Efficiency and Robustness of Three Metaheuristics in the Framework of Structural Optimization

Nikos D. Lagaros, Dimos C. Charmpis

Efficiency and robustness of three metaheuristics in the framework of structural optimization

Nikos D. Lagaros[1] and Dimos C. Charmpis[2]

[1] Institute of Structural Analysis & Seismic Research,
National Technical University Athens
Zografou Campus, Athens 15780, Greece
nlagaros@central.ntua.gr

[2] Department of Civil and Environmental Engineering,
University of Cyprus,
75 Kallipoleos Str., P.O. Box 20537, 1678 Nicosia, Cyprus
charmpis@ucy.ac.cy

Abstract. Due to the technological advances in computer hardware and software tools, structural optimization has been gaining continuously increasing interest over the last two decades. The purpose of the present work is to quantitatively compare three metaheuristic optimization algorithms, namely the Differential Evolution, Harmony Search and Particle Swarm Optimization methods, in the framework of structural optimization. The comparison of the optimizers is performed with reference to their efficiency (overall computing demands) and robustness (capability to detect near-optimal solutions). The optimum design of a real-world overhead traveling crane is used as the test bed application for conducting optimization test runs.

Keywords: Structural Optimization, Metaheuristics, Differential Evolution, Harmony Search, Particle Swarm Optimization.

1 Introduction

Structural optimization typically aims in detecting the optimum design by minimizing the cost (or weight) of a structure subject to certain behavioral constraints imposed by relevant design codes. Structural optimization applications are associated with multiple local minima, large and non-convex search spaces and several (usually conflicting) constraints to be satisfied. As a result, such applications do not favor the use of Mathematical Programming (gradient-based) algorithms, which exhibit a satisfactory local rate of convergence to the nearest optimum, but they cannot assure that the global optimum can be found when confronted with complex optimization problems. Therefore, various heuristic probabilistic-based search algorithms, which present a better global behavior and are less vulnerable to local optima, have been applied during the last decades to meet the high demands of structural optimization problems.

Several structural optimization applications have been reported, which utilize algorithms inspired by natural phenomena, such as Evolutionary Programming [1], Genetic Algorithms [2] and Evolution Strategies [3], among others. More recently, new metaheuristic optimizers have been developed based on the simulation of social interactions among members of a specific species looking for food or resources in general. One such method is Particle Swarm Optimization (PSO) [4], which is based on the behavior reflected in flocks of birds, bees and fish that adjust their physical movements to avoid predators and seek for food. Two other metaheuristics with growing interest within the structural optimization community are Differential Evolution (DE) [5,6] and Harmony Search (HS) [7,8].

The purpose of the present work is to provide quantitative comparison results on the performance of the three aforementioned metaheuristics (DE, HS and PSO) in the framework of structural optimization. A typical overhead traveling crane (the crane bridge span, the trolley, the end carriages and the runway beam span) was selected from the catalogues of GH Cranes SA (a manufacturer of elevation systems) as the test bed problem for conducting optimization test runs. A 3D CAD model was developed for the crane and its cross-sectional dimensions have been optimized using DE, HS and PSO subject to constraints imposed by Eurocode Standards.

2 The Metaheuristic Algorithms

This section provides a short description of the three metaheuristic optimization algorithms (DE, HS and PSO) assessed in the present work. All three algorithms are applied to an optimization problem, which can be formulated as follows:

$$\begin{aligned} &\min_{x \in F} f(\boldsymbol{x}) \\ &\textit{subject to } g(\boldsymbol{x}) \geq 0.0 \end{aligned} \tag{1}$$

where f denotes the objective function, $\boldsymbol{x}$ is the vector of design (decision) variables, F represents the design (search) space and g are the constraint functions of the problem.

2.1 Differential Evolution (DE)

In 1995, Storn and Price [6] proposed a new floating point evolutionary algorithm for global optimization and named it Differential Evolution (DE). DE has a special kind of differential operator, which is invoked to create new offspring from parent chromosomes instead of the classical crossover or mutation. DE is a novel parallel direct search method which utilizes NP parameter vectors $\boldsymbol{x}_{i,G}$ (i=1,..,NP) as a population for each generation G. The initial population is generated randomly. As a rule, a uniform probability distribution for all random decisions is assumed. In case a preliminary solution is available, the initial population is often generated by adding normally distributed random deviations to the nominal solution $\boldsymbol{x}_{\text{nom},0}$. The crucial idea behind DE is a new scheme for generating new vectors. DE generates new vectors by adding the weighted difference vector between two population members to

a third member. If the resulting vector yields a lower objective function value than a predetermined population member, then the newly generated vector replaces the vector with which it was compared. The comparison vector can - but needs not to be - part of the generation process mentioned above. The best parameter vector $\boldsymbol{x}_{\text{best},G}$ is evaluated for every generation G in order to keep track of the progress made during the optimization process. The extraction of distance and direction information from the population to generate random deviations results in an adaptive scheme with excellent convergence properties. Several variants of DE have been proposed so far, the two most popular of which are presented below as schemes DE1 and DE2.

Scheme DE1. The first variant of DE works as follows: for each vector $\boldsymbol{x}_{i,G}$, a trial vector $\boldsymbol{v}$ is generated according to:

$$\boldsymbol{v} = \boldsymbol{x}_{r1,G} + F \cdot (\boldsymbol{x}_{r2,G} - \boldsymbol{x}_{r3,G}) \tag{2}$$

The integers $r1$, $r2$ and $r3$ are chosen randomly from the interval [1,NP] and are different from the running index i. F is a real and constant factor, which controls the amplification of the differential variation ($\boldsymbol{x}_{r2,G} - \boldsymbol{x}_{r3,G}$). In order to increase the diversity of the parameter vectors, the vector $\boldsymbol{u} = [u_1, u_2, \ldots, u_D]^T$ is defined as follows:

$$u_j = \begin{cases} v_j \; for \; j = \langle n \rangle_D, \langle n+1 \rangle_D \ldots, \langle n+L-1 \rangle_D \\ (x_{i,G})_j \qquad\qquad otherwise \end{cases} \tag{3}$$

where the acute brackets $\langle \; \rangle_D$ denote the modulo function with modulus D.

Scheme DE2. In the second variant of DE, the trial vector $\boldsymbol{v}$ for each vector $\boldsymbol{x}_{i,G}$ is generated through the expression:

$$\boldsymbol{v} = \boldsymbol{x}_{i,G} + \lambda \times (\boldsymbol{x}_{best,G} - \boldsymbol{x}_{i,G}) + F \times \left(\boldsymbol{x}_{r2,G} - \boldsymbol{x}_{r3,G} \right) \tag{4}$$

which introduces an additional control variable λ. The idea behind λ is to provide a means to enhance the greediness of the scheme by involving the current best vector $\boldsymbol{x}_{\text{best},G}$. This feature can be useful for non-critical objective functions. The construction of $\boldsymbol{u}$ from $\boldsymbol{v}$ and $\mathbf{x}_{\text{i,G}}$, as well as the decision process, are identical to those of DE1.

2.2 Harmony Search (HS)

The Harmony Search (HS) algorithm was originally inspired by the improvisation process of Jazz musicians. Each musician corresponds to a decision variable; musical instrument's pitch range corresponds to decision variable's value range; musical harmony at certain time corresponds to the solution vector at certain iteration; and audience's aesthetics corresponds to the objective function. Just like musical harmony is improved as time passes, the solution vector is improved iteration by iteration.

HS utilizes a number of parameters: harmony memory size (HMS), harmony memory considering rate (HMCR), pitch adjusting rate (PAR), maximum improvisation (MI) and fret width (FW). HMS is the number of solution vectors

simultaneously handled by the algorithm. HMCR is the probability of choosing one value from the musician's memory ($0 \leq HMCR \leq 1$). Thus, (1-HMCR) is the rate with which HS picks one value randomly from the total value range. PAR is the rate with which HS tweaks the value originally picked from memory ($0 \leq RAR \leq 1$). Thus, (1-PAR) is the rate with which HS keeps the original value obtained from memory. MI is the number of iterations. HS improvises one harmony (= vector) at each iteration. Finally, FW refers to the term fret, which is the metallic ridge on the neck of a string instrument dividing the neck into fixed segments; each fret represents one semitone. In the context of the HS algorithm, frets are arbitrary points which divide the total value range into fixed segments and fret width (FW) is the length between two neighboring frets. Uniform FW is normally used in HS. Mahdavi et al. [7] suggested that PAR increases linearly and FW decreases exponentially with iterations:

$$PAR(I) = PAR_{\min} + (PAR_{\max} - PAR_{\min}) \times \frac{1}{MI} \tag{5}$$

$$FW(I) = FW_{\max} \exp\left[\ln\left(\frac{FW_{\min}}{FW_{\max}} \right) \frac{I}{MI} \right] \tag{6}$$

Once the optimization problem is formulated and the parameter values are set, a random tuning process is performed. The HS algorithm initially improvises many random harmonies. The number of random harmonies is at least HMS (this number can be more than HMS, such as two or three times HMS [8]). Then, top-HMS harmonies are selected as starting vectors. Musician's harmony memory (HM) can be considered as a matrix.

$$HM = \left[\begin{array}{cccc|c} x_1^1 & x_2^1 & x_3^1 \ldots & x_v^1 & f(x)^1 \\ x_1^2 & x_2^2 & \chi_3^2 \ldots & x_v^2 & f(x)^2 \\ \ldots & \ldots & \ldots & \ldots & \ldots. \\ x_1^{HMS} & x_2^{HMS} & x_3^{HMS} \ldots & x_v^{HMS} & f(x)^{HMS} \end{array} \right] \tag{7}$$

Once the HM is prepared, a new vector (=harmony) is improvised based on the following operators:

Memory Consideration: When HS determines the new value $\boldsymbol{x}_i$, it randomly picks the j-th value from HM = $\{x_i^1 \ldots x_i^{HMS}\}$ with probability HMCR. The appendix j is taken from a uniform distribution $U(0, 1)$:

$$\mathrm{j} \leftarrow \mathrm{int}(\mathrm{U}(0,1) \,.\, \mathrm{HMS}) + 1 \tag{8}$$

Pitch Adjustment: Once the value $\boldsymbol{x}_i^{new}$ is randomly picked from HM in the above memory consideration process, it can be further adjusted to neighboring values by adding a certain amount to the value, with probability of PAR. For a discrete variable, if $\boldsymbol{x}_i(k) = \boldsymbol{x}_i^{new}$, the pitch-adjusted value becomes $\boldsymbol{x}_i(k+m)$, where $m \in \{-1, 1\}$ normally; and for a continuous variable, the pitch-adjusted value becomes $x_i^{new} + \Delta$,

where $\Delta = U(0,1) * FW(i)$ normally. The above-mentioned three basic operations (random selection, memory consideration and pitch adjustment) can be expressed as follows:

$$\text{xj} \leftarrow \begin{cases} \begin{cases} xi \in \{xi(1),.....xi(k)....,xi(Ki)\} \\ xu \in [xi^L, xi^U] \end{cases} w.p.(1-HMCR) \\ xi \in HM = \{xi^1, xi^2, ..., xi^{HMS}\} w.p.HMCR(1-PAR) \\ \begin{cases} xi(k+m)if..xi(k) \in HM \\ xi + \Delta...if..xi \in HM \end{cases} w.p.HMCR * PAR \end{cases} \tag{9}$$

2.3 Particle Swarm Optimization (PSO)

In Particle Swarm Optimization (PSO), multiple candidate solutions coexist and collaborate simultaneously. Each solution is called a 'particle' that has a position and a velocity in the multidimensional design space. A particle 'flies' in the problem search space looking for the optimal position. As 'time' passes during its quest, a particle adjusts its velocity and position according to its own 'experience', as well as the experience of other (neighbouring) particles. Particle's experience is built by tracking and memorizing the best position encountered. As every particle remembers the best position it has visited during its 'flight', the PSO possesses a memory. A PSO system combines local search method (through self experience) with global search method (through neighbouring experience), attempting to balance exploration and exploitation.

Each particle maintains two basic characteristics, velocity and position, in the multi-dimensional search space that are updated as follows:

$$\boldsymbol{v}^j(t+1) = w\boldsymbol{v}^j(t) + c_1\boldsymbol{r}_1 \circ \left(\boldsymbol{x}^{\mathrm{Pb},j} - \boldsymbol{x}^j(t)\right) + c_2\boldsymbol{r}_2 \circ \left(\boldsymbol{x}^{\mathrm{Gb}} - \boldsymbol{x}^j(t)\right) \tag{10}$$

$$\boldsymbol{x}^j(t+1) = \boldsymbol{x}^j(t) + \boldsymbol{v}^j(t+1) \tag{11}$$

where w is the inertia weight parameter, $\boldsymbol{v}^j(t)$ denotes the velocity vector of particle j at time t, $\boldsymbol{x}_j(t)$ represents the position vector of particle j at time t, vector $\boldsymbol{x}^{\mathrm{Pb},j}$ is the personal best ever position of the j-th particle, and vector $\boldsymbol{x}^{\mathrm{Gb}}$ is the global best location found by the entire swarm. The acceleration coefficients c_1 and c_2 indicate the degree of confidence in the best solution found by the individual particle ($c1$ - cognitive parameter) and by the whole swarm (c_2 - social parameter), respectively, while r_1 and r_2 are two random vectors uniformly distributed in the interval [0,1]. The symbol "$\circ$" of Eq. (10) denotes the Hadamard product, i.e. the element-wise vector or matrix multiplication.

Fig. 1 depicts a particle's movement in a two-dimensional design space. The particle's current position $\boldsymbol{x}^j(t)$ at time t is represented by the dotted circle at the lower left of the drawing, while the new position $\boldsymbol{x}^j(t+1)$ at time $t+1$ is represented by the dotted bold circle at the upper right hand of the drawing. The figure shows how the

particle's movement is affected by: (i) it's velocity $\boldsymbol{v}^j(t)$; (ii) the personal best ever position of the particle, $\boldsymbol{x}^{\mathrm{Pb},j}$, at the right of the figure and (iii) the global best location found by the entire swarm, $\boldsymbol{x}^{\mathrm{Gb}}$, at the upper left of the figure.

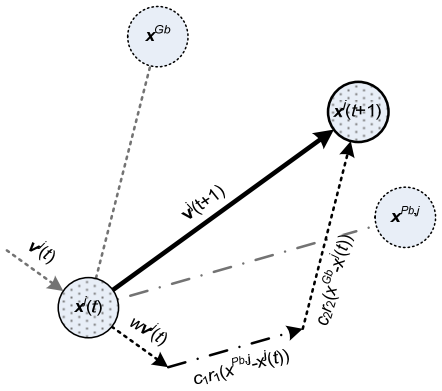

Fig. 1. PSO: Visualization of the particle's movement in a two-dimensional design space.

3 Test Case

The three metaheuristic algorithms described in the previous section have been applied for the sizing optimization of a travelling Overhead Crane (Fig. 2). The design variables (11 in total) are web, top and bottom flange sizes both of the box girder and the runway beams and the thickness (Fig. 3). The design variables of the box girder are the breadth of the top flange (DV1), the distance of the webs (DV2), the height of the girder (DV3) and the thicknesses of top flange, bottom flange, webs (DV4, DV5 and DV6, respectively). The design variables of the runway beams are the breadth of the bottom flange (DV7), the height of the beams (DV8) and the thicknesses of top flange, bottom flange, web (DV9, DV10 and DV11, respectively). The objective function of the optimization problem corresponds to the total mass of the crane system. The constrains of the problem are imposed based on the Structural Eurocodes (Chapter 2): (i) the von Misses equivalent stress must not exceed the prescribed stress limit state of the selected material used for the crane (S235) divided by the coefficient 1.1; (ii) the vertical deflection must not exceed 15mm (=L/600, where L is the span of the bridge); (iii) the vertical deflection must not exceed 15mm (=L/600). Constraints violation is taken into account by penalizing the objective function.

Each of algorithms DE (scheme DE2), HS and PSO was executed 1,000 times. The performance results obtained are summarized in Tables 1 and 2 (CoV is defined as the standard deviation divided by the mean).

DE was found to give the minimum mass for the test example considered, while PSO also yielded almost the same minimum result. In general, DE and PSO appear to be rather robust algorithms, since they yield solutions close to the best result achieved with low variance when compared between several test runs. HS provides solutions of lower quality and with larger variation between several test runs.

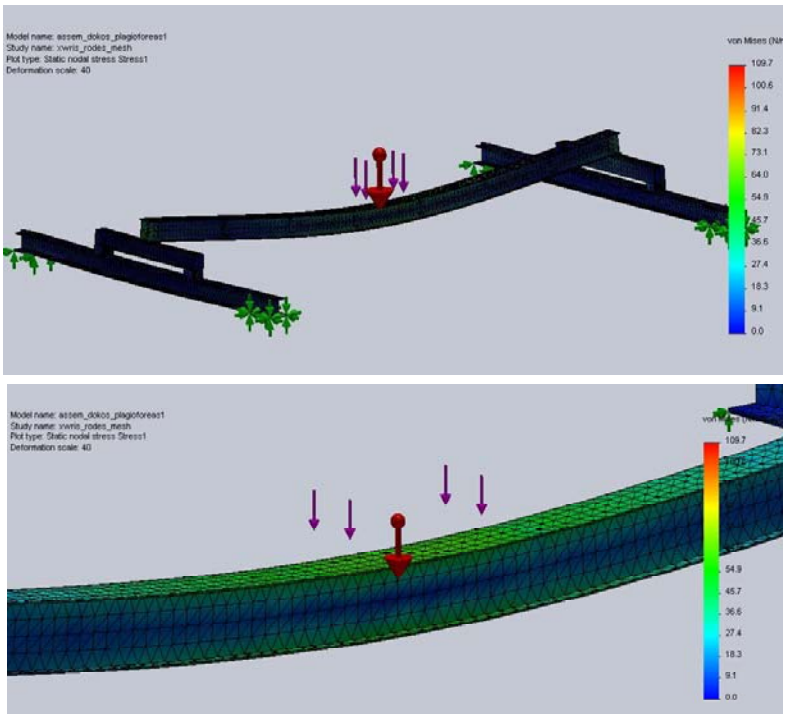

Fig. 2. Views of the crane and its finite element mesh.

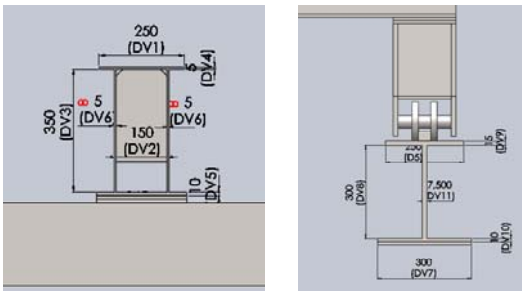

Fig. 3. The 11 design variables.

Table 1. Minimum objective function value achieved by the three metaheuristics.

Optimization Method	Minimum mass achieved (Kg)		
	mean (CoV)	min	max
PSO	2007.3 (0.33%)	1999.8	2071.7
DE	2001.1 (0.41%)	1998.1	2084.1
HS	2286.2 (2.43%)	2076.8	2322.3

Table 2. Computational cost induced by the three metaheuristics (a time-consuming structural analysis of the crane is performed at each iteration of the optimization algorithms).

Optimization Method	Number of iterations		
	mean (CoV)	min	max
PSO	60912.0 (33.8%)	33000	157500
DE	95613.8 (80.5)	5000	371000
HS	5477.5 (72.6)	4422	20100

In terms of computing demands, HS seems to be the most inexpensive of the three meatheuristics. DE appears to be the computationally most demanding algorithm, while PSO performance lies between the ones of HS and DE.

5 Conclusions

Differential Evolution (DE) is the optimization method that was found to give the best design vector, but required the largest amount of iterations for convergence. Harmony Search (HS) is the optimization method that provided the worst design vector, but converged faster than both DE and PSO. PSO yielded solutions of almost the same quality as DE, but with less iteration demands.

In general, it can be stated that efficiency is inversely proportional to robustness. A fast algorithm (HS) cannot provide the solution quality of a robust yet computationally expensive algorithm (DE). PSO seems to be a good compromise between HS and DE, although it exhibits substantially larger computational demands that HS.

References

1. Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial Intelligence through Simulated Evolution. John Wiley (1966).
2. Goldberg, D.E.: Genetic Algorithms in Search Optimization and Machine Learning. Addison Wesley (1989).
3. Rechenberg, I.: Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. (1971). Reprinted by Fromman-Holzboog (1973).
4. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks IV (1995).
5. Price, K., Storn, R., Lampinen J.: Differential Evolution - A Practical Approach to Global Optimization. Springer, Berlin (2005).
6. Storn, R., Price, K.: Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces". Technical Report TR-95-012, International Computer Science Institute (1995).
7. Mahdavi, M., Fesanghary, M., Damangir, E.: An improved harmony search algorithm for solving optimization problems. Applied Mathematics and Computation 188, 1567--1579 (2007).
8. Degertekin, S.: Optimum design of steel frames using harmony search algorithm. Structural and Multidisciplinary Optimization 36, 393--401(2008).