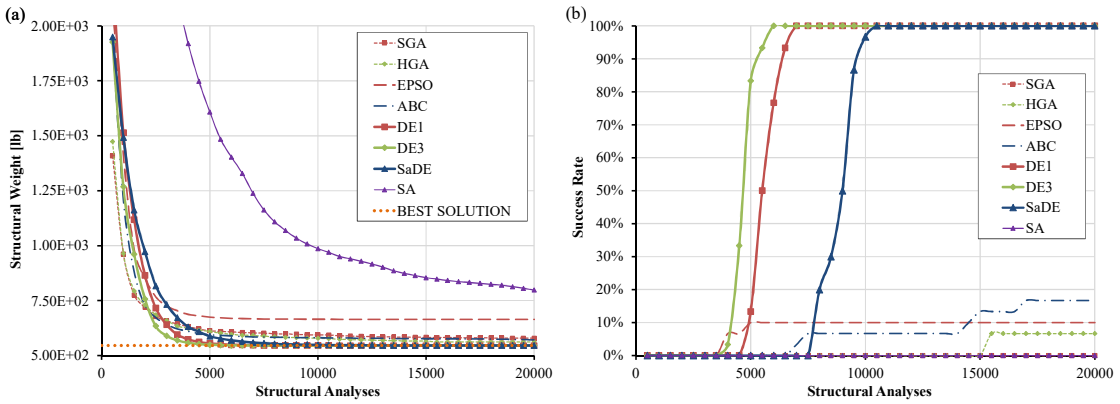

Fig. 2: Size optimization of the spatial 25-bar tower
(a) average progress of best solution (b) evolution of success rate.

Bar Group Areas [in^2]	Lee and Geem (HS) [6]	Sonmez (ABC-AP) [9]	Li et al. (HPSO) [10]	Li et al. (PSOPC) [10]	Lamberti (CMLPSA) [5]	Kaveh and Talatahari (HPSACO) [25]
1	0.047	0.011	0.010	0.010	0.0100	0.010
2	2.022	1.979	1.970	1.979	1.9870	2.054
3	2.950	3.003	3.016	3.011	2.9935	3.008
4	0.010	0.010	0.010	0.100	0.0100	0.010
5	0.014	0.010	0.010	0.100	0.0100	0.010
6	0.688	0.690	0.694	0.657	0.6840	0.679
7	1.657	1.679	1.681	1.678	1.6769	1.611
8	2.663	2.652	2.643	2.693	2.6621	2.678
Weight [lb]	544.365	545.206	545.238	547.965	545.163	544.991
Penalty	1.817E+04	-	-	-	2.007E+03	7.686E+04
Bar Group Areas [in^2]	Talatahari et al. (MSPSO) [13]	Kaveh et al. (HPSSO) [24]	Degertekin and Hayalioglu (TLBO) [15]	Degertekin (EHS) [7]	Degertekin (SAHS) [7]	Kaveh et al. (CSP) [26]
1	0.0100	0.0100	0.0100	0.010	0.010	0.010
2	1.9848	1.9907	2.0712	1.995	2.074	1.910
3	2.9956	2.9881	2.9570	2.980	2.961	2.798
4	0.0100	0.0100	0.0100	0.010	0.010	0.010
5	0.0100	0.0100	0.0100	0.010	0.010	0.010
6	0.6852	0.6824	0.6891	0.696	0.691	0.708
7	1.6778	1.6764	1.6209	1.679	1.617	1.836
8	2.6599	2.6656	2.6768	2.652	2.674	2.645
Weight [lb]	545.172	545.160	545.095	545.486	545.118	545.145
Penalty	-	2.026E+03	5.692E+04	-	6.161E+04	2.413E+04
Bar Group Areas [in^2]	Kaveh and Zakian (EBA) [20]	Camp (BB-BC) [17]	Kaveh and Talatahari (HBB-BC) [18]	Camp and Farshchin (TLBO) [16]	Kaveh and Talatahari (CSS) [21]	This study (DE1, DE3)
1	0.01000	0.010	0.010	0.0100	0.010	0.010
2	1.97889	2.092	1.993	1.9878	2.003	1.983
3	3.00472	2.964	3.056	2.9914	3.007	2.999
4	0.01000	0.010	0.010	0.0102	0.010	0.010
5	0.01000	0.010	0.010	0.0100	0.010	0.010
6	0.68880	0.689	0.665	0.6828	0.687	0.682
7	1.67834	1.601	1.642	1.6775	1.655	1.678
8	2.65270	2.686	2.679	2.6640	2.660	2.663
Weight [lb]	545.169	545.522	545.143	545.176	545.093	545.172
Penalty	2.000E+03	7.910E+04	4.538E+04	2.000E+03	2.820E+04	-

Note: The weight has been re-evaluated without rounding and may differ from the source. The penalty is evaluated using Eq. (3).

Table 3: Comparison of best designs obtained in the spatial 25-bar tower problem.

Algorithm	SGA		HGA		EPSO		ABC	
Weight [lb]	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.
	579.010	30.052	561.842	9.975	664.259	123.391	570.909	21.898
	Best	Worst	Best	Worst	Best	Worst	Best	Worst
	550.675	703.142	549.279	585.425	545.426	1155.675	547.124	627.713
Algorithm	DE1		DE3		SaDE		SA	
Weight [lb]	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.
	545.176	0.004	545.173	0.002	545.368	0.060	797.911	179.954
	Best	Worst	Best	Worst	Best	Worst	Best	Worst
	545.172	545.187	545.172	545.180	545.265	545.487	558.483	1129.935

Table 4: Statistical evaluation of algorithms' performance in the spatial 25-bar tower problem.

4.2 Planar 10-bar truss (D=10)

Fig. 3 shows the geometry and loads of a cantilever truss structure consisting of 10 bars and 6 nodes which has been analyzed by many researchers. Two loading cases are considered: in case I, $P_1 = 100$ kips and $P_2 = 0$; in case II, $P_1 = 150$ kips and $P_2 = 50$ kips. All bars are made of the same material with E = 10000 ksi and $\rho = 0.1$ lb/in^3. Cross-sectional areas of bars can vary between 0.10 and 35 in^2. The stress limit is set to ± 25 ksi, while the displacement of the free nodes must be less than 2 inches in all directions.

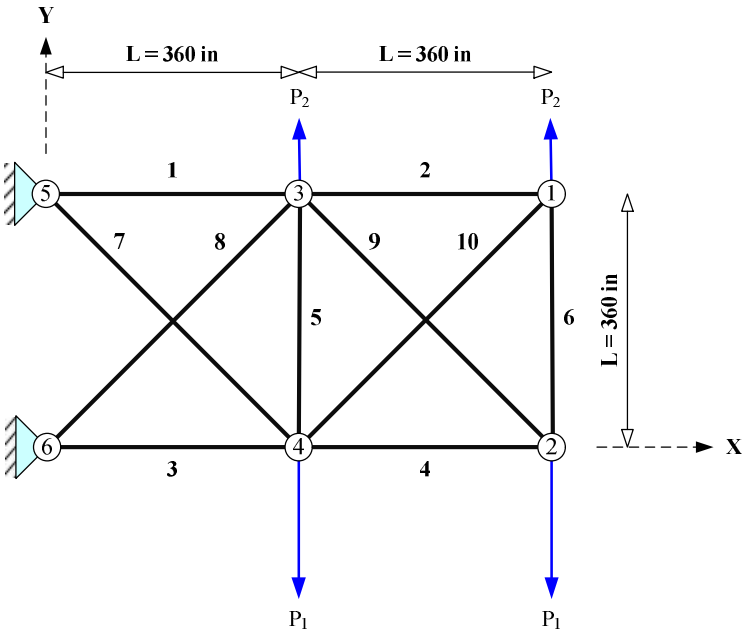

Fig. 3: Schematic of the planar 10-bar truss.

Regarding load case I, Table 5 shows that the best result was discovered by DE3 and used as reference (VTR = 5060.855 × 1.01 = 5111.464 lb). It is clear from Fig. 4 that EPSO, DE1 and DE3 variants find the optimum region very quickly, as 100% success rate is reached after only ~500D analyses. SaDE follows next, producing very good results but with slower convergence rate. Interestingly, SA may be the slowest to converge, but its final results are comparable to EPSO. HGA follows next, reaching ~60% at the end. The small difference between SGA and HGA during the first 10000 analyses is due to the coarser discretization of the search space (16 and 10 bits per variable for SGA and HGA, respectively). The SSRM is triggered at 10000 analyses while GDHC is introduced at 20000 analyses, as shown in Fig. 4a. Table 6 reveals the very small standard deviation of the results obtained by DE1 and DE3, as well as the small final ranges for all design variables.

Regarding load case II, Table 8 shows that the best solution was discovered by DE3 and used as reference (VTR = 4676.932 × 1.01 = 4723.701 lb). The average progress of the best solution and the evolution of the success rate are presented in Fig. 5, whereas a statistical analysis of the results is given in Table 7. The same conclusions can be drawn as in load case I. Once again, the final variable ranges for DE variants are very narrow.

Bar Group Areas [in^2]	Lee and Geem (HS) [6]	Sonmez (ABC-AP) [9]	Li et al. (HPSO) [10]	Perez and Behdinan (PSO) [11]	Wu and Cheng (AMPDE) [22]
1	30.150	30.548	30.704	33.500	30.378
2	0.102	0.100	0.100	0.100	0.100
3	22.710	23.180	23.167	22.766	23.468
4	15.270	15.218	15.183	14.417	15.196
5	0.102	0.100	0.100	0.100	0.100
6	0.544	0.551	0.551	0.100	0.533
7	7.541	7.463	7.460	7.534	7.437
8	21.560	21.058	20.978	20.467	21.084
9	21.450	21.501	21.508	20.392	21.433
10	0.100	0.100	0.100	0.100	0.100
Weight [lb]	5058.336	5060.888	5060.906	5024.248	5060.234
Penalty	1.907E+03	-	1.001E+03	2.696E+04	2.308E+03
Bar Group Areas [in^2]	Haftka and Grdal [44]	Kaveh et al. (HPSSO) [24]	Degertekin and Hayalioglu (TLBO) [15]	Lamberti and Pappalettere (IHS) [8]	Kaveh and Talatahari (HPSACO) [25]
1	30.520	30.53838	30.4286	30.5222	30.307
2	0.100	0.10000	0.1000	0.1000	0.100
3	23.200	23.15103	23.2436	23.2005	23.434
4	15.220	15.20566	15.3677	15.2232	15.505
5	0.100	0.10000	0.1000	0.1000	0.100
6	0.551	0.54890	0.5751	0.5513	0.524
7	7.457	7.46532	7.4404	7.4572	7.437
8	21.040	21.06437	20.9665	21.0367	21.079
9	21.530	21.52935	21.5330	21.5288	21.229
10	0.100	0.10000	0.1000	0.1000	0.100
Weight [lb]	5060.926	5060.864	5060.956	5060.930	5056.591
Penalty	1.108E+03	-	-	1.001E+03	1.992E+03
Bar Group Areas [in2]	Renwei and Peng (MP) [45]	Bureerat and Pholdee (ADEA) [23]	Camp and Farshchin (TLBO) [16]	Degertekin (SAHS) [7]	This study (DE3)
1	30.590	30.5139	30.6684	30.394	30.531
2	0.100	0.1000	0.1000	0.100	0.100
3	23.270	23.2052	23.1584	23.098	23.197
4	15.190	15.2084	15.2226	15.491	15.228
5	0.100	0.1000	0.1000	0.100	0.100
6	0.460	0.5318	0.5421	0.529	0.550
7	7.500	7.4585	7.4654	7.488	7.459
8	21.070	21.0512	21.0255	21.189	21.045
9	21.480	21.5391	21.4660	21.342	21.511
10	0.100	0.1000	0.1000	0.100	0.100
Weight [lb]	5062.781	5060.895	5060.975	5061.275	5060.855
Penalty	-	-	-	1.031E+03	-

Note: The weight has been re-evaluated without rounding and may differ from the source. The penalty is evaluated using Eq. (3).

Table 5: Comparison of best designs obtained in the planar 10-bar truss problem (load case I).

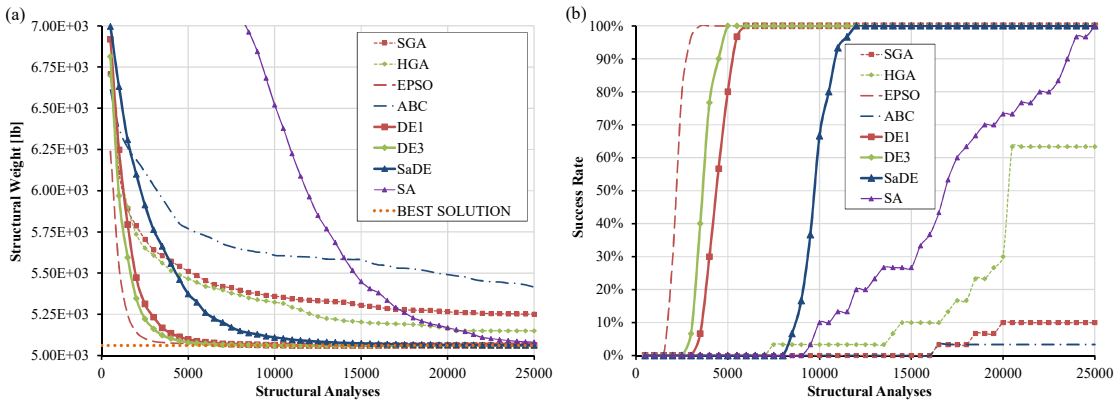

Fig. 4: Size optimization of the planar 10-bar truss (load case I)
(a) average progress of best solution (b) evolution of success rate.

Algorithm	SGA		HGA		EPSO		ABC	
Weight [lb]	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.
	5250.611	209.569	5149.860	117.664	5072.901	9.496	5415.715	276.013
	Best	Worst	Best	Worst	Best	Worst	Best	Worst
	5099.936	5743.367	5081.989	5628.621	5063.456	5101.853	5102.510	6289.445
Algorithm	DE1		DE3		SaDE		SA	
Weight [lb]	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.
	5060.865	0.006	5060.863	0.011	5062.677	2.885	5080.998	8.929
	Best	Worst	Best	Worst	Best	Worst	Best	Worst
	5060.858	5060.881	5060.855	5060.915	5061.195	5077.710	5064.229	5097.011

Table 6: Statistical evaluation of algorithms' performance in the planar 10-bar truss problem (load case I).

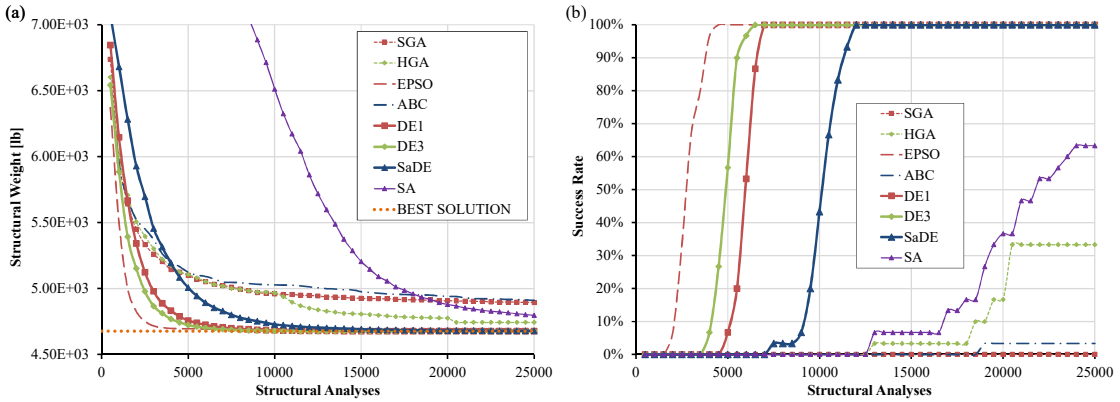

Fig. 5: Size optimization of the planar 10-bar truss (load case II)
(a) average progress of best solution (b) evolution of success rate.

Algorithm	SGA		HGA		EPSO		ABC	
Weight [lb]	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.
	4894.657	90.345	4743.481	32.227	4694.085	10.994	4910.675	111.381
	Best	Worst	Best	Worst	Best	Worst	Best	Worst
	4735.553	5079.055	4692.920	4805.366	4680.118	4717.370	4710.823	5219.890
Algorithm	DE1		DE3		SaDE		SA	
Weight [lb]	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.
	4676.946	0.008	4676.940	0.007	4679.069	0.749	4796.789	215.699
	Best	Worst	Best	Worst	Best	Worst	Best	Worst
	4676.937	4676.969	4676.932	4676.968	4677.916	4680.667	4682.242	5612.583

Table 7: Statistical evaluation of algorithms' performance in the planar 10-bar truss problem (load case II).

Bar Group Areas [in^2]	Lee and Geem (HS) [6]	Sonmez (ABC-AP) [9]	Li et al. (HPSO) [10]	Kaveh et al. (HPSSO) [24]	Kaveh and Talatahari (HPSACO) [25]	Degertekin and Hayalioglu (TLBO) [15]
1	23.250	23.4692	23.353	23.52377	23.1940	23.5240
2	0.102	0.1005	0.100	0.10000	0.1000	0.1000
3	25.730	25.2393	25.502	25.36864	24.5850	25.4410
4	14.510	14.3540	14.250	14.37799	14.2210	14.4790
5	0.100	0.1001	0.100	0.10000	0.1000	0.1000
6	1.977	1.9701	1.972	1.96973	1.9690	1.9950
7	12.210	12.4128	12.363	12.36780	12.4890	12.3340
8	12.610	12.8925	12.894	12.79722	12.9250	12.6890
9	20.360	20.3343	20.356	20.32577	20.9520	20.3540
10	0.100	0.1000	0.101	0.10000	0.1010	0.1000
Weight [lb]	4669.365	4677.089	4677.349	4676.949	4675.797	4678.315
Penalty	5.561E+03	-	1.025E+03	-	3.871E+03	-
Bar Group Areas [in^2]	Talatahari et al. (MSPSO) [13]	Talatahari et al. (PSO) [13]	Degertekin (SAHS) [7]	Degertekin (EHS) [7]	Bureerat and Pholdee (ADEA) [23]	This study (DE3)
1	23.4432	23.9324	23.5250	23.589	23.7697	23.515
2	0.1000	0.1000	0.1000	0.100	0.1001	0.100
3	25.3718	25.2478	25.4290	25.422	25.3328	25.293
4	14.1360	14.1791	14.4880	14.488	14.3954	14.385
5	0.1000	0.1000	0.1000	0.100	0.1004	0.100
6	1.9699	1.9701	1.9920	1.975	1.9714	1.970
7	12.4335	12.5097	12.3520	12.362	12.4120	12.389
8	13.0173	13.0379	12.6980	12.682	12.8414	12.831
9	20.2717	19.9002	20.3410	20.322	20.0824	20.325
10	0.1000	0.1000	0.1000	0.100	0.1000	0.100
Weight [lb]	4677.253	4677.974	4678.848	4679.015	4677.326	4676.932
Penalty	-	1.022E+03	-	-	1.005E+03	-

Note: The weight has been re-evaluated without rounding and may differ from the source. The penalty is evaluated using Eq. (3).

Table 8: Comparison of best designs obtained in the planar 10-bar truss problem (load case II).

4.3 Planar 17-bar truss (D=17)

The planar 17-bar with 9 nodes shown in Fig. 6 has been studied in [6], [46], [47] and [48]. All bars are made of the same material with E= 30000 ksi and ρ = 0.268 lb/in3. The structure is subject to a single vertical load of 100 kips at node 9. Cross-sectional areas of bars can vary between 0.1 and 50 in^2. The stress limit is 50 ksi for both tension and compression, while the displacement of free nodes must be less than ± 2 inches. Since no design variable linking was used, the problem dimensionality is D = 17.

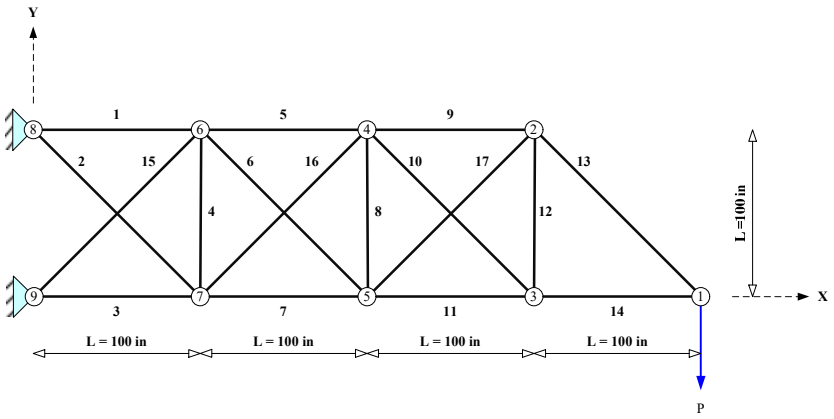

Fig. 6: Schematic of the planar 17-bar truss.

It can be seen from Table 9 that the best design was obtained by DE3. The corresponding structural weight was taken as target to compute VTR = 2581.895 × 1.01 = 2607.714 lb. The statistical data given in Table 10 confirm the robustness of DE1 and DE3 that achieved a standard deviation on optimized weight equal to 0.047 and 0.086 lb, respectively, converged practically to the same optimized weight. The other algorithms ranked in the following order: SaDE, EPSO, SA, HGA and ABC.

The average progress of the best solution and the variation of success rate of each algorithm with respect to the number of structural analyses are compared in Fig. 7. It can be seen that DE1, DE3 could reach 100% success after ~900D analyses. SaDE also reached 100 success but much later. EPSO was the fastest algorithm in the early stages of the optimization but about 20% of the runs failed to succeed at the end. SA achieved 65% success rate when measured at the end of the analyses.

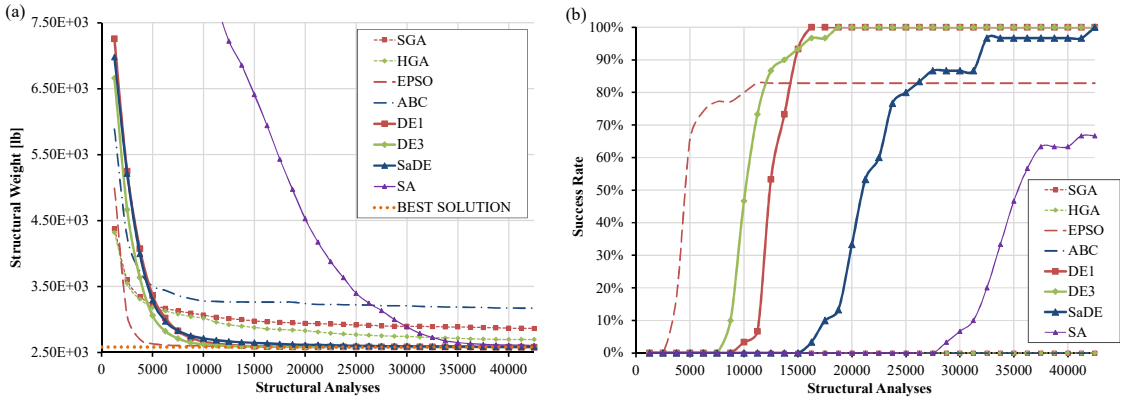

Fig. 7: Size optimization of the planar 17-bar truss
(a) average progress of best solution (b) evolution of success rate.

Bar Group Areas [in^2]	Lee and Geem (HS) [6]	Khot and Berke (OC) [46]	Barakat and Ibrahim (SCEO) [47]	Adeli and Kumar (GA) [48]	This study (DE3)
1	15.821	15.930	15.897700	16.02858206	15.911
2	0.108	0.100	0.100367	0.10695021	0.100
3	11.996	12.070	12.064600	12.18302437	12.053
4	0.100	0.100	0.100189	0.11005022	0.100
5	8.150	8.067	8.085990	8.41651683	8.076
6	5.507	5.562	5.562290	5.71486143	5.552
7	11.829	11.933	11.939700	11.33052266	11.963
8	0.100	0.100	0.100009	0.10540021	0.100
9	7.934	7.945	7.950370	7.30051460	7.958
10	0.100	0.100	0.100005	0.11470023	0.100
11	4.093	4.055	4.049050	4.04550809	4.057
12	0.100	0.100	0.100005	0.10075020	0.100
13	5.660	5.657	5.665820	5.61101122	5.651
14	4.061	4.000	3.988130	4.04550809	4.005
15	5.656	5.558	5.565300	5.15221030	5.565
16	0.100	0.100	0.100031	0.10695021	0.100
17	5.582	5.579	5.578710	5.28551057	5.571
Weight [lb]	2580.975	2581.923	2581.899	2543.573	2581.895
Penalty	1.439E+03	-	1.000E+03	1.797E+04	-

Note: The weight has been re-evaluated without rounding and may differ from the source. The penalty is evaluated using Eq. (3).

Table 9: Comparison of best designs obtained in the planar 17-bar truss problem.

Algorithm	SGA		HGA		EPSO		ABC	
Weight [lb]	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.
	2864.364	132.409	2697.876	53.008	2597.112	31.927	3170.541	188.595
	Best	Worst	Best	Worst	Best	Worst	Best	Worst
	2678.314	3320.587	2612.075	2803.950	2582.851	2757.438	2906.426	3724.046
Algorithm	DE1		DE3		SaDE		SA	
Weight [lb]	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.	Average	St. Dev.
	2581.992	0.047	2581.981	0.086	2584.694	4.388	2605.621	21.507
	Best	Worst	Best	Worst	Best	Worst	Best	Worst
	2581.919	2582.091	2581.895	2582.242	2582.511	2606.616	2586.249	2660.871

Table 10: Statistical evaluation of algorithms' performance in the planar 17-bar truss problem.

5 CONCLUSIONS

This study analyzed the relative performance of variants of Genetic Algorithms (SGA, HGA), Particle Swarm Optimization (EPSO), Artificial Bee Colony (ABC), Differential Evolution (DE1, DE3 and SaDE) and Simulated Annealing (SA) in sizing optimization problems of truss structures. The comparison was based on a framework that is both unbiased and meaningful. In order to assess the differences between algorithms we make use of rigorous statistical analysis with large samples, common computational budget depended on the problem dimensionality, careful accounting for all function evaluations and monitoring of the success rate of each algorithm during the analyses.

Regarding the algorithms, SGA provides a baseline performance whereas the HGA, which combines a search space reduction method (SSRM) with a local optimizer, showed improved performance with respect to SGA. The enhanced PSO algorithm (EPSO) resulted very competitive in the small to medium problems examined in this study. Its performance should also be examined in more difficult problems, which is a topic for further research. ABC is not easily trapped into local optima but has slow convergence because a single design variable is changed per mutant vector. The present results indicate that ABC could not compete with EPSO and DE variants. As far as it concerns DE variants, DE1 is based on rand/1/bin and DE3 is a stochastic mixture of rand/1/bin and best/1/bin. These variants combined very good convergence rate with robustness, stability and scalability. DE3 in particular showed improved convergence rate in the small-scale problems. A very greedy DE variant, based solely on best/1/bin, was also tested in the preliminary versions of this study and found to be very competitive only in small problems. Regarding SaDE, it appears that the advantage of its adaptive nature is traded-off by a generally small hysteresis (delay) in the convergence rate, as compared to DE1 and DE3. Note that the latter algorithms utilize fixed settings that were expected to work well based on the specific problem characteristics. This information may not be available for other problems, which makes SaDE and similar self-adapting algorithms a very good choice.

SA is the only method examined in this study which perturbs/improves a single solution. The lack of synergistic information within a population had the obvious effect that SA did not show explosive initial performance. Nevertheless, its progress was clear and it is evident that the Metropolis test (Eq. (27)) is a powerful method to escape local optima. For most of the small problems, this was enough to allow SA to rank among the best. Its performance should also be examined in more difficult problems, which is a topic for further research.

Overall, DE was definitely superior over the rest of the algorithms and found very competitive designs, which in some cases were even better than those reported in literature.

REFERENCES

[1] Feury, C., Geradin, M. (1978) "Optimality criteria and mathematical programming in structural weight optimization," Comput Struct 8(1) pp. 7–17.

[2] Eiben, A.E., Smith, J.E. (2003) *Introduction to Evolutionary Computing*. Springer, New York.

[3] Coello, C.A., Christiansen, A.D. (2000) "Multiobjective optimization of trusses using genetic algorithms," Comput Struct 75 pp. 647–60.

[4] Koumousis, V., Georgiou, P. (1994) "Genetic algorithms in discrete optimization of steel truss roofs," J Comput Civ Eng 8(3) pp. 309–25.

[5] Lamberti, L. (2008) "An efficient simulated annealing algorithm for design optimization of truss structures," Comput Struct 86 pp. 1936–53.

[6] Lee, K.S., Geem, Z.W. (2004) "A new structural optimization method based on the harmony search algorithm," Comput Struct 82 pp. 781–98.

[7] Degertekin, S.O. (2012) “Improved harmony search algorithms for sizing optimization of truss structures,” Comput Struct 92–93 pp. 229–41.
[8] Lamberti, L., Pappalettere, C. (2009) “An improved harmony-search algorithm for truss structure optimization,” *Proceedings of the twelfth international conference civil structural and environmental engineering computing*. Stirlingshire: Civil-Comp Press.
[9] Sonmez, M. (2011) “Articial bee colony algorithm for optimization of truss structures,” Appl Soft Comput 11 pp. 2406–18.
[10] Li, L.J., Huang, Z.B., Liu, F., Wu, Q.H. (2007) “A heuristic particle swarm optimizer for optimization of pin connected structures” Comput Struct 85 pp. 340–9.
[11] Perez, R.E., Behdinan, K. (2007) “Particle swarm approach for structural design optimization. Comput Struct 85 pp. 1579–88.
[12] Fourie, P.C., Groenwold, A.A. (2002) “The particle swarm optimization algorithm in size and shape optimization,” Struct Multidisc Optim 23(4) pp. 259–67.
[13] Talatahari, S., Kheirollahi, M., Farahmandpour, C., Gandomi, A.H. (2013) “A multi-stage particle swarm for optimum design of truss structures,” Neural Comput Appl 23: pp. 1297–309.
[14] Dimou, C.K., Koumousis, V.K. (2009) “Reliability based optimal design of truss structures using particle swarm optimization,” J Comput Civil Eng-ASCE 23 pp. 100–9.
[15] Degertekin, S.O., Hayalioglu, M.S. (2013) “Sizing truss structures using teaching-learning-based optimization,” Comput Struct 119 pp. 177–88.
[16] Camp, C.V., Farshchin, M. (2014) “Design of space trusses using modified teaching–learning based optimization,” Eng Struct 62 pp. 87–97.
[17] Camp, C. (2007) “Design of space trusses using Big Bang–Big Crunch optimization,” J Struct Eng 133(7) pp. 999–1008.
[18] Kaveh, A., Talatahari, S. (2009) “Size optimization of space trusses using Big Bang–Big Crunch algorithm,” Comput Struct 87 pp. 1129–40.
[19] Kaveh, A., Ilchi Ghazaan, M. (2015) “A comparative study of CBO and ECBO for optimal design of skeletal structures,” Comput Struct 153 pp. 137–47.
[20] Kaveh, A., Zakian, P. (2014) “Enhanced bat algorithm for optimal design of skeletal structures,” Asian J Civil Eng (BHRC) 15(2) pp. 179–212.
[21] Kaveh, A., Talatahari, S. (2010) “Optimal design of skeletal structures via the charged system search algorithm,” Struct Multidisc Optim 41 pp. 893–911.
[22] Wu, C-Y., Tseng, K-Y. (2010) “Truss structure optimization using adaptive multi-population differential evolution,” Struct Multidisc Optim 42, pp. 575–90.
[23] Bureerat, S., Pholdee, N. (2015) “Optimal truss sizing using an Adaptive Differential Evolution algorithm,” J Comput Civil Eng-ASCE, 04015019.
[24] Kaveh, A., Bakhshpoori, T., Afshari, E. (2014) “An efficient hybrid particle swarm and swallow swarm optimization algorithm,” Comput Struct 143 pp. 40–59.
[25] Kaveh, A., Talatahari, S. (2009) “Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures,” Comput Struct 87 pp. 267–83.
[26] Kaveh, A., Sheikholeslami, R., Talatahari, S., Keshvari-Ilkhichi, M. (2014) “Chaotic swarming of particles: A new method for size optimization of truss structures,” Adv Eng Softw 67 pp. 136–47.
[27] Coello C.A.C. (2002) “Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: survey of the state of the art,” Comput Methods Appl Mech Eng 191 pp. 1245–87.
[28] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P. (2002) *Numerical recipes in c++: the art of scientific computing*. Cambridge University Press.
[29] Holland, J.H. (1975) *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.
[30] Charalampakis, A.E., Koumousis, V.K. (2008) “Identification of Bouc–Wen hysteretic systems by a hybrid evolutionary algorithm,” J Sound Vib 314 pp. 571–85.
[31] Marano, G.C., Quaranta, G., Monti, G. (2011) “Modified genetic algorithm for the dynamic identification of structural systems using incomplete measurements,” Comp Aided Civ Infr Eng 26(2) pp. 92–110.
[32] Kennedy, J., Eberhart, R.C. (1995) “Particle swarm optimization,” *Proceedings of IEEE international conference on neural networks IV*. Perth Australia, IEEE press Piscataway, NJ, pp. 1942–8.
[33] Charalampakis, A.E., Dimou, C.K. (2010) “Identification of Bouc–Wen hysteretic systems using particle swarm optimization,” Comp Struct 88 pp. 1197–205.
[34] Charalampakis, A.E., Dimou, C.K. (2011) “Comparison of differential evolution, particle swarm optimization and genetic algorithms for the identification of Bouc-Wen hysteretic systems,” *Proceedings of the second international conference on soft computing technology in civil, structural and environmental engineering CSC2011*, Chania, Greece.

[35] Charalampakis, A.E., Dimou, C.K. (2015) "Comparison of evolutionary algorithms for the identification of Bouc-Wen Hysteretic Systems," J Comput Civil Engin ASCE 29(3) 04014053.
[36] Karaboga, D., Basturk, B. (2008) "On the performance of Artificial Bee Colony (ABC)," Appl Soft Comp 8(1) pp. 687–97.
[37] Storn, R., Price, K. (1997) "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," J Global Optim 11 pp. 341–59.
[38] Storn, R., Price, K., Lampinen, J.A. (2005) *Differential evolution – a practical approach to global optimization,* Springer.
[39] Qin, A., Huang, V.L., Suganthan, P.N. (2009) "Differential evolution algorithm with strategy adaptation for global numerical optimization," IEEE T Evolut Comput 13(2) pp. 398–417.
[40] Zhang, J., Sanderson, A.C. (2009) "JADE: adaptive differential evolution with optional external archive," IEEE T Evolut Comput 13(5) pp. 945–58.
[41] Yang, Z., Tang, K., Yao, X. (2008) "Self-adaptive differential evolution with neighborhood search," *Proceedings IEEE Congr Evolut Comput*, Hong Kong, China, pp. 1110–6.
[42] Kirkpatrick, S., Gerlatt, C.D., Vecchi, M.P. (1983) "Optimization by simulated annealing," Science 220 pp. 671–80.
[43] Balling, R.J. (1991) "Optimal steel frame design by simulated annealing," J Struct Engng 117(6) pp. 1780–95.
[44] Haftka, R., Grdal, Z. (1992) *Elements of structural optimization*. 3rd ed. Dordrecht: Kluwer Academic Publishers.
[45] Renwei, X., Peng, L. (1986) "An efficient method for structural optimization," Acta Mech Sinica 2(4) pp. 348–61.
[46] Khot, N.S., Berke, L. (1984) "Structural optimization using optimality criteria methods," In: Atrek E., Gallagher R.H., Ragsdell K.M., Zienkiewicz O.C., editors. *New directions in optimum structural design*. New York: John Wiley.
[47] Barakat, S., Ibrahim, H. (2011) "Application of shuffled complex evolution global optimization technique in the design of truss structures," *Proceedings of the Forth International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*, Kuala Lumpur, Malaisia.
[48] Adeli, H., Kumar, S. (1995) "Distributed genetic algorithm for structural optimization," J Aerospace Eng, ASCE 8(3) pp. 156–63.