Structural optimization with the multistrategy PSO-ES unfeasible local search operator

Marco Martino Rosso$^{1[0000-0002-9098-4132]}$, Angelo Aloisio$^{2[0000-0002-6190-0139]}$, Raffaele Cucuzza$^{1[0000-0002-9344-6006]}$, Rebecca Asso$^{1[0000-0002-6196-4052]}$, and Giuseppe Carlo Marano$^{1[0000-0001-8472-2956]}$

[1] DISEG, Department of Structural, Geotechnical and Building Engineering, Politecnico di Torino, Corso Duca Degli Abruzzi, 24, Turin 10128, Italy; Corresponding author: `marco.rosso@polito.it`,
[2] Civil Environmental and Architectural Engineering Department, Università degli Studi dell'Aquila, via Giovanni Gronchi n.18, L'Aquila, 67100, Italy.

Abstract. The convergence of meta-heuristic optimization algorithms is not mathematically ensured given their heuristic nature of mimicking natural phenomena. Nevertheless, in recent years, they have become very widespread tools due to their successful capability to handle hard constrained problems. In the present study, the Particle Swarm Optimization (PSO) algorithm is investigated. The most important state-of-art improvements (inertia weight, neighbourhood) have been implemented and an unfeasible local search operator based on self-adaptive Evolutionary Strategy (ES) algorithm has been proposed. Firstly, the current PSO-ES has been tested on literature constrained benchmark numerical problems compared with PSO which adopts the traditional penalty function approach. In conclusion, some constrained structural optimization truss design examples have been covered and critically discussed.

Keywords: structural optimization,self-adaptive evolutionary strategies (ES), structural benchmark, multistrategy particle swarm optimization

1 Particle swarm optimization introduction

A mathematical problem involving the minimization of at least one objective function (OF) $f(\boldsymbol{x})$ is denoted as an optimization problem, which may be constrained or not, depending on parameters gathered in a design vector $\boldsymbol{x}$ defined in a search space Ω. In recent years, metaheuristic algorithms and evolutionary algorithms (EAs) have been successfully employed in many engineering applications and structural optimization design tasks [1–8]. They do not require information from the OF gradient, a characteristic of the time and computationally expensive gradient-based approaches. In the EAs field, J. Holland firstly developed the population-based genetic algorithm (GA) [9, 10], which mimics the Darwinian Theory and genetics phenomena. R. Eberhart and J. Kennedy in [11] proposed the particle swarm optimization (PSO) algorithm in 1995, another famous population-based approach which mimics the food search behaviour of animals in the natural environment like fish schooling or birds flockings. In the mechanisms of the algorithm, each particle of the swarm act as an intelligent agent and explore the search space in order to improve the optimum solution of the optimization problem. At the beginning, the PSO has been able to solve unconstrained optimization only, and then different strategies have been later adopted to even solve constrained ones. In the next sections, after a brief review of the PSO, a particular focus on the novel multistrategy method has been discussed. Eventually, some literature constrained mathematical benchmark problems have been successfully solved by the enhanced PSO. In the final part, real-world structural optimization problems have been solved, comparing performance with other techniques.

2 The Particle Swarm Optimization (PSO) algorithm

The PSO algorithm is based on a population of N of intelligent agents whose position in the search space identifies a trial solution of the optimization problem. To explore the search space, the particles of the swarm flies independently, even if a global intelligent movement appears considering the entire swarm during the iterative optimization approach. The standard PSO formulation is based on the classical mechanics perspective, therefore each particle i in every generation k is fully characterized by its position ${}^{k}\boldsymbol{x}_i$ and velocity ${}^{k}\mathbf{v}_i$ in the search space. The next position of the particle is influenced by two kind of information gathered from the swarm: a self-memory allow the particle to remember its best visited position which acts as local attractor denoted as *pbest* ${}^{k}\boldsymbol{x}_i^{Pb}$. On the other hand, a global attractor denoted as *gbest* ${}^{k}\boldsymbol{x}^{Gb}$ is based on information shared among the particles of the entire swarm. To prevent the explosion of the velocity, this term has been clamped by an upper bound $\mathrm{v}^{\max} = \gamma(\boldsymbol{x}^u - \boldsymbol{x}^l)/\tau$, considering a time unit $\tau = 1$ to make it consistent with a physical velocity and $\gamma \in [0.1, 1]$ as suggested in [12]. The position and the velocity of each particle follow the below adjusting rules:

$$ {}^{(k+1)}\mathbf{v}_i = {}^{k}\mathbf{v}_i + c_1\, {}^{(k+1)}\boldsymbol{r}_{1i} * \left[{}^{k}\boldsymbol{x}_i^{Pb} - {}^{k}\boldsymbol{x}_i\right] + c_2\, {}^{(k+1)}\boldsymbol{r}_{2i} * \left[{}^{k}\boldsymbol{x}_i^{Gb} - {}^{k}\boldsymbol{x}_i\right], \quad (1) $$

$$ {}^{(k+1)}\boldsymbol{x}_i = {}^{k}\boldsymbol{x}_i + \tau\ {}^{(k+1)}\mathbf{v}_i \quad (\tau = 1), \quad (2) $$

where the symbol $*$ denotes the element-wise multiplication [13], whereas c_1 and c_2 denoted the cognitive and the social acceleration factor. To introduce a certain level of randomness in the above quite-deterministic update rules, two uniform sampled random scalars between 0 and 1, ${}^{(k+1)}\boldsymbol{r}_{1i}$ and ${}^{(k+1)}\boldsymbol{r}_{2i}$, have been considered to increase the domain exploration. The algorithm termination is usually set as the achievement of a priorly set number of iterations k_{max}. However, it is not easy to priorly estimate the correct number of maximum iteration because it is strongly problem-oriented [14]. Therefore, some other approaches can be based directly on the monitoring of the variation of the OF during the iterations. A stop criterion could be based on a predefined number of stagnations, which means that the OF registers small variations within a certain threshold

level for a certain number of iterations. [15] improved the standard PSO formulation introducing the inertia weight term ${}^{k}w$ applied to the previous iteration velocity to manipulate the inertial effect of each particle to the movement The hyperparameters of the PSO need to be fine tuned to reach the best performances. For example the population size defines the level of exploration of the search space and it is suggested to be a number comprise between 20 and 100 when the design vector size is less than 30 [12]. In the following implementations, as suggested in literature e.g. by [12], acceleration factors can be set as constant scalars equal to 2 and inertia weight . A fundamental aspect of the PSO is the information sharing among the agents, defined by the particles interconnection topology, also known as neighbourhood. If the information of every particle is shared with the entire swarm, it is denoted as fully connected or gbest topology. However, this strategy particularly suffers of premature entrapment convergence to local optima. Therefore, lbest models have been proposed to slow down the convergence ensuring enough exploration. Among the different implementations illustrated in [16], in this study, the ring topology has been adopted. Defined a neighbourhood radius and considering a particles indexing order, the information are shared only among the particles who belong to their neighbourhood. In [17], an example of multi-population PSO involves a dynamic topology adjustment during iterations.
Constraint handling in EAs is a challenging task especially because of unfeasible trial solutions. Numerous strategies have been developed and in [18] have been reconducted to five main typologies: penalty functions-based methods, methods based on special operators and representations, methods based on repair algorithms, methods based on the separation between OF and constraints, and hybrid methods. Due to its simplicit, the most adopted method is the penalty approach (death, static, dynamic or adaptive) which delivers an unconstrained version of the problem $\phi(\boldsymbol{x}) = f(\boldsymbol{x}) + H(\boldsymbol{x})$ with $H(\boldsymbol{x})$ as a penalty function [19]. For the preservation of swarm diversity and optimization performances, it is necessary to select the best approach to deal with constraints. Indeed, the death penalty approach does not represent at all the ideal solution because it brings a dreadful loss of information from unfeasible points [18]. In the structural optimization field [20] the static and dynamic penalty functions are the most widespread used constraint handling techniques. The static penalty function $H_s(\boldsymbol{x})$ depends on H_{NVC} the number of constraints that are violated by each particle and H_{SVC} the sum of all violated constraints:

$$H_s(\boldsymbol{x}) = w_1 H_{NVC}(\boldsymbol{x}) + w_2 H_{SVC}(\boldsymbol{x}) \quad ; \quad H_{SVC}(\boldsymbol{x}) = \sum_{p=1}^{n_p} \max\{0, g_p(\boldsymbol{x})\} \tag{3}$$

with w_1 and w_2 as static control parameters usually set to $w_1 = w_2 = 100$ [21]. In this study, $w_1 = 0$ and $1000 < w_2 < 10000$ have been assumed for penalty-based PSO adopted as a comparison with the enhanced multistrategy PSO. The dynamic penalty approach attempted to improve the static version by allowing a more relaxed constraint handling at the beginning and an increasingly penalty value approaching to the end of the k_{max} iterations, according to a ${}^{k}h$

is a dynamic penalty factor [21]:

$$\min_{\boldsymbol{x}\in\Omega}\{f(\boldsymbol{x})+{}^{k}hH_d(\boldsymbol{x})\} \quad \text{with} \quad {}^{k}h=\sqrt{k} \tag{4}$$

$$H_d(\boldsymbol{x})=\sum_{p=1}^{n_p}\theta_p(\boldsymbol{x})[\max\{0,g_p(\boldsymbol{x})\}]^{\gamma_p(\boldsymbol{x})} \quad ; \quad 10<H_d(\boldsymbol{x})<1000 \tag{5}$$

Usual values of the above factors are reported in [21]. A careful calibration of the penalty is crucial because an high value will reduce exploration and diversity, whereas a too low value will not contrast properly the constraint violation.

3 Multistrategy PSO

Considering the Newtonian dynamics-based PSO [11], an enhanced PSO, illustrated in Fig. 1, has been implemented with the most well-acknowledged literature improvements, implementing an additional unfeasible search feature to boost the optimization performance. The very beginning swarm is random sampled in the domain through the latin hypercube sampling. The OF and the level of violation of each constraint are then evaluated. From these evaluations, each particle is associated to a accomplish to a precise goal according to its position in the domain and its violation value. If it is located in the feasible region, its goal is to optimize the OF. On the contrary, when it is unfeasible, its goal is to minimize the envelope of the violation level of the violated constraints. The just explained mechanism inspired the notation of "multistrategy", since it breaks

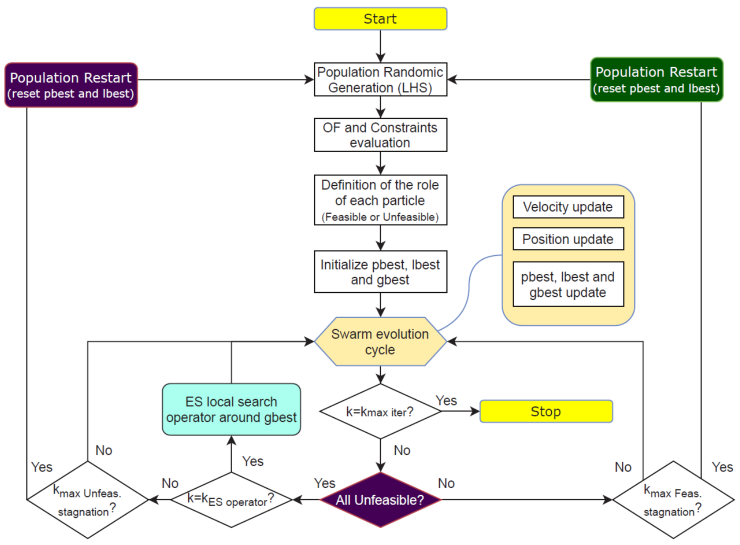
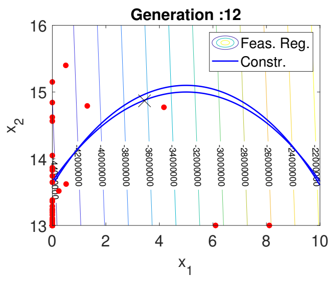
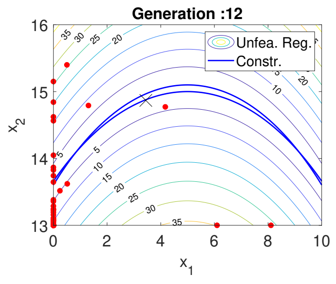

Start
Population Restart (reset pbest and lbest)
Population Randomic Generation (LHS)
Population Restart (reset pbest and lbest)
OF and Constraints evaluation
Definition of the role of each particle (Feasible or Unfeasible)
Velocity update
Position update
pbest, lbest and gbest update
Initialize pbest, lbest and gbest
Swarm evolution cycle
ES local search operator around gbest
k=kmax iter?
Yes
Stop
No
All Unfeasible?
Yes
No
k=kES operator?
Yes
No
kmax Unfeas. stagnation?
Yes
No
kmax Feas. stagnation?
No
Yes

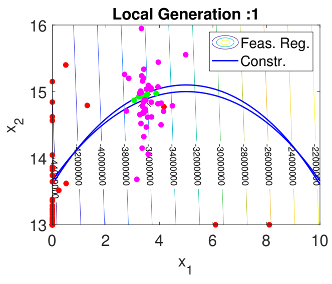
Fig. 1: Enhanced PSO multistrategy flowchart.



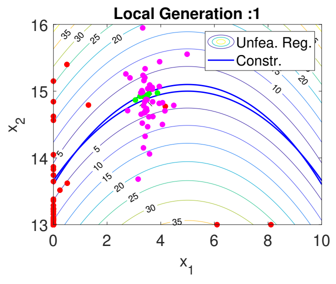
(a) *OF - Generation 12*



(b) *Constraints - Gen. 12*



(c) *OF - ES operator Gen. 1*



(d) *Constraints - ES Gen. 1*

Fig. 2: Example Problem g06 [22]; (a) and (b): the objective function and the constraints violation are depicted as contour plot. The black cross symbol stands for the unfeasible gbest, whereas the red and green dots indicates the swarm positions at a certain iteration instant. In (c) and (d), purple dots: local search population ES search operator; green dots: feasible points.

free from any arbitrary penalty factor or something else, but the PSO relies only on OF and constraint violations. As depicted in Fig. 1, after the definition of the aim of each agent of the first population, the swarm evolution cycle can begin with the usual evolutionary stages such as the Velocity update (1), and the Position update (2). The pbest (self-memory) of each particle is upgraded everytime a new better feasible position is explored, as well as the gbest. When $k_{\max}$ is reached, the evolutionary cycles stop. When the feasible region is very narrow, the swarm may stagnate without founding a region of feasible design parameters. However, in the multistrategy approach, the swarm has attempted to minimize the violations. This means that probably the gbest reaches an unfeasible position located enough close to the admissible feasible area of the domain. This aspect inspired us to enhance the local exploration to attempt to seek the feasible region. Therefore, after $k = k_{\text{ES operator}}$ stagnations, the swarm begins a local exploration adopting another methaheuristic algorithm developed by H.P. Schwefel and I. Rechenberg, the *Evolutionary Strategy* (ES) [9, 23]. This EAs is based on Darwinian Theory of Evolution. A parent population is sampled in the nearby of the unfeasible gbest point. Every memeber of the population generates an offspring by a mutation in its genome which is sampled from a normal Gaussian distribution $x_i + N(0, \sigma)$, ruled by a mutation step σ [23]. Finally, in every iteration only the best individuals will survive to the next generation among parents and offspring. The advantage of ES is related to a single parameter σ which should be tuned. In this study, the self adaptive ES (SA-ES [23]) has been taken into account. Thus, the genomes encodes the mutation step $(x_1, ..., x_n, \sigma)$ and it is indirectly evolved through the fittest individuals. It is even possible to consider a self-adaptive mutation step for each single gene $(x_1, ..., x_n, \sigma_1, ..., \sigma_n)$ [23]. In the current study the uncorrelated steps self adaptive ES has been selected to perform the local search in the nearby of the unfeasible gbest position $\boldsymbol{x}^{Gb,\text{unfea}}$ when the swarm stagnates $k_{\text{ES operator}} = 10$. The size of the parent local population has been set to $N_p = 50$, sampled with a multivariate Gaussian centered in the unfeasible gbest position, with standard deviation of each genome component equal to

$$\sigma_i = |\tau \cdot N(0, 1)| \tag{6}$$

where τ which is the learning rate parameter assumed as $1/\sqrt{N_p}$. $N_o = 100$ offspring have been generated foremost by randomly selected parents with this mutation scheme:

$$\sigma_{i,\text{off}} = \max\left(0, |\sigma_i + N(0, 1)|\right). \tag{7}$$

Next generations have been obtained by updating the covariance of the multivariate Gaussian mixture model. The mating pool is thus composed by parents and offspring, in a $\mu + \lambda$-ES strategy, from which the fittest N_p elements will survive. A maximum number of local iterations has been set to $k_{\text{max,Local}} = 50$, unless a feasible point has been found, becoming the new gbest for the PSO algorithm. Fig. 2 depicts an illustrative example of the ES unfeasible local search operator capabilities. Even though the above unfeasible local search, the admissible feasible area may have not been discovered after further $k_{\text{max Unfeas Stagn}} = 15$ stagnations, thus the PSO is completely restarted from the initial LHS sampling as shown in Fig. 1. On the other hand, if a feasible point is found, the PSO starts again to evolve setting it as the new gbest. When the PSO stagnates $k_{\text{max Feas Stagn}} = 50$ times on the same feasible gbest, the population is restarted with latin hypercube sampling to attempt to scan again the search space exploiting the available iterations left before the termination criterion. In the following, the enhanced multistrategy PSO has been applied to some mathematical benchmark problems and some real-world structural optimization problems.

4 Numerical benchmark problems

Implemented in Matlab environment, the proposed multistrategy PSO solved some constrained mathematical benchmark problems whose mathematical statements are reported in [24]. In total, 13 constrained problems have been considered, the ones with inequalities constraints only. For the sake of comparisons, the current multistrategy PSO has been compared with other PSO with classic penalty approaches proposed by [25]. The code has been adapted for the static and dynamic penalty approach as in (3) and (4). The penalty factors were problem-oriented adjusted and calibrated. The size of the swarm is $N = 100$ with the total maximum iterations of $k_{\max} = 500$ for all the PSOs implementations. Tab. 1 presents the results obtained by 50 execution performed in a independent way and illustrates comparisons such as worst and best obtained results, mean and standard deviation of the OF for the three different implemented PSOs. The results provided by the enhanced multistrategy PSO are in very good agreement with the theoretical global minimum solutions and in accordance with the other PSO penalty-based variants. These results demonstrate that the current multistrategy PSO implementation is effective to cope with constrained mathematical problems, without requiring troublesome calibrations of many arbitrary hyperparameters as it happens for penalty-based techniques.

Table 1: Numerical benchmark examples taken from [24], results for 50 runs.

Problem	**Algorithm**	**Real Optimum**	**best OF**	**worse OF**	**mean**	**std. dev.**
g01	PSO-ES	-15.000	-15.000	-12.002	-14.443	0.8948
	PSO-Static		-15.000	-12.000	-13.938	1.4333
	PSO-Dynamic		-15.000	-12.000	-13.920	1.4546
g02	PSO-ES	0.803619	0.803570	0.609630	0.758960	0.0636
	PSO-Static		0.801460	0.520130	0.710500	0.0736
	PSO-Dynamic		0.793580	0.382850	0.665970	0.0870
g04	PSO-ES	-30665.539	-30666.0	-30666.0	-30666.0	2.197E-05
	PSO-Static		-30666.0	-30665.0	-30665.0	0.8659
	PSO-Dynamic		-31207.0	-30137.0	-31138.2	252.203675
g06	PSO-ES	-6961.81388	-6961.8	-6958.4	-6960.7	0.9752
	PSO-Static		-6973.0	-6973.0	-6973.0	0.0000
	PSO-Dynamic		-6973.0	-6973.0	-6973.0	0.0000
g07	PSO-ES	24.3062091	24.426	27.636	25.413	1.1209
	PSO-Static		24.034	30.203	28.508	1.4351
	PSO-Dynamic		24.477	30.112	27.043	1.8821
g08	PSO-ES	0.095825	0.095825	0.095825	0.095825	6.96E-17
	PSO-Static		0.095825	0.095825	0.095825	6.77E-17
	PSO-Dynamic		0.095825	0.095825	0.095825	7.10E-17
g09	PSO-ES	680.6300573	680.640	680.980	680.730	0.0794
	PSO-Static		680.630	680.720	680.660	0.0175
	PSO-Dynamic		680.630	680.730	680.660	0.0189
g12	PSO-ES	1.000	1.000	1.000	1.000	0.0000
	PSO-Static		1.000	1.000	1.000	2.12E-15
	PSO-Dynamic		1.000	1.000	1.000	0.0000

5 Structural optimization examples

In this final paragraph, some acknowledged literature structural optimization benchmark problems have been solved by the proposed multistrategy PSO. The PSO-ES has been compared in the following with other penalty PSO implementations [25]. To make a more complete comparison with a completely different and independent approach, the GA from Matlab Optimization Toolbox has been also analysed. In particular, a spring design and two different spatial truss optimization problems have been considered [26]. They are configured as a size optimization task, where the main goal is usually to reduce the self-weight w of the structure respecting the safety constraints, which is related to the cost [7]. Denoting with ρ_i the unit weight, the purpose is to find the optimal cross-sectional areas A_i of every i-th structural element which are lower and upper bounded (box-type hyper-rectangle domain) $A_i \in [A_i^{\mathrm{LB}}, A_i^{\mathrm{UB}}]$. In general, two inequality constraints are considered: the assessment of the maximum allowable stress σ_{adm} (strength constraint) and the assessment of the codes admissible displacement δ_{adm} (deformation limitation). For the truss problem, the general optimization problem statement is:

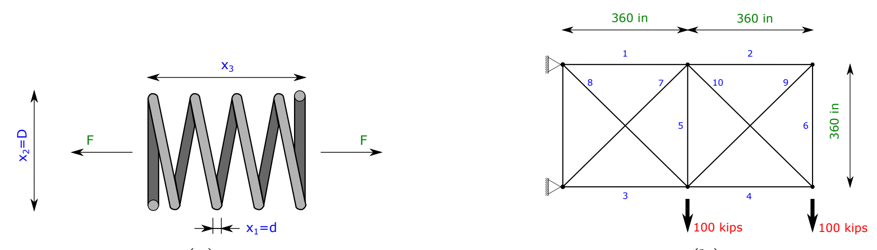
$$\begin{aligned} \min_{\boldsymbol{x}\in\Omega} \quad & f(\boldsymbol{x}) = \sum_{i=1}^{N_el} \rho_i L_i A_i \\ \text{s.t.} \quad & A_i^{\mathrm{LB}} \leq A_i \leq A_i^{\mathrm{UB}} \\ & \sigma_i \leq \sigma_{\mathrm{adm}} \\ & \delta \leq \delta_{\mathrm{adm}} \end{aligned} \tag{8}$$

in which the structure is composed by N_el members of actual length L_i. The structural steel mechanical properties are expressed in imperial units, i.e. the unique unit weight for each member equal to $\rho_i = \rho = 0.1$ lb/in^3 and the Young's modulus of 10^7 psi.

5.1 Tension/compression spring optimization benchmark

The first problem is referred to a well-acknowledged continuous constrained engineering problem which aims to find the minimum weight of a spring, as depicted in Fig. 3 (a). Under the action of an axial force F, some inequality constraints are posed by considering respectfulness of requirements related to the minimum deflection, on shear stress and on surge frequency. The design vector is composed of three parameters: the wire diameter ($x_1 = d$), the mean coil diameter $x_2 = D$ and the number of active coils x_3. The design parameters are bounded in the following intervals: $0.05 \leq x_1 \leq 2.0$, $0.25 \leq x_2 \leq 1.3$ and $2.0 \leq x_3 \leq 15.0$ The problem formulation is stated as [27]:

$$\begin{aligned} \min_{\boldsymbol{x}\in\Omega} \quad & f(\boldsymbol{x}) = (x_3+2)x_2x_1^2 \\ \text{s.t.} \quad & g_1(\boldsymbol{x}) = 1 - \frac{x_2^3x_3}{7.178x_1^4} \leq 0 \\ & g_2(\boldsymbol{x}) = \frac{4x_2^2 - x_1x_2}{12.566(x_2x_1^3) - x_1^4} + \frac{1}{5.108x_1^2} \leq 0 \\ & g_3(\boldsymbol{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \\ & g_4(\boldsymbol{x}) = \frac{x_2+x_1}{1.5} - 1 \leq 0 \end{aligned} \tag{9}$$



(a) (b)

Fig. 3: (a) Spring problem. (b) Ten bars truss problem.

Table 2: Results for 100 runs of tension/compression spring design.

	Cross Section [in^2]				
Parameter	**Solution** [27]	**Static Pen. PSO**	**Dynamic Pen. PSO**	**GA**	**PSO-ES**
1	0.051690	0.051690	0.051470	0.051770	0.051759
2	0.356750	0.356736	0.351474	0.358910	0.358400
3	11.287126	11.287891	11.603168	11.141985	11.191072
best OF [lb]	0.012665	0.012665	0.012666	0.012642	0.012665
worse OF [lb]	-	0.017416	0.015080	0.321591	0.012722
mean [lb]	-	0.012933	0.012928	0.023281	0.012714
std. dev. [lb]	-	0.000599	0.000445	0.042785	0.000013

In Tab. 2, the comparison results are illustrated. The theoretical optimal solution of the above-mentioned problem is reported in the first column of Tab. 2. All the considered algorithms provide great performances on this simple benchmark problem which has only three design parameters, even the penalty approaches. It is worth noting that the mean and the standard deviation of the proposed multistrategy PSO-ES appear to be lower than the other algorithms, demonstrating the reliability of the proposed method to solve engineering optimization tasks.

5.2 Ten bars truss design optimization

As depicted in Fig. 3 (b), a planar ten bars truss cantilever structure is now considered. The cantilever span is 720in (1in = 25.4mm) and 360 in depth with elements numbered from 1 to 10. The loads are two downward equal forces of 100 kips (1kips = 4.4482kN). The cross section areas to be optimized are considered as continuous variables within the admissible range $[0.1, 35]$ in^2. The maximum admissible stress is $\sigma_{\text{adm}} = \pm 25$ ksi, whereas the maximum displacement has been set to $\delta_{\text{adm}} = \pm 2$ in for every structural node. The population of the PSO has been set to 50 individuals and the stopping criterion has been set to a maximum number of iterations of 500 both for PSO-ES and GA. Since their dreadful results, for the penalty-based PSO the swarm size has been increased to 500 particles. 100 runs have been independently performed in total, calculating the mean and standard deviation of the OF. Tab. 3 illustrates the optimiza-

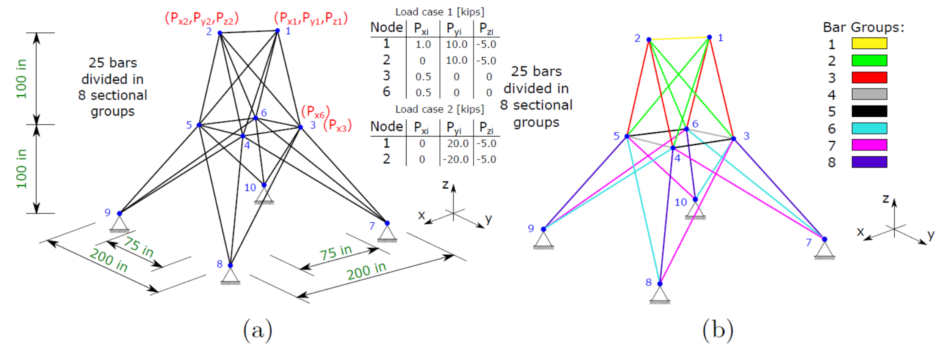
Table 3: Results for 100 runs of the optimization of the truss with 10 elements.

	Cross Section [in^2]				
Element	**Ref. Sol. from** [26]	**PSO-Static**	**PSO-Dynamic**	**GA**	**PSO-ES**
1	28.920	29.6888	30.3092	30.145	30.372
2	0.100	18.3211	14.7464	0.100	0.110
3	24.070	19.9891	16.5717	22.466	23.644
4	13.960	18.2381	25.1945	15.112	15.391
5	0.100	2.3404	4.5489	0.101	0.101
6	0.560	20.8674	26.1207	0.543	0.496
7	21.950	21.1805	32.2698	21.667	20.984
8	7.690	16.0851	0.2168	7.577	7.410
9	0.100	6.0845	7.5871	0.100	0.103
10	22.090	25.5632	23.524	21.695	21.378
best OF [lb]	5076.310	6141.986	6333.035068	5063.250	5063.328
worse OF [lb]	-	8415.134	8675.749551	5144.148	5229.108
mean [lb]	-	7294.455	7501.394582	5079.744	5076.473
std. dev. [lb]	-	516.7823	475.3885728	14.1194	24.8666

tion results comparing PSO-ES with static penalty and dynamic penalty PSO and, finally, with GA. The penalty approaches provide underwhelming results, demonstrating their unsuitability when solving structural optimization tasks. Conversely, the proposed multistrategy PSO-ES algorithm delivers excellent results in agreement with another completely independent implementation such as the GA.

5.3 Truss optimization with twenty-five bars

The last problem is depicted in Figure 4 (a), and it consists of a twenty-five bars three-dimensional truss tower structure with ten numbered structural nodes and under two different load cases. The footprint is a square shaped of side 200in, which tapers to 75 in at an elevation of 100 in, and finally reaches the maximum elevation from the ground at 200in. Since the cross section areas have been gathered into 8 groups as shown in Fig. 4 (b), the 8 design parameters are considered as continuous variables whose upper and lower bounds are defined by the interval $[0.01, 3.40]$ in^2. The maximum admissibile stress is $\sigma_{\text{adm}} = \pm 40$ ksi, whereas the maximum displacement is $\delta_{\text{adm}} = \pm 0.35$ in in every direction. The population of the PSO has been set to 50 individuals and the stopping criterion has been set to a maximum number of iterations of 500 both for PSO-ES and GA. Since their dreadful results, for the penalty-based PSO the swarm size has been increased to 500 particles. 100 runs have been independently performed calculating the mean and standard deviation of the OF. Tab. 4 summarizes the results comparing PSO-ES with static and dynamic penalty PSO and, finally, with GA. Even in this last example, the penalty approaches provide underwhelming results, demonstrating their unsuitability when solving structural optimization tasks. Conversely, the proposed multistrategy PSO-ES algorithm delivers excellent re-



(P_{x2}, P_{y2}, P_{z2})
(P_{x1}, P_{y1}, P_{z1})
(P_{x6})
(P_{x3})
25 bars divided in 8 sectional groups
100 in
100 in
75 in
200 in
75 in
200 in
z
x
y
(a)
25 bars divided in 8 sectional groups
Bar Groups:
1 🟨
2 🟩
3 🟥
4 ⬜
5 ⬛
6 🟦
7 🟪
8 🟪
(b)

Load case 1 [kips]

Node	P_{xi}	P_{yi}	P_{zi}
1	1.0	10.0	-5.0
2	0	10.0	-5.0
3	0.5	0	0
6	0.5	0	0

Load case 2 [kips]

Node	P_{xi}	P_{yi}	P_{zi}
1	0	20.0	-5.0
2	0	-20.0	-5.0

Fig. 4: (a) Twenty-five bars truss problem. (b) Eight bar groups representation.

Table 4: Results for 100 runs of the optimization of the truss with 25 elements.

	Cross Section [in^2]				
Element	**Ref. Sol. from** [26]	**PSO-Static**	**PSO-Dynamic**	**GA**	**PSO-ES**
1	28.920	29.6888	30.3092	30.145	30.372
2	0.100	18.3211	14.7464	0.100	0.110
3	24.070	19.9891	16.5717	22.466	23.644
4	13.960	18.2381	25.1945	15.112	15.391
5	0.100	2.3404	4.5489	0.101	0.101
6	0.560	20.8674	26.1207	0.543	0.496
7	21.950	21.1805	32.2698	21.667	20.984
8	7.690	16.0851	0.2168	7.577	7.410
9	0.100	6.0845	7.5871	0.100	0.103
10	22.090	25.5632	23.524	21.695	21.378
best OF [lb]	5076.310	6141.986	6333.035068	5063.250	5063.328
worse OF [lb]	-	8415.134	8675.749551	5144.148	5229.108
mean [lb]	-	7294.455	7501.394582	5079.744	5076.473
std. dev. [lb]	-	516.7823	475.3885728	14.1194	24.8666

sults in agreement with another completely independent implementation such as the GA.

6 Results analysis summary and discussion

Similarly to [28], in this last paragraph, the main outcomes and critical observations are finally discussed. Tab. 1 demonstrated the strength of the proposed PSO to deal with mathematical benchmark with inequality constraints. 50 independent runs have been performed and basic statistics have been extracted. On average, the mean value of the multistrategy PSO-ES approaches to the theoretical solutions in a more reliable way, since the standard deviations appear to be lower than the other penalty-based PSO. Indeed, these latter sometimes stall quite far from the theoretical solutions and often with almost nil standard deviation. This fact demonstrates the penalty-based PSO tendency to be trapped in local optima. The multistrategy PSO has been finally tested on structural optimization real-world case studies. Tabs. 2, 3 and 4 reported the results for 100 independent runs. These values demonstrated the dreadful performances of the penalty-based PSO algorithm to deal with this kind of problem. Conversely, the multistrategy PSO-ES behaves excellently, approaching very close to the reference solutions in a reliable way. For the sake of comparison, another EA has been tested in order to produce a valuable comparison with another completely different technique rather than the currently adopted PSO ones. Nevertheless, this last comparison further consolidates the multistrategy PSO-ES effectiveness to reach the optimal solution. Futhermore, this comparison with GA contributed pointing out the reliability of the proposed method, because the solutions provided lower standard deviations with respect to the GA in two problems out of three.

7 Conclusions

In the current study, the state-of-art improvements of the PSO have been implemented with a multistrategy approach [15, 16]. In addition, in place of the penalty functions method to cope with constraint handling, the information of violation was used to govern the optimization. When PSO excessively stagnates in the unfeasible region, the swarm begins a local search based on a self-adaptive ES, increasing the nearby exploration, hopefully close to the feasible region. This feature powers up the current algorithm with respect to other PSO. Moreover, outstanding results for real-world structural optimization benchmarks have been obtained. In all the problems, the PSO-ES provide solutions closer to the theretical one and in a reliable way, demonstrated by the less standard deviation with respect to the GA or the other PSO implementations. The hybridisation of the PSO with machine learning and probabilistic approaches, e.g. estimation distribution algorithm (EDA) [29], may represent very promising future studies to further improve the capabilities of swarm-based algorithms.

References

1. Fabio Di Trapani, Giovanni Tomaselli, Antonio Pio Sberna, Marco Martino Rosso, Giuseppe Carlo Marano, Liborio Cavaleri, and Gabriele Bertagnoli. Dynamic response of infilled frames subject to accidental column losses. In Carlo Pellegrino, Flora Faleschini, Mariano Angelo Zanini, José C. Matos, Joan R. Casas, and Alfred Strauss, editors, *Proceedings of the 1st Conference of the European Association on Quality Control of Bridges and Structures*, pages 1100–1107, Cham, 2022. Springer International Publishing.
2. Rebecca Asso, Raffaele Cucuzza, Marco Martino Rosso, Davide Masera, and Giuseppe Carlo Marano. Bridges monitoring: an application of ai with gaussian processes. In *14th International Conference on Evolutionary and Deterministic*

Methods for Design, Optimization and Control. Institute of Structural Analysis and Antiseismic Research National Technical University of Athens, 2021.

3. Angelo Aloisio, Dag Pasquale Pasca, Luca Battista, Marco Martino Rosso, Raffaele Cucuzza, Giuseppe Marano, and Rocco Alaggio. Indirect assessment of concrete resistance from fe model updating and young's modulus estimation of a multi-span psc viaduct: Experimental tests and validation. *Elsevier Structures*, 37:686–697, 01 2022.
4. Laura Sardone, Marco M. Rosso, Raffaele Cucuzza, Rita Greco, and Giuseppe Carlo Marano. Computational design of comparative models and geometrically constrained optimization of a multi domain variable section beam based on timoshenko model. In *14th International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control*. Institute of Structural Analysis and Antiseismic Research National Technical University of Athens, 2021.
5. Raffaele Cucuzza, Marco Martino Rosso, and Giuseppe Marano. Optimal preliminary design of variable section beams criterion. *SN Applied Sciences*, 3, 08 2021.
6. Raffaele Cucuzza, Carlo Costi, Marco Martino Rosso, Marco Domaneschi, Giuseppe Carlo Marano, and Davide Masera. Optimal strengthening by steel truss arches in prestressed girder bridges. *Proceedings of the Institution of Civil Engineers - Bridge Engineering*, 0(0):1–21, 0.
7. Marco M Rosso, Raffaele Cucuzza, Fabio Di Trapani, and Giuseppe C Marano. Nonpenalty machine learning constraint handling using pso-svm for structural optimization. *Advances in Civil Engineering*, 2021, 2021.
8. Marco Martino Rosso, Raffaele Cucuzza, Angelo Aloisio, and Giuseppe Carlo Marano. Enhanced multi-strategy particle swarm optimization for constrained problems with an evolutionary-strategies-based unfeasible local search operator. *Applied Sciences*, 12(5), 2022.
9. Rafael Martí, Panos M. Pardalos, and Mauricio G. C. Resende. *Handbook of Heuristics*. Springer Publishing Company, Incorporated, 1st edition, 2018.
10. Nikolaos D. Lagaros, Manolis Papadrakakis, and George Kokossalakis. Structural optimization using evolutionary algorithms. *Computers & Structures*, 80(7):571–589, 2002.
11. J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.
12. Giuseppe Quaranta, Walter Lacarbonara, and Sami Masri. A review on computational intelligence for identification of nonlinear dynamical systems. *Nonlinear Dynamics*, 99, 01 2020.
13. Vagelis Plevris. *Innovative Computational Techniques for the Optimum Structural Design Considering Uncertainties*. PhD thesis, Institute of Structural Analysis and Seismic Research, School of Civil Engineering, National Technical University of Athens (NTUA), 2009.
14. Bo Li and RenYue Xiao. The particle swarm optimization algorithm: How to select the number of iteration. pages 191 – 196, 12 2007.
15. Yuhui Shi and B.Gireesha Obaiahnahatti. A modified particle swarm optimizer. volume 6, pages 69 – 73, 06 1998.
16. Angelina Medina, Gregorio Toscano Pulido, and José Ramírez-Torres. A comparative study of neighborhood topologies for particle swarm optimizers. pages 152–159, 01 2009.

17. J.J. Liang and P.N. Suganthan. Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism. In *2006 IEEE International Conference on Evolutionary Computation*, pages 9–16, 2006.
18. Carlos A Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11):1245–1287, 2002.
19. Ahmad rezaee jordehi. A review on constraint handling strategies in particle swarm optimisation. *Neural Computing and Applications*, 26, 01 2015.
20. George G. Dimopoulos. Mixed-variable engineering optimization based on evolutionary and social metaphors. *Computer Methods in Applied Mechanics and Engineering*, 196(4):803–817, 2007.
21. Konstantinos Parsopoulos and Michael Vrahatis. *Particle Swarm Optimization Method for Constrained Optimization Problem*, volume 76, pages 214–220. 01 2002.
22. Petru-Aurelian Simionescu, DG Beale, and Gerry Vernon Dozier. Constrained optimization problem solving using estimation of distribution algorithms. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, volume 1, pages 296–302. IEEE, 2004.
23. Hans-Georg Beyer. Toward a theory of evolution strategies: Self-adaptation. *Evolutionary Computation*, 3(3):311–347, 1995.
24. Wen Long, Ximing Liang, Yafei Huang, and Yixiong Chen. A hybrid differential evolution augmented lagrangian method for constrained numerical and engineering optimization. *Computer-Aided Design*, 45(12):1562–1574, 2013.
25. Mahamad Alam. Codes in matlab for particle swarm optimization. 03 2016.
26. C.V. Camp and M. Farshchin. Design of space trusses using modified teaching–learning based optimization. *Engineering Structures*, 62-63:87–97, 2014.
27. Leticia Cagnina, S. Esquivel, and Carlos Coello. Solving engineering optimization problems with the simple constrained particle swarm optimizer. *Informatica (Slovenia)*, 32:319–326, 01 2008.
28. Pawan Bhambu, Sandeep Kumar, and Kavita Sharma. Self balanced particle swarm optimization. *International Journal of System Assurance Engineering and Management*, 9(4):774–783, August 2018.
29. Martin Pelikan, Mark W Hauschild, and Fernando G Lobo. Estimation of distribution algorithms. pages 899–928, 2015.