# An enhanced surrogate-assisted differential evolution for constrained optimization problems

**Rafael de Paula Garcia**
Federal University of Vicosa: Universidade Federal de Vicosa
**Beatriz Souza Leite Pires de Lima**
Federal University of Rio de Janeiro: Universidade Federal do Rio de Janeiro
**Afonso Celso de Castro Lemonge** ( afonso.lemonge@ufjf.edu.br )
Federal University of Juiz de Fora: Universidade Federal de Juiz de Fora https://orcid.org/0000-0001-9938-294X
**Breno Pinheiro Jacob**
Federal University of Rio de Janeiro

**Research Article**

**Keywords:** Constrained optimization problems, Evolutionary algorithms, Surrogate models, Constraint-handling techniques

**Posted Date:** June 4th, 2021

**DOI:** https://doi.org/10.21203/rs.3.rs-522951/v1

# An enhanced surrogate-assisted differential evolution for constrained optimization problems

**Rafael de Paula Garcia**[a,1], *rafael.pgarcia@ufv.br*
**Beatriz Souza Leite Pires de Lima**[a], *bia@coc.ufrj.br*
**Afonso Celso de Castro Lemonge**[b,*], *afonso.lemonge@ufjf.edu.br*
**Breno Pinheiro Jacob**[a], *breno@lamcso.coppe.ufrj.br*

[a]*PEC/COPPE/UFRJ – Post-Graduate Institute of the Federal University of Rio de Janeiro, Civil Engineering Dept.*
*Avenida Pedro Calmon, S/N, Cidade Universitária, Ilha do Fundão*
*21941-596 Rio de Janeiro, RJ, Brazil.*
*http://www.lamcso.coppe.ufrj.br*

[b]*UFJF – Federal University of Juiz de Fora, Applied and Computational Mechanics Dept.*
*Rua José Lourenço Kelmer, S/N*
*36036-330 Juiz de Fora, MG, Brazil*
*http://www.ufjf.br/mac/*

**Corresponding Author.*

[1]*Present address: UFV – Federal University of Viçosa, Department of Architecture and Urban Planning.*
*Avenida Peter Henry Rolfs, s/n, Campus Universitário*
*36570-900, Viçosa, MG, Brazil*

## Abstract

The application of Evolutionary Algorithms (EAs) to complex engineering optimization problems may present difficulties as they require many evaluations of the objective functions by computationally expensive simulation procedures. To deal with this issue, surrogate models have been employed to replace those expensive simulations. In this work, a surrogate-assisted evolutionary optimization procedure is proposed. The procedure combines the Differential Evolution method with a $k$-nearest neighbors ($k$–NN) similarity-based surrogate model. In this approach, the database that stores the solutions evaluated by the exact model, which are used to approximate new solutions, is managed according to a merit scheme. Constraints are handled by a rank-based technique that builds multiple separate queues based on the values of the objective function and the violation of each constraint. Also, to avoid premature convergence of the method, a strategy that triggers a random reinitialization of the population is considered. The performance of the proposed method is assessed by numerical experiments using 24 constrained benchmark functions and 5 mechanical engineering problems. The results show that the method achieves optimal solutions with a remarkably reduction in the number of function evaluations compared to the literature.

**Keywords**: *Constrained optimization problems; Evolutionary algorithms; Surrogate models; Constraint-handling techniques*

# 1 Introduction

EAs have become appealing to solve complex real-world engineering optimization problems, for which classical mathematical methods may no longer be effective. This includes, for instance, the design of offshore oil & gas production systems such as floating platforms, risers, mooring systems and submarine pipelines [1-3]: those problems are characterized by many design variables, defined on high-dimensional spaces, with objectives and constraints that are generally non-linear functions. However, the use of EAs for such problems can become impractical due to excessive computational costs: many evaluations of candidate solutions may be needed until a satisfactory solution is found, and each evaluation requires computationally expensive numerical simulations.

Those issues can be handled by building cheaper approximation models (usually referred as *surrogate* or *metamodels*) [4] to partially or totally replace the numerical simulation procedures, emulating or mimicking the behavior of the actual model as closely as possible by capturing the relationship between a given set of inputs and outputs. The goal is to significantly reduce computational costs, while presenting accurate results. In this sense, several surrogate models have been proposed. They may comprise polynomial approximation models [5,6]; radial basis functions [7-10]; response surface models [11]; artificial neural networks [12]; Gaussian process [13,14]; Kriging [15]; Support Vector Machines [16], and similarity-based models such as the nearest-neighbor algorithms [17-19]. Particularly, the ***k*-NN** model presents many advantages in terms of good performance and ease of implementation, as reported in [19-21].

Many applications of surrogate models with EAs have been presented in the literature. In [22] a polynomial model was used to approximate the objective function. In [23] a kriging metamodel, trained using few samples evaluated by the actual objective function, has been successfully used to reduce the number of function evaluations. The goal of reducing computational costs was also sought in [24] where the fitness function has been approximated by an online multi-layer neural network, and in [25] that presented a surrogate-assisted PSO algorithm. Surrogate models have also been employed to initialize the population of candidate solutions in evolutionary operators such as mutation and crossover [26,27], and to calculate approximations for the sum of violation terms [21]. The IEEE Congress on Evolutionary Computation (CEC) competitions have presented many articles that solve computationally expensive functions by employing metamodels with EAs. Regarding particularly the application to offshore oil & gas production systems mentioned above, surrogate models based on neural networks were used in [28,29] to avoid expensive nonlinear dynamic Finite Element analyses.

Among evolutionary algorithms, the Differential Evolution (DE) algorithm [30] has been widely used due to its outstanding performance. The association of DE with approximation models has been studied in some works, including neural networks [31] and neural networks of radial basis functions [32]. In [33] a simple kriging metamodel was proposed to comprise a surrogate model-based differential evolution (ESMDE). The proposed strategy was based on the population members of the current generation, to assist the DE to generate competitive offspring by using an appropriate parameter set along the different stages of the evolution.

A surrogate model for DE consisting of multiple local surrogate models was presented in [34], built from the feedback of the evolutionary process in diverse overlapped local regions of the search space. In [35] a multi-fidelity surrogate-model-based DE optimization strategy was developed, coupled to a data mining technique to handle the discrepancy between the low and high fidelity simulation models. A surrogate-model to assist the DE algorithm in generating competitive solutions along the evolutionary process was proposed in [36], using an adaptation scheme to adapt parameters of a Kriging model. Recently, a two-layer adaptive surrogate-assisted DE has been proposed in [37], where three different search strategies are adapted during the evolutionary process according to the feedback information, to measure the status of the algorithm approaching the optimal value. Local similarity-based surrogate models based on an r-nearest neighbors have been employed in [17] and [18] to solve complex structural optimization problems; the results indicated that the performance of DE has been significantly improved. In this context, in [38] we had presented preliminary studies leading to the optimization procedure called **SADE-kNN** (Surrogate-Assisted Differential Evolution with the *k*-NN surrogate model).

Now, this work presents and evaluates an improved surrogate-assisted evolutionary method, which will be referred to as **SADE-kNN-MCR**, intended to comprise a more efficient optimization tool to solve complex real-world engineering problems. This method introduces enhanced techniques, beginning with the Multiple Constraint Ranking (MCR) constraint-handling technique (CHT) [39]. The MCR comprises a rank-based approach that builds multiple separate queues, corresponding to the values of the objective function and the violation of each constraint. It has been devised to deal with complex engineering problems characterized by a large number of constraints that may present different units and/or different orders of magnitude, where the possible range of minimum and maximum violation values may not be known a-priori (thus standard normalization procedures cannot be applied).

The proposed **SADE-kNN-MCR** method also incorporates a merit-based scheme to manage the database used by the ***k*-NN** surrogate model to store the solutions evaluated by the exact model, privileging solutions able to better approximate new candidate solutions. Moreover, results of preliminary studies indicated that the previous **SADE-*k*NN** method presented by the authors in [38] is susceptible to premature convergence and/or stagnation around local optima. This is a known issue related to DE algorithms, as acknowledged in the literature [40,41]. To circumvent this issue, a strategy that triggers a random reinitialization of the population whenever it stagnates on local optima has been incorporated in the proposed method, comprising the variant called **SADE-kNN-MCR+**.

The performance of the proposed method is compared with other surrogate-assisted procedures recently presented in the literature, by applying them to several benchmark functions (including the G-suite from the IEEE competition on real parameter constrained optimization [42], and mechanical engineering problems), and assessing their relative performance by direct comparisons and nonparametric statistical tests.

This paper is organized as follows: Initially, Section 2 summarizes the Differential Evolution algorithm, the $k$-NN similarity-based metamodel, and the CHTs employed in surrogate-assisted evolutionary methods recently presented in the literature. Section 3 then describes the proposed surrogate-assisted differential evolution methods. The full sets of numerical experiments are described in Section 4. The results are presented in Sections 5 and 6. The former compares the variants of the proposed method (**SADE-kNN-MCR** and **SADE-kNN-MCR+**) with the **SADE-kNN** previously presented by the authors in [38]. Then, Section 6 presents comparisons with other methods recently presented in the literature that are listed in Section **2.4**. Final remarks and conclusions are presented in Section 7.

# 2 Related Works

## *2.1 Differential Evolution*

The DE algorithm, proposed by Storn and Price in the nineties [30], has been shown to be a very effective algorithm to solve complex real-world optimization problems such as system design [43] and robot manipulator design [44]. One of the main characteristics of the DE is its simplicity and ease of implementation. A detailed review of its basic concepts is provided by [45,46], along with a survey describing its major variants, covering several aspects related to its application to multiobjective, constrained, large scale and uncertain engineering optimization problems.

Four DE variants or "strategies" were proposed in [47,48], characterized by the procedures adopted for the selection of the individuals, and the type of recombination. In this work the first one (DE/rand/1/bin) will be adopted, as follows:

$$u_{i,j,G+1} = x_{r_1,j,G} + F(x_{r_2,j,G} - x_{r_3,j,G}) \tag{2.1}$$

where $F$ is the scale factor of the algorithm, and $r_1$, $r_2$ and $r_3$ are randomly selected individuals. A crossover operator CR is also defined, to combine the information of the solutions in the current population and the solutions generated by Equation 2.1.

## *2.2 The k-NN similarity-based metamodel*

As mentioned in the Introduction, due to this simplicity and good performance, the $k$-NN approximation method has been selected in this work to comprise a surrogate model coupled to the DE algorithm. In general, the $k$-NN method evaluates a solution by calculating the average of the objective function of the $k$-nearest solutions weighted by its distances. This technique can be classified in two categories:

- $k$-nearest neighbors ($k$-NN) proposed in [49], where the $k$ nearest candidate solutions are able to be selected; and
- $r$-nearest neighbors ($r$-NN) proposed in [50], for a given neighborhood controlled by $r$ defines the candidate solutions able for selection.

In this paper, the k-NN similarity-based metamodel is adopted. The solutions $(x, f(x))$ evaluated by the exact function are stored in a database as an ordered set $D$. Any solution $x_h$ generated during the evolution is evaluated by the following weighed average:

$$f(x_h) = \frac{\sum_{j=1}^{k} d(x_h, x_j)^{u} f(x_j)}{\sum_{j=1}^{k} d(x_h, x_j)^{u}} \tag{2.2}$$

where $d(x_h, x_j)$ is the Euclidean distance between $x_h$ and $x_j$, $k$ is the number of neighbor solutions, and $u = 2$.

The number of neighbors $k$ and the size of the database $D$ significantly impact the computational costs of the algorithm, since they determine the number of operations to perform the similarity comparisons and to calculate the approximations. Section 3.2 will present some considerations regarding the database management.

### *2.3 Constraint-Handling Techniques*

A general constrained optimization problem may be formally defined in a $q$-dimensional search space as follows:

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \leq 0, i = 1, \dots, m \\ & h_j(\boldsymbol{x}) = 0, j = 1, \dots, p \\ & l_k \leq x_k \leq u_k, k = 1, \dots, q \end{aligned} \tag{2.3}$$

where $\boldsymbol{x} = (x_1, x_2, x_3, \dots, x_q)$ is a vector of design variables whose components $x_i$ present lower and upper bounds $[l_k, u_k]$. The goal is to minimize the objective function $f(\boldsymbol{x})$, considering inequality and equality constraints (respectively $g_i(\boldsymbol{x}) \leq 0$ and $h_j(\boldsymbol{x}) = 0$) that define the feasible region.

Many classical CHTs transform this constrained problem into an unconstrained one. For this purpose, whenever any given constraint is violated a penalty functions $p(\boldsymbol{x})$ is added to the objective function $f(\boldsymbol{x})$; thus, the objective function and constraints are combined into a single value. Another approach is to maintain those values apart along the evolution. Deb [51] proposed a binary tournament selection method that ranks the solutions according to the values of the objective function and constraint violation. That method comprises a pairwise comparison where a feasible solution is always preferred over an infeasible one; if both solutions are feasible, the one having the best objective function value is preferred; and if both solutions are infeasible, the one having the lowest constraint violation value is preferred. Later, other methods have followed this approach, such as the Stochastic Ranking (SR) method [52], the GCR (Global Competitive Ranking) method [53], and the Ho and Shimizu ranking method (HSR) [54].

### *2.4 Surrogate-assisted methods for constrained problems*

Surrogate-assisted evolutionary methods to solve constrained problems have been presented in the literature, including the **ASRES** – Approximated Stochastic Ranking Evolution Strategy, which employs two levels of approximation along the evolution, to estimate the values of the objective and penalty functions [55].

The **SADE-kNN** method presented in [38] employed the binary tournament selection technique proposed by Deb [51]. Other methods recently presented in the literature also employed the DE algorithm associated with some of the CHTs described above, including for instance **SA-DECV** [21] and **SA-DECV_SR** [56]. These works presented methods that couple DE with the **k-NN** surrogate model to approximate both the values of the objective function and the sum of constraint violations. The former also employs Deb's feasibility rules of [51]; the latter work [56], after comparing different CHTs associated to the same surrogate model, concludes that the best performance is provided by the Stochastic Ranking (SR) method [52].

Recently, [57] presented the **e-DE** method following two different approaches for the crossover of solutions (with the goal of balancing population diversity and fast convergence). This method employs seven comparison criteria for feasible and unfeasible solutions, comprising an extension of Deb's feasibility rules.

## 3 A new surrogate-assisted differential evolution method with multiple constraint ranking

This section describes the main characteristic of the new surrogate-assisted evolutionary method proposed in this work: **SADE-kNN-MCR**. Those characteristics are related to the following aspects:

- The Multiple Constraint Ranking (MCR) technique, comprising a specialized CHT to deal with the large number of complex constraints that may be found in real-world engineering problems;
- A merit-based strategy to manage the database that stores the exact solutions employed by the surrogate model;
- A strategy that triggers a random reinitialization or "rebirth" of the population whenever it stagnates on local optima.

### *3.1 The Multiple Constraint-Handling technique (MCR)*

Recently, focusing on the solution of complex real-world engineering optimization problems, we had presented the Multiple Constraint Ranking technique (MCR) in association with a Genetic Algorithm [39]. Now, in this paper the MCR is associated with the DE algorithm, comprising the **SADE-kNN-MCR** method and thus replacing the pair-wise tournament method of [51] that had been employed in the **SADE-*k*NN** method described in Section 2.4 . In MCR, the fitness function *F* for each solution $x_i$ is evaluated as follows:

$$F(x_i) = \begin{cases} R_{N_{vi}} + \sum_{j=1}^{m} R_{\phi i}^{j}, & \text{if only infeasible solutions} \\ R_{fi} + R_{N_{vi}} + \sum_{j=1}^{m} R_{\phi i}^{j}, & \text{otherwise} \end{cases} \quad (3.1)$$

where rank $R_f$ sorts the value of the objective function $f$, and $R_{Nv}$ sorts the number of constraints violated. **MCR** extends the rank-based approach described in Section 2.3: while previous ranking methods such as the HSR [54] employ only one rank $R_\phi$ associated to all violation values (built as a single summation of the values for functions $v_j(x)$ of all constraints), here **MCR** splits $R_\phi$ into several ranks $R_\phi^j$, one for each constraint $j = 1, \ldots, m$, building multiple queues considering the objective function and the violation of each constraint. Algorithm 1 presents the pseudocode of the **MCR**.

The HSR and other previous rank-based techniques may face difficulties in complex problems characterized by several constraints with different units and/or different orders of magnitude; in those cases, the single rank of constraint violation values could be dominated by a given constraint with higher violation values. Of course, this would not be an issue for simpler cases where one knows a-priori the ranges of minimum and maximum violation values, and usual normalization procedures could be applied. However, for many real-world engineering applications such information is not available and cannot be estimated in advance [58,59,1,2,60]. With **MCR**, all types of constraints are given the same priority and importance in the evaluation of each solution, irrespective of the range of their violation values.

**Algorithm 1:** Pseudocode of the MCR CHT

```
Initialize population of x_i individuals (popsize);
for i = 1, popsize do
        Evaluate objective function and constraints of each individual x_i;
end
for each solution x_i do
        R_Nvi = ranking position of x_i according to its number of violations;
        for each constraint j do
                R_φi^j += position of x_i in rank according to violation in constraint j;
        end
        if There are feasible individuals in the population then
                R_fi = ranking position of x_i according to its objective function;
                F(x_i) = R_fi + R_Nvi + R_φi^j;
        else
                F(x_i) = R_Nvi + R_φi^j;
        end
end
```

The **MCR** presents other advantages: it does not require any user-defined parameter; also, when there are no feasible solutions in the population, $F$ depends only on the ranks associated to the constraints ($R_{Nv}$ and $R_\phi^j$), thus the computational costs associated to evaluations of the objective function $f$ is avoided.

### *3.2 Surrogate model: Database management*

A difficult and nontrivial task associated to the k-NN metamodel described in section 2.2 is the definition of an adequate database size to store the exact solutions. Extensive databases may slow down the evolution; many similar individuals can be generated, compromising the population diversity. On the other hand, small databases may not efficiently approximate the different solutions scattered through the search space. This requires an efficient database management considering the strategies for insertion, maintenance, and replacement of the solutions to keep the database size constant.

Usually, an insertion step choses the best solution of the population for evaluation by the exact model [61-63]. In this work, a merit-based database management strategy is employed to define how the solutions are replaced according to their contribution to the approximation, following some guidelines that have been suggested in [64]. Initially, a predetermined number of solutions are generated and evaluated by the exact function to compose the database $D$. Those with the best objective function values also compose the initial population (here it is assumed that the number of solutions in the population is less or equal than the number of solutions in the database). Evolution occurs considering the current population evaluated by the surrogate model, and the database solutions used in the approximation process are scored as indicated in Algorithm 2 (that shows the general structure of the metamodel including this merit-based strategy). After each generation, a percentage of solutions of the current population is randomly chosen; they are evaluated by the exact function and introduced into the database, replacing the lower scored solutions.

**Algorithm 2:** Pseudocode of the $k$-NN metamodel with the merit-based strategy

**for** $i = 1, popsize$ **do**
  Evaluate the similarity considering the solutions of database $D$;
  Select the most similar $k$ solutions;
  Define the updated fitness function according to the arithmetic average of the $k$ distance-weighted neighboring solutions
  Increase with +1 the score of the selected solutions of $D$;
  Decrease with -1 the score of the not selected solutions of $D$;
**end**
Evaluate a percentage of the population by the exact function;
Replace the lower scored solutions in D by the solutions evaluated by the exact function;

### *3.3 Avoiding stagnation around local optima*

A very common problem associated to optimization algorithms in general is the stagnation of solutions around a local minimum in the search space. For a wide class of problems, the distribution of the solutions in the search space may be a decisive factor for this stagnation behavior, when many evaluations of the objective function may be uselessly performed.

An example of this behavior may be observed in Table 1 that presents results of the application of the **SADE-kNN-MCR** to the G1 function (from the G-Suite benchmark functions proposed for the IEEE-CEC 2006 competition on real parameter constrained optimization [42]).

The success rate (SuccRate) of the algorithm is 0.84; that is, it finds the optimal solution -15.0000 in 21 out of the 25 runs. In those 21 successful runs, the algorithm performs an average of 3,783 evaluations of the objective function (succ_nfe). This value corresponds to approximately 0.76% of the maximum number of evaluations (500,000) that defines the stopping criterion of the algorithm. In the other 4 runs, the algorithm remains stagnated in the local optimum -12.0000 until it reaches the termination criterion of 500,000 evaluations. This way, the average number of evaluations considering all 25 runs (nfe) increases to 83,177.22. This does not accurately represent the performance of the algorithm: although it was able to reach the optimum with a small number of evaluations in most of the runs (21), the stagnation in the other 4 runs drastically increases the value of this metric (nfe).

**Table 1. Results for the G1 function**

| Func | Best | Average | Worst | succ_nfe | succRate | Nfe |
|---|---|---|---|---|---|---|
| **G1** | -15.0000 | -14.5999 | -12.0000 | 3,783 | 0.84 | 83,177.22 |

There are several strategies to avoid this problem; one of them consists in restarting the population along the evolution, generating new candidate solutions whenever the population converges to a local optimum. The strategy employed in this work maintains in the database the population from the previous generation, and triggers the reinitialization (or “rebirth”) of the population after a given number of generations in which stagnation of the best solution is observed. In the context of the **SADE-kNN-MCR** algorithm, this strategy may be particularly suited to avoid the frequent premature convergence behavior due to the following reasons: The new population may visit new regions of the search space, counteracting the tendency of being stuck in regions where it previously stagnated; and there is no additional computational cost, since the objective function of the new individuals from the restarted population are approximated by the k-NN metamodel using the values from the solutions stored in the k-NN database, and thus there is no need for costly extra evaluations of the objective function.

To summarize, the method proposed in this work (that combines the **DE** algorithm with a **k-NN** surrogate model and the **MCR** constraint-handling technique) is referred to as **SADE-kNN-MCR**. Its variation that includes the population reinitialization described in this section is called **SADE-kNN-MCR+**.

# 4 Numerical experiments

The performance of the proposed methods is compared with the other methods recently presented in the literature that have been mentioned in Section **2.4**: **ASRES**, **SADE-kNN, SA-DECV, SA-DECV_SR** and **e-DE.** For this purpose, two sets of benchmark numerical experiments are executed: the twenty-four G-Suite benchmark functions proposed for the IEEE-CEC 2006 competition on real parameter constrained optimization [42]; and a suite of five benchmark mechanical engineering problems [65]: Pressure vessel [66], Welded beam [67], Cantilever beam [68], Speed reducer and Spring design [65]. The main parameters related to these latter problems are presented in Table 2, where $n$ is the number of decision variables, LI is the number of linear inequality constraints, NI is the number of nonlinear inequality constraints. LE and NE are the number of linear and nonlinear equality constraints, respectively. The complete formulation and mathematical definition of these benchmark problems are already well documented in those references; thus, they will not be reproduced here.

**Table 2. Parameters of the engineering problems**

| Problem | n | Type of function | LI | NI | LE | NE |
|---|---|---|---|---|---|---|
| Pressure Vessel [66] | 4 | Quadratic | 3 | 1 | 0 | 0 |
| Welded Beam [67] | 4 | Quadratic | 6 | 1 | 0 | 0 |
| Tension/Compression Spring [65] | 3 | Quadratic | 1 | 3 | 0 | 0 |
| Cantilever Beam [68] | 10 | Quadratic | 6 | 5 | 0 | 0 |
| Speed Reducer [65] | 7 | Cubic | 1 | 10 | 0 | 0 |

To assess specifically the advantages of the **MCR** constraint-handling technique described in Section 3.1, special attention will be focused on the problems that are representative of many actual engineering problems, i.e. presenting constraints with different units and/or different orders of magnitude: functions G10 and G21 from the CEC2006 suite, the Pressure Vessel and the Welded Beam engineering problems. Lower and upper bounds for the constraint functions of most of those problems can be analytically calculated, as shown in Table 3; the differences in scale and magnitude are evident. The orders of magnitude have been numerically estimated for the Welded Beam problem that presents severely nonlinear constraint functions.

**Table 3. Lower and upper bounds for the constraints with different magnitudes**

| **G10** | | | **G21** | | |
|---|---|---|---|---|---|
| -0.95 | $\leq g_1 \leq$ | 4 | -1,000 | $\leq g_1 \leq$ | 640.227 |
| -3.45 | $\leq g_2 \leq$ | 3.975 | -48,250 | $\leq h_1 \leq$ | 52,500 |
| -10.9 | $\leq g_3 \leq$ | 8.9 | -47,625 | $\leq h_2 \leq$ | 39,448 |
| -10,065,000 | $\leq g_4 \leq$ | 1,748,999.187 | -0.303071 | $\leq h_3 \leq$ | 0.384611 |
| -11,227,500 | $\leq g_5 \leq$ | 11,227,500 | -0.4085 | $\leq h_4 \leq$ | 0.497 |
| -11,240,000 | $\leq g_6 \leq$ | 11,215,000 | -1.6449 | $\leq h_5 \leq$ | 1.7146 |
| **Pressure Vessel** | | | **Welded Beam** (Orders of magnitude) | | |
| -3.85375 | $\leq g_1 \leq$ | 4.9807 | $-10^4$ | $\leq g_1 \leq$ | $10^4$ |
| -1.90175 | $\leq g_2 \leq$ | 4.9046 | $-10^4$ | $\leq g_2 \leq$ | $10^6$ |
| -1,288,673.3 | $\leq g_3 \leq$ | 57,317,600 | $-10$ | $\leq g_3 \leq$ | $10$ |
| 40 | $\leq g_4 \leq$ | 230 | $-10$ | $\leq g_4 \leq$ | $10^2$ |
| | | | $-10$ | $\leq g_5 \leq$ | $10^{-1}$ |
| | | | $-10^{-1}$ | $\leq g_6 \leq$ | $10^4$ |
| | | | $-10^7$ | $\leq g_7 \leq$ | $10^3$ |

The remainder of this section will present the main parameters considered for the execution of the methods for those benchmark problems and will describe the approaches for the assessment of their performance.

### *4.1 Parameters of the algorithms*

Following the suggestion of [42], the algorithms are tested from the results of twenty-five independent executions performed for each benchmark problem. Also following the specifications of [42], the tolerances for satisfying the equality constraints, and for determining if the optimal solution $x^*$ has been found, are respectively 0.0001 and $f(x^*) - f(x) < 0.0001$. The maximum number of function evaluations is set to 500,000 for the G-Suite problems. Regarding the structural engineering problems, 10,000 function evaluations are considered to obtain the results presented in section 5.2 to compare the different **SADE-kNN** algorithms.

Numerical experiments have been conducted to set the value for the parameter that control the reinitialization of the population, i.e. the number of iterations in which stagnation is observed (as described in section 3.2). The results indicated that an appropriate value would be 1,000 iterations. Similar experiments have also indicated that an appropriate value for the percentage of solutions selected in the current population at each generation, for the evaluation of their objective function and introduction in the database (as described in section 3.2), is twenty-two percent. Table 4 presents the values for the remaining parameters employed for the **SADE-kNN** algorithm, and also the parameters that have been used by the other algorithms being compared.

**Table 4. Parameters of the algorithms**

| | | **SADE-kNN** | **SA-DECV** [21] | **SA-DECV-SR** [56] | **ASRES** [55] | **e-DE** [57] |
|---|---|---|---|---|---|---|
| Differential Evolution | Pop. size ($N$) | 30 | 90 | 90 | 200 | 50 |
| | Runs | 25 | 30 | 25 | 25 | 25 |
| | CR | 0.9 | 1 | 1 | - | 0.9 |
| | F | 0.8 | 0.9 | 0.9 | - | 0.8 |
| k-NN | $k$ (number of neighbors) | 2 | $n$ (problem dimension) | $n$ (problem dimension) | $N$ $N$ regression model | - |
| | $D$ (database size) | 1.5*$N$ | 2*$N$ | 2*$N$ | $N$ | - |

### *4.2 Performance Assessment*

The performance of the methods is assessed by direct comparisons and by nonparametric statistical tests, as described next.

#### *4.2.1 Direct Comparisons*

Direct comparisons are performed by observing the most relevant performance measures collected in tables: best, average, worst results for 25 runs and average of the number of function evaluations to find a feasible solution, considering only the successful runs (*succ_nfe*); ratio between the number runs in which the algorithms find the optimum solution and the total number

of runs (*succRate*); average of the number of function evaluations (*nfe*) (when *succRate* = 1, then *succ_nfe* = *nfe*); ratio of runs where the algorithm finds a feasible solution (*feasRate*).

### *4.2.2 Sign Test*

The Sign Test (ST) is a statistical procedure that indicates if the results of two methods are statistically different. It employs pairwise comparisons that count the number of cases on which a method wins. The method that provides a better result for a given problem receives a positive sign, while the other receives a negative sign. The significance of the difference between these pairwise comparisons (or *significance level*) is tested by calculating probability values (*p*-values) [69] from the totalized signs. If the *p*-value is higher than a given threshold value $\alpha$, then the null hypothesis ($H_0$) that both methods are equivalent is accepted; otherwise $H_0$ is rejected, and the performance of the methods is different. The most commonly used threshold values are 0.01 and 0.05 [70-73], representing a significance level of respectively 1% and 5%, i.e., the chance of rejecting the null hypothesis when in fact it is correct.

### *4.2.3 Performance Profiles*

The comparisons also use another nonparametric statistical test, specifically the Performance Profiles (PPs) that were introduced in [74] to evaluate and compare the performance of a set of *S* solvers for a given set of optimization problems *P*. The PPs are a useful tool for the easy visualization and interpretation of results from computational experiments. Briefly speaking, they comprise graphs with several curves (one for each solver $s \in S$ considering all problems $p \in P$), built considering a given metric $m_{p,s}$ . This metric can be the number of evaluations of the objective function; or a statistical parameter such as mean or best value computed from objective function values obtained in several independent runs of the method. The *performance ratio* $r_{p,s,}$ is then calculated dividing this metric $m_{p,s}$ by the corresponding value of the metric obtained by the solver that performed best on problem *p*. An overall assessment of the performance of solver *s* may then be given by:

$$\rho_s(\tau) \quad = \quad \frac{\text{size}\left\{ p \in P : r_{p,s} \le \tau \right\}}{n_p} \tag{4.1}$$

where $n_p$ is the total number of problems in set *P*; and $\rho_s(\tau) \in \Re \rightarrow [0, 1]$ may be seen as the cumulative distribution function for the performance ratio $r_{p,s}$; that is, the probability of solver *s* to have its ratio $r_{p,s}$ within a factor $\tau$ of the best possible ratio.

For $\tau$=1, $\rho_s(1)$ indicates the accuracy of solver *s*, being a normalized measure of the number of times it was able to provide the best value for the considered metric, or the probability that it will win over all other solvers in set *S*; $\rho_s(1) = 1$ indicates that this solver wins over all other solvers in all problems *P*.

For larger values of $\tau$, $\rho_s(\tau)$ indicates the robustness of the solver, i.e., the number of times it is able to provide feasible solutions; considering an upper limit $r_M$ as an arbitrarily large value assigned to $r_{p,s}$ when solver *s* is not able to find a solution for problem *p*, then $\rho_s(r_M) = 1$ and the probability that method *s* solves a problem is given by $\rho_s^* = \lim_{\tau \to r_M} \rho_s(\tau)$.

Thus, the PPs are curves $[\rho_s(\tau), \tau]$, nondecreasing, piecewise constant functions, continuous from the right at each discontinuity point. A detailed explanation including an illustrative example of the assembly of a PP may be found in [39].

# 5 Comparing the SADE-kNN methods

Initially, this section compares the proposed methods (**SADE-kNN-MCR** and **SADE-kNN-MCR+**) with the **SADE-kNN** method previously presented by the authors in [38]. The performance is compared by their application to the sets of numerical experiments listed at the beginning of Section 4: the G-Suite benchmark functions proposed for the IEEE-CEC 2006 competition and the suite of five benchmark mechanical engineering problems.

## *5.1 G-Suite functions*

### *5.1.1 Direct Comparisons*

Table 5 presents the direct comparisons between the methods, in terms of the performance measures defined in Section 4.2.1, statistically evaluated from the results of the 25 independent runs. Those measures are: best, average and worst solution; *succ_nfe*: average number of function evaluations considering only the successful runs; *succRate*: ratio between successful and total runs; *nfe*: average number of evaluations considering all 25 runs; *feasRate*: ratio of feasible runs and total runs. Results for functions G20 and G22, for which no algorithm has currently found a feasible solution, are not presented. Values in bold indicate the results in which **SADE-kNN-MCR+** finds better or equal solutions than the other algorithms. For problems in which all three methods reached at least one successful run, the cells corresponding to *succ_nfe* shows in parenthesis the percentage values corresponding to the reduction/increase of the *succ_nfe* values provided by **SADE-kNN-MCR+** in relation to the smaller value observed for the other methods.

**Table 5. Results obtained by the SADE-kNN algorithms for the G-Suite problems**

| Function [best known] | | SADE-kNN | SADE-kNN MCR | SADE-kNN MCR+ |
|---|---|---|---|---|
| G1 [-15.0000] | best | -15 | -15 | **-15** |
| | average | -14.5599 | -14.5999 | **-15** |
| | worst | -12 | -12 | **-15** |
| | succ_nfe | 3,722.4 | 3,783 | **2,528.5 (-32%)** |
| | succRate | 0.84 | 0.84 | **1** |
| | nfe | 83,127.32 | 83,177.22 | **2,528.5** |
| | feasRate | 1 | 1 | **1** |
| G2 [-0.8036] | best | -0.7429 | -0.7727 | **-0.8036** |
| | average | -0.6367 | -0.7458 | **-0.7729** |
| | worst | -0.5277 | -0.6532 | **-0.7186** |
| | succ_nfe | - | **-** | **28,632.7** |
| | succRate | 0 | 0 | **0.16** |
| | nfe | 500,000 | 500,000 | **424,581.2** |
| | feasRate | 1 | 1 | **1** |

| Function [best known] | | SADE-kNN | SADE-kNN MCR | SADE-kNN MCR+ |
|---|---|---|---|---|
| G3 [-1.0005] | best | -0.4515 | -0.1687 | **-1.0005** |
| | average | -0.0399 | -0.0288 | **-0.4379** |
| | worst | 0.0000 | 0.0000 | **-0.0487** |
| | succ_nfe | - | **-** | **169,089.4** |
| | succRate | 0 | 0 | **0.24** |
| | nfe | 500,000 | 500,000 | **380,873.3** |
| | feasRate | 1 | 0.68 | **1** |
| G4 [-30,665.5386] | best | -30,665.5386 | -30,665.5386 | **-30,665.5386** |
| | average | -30,665.5386 | -30,665.5386 | **-30,665.5386** |
| | worst | -30,665.5386 | -30,665.5386 | **-30,665.5386** |
| | succ_nfe | 2,597.60 | 2,457.12 | **1,208.2 (-51%)** |
| | succRate | 1 | 1 | **1** |
| | nfe | 2,597.60 | 2,457.12 | **1,208.2** |
| | feasRate | 1 | 1 | **1** |
| G5 [5,126.49] | best | 5,126.49 | 5,126.49 | **5,126.49** |
| | average | 5,126.49 | 5,165.92 | **5,126.49** |
| | worst | 5,126.4977 | 6,112.22 | **5,126.49** |
| | succ_nfe | 17,809.7 | 26,003.9 | **14,612.6 (-18%)** |
| | succRate | 0.88 | 0.96 | **1** |
| | nfe | 74,855.24 | 44,963.92 | **14,612.6** |
| | feasRate | 0.96 | 1 | **1** |
| G6 [-6,961.8138] | best | -6,961.8138 | -6,961.8138 | **-6,961.8138** |
| | average | -6,961.8138 | -6,961.8138 | **-6,961.8138** |
| | worst | -6,961.8138 | -6,961.8138 | **-6,961.8138** |
| | succ_nfe | 1,234.88 | 1,134.76 | **655.5 (-42%)** |
| | succRate | 1 | 1 | **1** |
| | nfe | 1,234.88 | 1,134.76 | **655.5** |
| | feasRate | 1 | 1 | **1** |
| G7 [24.3062] | best | 24.3073 | 24.3082 | **24.3062** |
| | average | 28.2727 | 48.2589 | **24.3062** |
| | worst | 121.7574 | 512.2665 | **24.3062** |
| | succ_nfe | - | **-** | **64,067.2** |
| | succRate | 0 | 0 | **1** |
| | nfe | 500,000 | 500,000 | **64,067.2** |
| | feasRate | 1 | 1 | **1** |
| G8 [-0.09582] | best | -0.09582 | -0.09582 | **-0.09582** |
| | average | -0.09582 | -0.09582 | **-0.09582** |
| | worst | -0.09582 | -0.09582 | **-0.09582** |
| | succ_nfe | 291.76 | 326.48 | **182.8 (-37%)** |
| | succRate | 1 | 1 | **1** |
| | nfe | 291.76 | 326.48 | **182.8** |
| | feasRate | 1 | 1 | **1** |
| G9 [680.630] | best | 680.638 | 680.6353 | **680.630** |
| | average | 681.287 | 680.6783 | **680.630** |
| | worst | 682.476 | 680.7954 | **680.630** |
| | succ_nfe | - | - | **38226.8** |
| | succRate | 0 | 0 | **1** |
| | nfe | 500,000 | 500,000 | **38226.8** |
| | feasRate | 1 | 1 | **1** |
| G10 [7,049.2480] | best | 7,049.249 | 7,049.249 | **7,049.2480** |
| | average | 7,278.785 | 7,198.919 | **7,049.2480** |
| | worst | 11,100.000 | 8,332.225 | **7,049.2480** |

| Function [best known] | | SADE-kNN | SADE-kNN MCR | SADE-kNN MCR+ |
|---|---|---|---|---|
| | succ_nfe | - | **-** | **20,198.4** |
| | succRate | 0 | 0 | **1** |
| | nfe | 500,000 | 500,000 | **20,198.4** |
| | feasRate | 1 | 0.96 | **1** |
| G11 [0.7499] | best | 0.74990 | 0.74990 | **0.74990** |
| | average | 0.9695 | 0.7736 | **0.74990** |
| | worst | 1.0000 | 1.0000 | **0.74990** |
| | succ_nfe | 2,995.0 | 2,941.57 | **1,334.2 (-55%)** |
| | succRate | 0.12 | 0.56 | **1** |
| | nfe | 440,363.28 | 221,649.32 | **1,334.2** |
| | feasRate | 1 | 0.76 | **1** |
| G12 [-1.0000] | best | -1 | -1 | **-1** |
| | average | -1 | -1 | **-1** |
| | worst | -1 | -1 | **-1** |
| | succ_nfe | 385.96 | 439.52 | **211.5 (-45%)** |
| | succRate | 1 | 1 | **1** |
| | nfe | 385.96 | 439.52 | **211.5** |
| | feasRate | 1 | 1 | **1** |
| G13 [0.05394] | best | 0.05394 | 0.05394 | **0.05394** |
| | average | 0.3534 | 0.2930 | **0.05394** |
| | worst | 1.000 | 1.000 | **0.05394** |
| | succ_nfe | **43,906.85** | 55,912.83 | 48,682.1 (+10%) |
| | succRate | 0.28 | 0.48 | **1** |
| | nfe | 372,297.24 | 286,840.32 | **48,682.1** |
| | feasRate | 1 | 0.96 | **1** |
| G14 [-47.76] | best | -47.76 | -47.76 | **-47.76** |
| | average | -47.595 | -47.513 | **-47.76** |
| | worst | -43.845 | -43.844 | **-47.76** |
| | succ_nfe | 55,179.31 | 100,064.78 | **54,356.1 (-1%)** |
| | succRate | 0.88 | 0.56 | **1** |
| | nfe | 108,558.52 | 276,038.16 | **54,356.1** |
| | feasRate | 1 | 1 | **1** |
| G15 [961.7150] | best | 961.7150 | 961.7150 | **961.7150** |
| | average | 961.7150 | 961.7150 | **961.7150** |
| | worst | 961.7150 | 961.7150 | **961.7150** |
| | succ_nfe | 1,1431.3 | 1,7245.28 | **5,412.4 (-53%)** |
| | succRate | 0.92 | 1 | **1** |
| | nfe | 54,909.56 | 17,245.28 | **5,412.4** |
| | feasRate | 0.92 | 1 | **1** |
| G16 [-1.9051] | best | -1.9051 | -1.9051 | **-1.9051** |
| | average | -1.9051 | -1.9051 | **-1.9051** |
| | worst | -1.9051 | -1.9051 | **-1.9051** |
| | succ_nfe | 4,633.36 | 4,330.66 | **1,447.8 (-67%)** |
| | succRate | 1 | 0.96 | **1** |
| | nfe | 4,633.36 | 2,4157.6 | **1,447.8** |
| | feasRate | 1 | 0.96 | **1** |
| G17 [8,853.53] | best | 8,853.53 | 8,853.53 | **8,853.53** |
| | average | 8,868.96 | 8,910.50 | **8,860.07** |
| | worst | **8,927.59** | 9,011.91 | 8,938.51 |
| | succ_nfe | **69,887.31** | 70,850.44 | 258,603.70 (+73%) |
| | succRate | **0.76** | 0.40 | 0.68 |
| | nfe | **173,115.36** | 345,508.28 | 304584 |

| Function [best known] | | SADE-kNN | SADE-kNN MCR | SADE-kNN MCR+ |
|---|---|---|---|---|
| | feasRate | **0.96** | **0.96** | 0.84 |
| G18 [-0.8660] | best | -0.866 | -0.866 | **-0.866** |
| | average | -0.8654 | -0.8649 | **-0.866** |
| | worst | -0.8536 | -0.8597 | **-0.866** |
| | succ_nfe | 253,743.70 | 85,907.04 | **5,855.1 (-93%)** |
| | succRate | 0.40 | 0.92 | **1** |
| | nfe | 401,500.28 | 119,034.84 | **5,855.1** |
| | feasRate | 1 | 1 | **1** |
| G19 [32.6555] | best | 32.6632 | 32.659 | **32.6569** |
| | average | 39.2811 | 38.080 | **32.6586** |
| | worst | 81.044 | 70.502 | **32.6676** |
| | succ_nfe | - | **-** | - |
| | succRate | 0 | 0 | **0** |
| | nfe | 500,000 | 500,000 | **500,000** |
| | feasRate | 1 | 1 | **1** |
| G21 [193.7245] | best | 193.7546 | 193.7546 | **193.7245** |
| | average | **193.7546** | **193.7546** | 199.4284 |
| | worst | **193.7546** | **193.7546** | 324.7028 |
| | succ_nfe | - | **-** | **122454** |
| | succRate | 0 | 0 | **0.6** |
| | nfe | 500,000 | 500,000 | **258,371.4** |
| | feasRate | 0.64 | 0.92 | **0.92** |
| G23 [-400.055] | best | -400.055 | -400.055 | **-400.055** |
| | average | -400.055 | -154.191 | **-400.055** |
| | worst | -400.055 | -0.0024 | **-400.055** |
| | succ_nfe | 68,852.00 | 88,843.33 | **51,922.6 (-25%)** |
| | succRate | 0.04 | 0.24 | **1** |
| | nfe | 482,757.88 | 401,325.76 | **51,922.6** |
| | feasRate | 0.04 | 0.96 | **1** |
| G24 [-5.5080] | best | -5.508 | -5.508 | **-5.508** |
| | Mean | -5.5080 | -5.5080 | **-5.508** |
| | worst | -5.5080 | -5.5080 | **-5.508** |
| | succ_nfe | 765.20 | 710.56 | **370 (-48%)** |
| | succRate | 1 | 1 | **1** |
| | nfe | 765.20 | 710.56 | **370** |
| | feasRate | 1 | 1 | **1** |

It can be seen in Table 5 that the **SADE-kNN-MCR+** method was able to find the optimal solution for 21 out of the 22 functions. Although it had not reached the known optimum for problem G19, the error is relatively small. Also, the fact that the best and worst solutions are very close (that is, the results found in the twenty-five runs present a small variation) indicate the robustness of the method. Considering these 21 functions for which the **SADE-kNN-MCR+** was able to reach the optimal solution, it frequently (except for function G17) obtained the highest success rate (*succRate*), indicating its best performance in terms of the greater number of runs in which it can find the optimum**.**

The other methods were able to reach the global optimum for only 15 out of these 21 functions (failing in functions G2, G3, G7, G10, G19 and G21). Considering those 15 functions for which all methods found the optimal solution, the highest *succRate* (always equal to 1) is

again obtained by the **SADE-kNN-MCR+**. The feasibility rate (*feasRate*) obtained by the **SADE-kNN-MCR+** is again always equal to 1.

Considering the 6 problems for which only **SADE-kNN-MCR+** was able to find the optimal solution, in three of them (G7, G9 and G10), it obtained a success rate of 1 (that is, all runs reached the optimal solution).

Regarding the G10 and G21 problems that present constraints with different units and/or different orders of magnitude, shown in Table 3: for function G10, the enhanced performance of the **SADE-kNN-MCR** method can be measured by the better *average* and *worst* metrics compared with the original **SADE-kNN**; then, the association of the "rebirth" strategy makes the **SADE-kNN-MCR+** obtain the optimal solution in all runs. For function G21, the enhanced performance of the **SADE-kNN-MCR** and **SADE-kNN-MCR+** methods is reflected by their higher number of feasible runs (*FeasRate*).

Besides the increased performance in finding optimal solutions as commented above, the **SADE-kNN-MCR+** also reduces the average number of function evaluations to find a feasible solution (*succ_nfe*). To assess this behavior, in the cells corresponding to *succ_nfe* for **SADE-kNN-MCR+** and considering only the 15 test problems for which all three methods reached at least one successful run, Table 5 shows percentage values in parenthesis that correspond to the ratio between the *succ_nfe* values provided by **SADE-kNN-MCR+**, and the smaller *succ_nfe* value observed for the other methods. The number of evaluations is reduced for 13 out of those 15 problems. For the other two problems (G13 and G17), although requiring a higher number of evaluations, **SADE-kNN-MCR+** presents better success rates; for the G13 function, for instance, it finds the optimal solution in 100% of the runs, while **SADE-kNN** succeeds in only 28%.

### *5.1.2 Sign Test*

Table 6 summarizes the **SuccRate** values for all methods, considering the 15 test problems for which all three methods reached at least one successful run. For all those problems (except G17) **SADE-kNN-MCR+** found the optimal solution for 100% of the runs, while the other methods reached **SuccRate** = 1 for only 6 out of those 15 problems. The results of the Sign Test from the pairwise comparison between **SADE-kNN-MCR+** and each other method are shown in the last three lines of Table 6 (discarding the draws). The number of wins and losses confirms the observations from Table 5, i.e., the **SADE-kNN-MCR+** outperforms all other methods. All *p*-values (computed according to the procedure described in Section 4.2.2) are small enough to reject the null hypothesis, meaning the pairwise compared samples are not statistically equivalent in all cases. This demonstrates that **SADE-kNN-MCR+** outperforms all other methods.

**Table 6. Success Rate**

| | Problem | SADE-kNN | SADE-kNN-MCR | SADE-kNN-MCR+ |
|---|---|---|---|---|
| 1 | G1 | 0.84 | 0.84 | **1** |
| 2 | G4 | **1** | **1** | **1** |
| 3 | G5 | 0.88 | 0.96 | **1** |
| 4 | G6 | **1** | **1** | **1** |
| 5 | G8 | **1** | **1** | **1** |
| 6 | G11 | 0.12 | 0.56 | **1** |
| 7 | G12 | **1** | **1** | **1** |
| 8 | G13 | 0.28 | 0.48 | **1** |
| 9 | G14 | 0.88 | 0.56 | **1** |
| 10 | G15 | 0.92 | **1** | **1** |
| 11 | G16 | 1 | 0.96 | **1** |
| 12 | G17 | **0.76** | 0.40 | 0.68 |
| 13 | G18 | 0.40 | 0.92 | **1** |
| 14 | G23 | 0.04 | 0.24 | **1** |
| 15 | G24 | **1** | **1** | **1** |
| SADE-kNN-MCR+ Wins (+) | | 8 | 9 | |
| SADE-kNN-MCR+ Losses (-) | | 1 | 0 | |
| p-values | | 0.019531 | 0.001953 | |

### *5.1.3 Performance Profiles*

The **SuccRate** values presented in Table 6 were also used as the metric for the generation of the performance profiles, according to the procedure described in Section 4.2.3 . The PPs shown in Figure 1 again indicate that **SADE-kNN-MCR+** is the best performer, since its curve reaches $\rho(\tau) = 1$ with the lowest $\tau$ value. A global indication of the performance of the methods may also be observed in Figure 2 that contains a bar chart where the y axis indicates the areas under the PP curves, again confirming the better accuracy and robustness of **SADE-kNN-MCR+**.

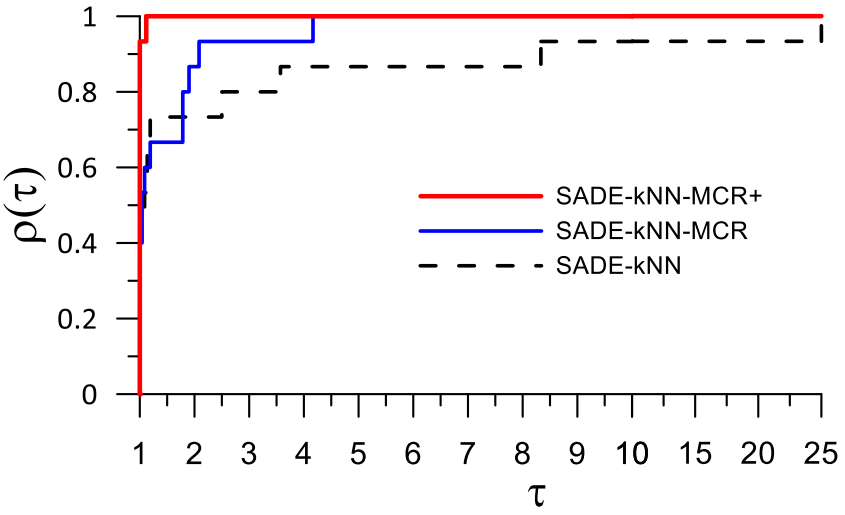


**Figure 1. Performance Profiles – SADE-kNN methods**

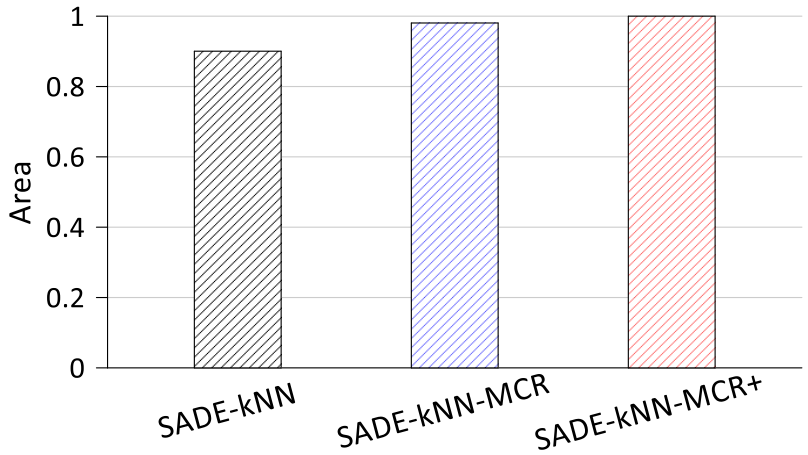


**Figure 2. Global performance – SADE-kNN methods**

### *5.2 Suite of structural engineering problems*

Table 7 compares the results of the methods for the suite of five benchmark mechanical engineering problems shown in Table 2. All algorithms provided feasible solutions in all runs. Values in boldface indicate the best results amongst the algorithms.

**Table 7. Results for the suite of engineering problems – SADE-kNN methods**

| | | SADE-kNN-MCR+ | SADE-kNN-MCR | SADE-kNN |
|---|---|---|---|---|
| **Pressure Vessel** | Best | **6,059.7143** | **6,059.7143** | 6,059.7180 |
| | Average | **6,176.5486** | 6,577.5020 | 6,276.0322 |
| | Worst | **6,820.4100** | 7,332.8415 | 7,332.8529 |
| **Welded Beam** | Best | **1.724852** | **1.724852** | 1.726247 |
| | Average | **1.724852** | 1.802360 | 1.784099 |
| | Worst | **1.724852** | 2.497832 | 2.319204 |
| **Cantilever** | Best | **64,578.194** | **64,578.194** | 65,036.880 |
| | Average | **66,581.615** | 66,815.056 | 70,160.740 |
| | Worst | **68,597.130** | 68,597.130 | 74,223.488 |
| **Speed Reducer** | Best | **2,996.3481** | **2,996.3481** | **2,996.3481** |
| | Average | **2,996.3481** | 3,002.1815 | 2,999.8637 |
| | Worst | **2,996.3481** | 73,035.625 | 3,035.6255 |
| **Tension/Compr. Spring** | Best | **0.012665** | **0.012665** | 0.012675 |
| | Average | 0.013285 | **0.013095** | 0.012952 |
| | Worst | 0.025651 | **0.017773** | 0.015754 |

Regarding the "best" solutions, both **SADE-kNN-MCR+** and **SADE-kNN-MCR** are the best performers, except for the Speed Reducer problem where all of them achieved the same solution. It is important to stress that, for the Pressure Vessel and Welded Beam problems that present constraints with different units and/or different orders of magnitude, the better

performance of the MCR-based methods compared with the **SADE-kNN** algorithm may be due to the increased efficiency of the **MCR** constraint-handling method in dealing with this type of problems, as indicated in [39]. The increased robustness of the **SADE-kNN-MCR+** compared with the other methods is demonstrated by the average and worst results, since it presents the best performance for all problems except the Tension/Compression Spring.

Here, since the number of examples (only five benchmark problems) is low, statistical performance assessment methods such as the sign test cannot be adequately employed. Even so, it may be illustrative to compare the performance profiles for the methods. Since **SADE-kNN-MCR+** and **SADE-kNN-MCR** found the same best values for all problems, the average was employed as the comparison metric to be evaluated by the performance profiles. Figure 3 shows that **SADE-kNN-MCR+** finds best values for 80% of the problems with $\rho(1) = 0.8$. Altogether, **SADE-kNN-MCR+** is the best overall performer, as also indicated by Figure 4.

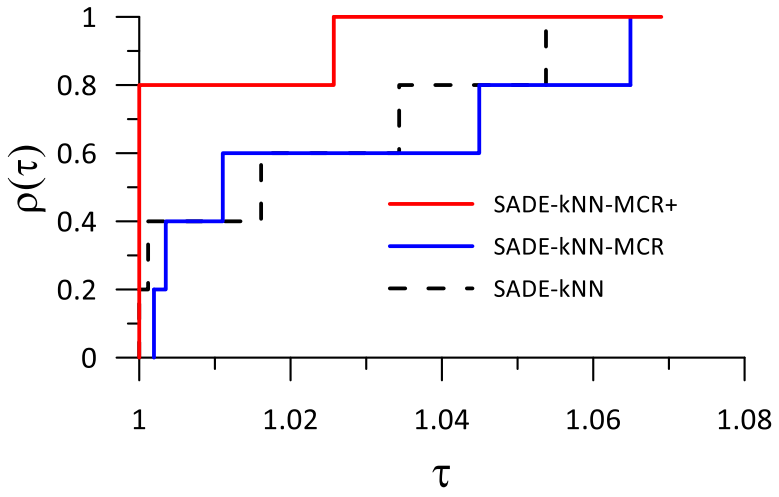


**Figure 3. Performance Profiles – SADE-kNN methods applied to the engineering problems. Metric: average.**

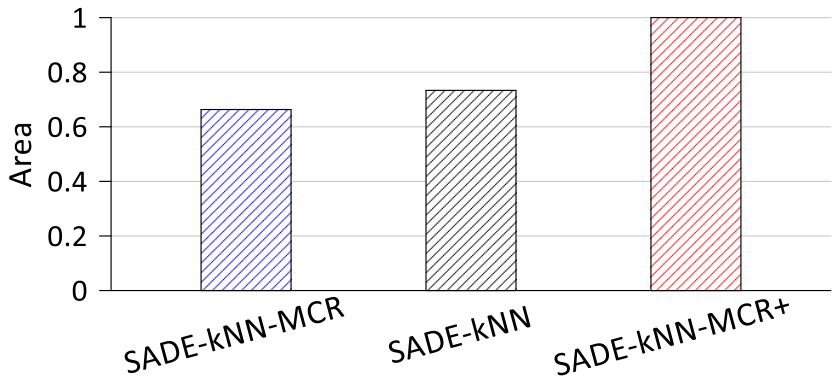


**Figure 4. Global performance – SADE-kNN methods applied to the engineering problems. Metric: average.**

# 6 Comparison with other recent methods

Since in the previous section **SADE-kNN-MCR+** had demonstrated its better performance amongst the proposed **SADE** methods, this section now compares its results with those obtained by other methods recently presented in the literature. Those methods have been mentioned in Section **2.4**: **ASRES**, **SA-DECV, SA-DECV_SR** and **e-DE** which employ different surrogate models and/or different constraint handling techniques. The performance of the methods is compared firstly by their application to the IEEE-CEC 2006 G-Suite benchmark functions, and then to the benchmark of mechanical engineering problems as described in the beginning of Section 4.

## *6.1 G-Suite functions*

### *6.1.1 Direct comparisons*

Table 8 compares the results of the methods, in terms of the available main statistical parameters that characterize their performance: the best solution found amongst the runs; the average number of function evaluations considering the successful runs *succ_nfe*; and the ratio between successful and total runs *SuccRate*. The cells corresponding to *succ_nfe* shows also, in parenthesis, the percentage values corresponding to the reduction/increase of the *succ_nfe* values provided by **SADE-kNN-MCR+** in relation to the smaller value observed for the other methods.

**Table 8. Comparison of Results**

| | | **SADE-kNN-MCR+** | **SA-DECV** [21] | **SA-DECV_SR** [56] | **ASRES** [55] | **e-DE** [57] |
|---|---|---|---|---|---|---|
| G1 [-15.0000] | Best | **-15** | **-15** | **-15** | **-15** | **-15** |
| | succ_nfe | **2,528.5 (-68%)** | 7,972 | 15,730.00 | 35,406 | 64,274 |
| | SuccRate | **1** | 0.83 | 0.84 | **1** | **1** |
| G2 [-0.8036] | Best | **-0.8036** | -0.713 | -0.6252 | -0.739 | **− 0.8036** |
| | succ _nfe | **28,632.7 (-85%)** | - | - | - | 192,297 |
| | SuccRate | 0.16 | 0 | 0 | 0 | **0.92** |
| G3 [-1.0005] | Best | **-1.0005** | -0.333 | -0.3127 | -0.998 | **− 1.0005** |
| | succ _nfe | 169,089.4 (+80%) | - | - | - | **33,066** |
| | SuccRate | 0,24 | 0 | 0 | 0 | **1** |
| G4 [-30,665.5386] | Best | **-30,665.5386** | -30,665.539 | -30,665.5387 | -30,665.539 | **− 30,665.5386** |
| | succ _nfe | **1,208.2 (-76%)** | 5,113 | 7,175 | 15,104 | 43,942 |
| | SuccRate | **1** | **1** | 0,8 | **1** | **1** |
| G5 [5,126.49] | Best | **5,126.49** | **5,126.49** | **5,126.49** | **5,126.49** | **5,126.49** |
| | succ _nfe | **14,612.6 (-24%)** | 69,418 | 21,960 | 19,281 | 152,110 |
| | SuccRate | **1** | **1** | 0.96 | **1** | **1** |
| G6 [-6,961.8138] | Best | **-6,961.8138** | -6,961.814 | **-6,961.8138** | -6,961.814 | **− 6,961.8138** |
| | succ _nfe | **655.5** | 7,335 | 5,734 | 9,603 | 42,098 |

| | | **SADE-kNN-MCR+** | **SA-DECV** [21] | **SA-DECV_SR** [56] | **ASRES** [55] | **e-DE** [57] |
|---|---|---|---|---|---|---|
| | | **(-86%)** | | | | |
| | SuccRate | **1** | **1** | **1** | **1** | **1** |
| G7 [24.3062] | Best | **24.3062** | 24.409 | **24.3062** | **24.306** | **24.3062** |
| | succ _nfe | 64,067.2 (+54%) | - | **29,440** | 77,674 | 99,614 |
| | SuccRate | **1** | 0 | 0.04 | 0.08 | **1** |
| G8 [-0.09582] | Best | **-0.09582** | **-0.096** | **-0.09582** | **-0.096** | **− 0.09582** |
| | succ _nfe | **182.8 (-66%)** | 676 | 539.3 | 1,027 | 6,254 |
| | SuccRate | **1** | **1** | 0.96 | **1** | **1** |
| G9 [680.630] | Best | **680.630** | **680.630** | **680.630** | **680.63** | **680.630** |
| | succ _nfe | **38,226.8 (+79%)** | 9,831 | 7,994 | 30,618 | 29,446 |
| | SuccRate | **1** | 0.2 | 0.84 | **1** | **1** |
| G10 [7,049.2480] | Best | **7,049.2480** | 7,260.041 | **7,049.2480** | 7,049.408 | **7,049.2480** |
| | succ _nfe | **20,198.4 (-56%)** | - | 45,560 | - | 135,934 |
| | SuccRate | **1** | 0 | 0.04 | 0 | **1** |
| G11 [0.7499] | Best | **0.7499** | **0.75000** | **0.7499** | **0.75** | **0.7499** |
| | succ _nfe | 1,334.2 | 10,478 | 10,580 | 2,792 | 24,566 |
| | SuccRate | 1 | 1 | 0.8 | 1 | 1 |
| G12 [-1.0000] | Best | **-1** | **-1** | **-1** | **-1** | **-1** |
| | succ _nfe | **211.5 (-68%)** | 1,139 | 660.6 | 2,996 | 1,354 |
| | SuccRate | **1** | **1** | 0.64 | **1** | **1** |
| G13 [0.05394] | Best | **0.05394** | 0.453 | 0.337683 | **0.054** | **0.05394** |
| | succ _nfe | 48,682.1 (+77%) | - | - | **11,292** | 303,741 |
| | SuccRate | **1** | 0 | 0 | 0.84 | 0.44 |
| G14 [-47.76] | Best | **-47.76** | -47.24 | -47.48 | **-47.76** | **− 47.76** |
| | succ _nfe | **54,356.1 (-41%)** | - | - | 92,820 | 92,730 |
| | SuccRate | **1** | 0 | 0 | 0.08 | **1** |
| G15 [961.7150] | Best | **961.715** | **961.715** | **961.715** | **961.715** | **961.715** |
| | succ _nfe | **5,412.4 (-36%)** | 46,105 | 12,430 | 8,519 | 95,022 |
| | SuccRate | **1** | 0.63 | 0.64 | **1** | **1** |
| G16 [-1.9051] | Best | **-1.905** | **-1.905** | **-1.905** | **-1.905** | **− 1.905** |
| | succ _nfe | **1,447.8 (-85%)** | 10,149 | 9,384 | 16,179 | 24,410 |
| | SuccRate | **1** | **1** | **1** | **1** | **1** |
| G17 [8,853.53] | Best | **8,853.53** | 8,872.62 | 8,857.65 | **8,853.54** | **8,853.53** |
| | succ _nfe | 258,603.71 (+92%) | - | - | **21,490** | 264,232 |
| | SuccRate | 0.68 | 0 | 0 | **0.76** | 0.44 |
| G18 [-0.8660] | Best | **-0.8660** | **-0.8660** | **-0.8660** | **-0.8660** | **− 0.8660** |
| | succ _nfe | **5,855.1 (-80%)** | 32,751 | 29,830 | 40,840 | 140,458 |
| | SuccRate | **1** | 0.03 | 0.76 | 0.92 | **1** |

| | | SADE-kNN-MCR+ | SA-DECV [21] | SA-DECV_SR [56] | ASRES [55] | e-DE [57] |
|---|---|---|---|---|---|---|
| G19 [32.6555] | Best | 32.6569 | 35.404 | **32.6556** | 32.665 | **32.6555** |
| | succ _nfe | - | - | 53,740 | - | 207,286 |
| | SuccRate | 0 | 0 | 0.04 | 0 | **1** |
| G21 [193.7245] | Best | **193.7245** | 193.732 | 193.7267 | - | **193.7245** |
| | succ _nfe | 122,454 (+31%) | - | - | - | **84,500** |
| | SuccRate | 0.6 | 0 | 0 | 0 | **0.72** |
| G23 [-400.055] | Best | **-400.055** | -308.803 | -365.2985 | -348.816 | **-400.055** |
| | succ _nfe | 51,922.6 (+10%) | - | - | - | **46,922** |
| | SuccRate | **1** | 0 | 0 | 0 | **1** |
| G24 [-5.5080] | Best | **-5.508** | **-5.508** | **-5.508** | **-5.508** | **− 5.508** |
| | succ _nfe | **370 (-88%)** | 3,444 | 3,015 | 3,638 | 19,678 |
| | SuccRate | **1** | **1** | **1** | **1** | **1** |

In Table 8, it is interesting to compare the performance between the **SA-DECV** and the **SA-DECV-SR** methods. The latter is a variation that uses a rank-based constraint handling technique, instead of simple feasibility rules; thus, unlike the **SA-DECV**, the **SA-DECV-SR** provides optimal solutions to problems G7, G10 and G19. It also improves the solution found by **SA-DECV** in problem G21. **SA-DECV-SR** also produced a slight improvement in the number of optimal executions.

However, the performance of these algorithms is still inferior to those presented by **SADE-kNN-MCR+** and **e-DE**: **SA-DECV** methods failed in functions G13, G14, G17 and G23, while **SADE-kNN-MCR+** and **e-DE** found optimal solution for most problems. Moreover, they found good solutions in most executions, as indicated by their *SuccRate*.

Considering the problems that present constraints with different orders of magnitude, such as G10 and G21 problems, **SADE-kNN-MCR+** presents an enhanced performance. For function G10, besides finding the optimal solution in all runs, it also requires the smaller number of function evaluations.

Table 9 summarizes the main results of Table 8, collecting the number of functions for which each method was able to find the optimal solutions in at least one run, and required the lower number of function evaluations to find the optimal solution.

**Table 9. Summary of some performance measures**

| N. of functions where: | SADE-kNN-MCR+ | SA-DECV | SA-DECV-SR | ASRES | e-DE |
|---|---|---|---|---|---|
| Found optimal solution in at least one run | 21 | 12 | 15 | 16 | **22** |
| Required lower number of function evaluations | **14** | 0 | 2 | 2 | 4 |

A more comprehensive comparison of the performance of all methods will be presented in terms of nonparametric statistical tests previously described in section 4.2: the sign test, and the performance profiles.

### *6.1.2 Sign Test*

Table 10 presents the results of the Sign Test based on the pairwise comparison between **SADE-kNN-MCR+** and each other method in terms of the average number of function evaluations (*succ_nfe*). The number of wins and losses confirms the observations based on Table 8, i.e., the **SADE-kNN-MCR+** outperforms all other methods. The statistical assessment in terms of the *p*-values indicates that all of them are below the threshold, which confirms that the superiority of the **SADE-kNN-MCR+** is statistically significant.

**Table 10. Sign Test, Pairwise Comparisons**

| SADE-kNN-MCR+ **versus** | SA-DECV | SA-DECV-SR | ASRES | e-DE |
|---|---|---|---|---|
| # of wins / # of losses<br>(p-value) | 20 / 1<br>(0.00001) | 19 / 3<br>(0.0004) | 18 / 3<br>(0.0007) | 17 / 5<br>(0.0084) |

Regarding the success rate (*succRate*), both **SADE-kNN-MCR+** and **e-DE** reach 100% of optimal executions in 16 problems. **e-DE** gets more optimal runs in 4 out of the remaining 6. However, this leads to a p-value of 0.3437, not small enough to indicate that there is a statistically significant difference between these two methods regarding this particular metric.

### *6.1.3 Performance Profiles*

The *succ_nfe* values were also used as the metric for the generation of the performance profiles shown in Figure 5, according to the procedure described in Section 4.2.3 . A global indication of the performance of the methods may also be observed in the bar chart of Figure 6 where the y axis indicates the areas under the PP curves.

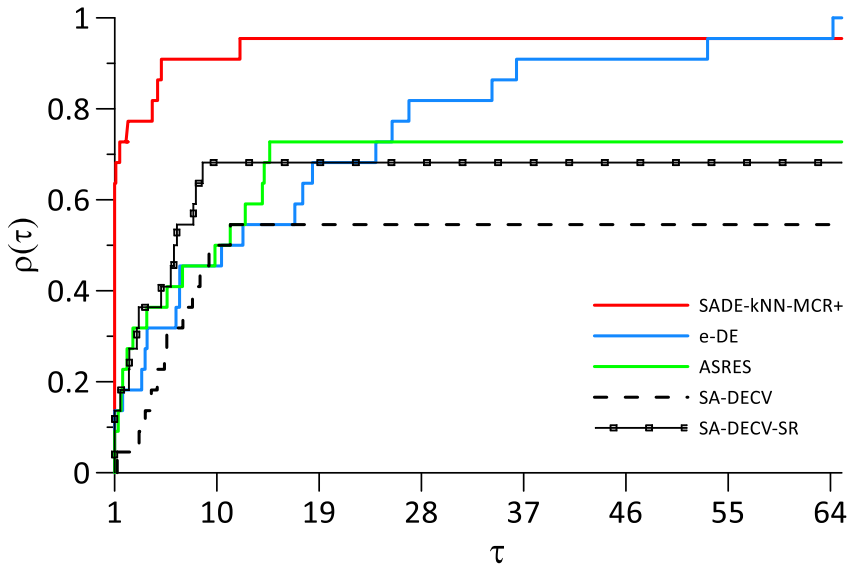


**Figure 5. Performance Profiles**

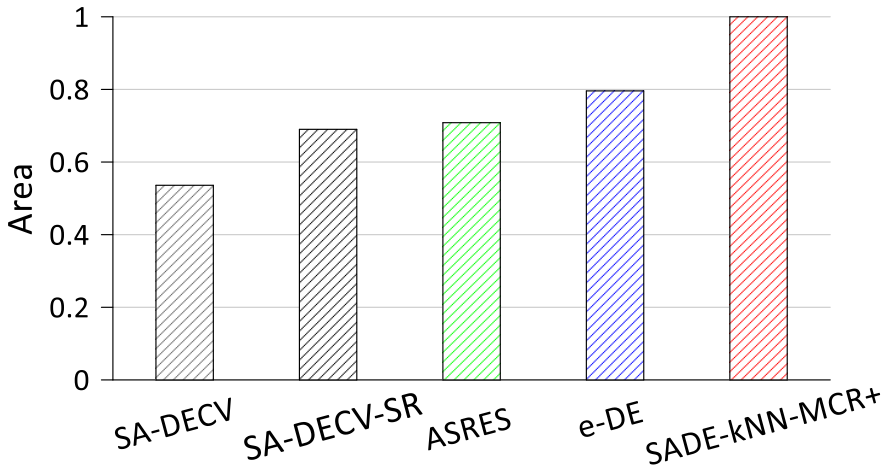


**Figure 6. Global performance of the methods**

In Figure 5 it is interesting to observe that, for smaller values of τ, the curve corrresponding to the **e-DE** method is located below those for the **SA-DECV-SR** and **ASRES** algorithms; however, since **e-DE** is able to find feasible solutions for more problems than **SA-DECV**, **SA-DECV-SR** and **ASRES**, it ends up with the second larger area in the bar chart of Figure 6. The overall best performer is **SADE-kNN-MCR+**, since its curve approaches $\rho(\tau) = 1$ with the lowest $\tau$ value and presents the larger area in the bar chart of Figure 6, reflecting the fact that it requires fewer evaluations for the greater number of problems.

### *6.2 Suite of structural engineering problems*

Finally, this section compares the results for some of the benchmark engineering problems shown in Table 2. Here the proposed **SADE-kNN-MCR+** method is compared only with **SA-DECV-SR**, since results for the other methods could not be found in the literature. Anyway, this is a useful comparison since both methods share the following characteristics: 1) **DE** is used as the search algorithm; 2) Some of the solutions are approximated by a similarity-based surrogate model; and 3) Rank-based constraint handling techniques are employed.

For these comparisons, the maximum number of function evaluations is set to 30,000 function evaluations for the Welded Beam problem; 20,000 for the Pressure Vessel; and 5,000 for the Tension/Compression Spring problem.

Table 11 compares the statistical analysis of the results for the three benchmark problems considering 25 runs: the best, average and worst solutions amongst all runs; and the average number of function evaluations considering the successful runs. Unlike section 5.2, where optimal reference values were not used as a stopping criterion, the results presented in parentheses were considered as optimal in order to compare the average number of function evaluations that each algorithm required to reach it.

**Table 11. Results for the suite of engineering problems**

| Prob. | Algorithm | Best | Average | Worst | Succ_nfe |
|---|---|---|---|---|---|
| Welded Beam (1.7248523) | **SADE-kNN-MCR+** | **1.7248523** | **1.7248523** | **1.7248523** | **3,983.0** |
| | **SA-DECV-SR** | 1.72486 | 1.72496 | 1.72595 | 6,971 |
| Pressure Vessel (6,059.71433) | **SADE-kNN-MCR+** | **6,059.71436** | **6,062.20310** | **6,090.52622** | 6,719.7 |
| | **SA-DECV-SR** | **6,059.7** | 6,296.0336 | 6,820.44 | **6,139** |
| Tension/Compression Spring (0.01266523) | **SADE-kNN-MCR+** | 0.012732 | **0.012765** | **0.012902** | **1,409.4** |
| | **SA-DECV-SR** | **0.0127127** | 0.013047148 | 0.0143943 | 1,617 |

For the Welded Beam problem, **SADE-kNN-MCR+** provided the better solution as indicated in the "***best***" column of Table 11. It also obtained this optimal solution in all runs, as indicated by the worst value, which is equal to the best one. Also, its average number of function evaluations (3,983.0) is nearly half of the value required by the **SA-DECV-SR**.

Considering the Pressure Vessel problem, **SADE-kNN-MCR+** also provided better solutions regarding all statistical measures. It obtained the accuracy required for the optimal solution in 20 out of the 25 runs, employing an average number of 6,719.7 function evaluations. In [56] it is not clear whether the value of 6,139 evaluations refers to the number of function evaluations considering only the optimal runs; anyway, it can be concluded that **SADE-kNN-MCR+** is the most efficient algorithm in obtaining the best solutions considering all runs.

Finally, for the Tension/Compression Spring problem, **SADE-kNN-MCR+** provided better results regarding all measures (average, worst, and number of function evaluations), except for the *best* solution. The best solution provided by **SADE-kNN-MCR+** (0.012732) reached the first termination criterion ($f(x^*) - f(x) < 0.0001$), thus stopping the evolution process.

Table 12 presents the values of the variables corresponding to the optimal solutions provided by **SADE-kNN-MCR+ .** Again, the number of problems is low, thus the sign test cannot be adequately employed. However, it may be illustrative to compare the performance profiles for the **SADE-kNN-MCR+** and **SA-DECV_SR** algorithms. Figures 8 and 9 show respectively the individual and global performance using as metric the *succ_nfe* values presented in Table 11. Those results show that **SADE-kNN-MCR+** is the best performer, since not only it gives better results for two problems ($\rho(1) = 0.666$), but still finds optimal executions in all of them (there is $\tau$ such that $\rho(\tau) = 1$).

**Table 12. Values of the variables corresponding to the optimal solutions provided by SADE-kNN-MCR+**

| Welded Beam | | Pressure Vessel | | Tens./Compr. Spring | |
|---|---|---|---|---|---|
| H | 0.205730 | $T_s$ | 0.8125 | d | 0.050016 |
| L | 3.470489 | $T_h$ | 0.4375 | D | 0.317587 |
| T | 9.036624 | R | 42.098446 | N | 14.025337 |
| B | 0.205730 | L | 176.636597 | | |
| **f(x*)** | **1.724852** | | **f(x*) = 6,059.714361** | | **f(x*) = 0.012732** |

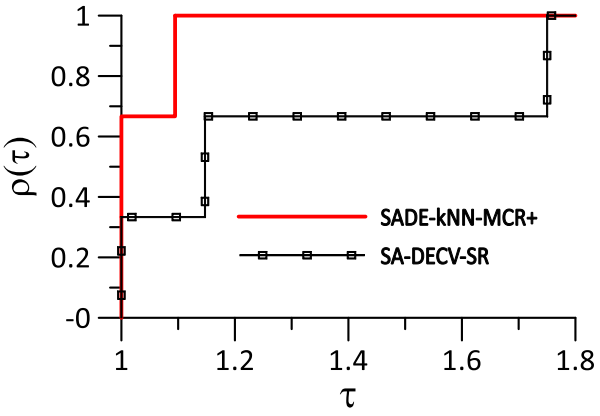


**Figure 7. PPs comparing SADE-kNN-MCR+ and SA-DECV-SR**

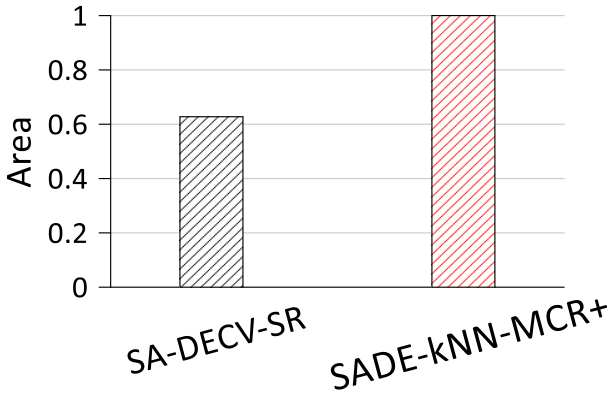


**Figure 8. Global performance of SADE-kNN-MCR+ and SA-DECV-SR**

## 7 Conclusions and extensions

This work presented a new merit-based surrogate-assisted evolutionary method designed for applications to complex real-world constrained optimization problems. Besides presenting a very good accuracy in finding optimal solutions, the main goal of the method is to reduce the number of objective function evaluations.

Those goals of accuracy and computational cost savings are pursued by the following approaches: **a)** employing the Multiple Constraint Ranking (**MCR**) technique to deal with the large number of constraints with different orders of magnitude and/or different units that are common in the aforementioned complex engineering problems; **b)** devising an enhanced approximation model based on the *k*-nearest-neighbor (***k*-NN**) algorithm associated to a merit-based database management procedure; and **c)** implementing a strategy that triggers a random reinitialization of the population whenever it stagnates on local optima. These approaches have the potential to present significant cost savings by reducing the number of objective function evaluations: the **MCR** does not require evaluations when there is no feasible solution in the population, and the merit scheme of the **k-NN** model evaluates only part of the population.

Two variants of the proposed method (**SADE-kNN-MCR** and **SADE-kNN-MCR+**) were applied to a set of 24 constrained benchmark problems and 5 mechanical engineering problems. Based on direct comparisons of different performance measures; nonparametric statistical sign tests; and performance profile graphs, the results were compared with those provided by other algorithms recently presented in the literature that also employ surrogate models and constraint handling techniques (**SADE-kNN, ASRES**, **SA-DECV, SA-DECV_SR** and **e-DE**). The comparisons have shown that both variants of the proposed method presented superior accuracy and drastically reduced the number of function evaluations. The best performer is **SADE-kNN-MCR+** since in several executions it was able to escape from local optima.

This method may be particularly suited for real-life engineering problems such as, for instance, the design of offshore systems that have been addressed in [58,59,1-3]. Those are

complex engineering problems defined on high-dimensional spaces, with many design variables, nonlinear objective and constraint functions (the latter comprised by constraints with different orders of magnitude and/or different units), and requiring evaluations by expensive Finite Element nonlinear dynamic analyses; thus, reducing the number of evaluations is a critical issue. The potential of the **SADE-kNN-MCR+** method to solve such practical engineering problems is indicated by its performance in the Welded Beam and Pressure Vessel problems, which are also characterized by severely nonlinear constraint functions with diverse magnitudes. Thus, extensions of this work will incorporate this method into optimal design procedures for mooring systems, submarine pipelines and risers that have been considered in [1,3,60]. It is expected that the resulting procedures will lead to increased efficiency in terms of accuracy, computational costs, and more importantly to better engineering design solutions.

## Declarations

### *Funding*

The authors acknowledge the support of the Brazilian funding agencies CNPq (grant numbers 306069/2016, 306186-2017, 308873/2017-3); FAPERJ (grant numbers E-26/200.314/2016, E-26/202.767/2017); and CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), finance code 001.

### *Conflicts of interests*

The authors declare that they have no conflict of interest.

### *Contributions*

The authors contributed to each part of this paper equally. The authors read and approved the final manuscript.

### *Ethical approval*

This paper does not contain any studies with human participants or animals performed by any of the authors.

## References

1. Vieira IN, de Lima BSLP, Jacob BP (2012) Bio-inspired algorithms for the optimization of offshore oil production systems. International Journal for Numerical Methods in Engineering 91 (10):1023-1044. doi:https://doi.org/10.1002/nme.4301

2. de Lucena RR, Baioco JS, de Lima BSLP, Albrecht CH, Jacob BP (2014) Optimal design of submarine pipeline routes by genetic algorithm with different constraint handling techniques. Advances in Engineering Software 76:110-124. doi:https://doi.org/10.1016/j.advengsoft.2014.06.003

3. Monteiro BdF, de Pina AA, Baioco JS, Albrecht CH, de Lima BSLP, Jacob BP (2016) Toward a methodology for the optimal design of mooring systems for floating offshore platforms using evolutionary algorithms. Marine Systems & Ocean Technology 11:55-67. doi:https://doi.org/10.1007/s40868-016-0017-8

4. Queipo NV, Haftka RT, Shyy W, Goel T, Vaidyanathan R, Tucker PK (2005) Surrogate-based analysis and optimization. Paper presented at the Progress in aerospace sciences,

5. Saunders C, Gammerman A, Vovk V (1998) Ridge regression learning algorithm in dual variables.

6. Hendrickx W, Gorissen D, Dhaene T Grid enabled sequential design and adaptive metamodeling. In: Simulation Conference, 2006. WSC 06. Proceedings of the Winter, 2006. IEEE, pp 872-881

7. Kybic J, Blu T, Unser M (2002) Generalized sampling: a variational approach. I. Theory. IEEE Transactions on Signal Processing 50:1965-1976

8. Kybic J, Blu T, Unser M (2002) Generalized sampling: a variational approach. II. Applications. IEEE Transactions on Signal Processing 8:1977-1985

9. Mullur AA, Messac A (2006) Metamodeling using extended radial basis functions: a comparative approach. Engineering with Computers:203

10. Lowe D (1988) Multi-variable functional interpolation and adaptive networks. Complex Systems:321-355

11. Myers RH, Montgomery DC, Anderson-Cook CM (2016) Response surface methodology: process and product optimization using designed experiments. John Wiley & Sons,

12. Ferrari S, Stengel RF (2005) Smooth function approximation using neural networks. IEEE Transactions on Neural Networks 16:24-38

13. Emmerich MT, Giannakoglou KC, Naujoks B (2006) Single-and multiobjective evolutionary optimization assisted by gaussian random field metamodels. IEEE Transactions on Evolutionary Computation 10

14. Ulmer H, Streichert F, Zell A Evolution strategies assisted by Gaussian processes with improved preselection criterion. In: Evolutionary Computation, 2003. CEC'03. The 2003 Congress on, 2003. IEEE, pp 692-699

15. Simpson TW, Mauery TM, Korte JJ, Mistree F (2001) Kriging models for global approximation in simulation-based multidisciplinary design optimization. AIAA journal 39:2233-2241

16. Kecman V (2005) Support vector machines--an introduction. Support vector machines: theory and applications:605-605

17. Krempser E, Bernardino HS, Barbosa HJ, Lemonge AC Differential evolution assisted by surrogate models for structural optimization problems. In: Proceedings of the international conference on computational structures technology (CST), 2012.

18. Krempser E, Bernardino HS, Barbosa HJ, Lemonge AC (2017) Performance evaluation of local surrogate models in differential evolution-based optimum design of truss structures. Engineering Computations 2 (34):499-547

19. Fonseca LG, Barbosa HJC, Lemonge ACC (2009) A similarity-based surrogate model for enhanced performance in genetic algorithms. Opsearch 46 (1):89-107

20. Liu Y, Sun F (2011) A fast differential evolution algorithm using k-Nearest Neighbour predictor. Expert Systems with Applications 38(4):4254-4258

21. Miranda-Varela M-E, Mezura-Montes E (2016) Surrogate-assisted Differential Evolution with an Adaptive Evolution Control Based on Feasibility to Solve Constrained Optimization Problems. Paper presented at the Fifth International Conference on Soft Computing for Problem Solving,

22. Grefenstette JJ, Fitzpatrick JM Genetic search with approximate fitness evaluations. In: Proc. Of the Intl. Conf. on Genetic Algorithms and Their Applications, 1985. pp 112-120

23. Ratle A Optimal sampling strategies for learning a fitness model. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), 1999. IEEE, pp 2078-2085

24. Hong YS, Lee H, Tahk MJ (2003) Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks. Engineering Optimization 35 (1):91-102

25. Sun C, Jin Y, Zeng J, Yu Y (2014) A two-layer surrogate-assisted particle swarm optimization algorithm. Soft Computing 19 (6):1461-1475. doi:10.1007/s00500-014-1283-z

26. Anderson KS, Hsu Y Genetic crossover strategy using an approximation concept. In: In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), 1999. IEEE, pp 527-533

27. Rasheed K, Vattam S (2002) Comparison of methods for using reduced models to speed up design optimization. Paper presented at the 4th Annual Conference on Genetic and Evolutionary Computation,

28. de Pina AC, Albrecht CH, de Lima BSLP, Jacob BP (2014) Wavelet network meta-models for the analysis of slender offshore structures. Engineering Structures 68:71-84. doi:https://doi.org/10.1016/j.engstruct.2014.02.039

29. de Pina AC, Monteiro BdF, Albrecht CH, de Lima BSLP, Jacob BP (2014) ANN and wavelet network meta-models for the coupled analysis of floating production systems. Applied Ocean Research 48:21--32. doi:https://doi.org/10.1016/j.apor.2014.07.009

30. Storn R, Price K (1997) Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11:341-359

31. Wang YS, Shi YJ, Yue BX, Teng HF An efficient differential evolution algorithm with approximate fitness functions using neural networks. In: International Conference on Artificial Intelligence and Computational Intelligence, Heidelberg, Berlin., 2010. Springer, pp 334-341

32. Pahner U, Hameyer K Adaptive coupling of differential evolution and multiquadrics approximation for the tuning of the optimization process. In: IEEE Transactions on magnetics, 2000. pp 1047-1051.

33. Mallipeddi R, Lee M (2015) An evolving surrogate model-based differential evolution algorithm. Applied Soft Computing 34:770-787. doi:10.1016/j.asoc.2015.06.010

34. Jin C, Qin AK, Tang K (2015) Local ensemble surrogate assisted crowding differential evolution. In 2015 IEEE Congress on Evolutionary Computation (CEC). IEEE,

35. Liu B, Koziel S, Zhang Q (2016) A multi-fidelity surrogate-model-assisted evolutionary algorithm for computationally expensive optimization problems. Journal of computational science, vol 12.

36. Awad NH, Ali MZ, Mallipeddi R, Suganthan PN (2018) An improved differential evolution algorithm using efficient adapted surrogate model for numerical optimization. Information Sciences 451-452:326-347. doi:10.1016/j.ins.2018.04.024

37. Yang Z, Qiu H, Gao L, Jiang C, Zhang J (2019) Two-layer adaptive surrogate-assisted evolutionary algorithm for high-dimensional computationally expensive problems. Journal of Global Optimization 74 (2):327-359. doi:10.1007/s10898-019-00759-0

38. Garcia RdP, de Lima BSLP, Lemonge ACdC (2017) A Surrogate Assisted Differential Evolution to Solve Constrained Optimization Problems. Paper presented at the IEEE Latin American Conference on Computational Intelligence (LA-CCI), Arequipa, Peru,

39. Garcia RdP, de Lima BSLP, Lemonge ACdC, Jacob BP (2017) A rank-based constraint handling technique for engineering design optimization problems solved by genetic algorithms. Computers and Structures 187:77-87. doi:https://doi.org/10.1016/j.compstruc.2017.03.023

40. Lampinen J, Zelinka I On stagnation of the differential evolution algorithm. In: Proceedings of MENDEL, 2000. pp 76-83

41. Hrstka O, Kučerová A (2004) Improvements of real coded genetic algorithms based on differential operators preventing premature convergence. Advances in Engineering Software 35(3-4):237-246

42. Liang JJ, Runarsson TP, Mezura-Montes E, Clerc M, Suganthan PN, Coello CAC, Deb K (2006) Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. Journal of Applied Mechanics 41(8):8-31

43. Storn R System design by constraint adaptation and differential evolution. In: IEEE Transactions on Evolutionary Computation, 1999. pp 22-34

44. Bergamaschi PR, Saramago SDFP, dos Santos Coelho L (2008) Comparative study of SQP and metaheuristics for robotic manipulator design. Applied numerical mathematics 58(9):1396-1412

45. Das S, Suganthan PN (2011) Differential evolution: A survey of the state-of-the-art. Paper presented at the IEEE transactions on evolutionary computation,

46. Wu G, al. e (2018) Ensemble of differential evolution variants. Information Sciences:172-186

47. Kenneth V (1999) Price, An introduction to differential evolution, New ideas in optimization. McGraw-Hill Ltd., UK, Maidenhead, UK,

48. Price K, Storn R, M, Lampinen JA (2006) Differential evolution: a practical approach to global optimization. Springer Science \& Business Media,

49. Shepard D A two-dimensional interpolation function for irregularly-spaced data. In: ACM (ed) Proceedings of the 1968 23rd ACM national conference, 1968. pp 517-524

50. Hu H, Lee DL (78-91) Range nearest-neighbor query. IEEE Transactions on Knowledge and Data Engineering 18

51. Deb K (2000) An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering 186:311-338

52. Runarsson T, Yao X (2000) Stochastic ranking for constrained evolutionary optimization. IEEE Trans Evol Comput 4:284–294

53. Runarsson TP, Yao X Continuous selection and self-adaptive evolution strategies. In: Proceedings of the 2002 Congress on Evolutionary Computation - CEC'02, 2002. pp 279-284

54. Ho PY, Shimizu K (2007) Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme. Information Sciences 177 (14):2985-3004. doi:10.1016/j.ins.2007.01.011

55. Runarsson TP (2006) Approximate Evolution Strategy using Stochastic Ranking. Paper presented at the 2006 IEEE International Conference on Evolutionary Computation, Vancouver, BC, Canada, July 16-21

56. Miranda-Varela M-E, Mezura-Montes E (2018) Constraint-handling techniques in surrogate-assisted evolutionary optimization. An empirical study. Applied Soft Computing 73:215-229. doi:10.1016/j.asoc.2018.08.016

57. Yu X, Lu Y, Wang X, Luo X, Cai M (2019) An effective improved differential evolution algorithm to solve constrained optimization problems. Soft Computing 23 (7):2409-2427. doi:https://doi.org/10.1007/s00500-017-2936-5

58. de Lima BSLP, Jacob BP, Ebecken NFF (2005) A hybrid fuzzy/genetic algorithm for the design of offshore oil production risers. International Journal for Numerical Methods in Engineering 64 (11):1459-1482. doi:https://doi.org/10.1002/nme.1416

59. de Pina AA, Albrecht CH, de Lima BSLP, Jacob BP (2011) Tailoring the particle swarm optimization algorithm for the design of offshore oil production risers. Optimization and Engineering 12 (1-2):215-235. doi:https://doi.org/10.1007/s11081-009-9103-5

60. Baioco JS, de Lima Jr. MHA, Albrecht CH, de Lima BSLP, Jacob BP, Rocha DM (2018) Optimal Design of Submarine Pipelines by a Genetic Algorithm with Embedded On-Bottom Stability Criteria. Mathematical Problems in Engineering 2018:1-21. doi:https://doi.org/10.1155/2018/1781758

61. Regis RG (2014) Evolutionary programming for high-dimensional constrained expensive black-box optimization using radial basis functions. IEEE Transactions on Evolutionary Computation 18:326-347

62. Regis RG, Shoemaker CA (2007) A stochastic radial basis function method for the global optimization of expensive functions. INFORMS Journal on Computing 19:497-509

63. Elsayed SM, Ray T, Sarker RA A surrogate-assisted differential evolution algorithm with dynamic parameters selection for solving expensive optimization problems. In: Evolutionary Computation (CEC), 2014 IEEE Congress on, 2014. IEEE, pp 1062-1068

64. da Silva AF (2016) Optimization of constrained problems using Particle Swarm algorithm aided by surrogate models [in Portuguese]. Doctoral thesis - Federal University of Rio de Janeiro. Advisors: de Lima BSLP, Lemonge ACdC, Rio de Janeiro

65. Gandomi AH, Yang X-S (2011) Benchmark Problems in Structural Optimization. In: Koziel S, Yang X-S (eds) Comput. Optimization, Methods and Algorithms. Springer-Verlag, Berlin Heidelberg, pp 259–281

66. Sandgren E (1988) Nonlinear integer and discrete programming in mechanical design. Paper presented at the ASME Design Technology Conference, Kissimee, FL,

67. Deb K (1991) Optimal design of a welded beam via genetic algorithms. AIAA Journal 29 (11):2013-2015

68. Erbatur F, Hasançebi O, Tütüncü I, Kılıç H (2000) Optimal design of planar and space structures with genetic algorithms. Computers & Structures 75(2):209-224

69. Gibbons JD, Chakraborti S (2003) Nonparametric Statistical Inference. Statistics: a Series of Textbooks and Monographs, Fourth edn. Marcel Dekker, Inc., New York - Basel

70. Fisher RA Theory of statistical estimation. In: Mathematical Proceedings of the Cambridge Philosophical Society, 1925. Cambridge University Press,

71. Corder GW, Foreman DI (2009) Nonparametric statistics: an introduction. In: Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach. John Wiley & Sons, Hoboken, NJ, USA, pp 101-111.

72. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation 1 (1):3-18. doi:10.1016/j.swevo.2011.02.002

73. Lynn N, Suganthan PN (2015) Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation. Swarm and Evolutionary Computation 24:11-24. doi:10.1016/j.swevo.2015.05.002

74. Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. Mathematical Programming 91 (2):201-213. doi:10.1007/s101070100263