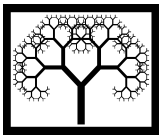
Paper 49



Proceedings of the Eighth International Conference
on Engineering Computational Technology,
B.H.V. Topping, (Editor),
Civil-Comp Press, Stirlingshire, Scotland

Differential Evolution Assisted by Surrogate Models for Structural Optimization Problems

E. Krempser[1], H.S. Bernardino[1], H.J.C. Barbosa[1,2] and A.C.C. Lemonge[2]
[1] National Laboratory for Scientific Computing -- LNCC / MCTI
Petropolis RJ, Brazil
[2] Federal University of Juiz de Fora - UFJF, MG, Brazil

Abstract

Differential evolution (DE) is a popular computational method used to solve optimization problems with several variants available in the literature. Here, the use of a similarity-based surrogate model is proposed in order to improve DE's overall performance in computationally expensive problems. The offspring are generated by means of different variants, and only the best one, according to the surrogate model, is evaluated by the simulator. The problem of weight minimization of truss structures is used to assess the performance of the proposed procedure. The surrogate assisted DE techniques presented here are compared to standard versions of DE using different variants. The experiments are composed by six different optimization problems involving five structures with continuous as well as discrete design variables.

Keywords: differential evolution, surrogate model, structural optimization, nearest neighbors, linear regression.

1 Introduction

Due to the increasing competitiveness in industry, as well as the identification of new scientific problems, significant effort has been invested in recent years to develop effective computational techniques to deal with those challenges. The complexity of current structural design, optimization, and identification problems provides the motivation for the design and analysis of novel optimization techniques.

Nature inspired metaheuristics can help overcome the challenges presented by multiple objectives, mixed types of design variables, low regularity of the objective functions, a large number of nonlinear implicit constraints, and expensive and/or unreliable gradients. However they often require a large number of fitness and constraint evaluations. Due to the size and complexity of current simulation models, large (sometimes

prohibitive) computational times are often required. As a result, the idea of incorporating surrogate/metamodeling techniques becomes attractive.

Among the many metaheuristics available, differential evolution (DE), inspired by Darwinian evolution, is a relatively new optimization technique which has generated interest in a number of researchers from different fields. In DE the population of candidate solutions moves in the search space by means of the addition of differences between other candidate vectors. Several variants of DE can be found in the literature. Although good solutions can be obtained, DE requires, similarly to other nature inspired techniques, many calls to the objective function evaluator. This becomes a serious drawback to their application in situations where expensive simulations are required. The user's computational budget then places a strong limit to the number of calls to the expensive simulation model, making it necessary to modify the search process in order to increase the convergence speed of the optimization procedure.

One way to alleviate that situation is to use a surrogate model (or metamodel), replacing the computationally intensive original simulator evaluation by a relatively inexpensive approximation of the objective function value and constraint checking. Here the use of local surrogate models in order to improve DE's overall performance in structural optimization problems is proposed.

In this paper, a similarity-based surrogate model (SBSM) is applied to DE so that, using a fixed number of expensive simulations, DE is allowed to perform additional (approximate) fitness function evaluations in order to (hopefully) obtain a final solution which is better than the one DE would find using only that fixed amount of simulations.

2 Structural Optimization Problems

The main types of structural optimization problems are usually referred to as sizing, configuration, or topology optimization. In sizing optimization problems, characteristics of the cross-section of the bars, such as the sectional areas of the members of a truss $\mathbf{a} = \{A_1, A_2, ..., A_n\}$ are used as design variables, (which can be continuous or discrete, when chosen from a list of commercially available sizes) and one is interested in finding $\mathbf{a}$ which minimizes the weight of the truss structure

$$w(\mathbf{a}) = \sum_{k=1}^{n} \gamma A_k \left(\sum_{j=1}^{N_k} L_j \right) \tag{1}$$

subject to the normalized stress (s) and displacements (u) constraints

$$\frac{|s_{j,l}|}{s_{adm}} - 1 \leq 0 \qquad \frac{|u_{i,l}|}{u_{adm}} - 1 \leq 0 \qquad 1 \leq j \leq N, \quad 1 \leq i \leq M, \quad 1 \leq l \leq N_L \tag{2}$$

where γ is the specific weight of the material, L_j is the length of j-th bar of the structure, u_i and s_j are respectively the nodal displacement of the i-th translational

degree of freedom and the stress of the j-th bar, s_{adm} is the allowable stress for the material, and u_{adm} is the maximum displacement for each nodal point. M is the number of translational degrees of freedom, N is the total number of bars in the truss structure, N_k is the number of members in the k-th group which share the same cross-sectional area, and N_L is the number of load cases applied to the structure.

Although the function w from Eq. (1) is linear, the $m = N_L \times (N + M)$ constraints in (2) are complex implicit functions of the design variables $\mathbf{a}$ and require the solution of the equilibrium equations of the discrete model given by $\mathbf{K(a)u}_l = \mathbf{f}_l$, $1 \leq l \leq N_L$. $\mathbf{K}$is the symmetric and positive definite stiffness matrix of the structure, derived in the finite element formulation by assembling each matrix contribution K_j of the j-th bar, which is a linear function of $\mathbf{a}$. The vector of nodal displacements is denoted by $\mathbf{u}_l$, and $\mathbf{f}_l$ is the vector of applied nodal forces for the l-th load condition. For each one of the load conditions, the system is solved for the displacement field $\mathbf{u}_l = [\mathbf{K}(\mathbf{a})]^{-1}\mathbf{f}_l$ and the stress in the j-th bar is calculated according to Hooke's Law as $s_{j,l} = E\varepsilon(\mathbf{u}_l)$, where E is the Young's modulus and ε is the unit change in length of the bar.

3 Adaptive Penalty Method

The application of evolutionary algorithms to constrained optimization problems requires a constraint handling technique which is a fundamental factor for a good performance of the algorithm. Penalty techniques are widely used in nature inspired metaheuristics. This is due to the fact that often the search using feasible as well as infeasible elements increases the chances of getting better results. Although conceptually simple, they require adequate values for (problem dependent) penalty parameters so that a good performance is attained. The adaptive penalty method (APM), proposed in [4], aims at relieving the user from the task of defining good values for the penalty coefficients by automatically setting those values using feedback from the search process. The idea is to observe how each constraint is being violated by the candidate solutions of a given population and then set a higher penalty coefficient to those constraints which seem to be harder to satisfy. Furthermore, the APM has been shown to be quite effective within Genetic Algorithms (GA) [11] and DE [16]. The quantities to be computed are the average value of the objective function of the elements of a given population, and the average violation of the j-th constraint in a given population.

The class of constrained optimization problems solved here can be written as

$$
\begin{aligned}
&\text{minimize } f(x) = w(x) \\
&\text{subject to:} \\
&\qquad g(x) \leq 0, \quad x_i^L \leq x_i \leq x_i^U, \quad \forall i = 1, \ldots, n \\
&\text{where:} \\
&\quad f : \Re^n \to \Re, \quad g : \Re^n \to \Re^m,
\end{aligned}
$$

where $w(\cdot)$ is defined by Equation 1 and $g(\cdot)$ is the vector of m constraints in Equa-

tion 2. Defining the amount of violation of the j-th constraint as

$$v_j(x) = max\{0, g_j(x)\},$$

then APM defines the fitness of a given candidate solution x as

$$F(x) = \begin{cases} f(x) & \text{if } x \text{ is feasible} \\ \overline{f}(x) + \sum_{j=1}^{m} k_j v_j(x) & \text{otherwise} \end{cases} \tag{3}$$

with

$$\overline{f}(x) = \begin{cases} f(x) & \text{if} \quad f(x) > \langle f(x) \rangle \\ \langle f(x) \rangle & \text{otherwise} \end{cases} \tag{4}$$

The penalty coefficient k_j corresponding to the $j-$th constraint is defined at every generation by

$$k_j = |\langle f(x) \rangle| \frac{\langle v_j(x) \rangle}{\sum_{l=1}^{m} [\langle v_l(x) \rangle]^2} \tag{5}$$

where $\langle f(x) \rangle$ is the average fitness in the current population and $\langle v_l(x) \rangle$ is the violation of the l-th constraint averaged over the current population.

4 Differential Evolution

Originally proposed by Storm and Price [17], DE is a simple and effective algorithm for global optimization in $\Re^n$. A pseudo-code is presented in Algorithm 1. The basic operation performed is the addition to each design variable in a given candidate solution of a term which is the scaled difference between the values of such variable in other candidate solutions in the population. The number of differences applied, the way in which the individuals are selected and the distribution of recombination determine the DE variant (also called DE strategy). The DE variants considered here, proposed in [13], modify the way that the individuals are selected to participate in the mutation as follows:

- **DE/best/1/bin:** Uses the best individual in the population $x_{best,j,G}$ as base vector in the mutation, leading to $u_{i,j,G+1} = x_{best,j,G} + F.(x_{r_1,j,G} - x_{r_2,j,G})$, where r_1 and r_2 are randomly selected individuals.

- **DE/target-to-best/1/bin:** This variant uses the best individual of the population and the target individual (the one that will be used in the comparison after the mutation, also called current individual), leading to $u_{i,j,G+1} = x_{i,j,G} + F.(x_{best,j,G} - x_{i,j,G}) + F.(x_{r_1,j,G} - x_{r_2,j,G})$

- **DE/target-to-rand/1/bin:** This one modifies the previous variant by using a randomly selected individual (r_3) instead of the best one $u_{i,j,G+1} = x_{i,j,G} + F.(x_{r_3,j,G} - x_{i,j,G}) + F.(x_{r_1,j,G} - x_{r_2,j,G})$

Algorithm 1: Algorithm DE/rand/1/bin.

input : NP (population size), GEN (# of generations), F (mutation scaling), CR (crossover rate)

```
1  G ← 0;
2  CreateRandomInitialPopulation(NP);
3  for i ← 1 to NP do
4  |  Evaluate f(x_{i,G}); /* x_{i,G} is an individual in the
   |  population */
5  for G ← 1 to GEN do
6  |  for i ← 1 to NP do
7  |  |  SelectRandomly(r1, r2, r3) ; /* r1 ≠ r2 ≠ r3 ≠ i */
8  |  |  jRand ←RandInt(1, N) ; /* N is the number of
   |  |  variables */
9  |  |  for j ← 1 to N do
10 |  |  |  if Rand(0, 1) < CR or j = jRand then
11 |  |  |  |  u_{i,j,G+1} = x_{r3,j,G} + F.(x_{r1,j,G} − x_{r2,j,G});
12 |  |  |  else
13 |  |  |  |  u_{i,j,G+1} = x_{i,j,G};
14 |  |  if f(u_{i,G+1}) ≤ f(x_{i,G}) then
15 |  |  |  x_{i,G+1} = u_{i,G+1};
16 |  |  else
17 |  |  |  x_{i,G+1} = x_{i,G};
```

5 Surrogate-assisted Differential Evolution

Replacing the original evaluation function (a complex computer simulation) by a substantially less expensive approximation is known as surrogate modeling, or metamodeling. Although this idea appeared early in the evolutionary computation literature [7], there are not many works combining DE with surrogate models. Wang et al. [18] use an artificial neural network (ANN) as a surrogate model, which is trained by the initial population, exactly evaluated. The ANN is then used to evaluate the individuals in the following generations where only the best individual is exactly evaluated and stored. After a fixed number of generations the population is evaluated by the original objective function and the ANN is retrained. Pahner & Hameyer [12], had previously proposed a similar process using radial basis function neural networks (RBFN). Zhang & Sanderson [19] modify jDE [14] by generating multiple offspring for each parent and choosing the best one in a special tournament which takes into account a measure of the RBFN surrogate model accuracy.

In contrast to "eager" learning algorithms, such as neural networks and polynomial response surfaces, which generate a model and then discard the inputs, the Similarity-

Based Surrogate Models store their inputs and defer processing until a prediction of the fitness value of a new candidate solution is requested. Thus, SBSMs can be classified as "lazy" learners or memory-based learners [2].

The Nearest Neighbors technique, a type of SBSM, was previously explored in [5]. Here, two variants of this method are studied: (i) k-Nearest Neighbors (k-NN) [15], in which the k nearest candidate solutions are selected; and (ii) r-Nearest Neighbors (r-NN) [10], where all points in a given neighborhood are used.

Given a candidate solution x and the archive $\mathcal{D} = \{(x_i, f(x_i)), i = 1, \ldots, \eta\}$ containing the solutions already exactly evaluated , the following approximation is considered:

$$f(x) \approx \widehat{f}(x) = \frac{\sum_{j=1}^{|\mathcal{N}|} s(x, x_j^{\mathcal{N}})^p f(x_j^{\mathcal{N}})}{\sum_{j=1}^{|\mathcal{N}|} s(x, x_j^{\mathcal{N}})^p}$$

where the $x_j^{\mathcal{N}} \in \mathcal{N}$ are the nearest neighbors of x, $s(x, x_j^{\mathcal{N}})$ is a similarity measure between x and $x_j^{\mathcal{N}}$, and p is set to 2. Here, $s(x, x_j^{\mathcal{N}}) = 1 - \left(d_E(x, x_j^{\mathcal{N}})\right) / \left(d_E(x^u, x^l)\right)$, where $d_E(x, x_j^{\mathcal{N}})$ is the Euclidean distance between x and $x_j^{\mathcal{N}}$, and x^u and x^l are the upper and lower bounds of the search space, respectively. If $x = x_i$ for some $x_i \in \mathcal{D}$ then $\widehat{f}(x) = f(x_i)$. When one uses the k-NN technique, $\mathcal{N}$ is composed by the k elements in the set $\mathcal{D}$ most similar to x. On the other hand, when the r-NN technique is used, $\mathcal{N}$ contains the elements from the set $\mathcal{D}$ which belong to the hyperbox centered in x such that its i-th "side" has length $2r\left(x_i^u - x_i^l\right)$.

For both querying techniques considered here, only one parameter is needed to be set by the user: the number of nearest neighbors k, or the length of the "size" of the hyperbox. When there are no candidate solutions inside the hyperbox, the two nearest neighbors are used by the surrogate function.

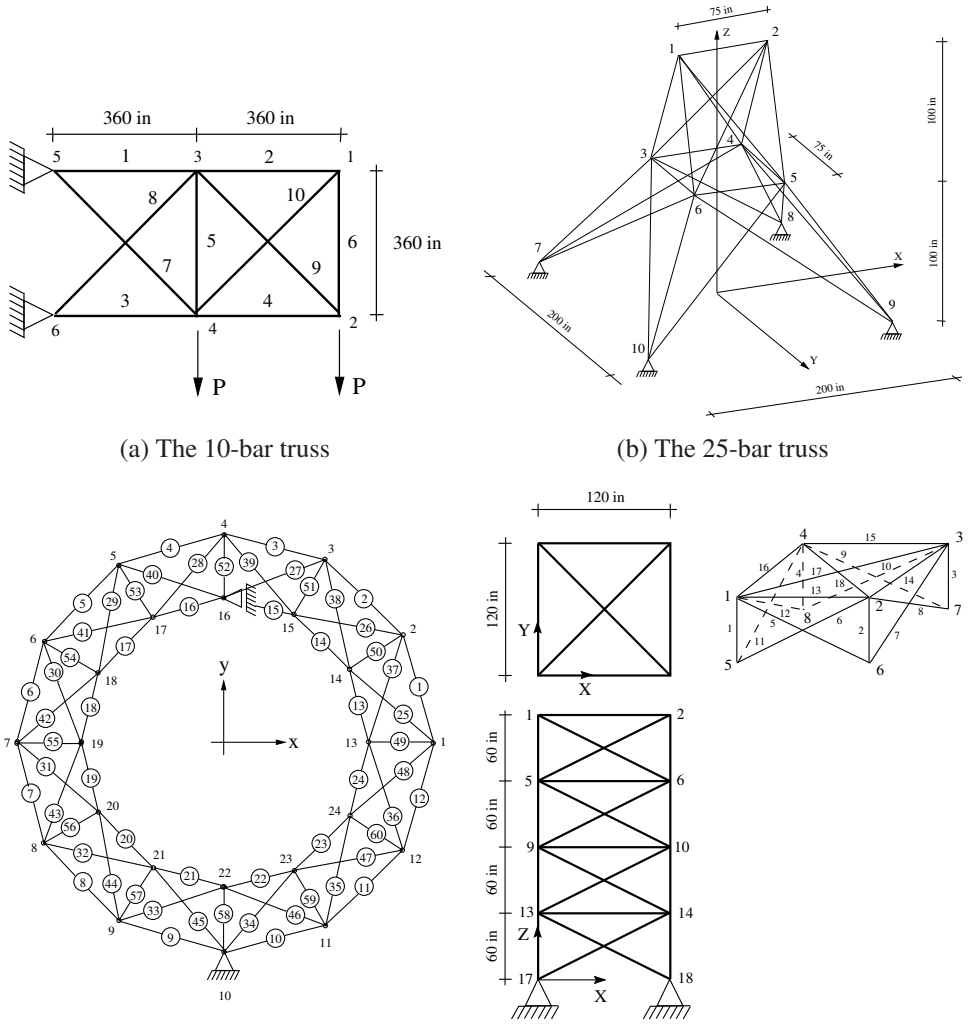
All candidate solutions from the initial population are evaluated exactly and compose the archive $\mathcal{D}$ which defines the surrogate model. All new candidate solutions are evaluated by the surrogate model and the best one is evaluated by the exact function. If the exact evaluation of the offspring is better than that of the parent, the parent is replaced. It is important to note that every individual exactly evaluated is stored in the archive $\mathcal{D}$ used by the surrogate model.

Finally, all surrogate models used in the works mentioned above are based on neural networks, which are much more computationally expensive than the SBSM method proposed here.

6 Computational Experiments

Five structures from the literature are considered here, namely the 10-, 25-, 60-, 72-, and 942-bar trusses, to assess the performance of the proposed algorithm. The Figures 1 and 2 display a schematic view of the structures considered.

All DE variants, as presented in Algorithm 1 from Section 4, use the same parameter values as follows: F = 0.8, CR = 0.9 and NP = 50. The standard DE variants



(a) The 10-bar truss

(b) The 25-bar truss

(c) The 60-bar truss

(d) The 72-bar truss

Figure 1: Images of the bar truss structures used in the computational experiments.

DE/rand/1/bin, DE/best/1/bin, DE/target-to-best/1/bin and DE/target-to-rand/1/bin are labeled as "Rand", "Best", "Target-to-best" and "Target-to-rand" respectively. The DE assisted by a surrogate model are labeled as "SMDE" followed by a parameter of its similarity model. For all cases, 12,000 objective function evaluations were used, and 100 independent runs were performed.

In the following, a brief description of the five structural design optimization problems considered is presented.

6.1 Test-Problems

The 10-bar Truss Design

This test problem corresponds to the weight minimization of the 10-bar truss shown in the Figure 1a. The stress in each member is limited to $\pm$ 25ksi, and the displacements

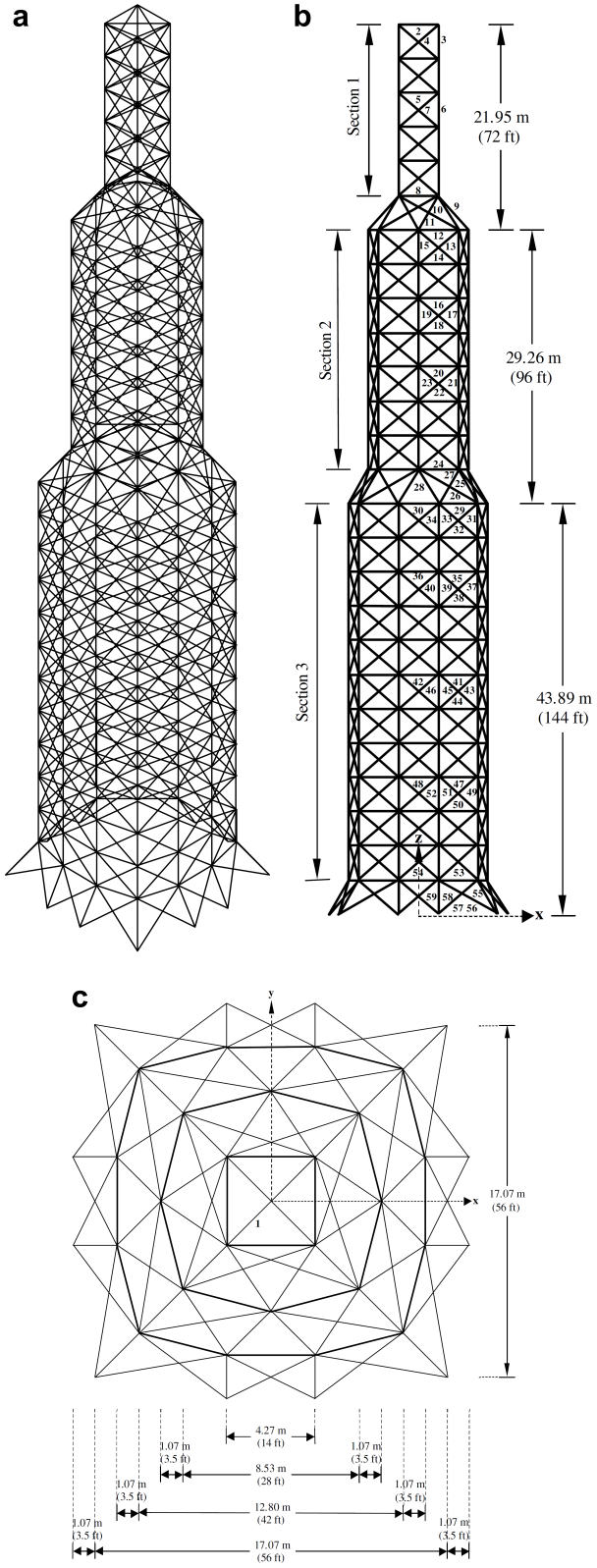


Figure 2: Images of the 942-bar truss structures used in the computational experiments (adapted from [9]).

node	F_x	F_y	F_z
1	1	-10.0	-10.0
2	0	-10.0	-10.0
3	0.5	0	0
6	0.6	0	0

group	connectivities
A_1	1-2
A_2	1-4, 2-3, 1-5, 2-6
A_3	2-5, 2-4, 1-3, 1-6
A_4	3-6, 4-5
A_5	3-4, 5-6
A_6	3-10, 6-7, 4-9, 5-8
A_7	3-8, 4-7, 6-9, 5-10
A_8	3-7, 4-8, 5-9, 6-10

Table 1: Loading data (kips) and member grouping for the 25-bar truss.

at the nodes are limited to 2 in., in the x and y directions. The design variables are the cross-sectional areas of the bars. The material has $\gamma = 0.1$ lb/in^3, and $E = 10^4$ ksi. Vertical downward loads of 100 kips are applied at nodes 2 and 4. Two cases have been considered: discrete and continuous design variables. For the discrete case the values of the cross-sectional areas (in^2) are chosen from the set $\mathcal{S}$: 1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97, 11.50, 13.50, 13.90, 14.20, 15.50, 16.00, 16.90, 18.80, 19.90, 22.00, 22.90, 26.50, 30.00, 33.50. For the continuous case the cross sectional areas are in the range $[0.1, 40]$ in^2.

The 25-bar Truss Design

In this test-problem, the weight of a truss with 25 bars, shown in the Figure 1b, is to be minimized. The constraints require that the maximum stresses in the members remain in the interval $[-40, 40]$ ksi and that the maximum displacements at the nodes 1 and 2 be limited to 0.35 in, in both the x and y directions. The design variables are the cross-sectional areas of the bars to be chosen from the set (in square inches): 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2, and 3.4. The design variables are linked in eight groups detailed in the Table 1. The material has $\gamma = 0.1$ lb/in^3 and $E = 10^4$ ksi. The loading data is listed in the Table 1.

The 60-bar Trussed Ring

This test-problem corresponds to the weight minimization of a 60-bar trussed ring depicted in the Figure 1c (not to scale) under three load cases as given in Table 2. Also in Table 2 is the grouping of the cross-sectional areas in 25 design variables. The outer radius of the ring is 100 in and the inner radius is 90 in. The material has $E = 10^4$ ksi and $\gamma = 0.1$ lb/in^3. There are 198 constraints where 180 refer to allowable stress (σ_i = 60 ksi, i = 1 to 60), and 18 refer to displacement constraints along both the x and y directions with magnitude: 1.75 in at node 4, 2.25 in at node 13, and 2.75 in at node 19. For the continuous case the cross-sectional areas of the bars are assumed to be in the range $[0.5, 5]$ in^2.

node	F_x	F_y
load case 1		
1	-10.0	0
7	9.0	0
load case 2		
15	-8.0	3.0
18	-8.0	3.0
load case 3		
22	-20.0	10.0

Group	bars	Group	bars
A_1	49 to 60	A_{14}	25, 37
A_2	1, 13	A_{15}	26, 38
A_3	2, 14	A_{16}	27, 39
A_4	3, 15	A_{17}	28, 40
A_5	4, 16	A_{18}	29, 41
A_6	5, 17	A_{19}	30, 42
A_7	6, 18	A_{20}	31, 43
A_8	7, 19	A_{21}	32, 44
A_9	8, 20	A_{22}	33, 45
A_{10}	9, 21	A_{23}	34, 46
A_{11}	10, 22	A_{24}	35, 47
A_{12}	11, 23	A_{25}	36, 48
A_{13}	12, 24		

Table 2: Loading data (kips) and member grouping for the ring structure.

The 72-bar Truss

The Figure 1d presents the 72-bar truss structure where the design variables are the cross-sectional areas of the bars, the minimum value for each one being 0.1 in^2. The 72 design variables are linked in sixteen groups detailed in the Table 3. The material has $\gamma = 0.1$ lb/in^3 and $E = 10^4$ ksi. Two load cases (see Table 3) are considered. Displacements at the nodes 1 to 16 along the x and y directions are constrained to a maximum of 0.25 in, and the stress in each bar is restricted to the range $[-25, 25]$ ksi.

The 942-bar Truss Design

In this truss tower [8], the symmetry of the tower around x and y-axes is employed to group the 942 truss members into 59 independent size variables. The tower is subject to a single loading condition consisting of both horizontal and vertical loads, as follows: (i) the vertical loads in the z direction are -3.0 kips (-13.344 kN), -6.0 kips (-26.688 kN), and -9.0 kips (-40.032 kN) at each node in the first, second and third sections, respectively, (ii) the lateral loads in the y direction are 1.0 kips (4.448 kN) at all nodes of the tower, and (iii) the lateral loads in the x direction are 1.5 kips (6.672 kN) and 1.0 kips (4.448 kN) at each node on the left and right sides of the tower, respectively.

The cross-sectional areas available are taken as continuous values in the range $[1, 200]$ in^2. The constraints for this problem include a maximum stress of 25.0 ksi (170 MPa) both in tension and compression for all members, and a limit of 15 in for the displacements of the top nodes in any global direction. The modulus of elasticity is E = 10,000 ksi and the density is ρ = 0.1 lb/in^3 (27.15 kN/m^3). This problem has been previously studied in [1, 9, 8], where more detailed information can be found.

node	F_x	F_y	F_z
load case 1			
1	5	5	-5
load case 2			
1	0	0	-5
2	0	0	-5
3	0	0	-5
4	0	0	-5

group	members
A_1	1, 2, 3, 4
A_2	5, 6, 7, 8, 9, 10, 11, 12
A_3	13, 14, 15, 16
A_4	17,18
A_5	19, 20, 21, 22
A_6	23, 24, 25, 26, 27, 28, 29, 30
A_7	31, 32, 33, 34
A_8	35,36
A_9	37, 38, 39, 40
A_{10}	41, 42, 43, 44, 45, 46, 47, 48
A_{11}	49, 50, 51, 52
A_{12}	53,54
A_{13}	55, 56, 57, 58
A_{14}	59, 60, 61, 62, 63, 64, 65, 66
A_{15}	67, 68, 69, 70
A_{16}	71,72

Table 3: Loading data (kips) and member grouping for the 72-bar truss.

6.2 Performance Profiles

An overall comparison of the algorithms is given by means of performance profiles [6, 3]. The performance indicator (to be maximized) $t_{p,a}$ of algorithm $a \in A$ when applied to test-problem $p \in P$ is the inverse of the minimum objective function value found by algorithm a in test-problem p averaged over 100 runs. A performance ratio can be defined as $r_{p,a} = t_{p,a}/\min\{t_{p,a} : a \in A\}$. Denoting cardinality of a set by $|.|$, performance profiles, are defined as [6]

$$\rho_a(\tau) = \frac{1}{n_p} \left| \{p \in P : r_{p,a} \leq \tau\} \right|.$$

Also, the area under the curves (AUC) of the performance profiles were used to describe the results.

Although performance profiles allow the direct comparison of all algorithms against all problems, we have separated the algorithms in 4 groups considering the surrogate model used, that is, baseline, SMDEs with k-NN, SMDEs with r-NN, and SMDE with LR. The algorithm with best AUC in each group is selected to the final comparison. These preliminary comparisons followed by a final analysis with the best performing techniques make possible not only to choose the best overall technique but also the best algorithm with respect to each group. The following sections discuss the results obtained in our experiments.

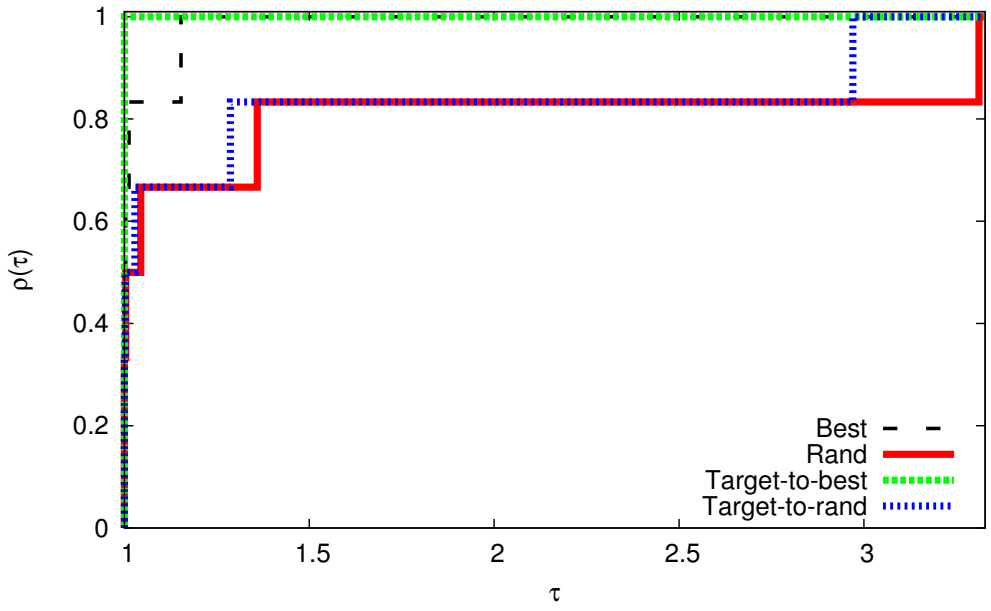


Figure 3: Performance profiles for baseline algorithms.

6.3 Preliminary Comparisons

Baseline DEs
The results are presented in figures 3 and 4 where a performance profile and its corresponding integral (area under the curve) show that the Target-to-best variant had the best performance.

SMDE with k-NN
The application of the k-Nearest Neighbors (k-NN) as a surrogate model in DE was evaluated with different values for the parameter k. The values 2, 4, 8, and 16 were considered and the results are presented in the figures 5, and 6, where one can observe that the best result was obtained with $k = 2$.

SMDE with r-NN
Now the r-Nearest Neighbors (r-NN) surrogate model is applied with DE using different values for the r parameter, namely 0.1, 0.01, and 0.001. The results are presented in the figures 7 and 8, and the best performance was obtained with $r = 0.001$.

6.4 Final Analysis

The last comparison was performed among the best baseline algorithm and the SMDE with the best parameters k and r following the results presented in the previous sec-

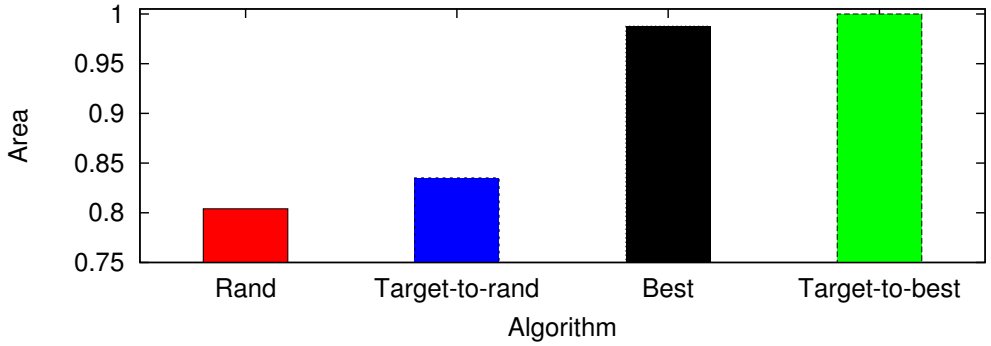


Figure 4: Area under the curve of the performance profiles of Figure 3.

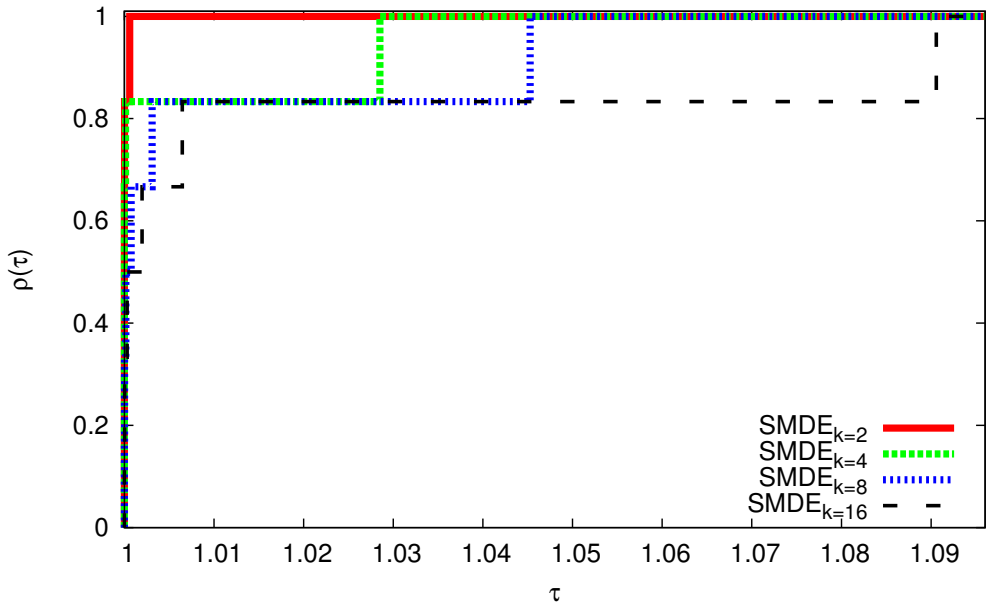


Figure 5: Performance profiles for SMDE with the k-NN.

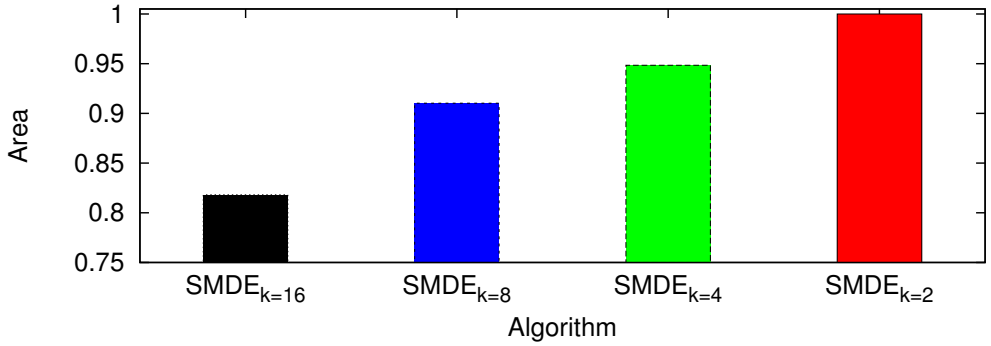


Figure 6: Area under the curve of the performance profiles of Figure 5.

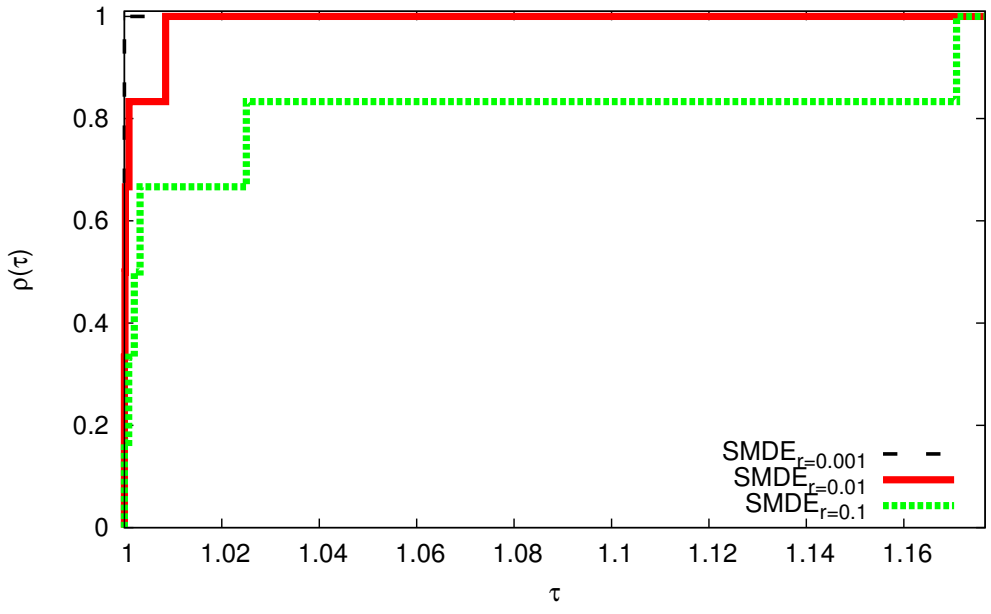


Figure 7: Performance profiles for SMDE with the r-NN.

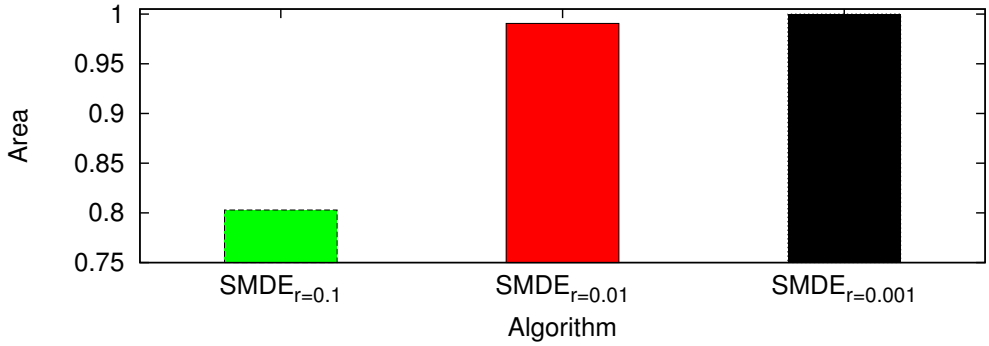


Figure 8: Area under the curve of the performance profiles of Figure 7.

tions. Another experiment evaluated was the SMDE with a liner regressor as a surrogate model. The results are presented in the figures 9 and 10. Also the results of the best performing techniques analyzed here can be found in Table 4 which displays the best, median, average, standard deviation (std), worst, and the number of exact objective function evaluations allowed (NEvals). Furthermore, the values obtained in [16] using DE with dynamic choice of variants (DUVDE), a genetic algorithm with APM [4, 11], simulated annealing (SA) [9], and an evolution strategy (ES) [8] were included. The best results with respect to each case are in boldface, the best performing algorithms are highlighted with a gray background, and an * indicates that the difference observed is not statistically significant with respect to the results found by the best performing technique. Algorithms with best median and average values are preferable. A pair of sets of results are statistically significantly different when the p-value from the non-parametric Kruskal-Wallis test [1] is less than 0.05.

The results presented in the figures indicate that SMDE obtained the best behavior when compared with the baseline algorithm. This result shows that the proposed algorithm is able to alleviate the user from the task of variant definition. In addition, it was also able to improve the original DE without increasing the computational cost.

[1]The sample was the solution obtained in each independent run and the open-source SciPy software (`http://www.scipy.org`) was used in the statistical tests.

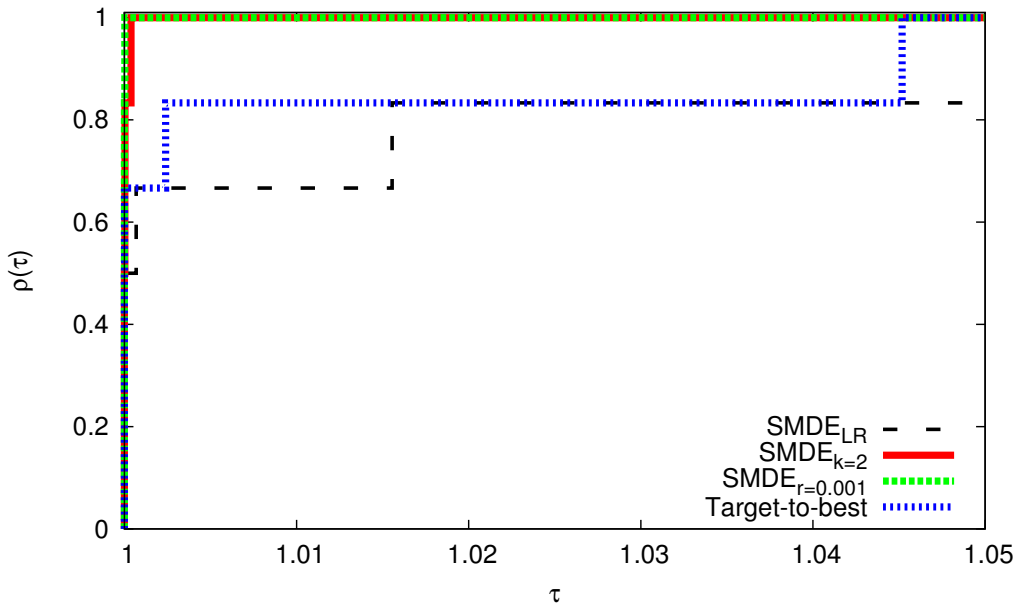


Figure 9: Performance profiles for best algorithms.

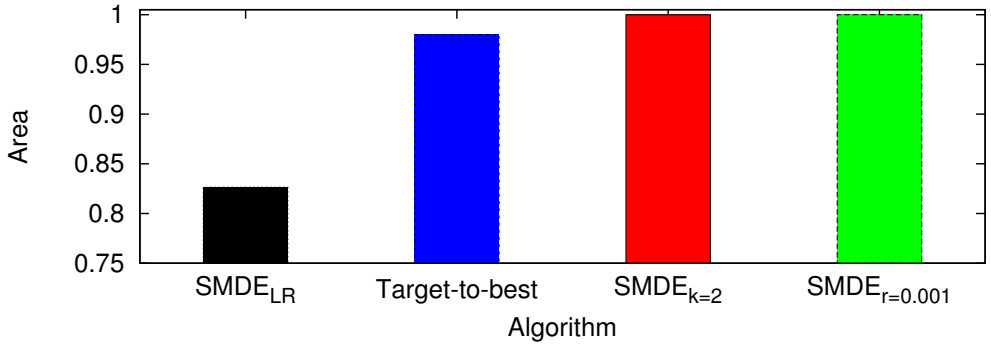


Figure 10: Area under the curve of the performance profiles of Figure 9.

	Best	Median	Average	std	Worst	NEvals
10-bar – Discrete case						
Target-to-best	**5490.74**	**5490.74**	5575.58	$2.77e+02$	6969.66	12000
SMDE k=2	**5490.74**	**5490.74**	5582.45	$2.75e+02$	6872.99	12000
SMDE r=0.001	**5490.74**	**5490.74**	5582.48	$2.75e+02$	6872.99	12000
SMDE LR	**5490.74**	**5490.74**	**5492.63**	$7.88e+00$	**5538.09**	12000
DUVDE[16]	5562.35	–	5564.90	$6.00e-01$	5565.04	24000
APM[11]	5490.74	–	5545.48	–	5567.84	90000
10-bar – Continuous case						
Target-to-best	**5060.85**	**5060.86**	5062.92	$5.31e+00$	5076.67	12000
SMDE k=2	**5060.85**	5061.00	5064.01	$6.15e+00$	5077.05	12000
SMDE r=0.001	5060.86	5061.00	5063.85	$6.03e+00$	5077.00	12000
SMDE LR	5060.87	5060.98	**5062.25**	$4.26e+00$	5076.69	12000
DUVDE[16]	**5060.85**	–	5067.18	$7.94e+00$	**5076.66**	280000
APM[11]	5069.08	–	5091.43	–	5117.39	280000
25-bar						
Target-to-best*	**484.85**	**484.85**	485.19	$2.51e+00$	509.60	12000
SMDE k=2	**484.85**	**484.85**	485.71	$3.21e+00$	515.39	12000
SMDE r=0.001	**484.85**	**484.85**	485.71	$3.21e+00$	515.39	12000
SMDE LR	**484.85**	**484.85**	**484.88**	$1.13e-01$	**485.91**	12000
DUVDE[16]	485.90	–	498.44	$7.66e+00$	507.77	20000
APM[11]	485.85	–	485.97	–	490.74	20000
60-bar						
Target-to-best	309.64	311.90	312.08	$1.31e+00$	317.90	12000
SMDE k=2*	309.27	311.27	312.45	$4.71e+00$	347.97	12000
SMDE r=0.001	**309.14**	**311.15**	312.43	$4.79e+00$	348.17	12000
SMDE LR	312.80	315.99	316.39	$2.48e+00$	329.64	12000
DUVDE[16]	309.44	–	**311.54**	$1.46e+00$	**314.70**	150000
APM[4]	311.87	–	333.01	–	384.19	800000
72-bar						
Target-to-best*	379.63	379.68	**379.70**	$6.15e-02$	**379.94**	12000
SMDE k=2*	**379.62**	379.69	379.79	$4.26e-01$	383.32	12000
SMDE r=0.001	379.63	**379.68**	379.77	$4.63e-01$	383.65	12000
SMDE LR	379.73	379.94	380.42	$4.17e+00$	421.81	12000
DUVDE[16]	379.66	–	380.42	$5.72e-01$	381.37	35000
APM[11]	387.04	–	402.59	–	432.95	35000
942-bar – Discrete case using integer values						
Target-to-best	158128.48	209841.59	217807.88	$3.35e+04$	363176.31	12000
SMDE k=2*	156559.70	**200771.49**	207290.41	$3.18e+04$	**339646.14**	12000
SMDE r=0.001	156559.70	**200771.49**	**207285.69**	$3.18e+04$	**339646.14**	12000
SMDE LR	206075.45	279905.43	280842.06	$3.12e+04$	350616.86	12000
SA [9]	143436.02	–	–	–	–	39834
ES [8]	**141241.00**	–	–	–	–	150000

Table 4: Weights found for all problems. The best results for each case are in boldface, the best performing algorithms are highlighted with a gray background, and an * indicates that the difference to the best performing technique is not statistically significant.

7 Concluding Remarks

In this paper, differential evolution is enhanced by means of similarity-based surrogate models, namely nearest neighbor techniques and local linear regression, in order to improve DE's performance when solving optimization problems involving computationally expensive objective functions and/or constraints evaluations.

Computational experiments were performed to assess the performance of the proposed procedure using structural problems from the literature where a maximum number of exact objective function evaluations is prescribed. The results show that the use of a similarity-based surrogate model improves the performance of DE for most test-problems, specially when using r-nearest neighbors with $r = 0.001$. The use of a simple local linear regressor produced relatively lower quality results in most problems, although producing the best results in 2 of the 6 test-problems.

It is important to notice that the results from the literature which presented a better (smaller) final weight were obtained with a much larger (often one-order magnitude) number of simulations.

In addition, the proposed technique alleviates the user from the task of defining a priori which type of variant to use in the DE.

As a future work, ways to improve the accuracy of the meta-model will be studied, and other test-problems and benchmarks will be considered.

Acknowledgments

The authors acknowledge the support from CNPq (grants 308317/2009-2, 140785/2009-4, 306815/2011-7, and 300192/2012-6), FAPERJ (E-26/100.308/2010), and FAPEMIG (PPM 528-11).

References

[1] H. Adeli, N.T. Cheng, "Concurrent Genetic Algorithms for Optimization of Large Structures", *Journal of Aerospace Engineering*, 7(3): 276–296, 1994.

[2] D.W. Aha, "Editorial", *Artif. Intell. Rev.*, 11(1-5): 1–6, 1997, ISSN 0269-2821, Special issue on lazy learning.

[3] H.J.C. Barbosa, H.S. Bernardino, A.M.S. Barreto, "Using Performance Profiles to Analyze the Results of the 2006 CEC Constrained Optimization Competition", in *Proc. of the World Congress on Computational Intelligence – WCCI / CEC*. IEEE, Barcelona, Spain, July 2010.

[4] H.J.C. Barbosa, A.C.C. Lemonge, "An New Adaptive Penalty Scheme for Genetic Algorithms", *Information Sciences*, 156: 215–251, 2003.

[5] H.S. Bernardino, H.J.C. Barbosa, L.G. Fonseca, "Surrogate-assisted clonal selection algorithms for expensive optimization problems", *Evolutionary Intelligence*, 4: 81–97, 2011.

[6] E. Dolan, J.J. Moré, "Benchmarcking optimization software with performance profiles", *Math. Programming*, 91(2): 201–213, 2002.

[7] J. Grefenstette, J. Fitzpatrick, "Genetic search with approximate fitness evaluations", in *Proc. of the Intl. Conf. on Genetic Algorithms and Their Applications*, pages 112–120. Lawrence Erlbaum, 1985.

[8] O. Hasançebi, "Adaptive evolution strategies in structural optimization: Enhancing their computational performance with applications to large-scale structures", *Computers & Structures*, 86(1-2): 119 – 132, 2008, ISSN 0045-7949.

[9] O. Hasançebi, F. Erbatur, "On efficient use of simulated annealing in complex structural optimization problems", *Acta Mechanica*, 157: 27–50, 2002, ISSN 0001-5970.

[10] H. Hu, D.L. Lee, "Range nearest-neighbor query", *IEEE Transactions on Knowledge and Data Engineering*, 18(1): 78 – 91, 2006.

[11] A.C.C. Lemonge, H.J.C. Barbosa, "An Adaptive Penalty Scheme for Genetic Algorithms in Structural Optimization", *International Journal for Numerical Methods in Engineering*, 59: 703–736, 2004.

[12] U. Pahner, K. Hameyer, "Adaptive coupling of differential evolution and multiquadrics approximation for the tunning of the optimization process", *IEEE Transactions on Magnetics*, 36(4): 347–367, 2000.

[13] K.V. Price, "An Introduction to Differential Evolution", *New Ideas in Optimization*, pages 79–108, 1999.

[14] A.K. Qin, P.N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization", in *Proc. of the Congress on Evolutionary Computation – CEC*, Volume 2, pages 1785 – 1791. IEEE, 2005.

[15] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data", in *Proc. of the 1968 23rd ACM National Conference*, pages 517–524. ACM Press, New York, NY, USA, 1968.

[16] E.K. Silva, H.J.C. Barbosa, A.C.C. Lemonge, "An adaptive constraint handling technique for differential evolution with dynamic use of variants in engineering optimization", *Optimization and Engineering*, 12: 31–54, 2011.

[17] R. Storn, K.V. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", *Journal of Global Optimization*, 11: 341–359, 1997.

[18] Y. Wang, Y. Shi, B. Yue, H. Teng, "An efficient differential evolution algorithm with approximate fitness functions using neural networks", in *Proceedings of the 2010 international conference on Artificial intelligence and computational intelligence: Part II*, AICI'10, pages 334–341. Springer-Verlag, Berlin, Heidelberg, 2010.

[19] J. Zhang, A.C. Sanderson, "DE-AEC: A differential evolution algorithm based on adaptive evolution control", in *Proc. of the Congress on Evolutionary Computation – CEC*, pages 3824 – 3830. IEEE, 2007.