An enhanced surrogate-assisted differential evolution for constrained optimization problems

Rafael de Paula Garcia1,3 · Beatriz Souza Leite Pires de Lima1 · Afonso Celso de Castro Lemonge2 · Breno Pinheiro Jacob1

Accepted: 13 January 2023 / Published online: 3 February 2023
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2023

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 and gas production systems such as floating platforms, risers, mooring systems and submarine pipelines (Vieira et al. 2012; Lucena et al. 2014; Monteiro et al. 2016): those problems are characterized by many design variables, defined on high-dimensional spaces, with objectives and constraints that are generally nonlinear 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) (Queipo et al. 2005) 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 (Saunders et al. 1998; Hendrickx and Gortissen 2006); radial basis functions (Kybic et al. 2002a, 2002b; Mullur and Messac 2006; Lowe 1988); response surface models (Myers et al. 2016); artificial neural networks (Ferrari and Saucier 1998; Hendrickx and Gorissen 2006); radial basis functions (Kybic et al. 2002a, 2002b; Mullur and Messac 2006; Lowe 1988); and similarity-based models such as the nearest-neighbor (NN) algorithms (Krempser et al. 2012, 2017; Fonseca et al. 2009). Particularly, the k-NN model presents many advantages in terms of good performance and ease of implementation, as reported in Fonseca et al. (2009); Liu and Sun (2011; Miranda-Varela and Mezura-Montes 2016).

Many applications of surrogate models with EAs have been presented in the literature. In (Grefenstette and Fitzpatrick 1985) a polynomial model was used to approximate the objective function. In (Ratle and Optimal sampling strategies for learning a fitness model. 1999) 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 Hong et al. (2003) where the fitness function has been approximated by an online multi-layer neural network, and in Sun et al. (2014) that presented a surrogate-assisted PSO algorithm. Surrogate models have also been employed in association with the genetic crossover operator (Anderson and Hsu 1999), to approximate the objective function (Rasheed and Vattam 2002), and to calculate approximations for the sum of violation terms (Miranda-Varela and Mezura-Montes 2016). 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 Pina et al. 2014a; Pina et al. 2014b) to avoid expensive nonlinear dynamic Finite Element analyses.

Among evolutionary algorithms, the differential evolution (DE) algorithm (Storn and Price 1997) 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 (Wang et al. 2010) and neural networks of radial basis functions (Pahner and Hameyer 2000). In (Mallipeddi and Lee 2015) a simple kriging metamodel was employed to comprise an “evolving surrogate model-based differential evolution” (ESMDE). As the population evolves over generations, the surrogate model also evolves using the appropriate parameter setting during different stages of the evolution, thus assisting the DE in generating competitive offspring. A surrogate model for DE consisting of multiple local surrogate models was presented in Jin et al. (2015), built from the feedback of the evolutionary process in diverse overlapped local regions of the search space. In (Liu et al. 2016) 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 Awad et al. (2018), using an adaptation scheme to adapt parameters of a Kriging model. Recently, a two-layer adaptive surrogate-assisted DE has been proposed in Yang et al. (2019), 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 Krempser et al. (2012) and (Krempser et al. 2017) to solve complex structural optimization problems; the results indicated that the performance of DE has been significantly improved. In this context, in Garcia et al. 2017a 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, intended to comprise a more efficient optimization tool to solve complex real-world engineering problems. The proposed method is comprised by two variants, referred to as SADE-kNN-MCR and SADE-kNN-MCR+. Both incorporate 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. Compared to the authors’ previous work (Garcia et al. 2017a), the proposed method introduces novel characteristics, beginning with its association with the multiple constraint ranking (MCR) constraint-handling technique (Garcia et al. 2017b). 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). Moreover, results of preliminary studies indicated that the previous SADE-kNN method presented by the authors in Garcia et al. 2017a 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 (Lampinen and Zelinka 2000; Hrstka and Kučerová 2004). To circumvent this issue, the variant called SADE-kNN-MCR+ incorporates a strategy that triggers a random reinitialization or “rebirth” of the population whenever it stagnates on local optima.

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 (Liang et al. 2006), and mechanical engineering problems), and assessing their relative performance by direct comparisons and nonparametric statistical tests.

This paper is organized as follows: Initially, Sect. 2 summarizes the differential evolution algorithm, the k-NN similarity-based metamodel, and a review of some surrogate-assisted methods for constrained problems presented in the literature. Section 3 describes the variants of the proposed surrogate-assisted differential evolution method (SADE-kNN-MCR and SADEX-kNN-MCR+). The full sets of numerical experiments are described in Sect. 4. The results are presented in Sects. 5 and 6. The former compares both variants of the proposed method with the original SADEx-kNN previously presented by the authors in Garcia et al. 2017a. Then, Sect. 4 presents comparisons with other methods recently presented in the literature that are listed in Sect. 0. Final remarks and conclusions are presented in Sect. 7.

2 Related works

2.1 Differential evolution

The DE algorithm, proposed by Storn and Price (Storn and Price 1997), has been shown to be a very effective algorithm to solve complex real-world optimization problems such as system design (Storn and System design by constraint adaptation and differential evolution. 1999) and robot manipulator design (Bergamaschi et al. 2008). One of the main characteristics of DE is its simplicity and ease of implementation. A detailed review of its basic concepts is provided by Das and Suganthan (2011); Wu et al. 2018), 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 were proposed in Kenneth (1999); Price et al. 2006), characterized by the procedures adopted for the selection of candidate solutions, and types of recombination. In this work the first variant (DE/rand/1/bin) will be adopted. Initially, vectors \( x_i \) (\( i = 1, \ldots, N \)) that comprise the individuals of the population are randomly created. Then, the mutation operators defined by the following expression is applied:

\[
 u_i = x_{r_1} + F(x_{r_2} - x_{r_3})
\]  

(2.1)

This operator provides the “mutant vector” \( u_i \), given by the perturbation of a “parent” vector \( x_{r_1} \) with the difference between two other vectors \( x_{r_2} \) and \( x_{r_3} \) (all randomly selected from the population such that \( x_{r_1} \neq x_{r_2} \neq x_{r_3} \neq x_i \)), affected by the parameter \( F \) that is a user-defined scale factor.

A crossover operator is also defined, to generate trial vectors \( v_i \) that combine the solutions in the current population \( x_i \) and the mutant solutions \( u_i \) generated by Eq. 2.1. Considering that \( D \) is the dimension of each vector, and taking a user-defined crossover rate \( CR \in [0,1] \), this operator is expressed as:

\[
 v_{i,d} = \begin{cases} u_{i,d} & \text{if } \text{rand}(d) \leq CR \text{ or } d = \text{drand} \\ x_{i,d} & \text{if } \text{rand}(d) > CR \text{ and } d \neq \text{drand} \end{cases}
\]  

(2.2)

where \( d = 1,2,\ldots,D; \text{rand}(d) \) is a random number between 0 and 1; and \( \text{drand} \) is an integer value randomly selected in the range \( [1,D] \) to ensure that at least one variable of the mutant vector \( u_i \) will be copied into the trial vector \( v_i \). After crossover, a greedy selection is performed by comparing the objective function values of solutions \( v_i \); among \( x_i \); only the best one is maintained in the population.

2.2 The k-NN similarity-based metamodel

Similarity metamodels evaluate a solution by calculating the average of the objective function of the nearest solutions weighted by its distances. This technique can be classified in two categories, according to the criterion that defines the candidate solutions able for selection:

- \( k \)-nearest neighbors (k-NN) (Shepard 1968) that consider the \( k \) nearest candidate solutions;
- \( r \)-nearest neighbors (r-NN) (Hu and Lee 2006) that consider a given neighborhood controlled by a range \( r \).

In the k-NN similarity-based metamodel, which is adopted in this paper, the solutions \( (x_i, f(x_i)) \) evaluated by the exact function are stored in a database as an ordered set \( D \). Any solution \( x \) generated during the evolution is evaluated by the following weighed average:
where \( \text{dist}(x, x_a) \) is the Euclidean distance between \( x \) and \( x_a \), \( x_a \) is a solution stored in the database \( D \), and \( k \) is the number of neighbor solutions.

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 1.1 will present some considerations regarding the database management.

As mentioned in Sect. 1, various kinds of surrogate models have been employed in association with EAs. Following the previous SADE-kNN method presented in Garcia et al. 2017a, in this work we will continue using the \( k \)-NN approximation to comprise the improved SADE-kNN-MCR and SADE-kNN-MCR + methods. This is due to several reasons: initially, because this model has been widely recognized in the literature for its simplicity and good performance (Fonseca et al. 2009; Liu and Sun 2011; Miranda-Varela and Mezura-Montes 2016). Then, because it is particularly suited to the association with the merit-based database management that will be described shortly in Sect. 1.1: while standard \( k \)-NN implementations require additional computation costs to assemble the database and perform the neighborhood comparisons, the adopted database management procedure keeps the database size constant, storing only the solutions that provide most accurate approximations.

This helps to maintain an accurate balance between accuracy and computational efficiency. Finally, another reason to use the \( k \)-NN model is that it is also particularly suited to the “rebirth” technique that will be described in Sect. 1.2 to avoid stagnation around local optima. It will be seen that this association enhances the main feature of the \( k \)-NN: the use of similarity for approximations, to benefit solutions more similar to the current population, thus increasing the accuracy of the model.

\[ f(x) = \frac{\sum_{a=1}^{k} \text{dist}(x, x_a)^2 f(x_a)}{\sum_{a=1}^{k} \text{dist}(x, x_a)^2} \quad (2.3) \]

where \( x = (x_1, x_2, \ldots, x_D) \) is the vector of design variables that characterize a given candidate solution, with components \( x_j \) presenting lower and upper bounds \([lb_d, ub_d]\). The goal is to minimize the objective function \( f(x) \), considering inequality and equality constraints (respectively \( g_j(x) \leq 0 \) and \( h_j(x) = 0 \)) that define the feasible region.

Many classical constraint-handling techniques (CHTs) transform this constrained problem into an unconstrained one. For this purpose, whenever any given constraint is violated a penalty function \( p(x) \) is added to the objective function \( f(x) \); thus, the objective function and constraints are combined into a single value. Another approach is to maintain those values apart along the evolution. Deb (Deb 2000) 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 rank-based approach, such as the Stochastic Ranking (SR) method (Runarsson and Yao 2000), the GCR (Global Competitive Ranking) method (Runarsson 2002), and the Ho and Shimizu ranking method (HSR) (Ho and Shimizu 2007).

### 2.3.2 Brief review of some methods

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 (Runarsson 2006).

The SADE-kNN method presented in Garcia et al. 2017a employed the binary tournament selection technique proposed by Deb (Deb 2000). 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 (Miranda-Varela and Mezura-Montes 2016) and SA-DECV_SR (Miranda-Varela and Mezura-Montes 2018). 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 Deb (2000); the latter work (Miranda-Varela and Mezura-Montes 2018), after comparing different CHTs associated to the same surrogate model, concludes that the best performance is provided by the stochastic ranking (SR) method (Runarsson and Yao 2000).
Recently, Yu et al. 2019 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 Description of the proposed method

This section describes the main characteristics of the new surrogate-assisted evolutionary method proposed in this work. The method is comprised by two variants, referred to as \textit{SADE-kNN-MCR} and \textit{SADE-kNN-MCR +}. Those characteristics are related to the following aspects:

Both variants include the multiple constraint ranking (MCR) technique, described in Section 3.1, comprising a specialized CHT to deal with the large number of complex constraints that may be found in real-world engineering problems.

Both variants also include the merit-based strategy described in Sect. 1.1 to manage the database that stores the exact solutions employed by the \textit{k-NN} surrogate model.

Finally, the variant \textit{SADE-kNN-MCR +} adds the strategy described in Sect. 1.2 that triggers a random reinitialization or “rebirth” of the population whenever it stagnates on local optima.

#### 3.1 The multiple constraint ranking technique (MCR)

Recently, focusing on the solution of complex real-world engineering optimization problems, we had introduced the multiple constraint ranking technique (MCR) (Garcia et al. 2017b) as a CHT working in association with a genetic algorithm. Now, in this work we will derive an association of MCR with the DE algorithm. As mentioned before, this is one of the main novelties of the variants of the proposed method (\textit{SADE-kNN-MCR} and \textit{SADE-kNN-MCR +}) compared to the original \textit{SADE-kNN} method described in Sect. 2.3.2 that employed as a CHT the pairwise tournament method of Deb (2000).

Section 2.3.1 have described some CHTs that follows a rank-based approach, including HSR (Ho and Shimizu 2007) that employed a single rank $R_{\phi}$ for all violation values (built as a single sum considering violations in all constraints).

MCR extends this approach by splitting $R_{\phi}$ into several ranks $R_{j\phi}$, one for each constraint $j = 1, \ldots, p$, and builds multiple queues considering the objective function values and the violations of each constraint. In MCR, the fitness function $F$ for each solution $x$ is evaluated as follows:

$$
F(x) = \begin{cases} 
R_{N_v} + \sum_{j=1}^{p} R_{j\phi}, & \text{if only infeasible solutions} \\
R_{f} + R_{N_v} + \sum_{j=1}^{p} R_{j\phi}, & \text{otherwise}
\end{cases}
$$

(3.1)

where $R_{f}$ ranks the value of the objective function $f$, and $R_{N_v}$ ranks the number of constraints violated.

Algorithm 1 presents the pseudocode of MCR. As seen in row 1, firstly $N$ individuals of the population are randomly created. Then, all individuals have their objective function and constraint violations evaluated (rows 2–4). The fitness function of each individual is evaluated in rows 10 or 12 (depending on whether there are feasible individuals in the population or not) by summing their ranking positions $R_{N_v}, R_{j\phi}$ and $R_{f}$ that have been calculated in rows 6, 7 and 9 respectively. It is interesting to remark that the two “for” loops of rows 2 and 5 cannot be merged into one, since the values of the objective function and constraint violations for the entire population are required before calculating the ranking positions at rows 6, 7 and 9.
As stated in Garcia et al. 2017b, HSR and other previous rank-based techniques may face difficulties in complex problems characterized by constraints with different units and/or different orders of magnitude. This is the case of many practical applications in the offshore oil industry, such as the optimization of risers connected to floating platforms (Vieira et al. 2012; Lima et al. 2005; Pina et al. 2011), or the optimization of subsea pipeline routes (Lucena et al. 2014; Baioco and de Lima Jr. MHA, Albrecht CH, de Lima BSLP, Jacob BP, Rocha DM 2018). For instance, in this latter application many disparate constraints are specified, such as declivity (measured in degrees); minimum radius of curvature (in meters); structural constraints such as on-bottom stability (DNVGL 2017a) and fatigue due to vortex-induced vibrations (VIV) in free spans (DNVGL 2017b). Constraints may be defined even as non-dimensional quantities, such as the number of identified interferences of the route with seabed obstacles (subsea equipment, flowlines, other pre-existent pipelines, regions with corals or geotechnical hazards). In such cases, if just one single rank $R_k$ of constraint violation values is built (as in the case of HSR), it could be dominated by a given constraint with higher violation values.

Of course, this wouldn’t be an issue for simpler cases where one knows in advance the ranges of minimum and maximum violation values, and normalization procedures could be applied. However, for many real-world engineering applications such as those offshore applications listed above, this information is not available and cannot be estimated in advance. With MCR, all types of constraints are given the same priority and importance in the evaluation of each solution, regardless of the range of their violation values.

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 ($R_{x_k}$ and $R_{x_j}$); thus, the computational costs associated to evaluations of the objective function $f$ are avoided.

### 3.2 Surrogate model: Merit-based database management

A difficult and nontrivial task associated to the $k$-NN metamodel described in Sect. 2.2, is the definition of an adequate database size $\mathcal{D}$ 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 chooses the best solution of the population for evaluation by the exact model (Regis 2014; Regis and Shoemaker 2007; Elsayed et al. 2014). 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 Silva (2016). Initially, a predetermined number of solutions are generated and evaluated by the exact function to compose the database $\mathcal{D}$. Those with the best objective function values also compose the initial population $N$ (here it is assumed that $N \leq D$). Evolution occurs considering the current population evaluated by the surrogate model. The database solutions used in the approximation process are scored according to their selection as the $k$-nearest neighbors: the selected solutions have their score increased, while the others have their score decreased.

After each generation, a given number of lower scored solutions (corresponding to a user-defined parameter $\%N$ as a percentage of the solutions on the current population) are replaced by solutions evaluated using the exact function and introduced into the database.

Algorithm 2 shows the general structure of the metamodel including this merit-based strategy. The main loop of the algorithm begins at row 1 for all solutions that comprise the population. Then, the loop shown at rows 2–4 calculates the Euclidean distance between the current solution $x_i$ and each exact solution $x_{\phi}$ stored in the database $\mathcal{D}$. The database is then sorted according to those distances (row 5), and the $k$-nearest neighbors of $x_i$ (selected in row 6) are used to calculate the objective function of the current solution $x_i$ according to Eq. (2.3) (row 7). Next, the score of all solutions stored in the database $\mathcal{D}$ is updated in rows 10 and 13, respectively, to increase the score of the selected $k$-nearest neighbors, and to decrease the score for the other solutions that have not been selected. Finally, at row 16 after the main loop, a given percentage of the lower scored solutions are replaced in the database for solutions that have been evaluated using the exact function.

This process enhances the main feature of the $k$-NN model: the use of similarity for approximations. It keeps a memory of the most promising regions, and, while allowing new regions of the search space to be visited (and benefiting from them), there remains a tendency to return to the best regions already known. As a result, this similarity process benefits solutions more similar to the current population, thus increasing the accuracy of the model.
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 (Liang et al. 2006)). 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).

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 frequent premature convergence behavior. Firstly, because the new population may visit new regions of the search space, counteracting the tendency of being stuck in regions where it previously stagnated. Also, because it does not introduce additional computational costs 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 already stored in the database), therefore not requiring costly extra evaluations of the exact function.

4 Summary of the proposed method

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, Algorithm 3 presents the pseudocode that describes its variants SADE-kNN-MCR and SADE-kNN-MCR + . Both methods begin by randomly initializing (row 1) and evaluating (row

| Func | Best | Average | Worst | succ_nfe | SuccRate | Nfe   |
|------|------|---------|-------|----------|----------|------|
| G1   | -15.0000 | -14.5999 | -12.0000 | 3,783 | 0.84 | 83,177.22 |
2) the objective function of the individuals stored in the database $D$. Then, as seen in row 3, the $N$ individuals with the best objective function values are selected to comprise the initial population. The evolutive process is defined by the loop beginning in row 4. At each step, while the maximum number of objective function evaluations is not reached, firstly the DE operators defined in Sect. 0 are applied to generate the new population (row 5). Then, the objective function $f(x)$ and fitness function $F(x)$ are evaluated according to Algorithms 2 and 1 (rows 6 and 7). For the \textbf{SADE-kNN-MCR+} variant, the algorithm checks if the population has stagnated around a local optimum (row 9). If this is the case, the current population is randomly reinitialized (row 10) according to the “rebirth” procedure described in Sect. 1.2; the objective function $f(x)$ and fitness function $F(x)$ of the new population are then evaluated (rows 11 and 12) according to Algorithms 2 and 1.

The twenty-four G-Suite benchmark functions proposed for the IEEE-CEC 2006 competition on real parameter constrained optimization (Liang et al. 2006); A suite of five benchmark mechanical engineering problems (Gandomi and Yang 2011): pressure vessel (Sandgren 1988), welded beam (Deb 1991), cantilever beam (Erbatur et al. 2000), speed reducer and spring design (Gandomi and Yang 2011). 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 problems are already well documented in those references; thus, they will not be reproduced here.

To illustrate the advantages of the MCR constraint-handling technique described in Sect. 3.1, the comparisons presented later in Sects. 3 and 4 will highlight the improved performance of the \textbf{SADE-kNN-MCR} and \textbf{SADE-kNN-MCR+} variants when applied to the following problems that present constraints with different

\begin{algorithm}
1 Initialize individuals randomly in $D$
2 Evaluate individuals in $D$
3 Set Current_population with the best individuals of $D$
4 While maximum number of objective function evaluation is not reached do
5 Generate New_population applying DE operators on the Current_population
6 Evaluate $f(x)$, according to Algorithm 2
7 Evaluate $F(x)$, according to Algorithm 1
8 if $\text{SADE-kNN-MCR+}$ then
9 if stagnated in a local optimum then
10 Reinitialize Current_population randomly
11 Evaluate $f(x)$ according to Algorithm 2
12 Evaluate $F(x)$ according to Algorithm 1
13 end
14 end
\end{algorithm}

\section{5 Numerical experiments}

\subsection{5.1 Benchmark problems}

The performance of the proposed method is compared with the other methods recently presented in the literature that have been mentioned in Sect. 2.3: ASRES, SADE-kNN, SA-DECV, SA-DECV_SR and \textit{e-DE}. For this purpose, two sets of benchmark numerical experiments are executed:

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
Problem & n & Type of function & LI & NI & LE & NE \\
\hline
Pressure Vessel (Sandgren 1988) & 4 & Quadratic & 3 & 1 & 0 & 0 \\
Welded Beam (Deb 1991) & 4 & Quadratic & 6 & 1 & 0 & 0 \\
Tension/Compression Spring (Gandomi and Yang 2011) & 3 & Quadratic & 1 & 3 & 0 & 0 \\
Cantilever Beam (Erbatur et al. 2000) & 10 & Quadratic & 6 & 5 & 0 & 0 \\
Speed Reducer (Gandomi and Yang 2011) & 7 & Cubic & 1 & 10 & 0 & 0 \\
\hline
\end{tabular}
\caption{Parameters of the engineering problems}
\end{table}
orders of magnitude (representative of many actual engineering problems as described in Section 3.1): functions G10 and G21 from CEC2006, the pressure vessel and the welded beam engineering problems.

Table 3 shows the lower and upper bounds for the constraint functions of those problems: the differences in scale and magnitude are evident. It is interesting to observe that these bounds are analytically determined only for the G10 and G21 functions, and for the Pressure Vessel problem. For the welded beam problem, which presents severely nonlinear constraint functions, orders of magnitude have been numerically estimated.

Following the suggestion of Liang et al. (2006), the algorithms are tested from the results of twenty-five independent executions performed for each benchmark problem. Also following the specifications of Liang et al. (2006), 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 Sect. 3.2 to compare the different SADE-kNN algorithms.

5.2 Parameters of the algorithms

Table 4 presents the values of the parameters that the proposed method share in common with the other algorithms being compared.

The parameters that are specific for the SADE-kNN-MCR and SADE-kNN-MCR + variants have been defined from numerical experiments. As described in Sect. 1.2, the strategy that controls the reinitialization of
the population to avoid stagnation relies on a user-defined parameter that is the number of generations in which stagnation is observed before triggering the population “rebirth”. The results of the experiments indicated that an appropriate value for this parameter would be 1,000 generations.

Regarding the merit-based database management procedure described in Sect. 1.1, similar experiments have also indicated that an appropriate value for the percentage of the population (%N) comprising the lower-score solutions to be replaced by the exact evaluation of their objective function and introduction in the database at each generation is 22%.

It is interesting to mention that the MCR constraint-handling technique described in Sect. 3.1 has no adjustable parameters.

5.3 Performance assessment

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

5.3.1 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).

5.3.2 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) (Gibbons and Chakraborti 2003) from the totalized signs. If the p-value is higher than a given threshold value ϱ, then the null hypothesis (H₀) that both methods are equivalent is accepted; otherwise H₀ is rejected, and the performance of the methods is different. The most commonly used threshold values are 0.01 and 0.05 (Fisher and Theory of statistical estimation. 1925; Corder and Foreman 2009; Derrac et al. 2011; Lynn and Suganthan 2015), representing a significance level of, respectively, 1% and 5%, i.e., the chance of rejecting the null hypothesis when in fact it is correct.

5.3.3 Performance profiles

The comparisons also use another nonparametric statistical test, specifically the performance profiles (PPs) that were introduced in Dolan and Moré (2002) 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 ∈ S considering all problems p ∈ P), built considering a given metric mp,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 rp,s, is then calculated dividing this metric mp,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:

\[ \rho_s(\tau) = \frac{\text{size}\{p \in P : r_p,s \leq \tau\}}{n_p} \] (4.1)

where \( n_p \) is the total number of problems in set P and \( \rho_s(\tau) \in \mathbb{R} \to [0, 1] \) 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:

\[ \rho_s^* = \lim_{\tau \to r_M} \rho_s(\tau). \]

Overall, the highest the \( \rho_s \) values, the better the method is, so a quick visual assessment of the performance of all methods may be performed simply by observing which one has the curves above all others. Conversely, in some cases a more precise numerical indication may be provided by calculating the areas under the curves and then drawing a
| 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              |
| 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                | 5,126.49 | 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       |
| Function | feasRate | SADE-kNN | SADE-kNN MCR | SADE-kNNMCR |
|----------|----------|----------|--------------|-------------|
|          | 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       |
| G8 [-0.09582] | feasRate | 1        | 1            | 1           |
|          | best     | 680.638  | 680.6353     | 680.630     |
|          | average  | 681.287  | 680.6783     | 680.630     |
|          | worst    | 682.476  | 680.7954     | 680.630     |
|          | succ_nfe | -        | -            | 38,226.8    |
|          | succRate | 0        | 0            | 1           |
|          | nfe      | 500,000  | 500,000      | 38,226.8    |
| G9 [680.630] | feasRate | 1        | 1            | 1           |
|          | 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  |
|          | succ_nfe | -        | -            | 20,198.4    |
|          | succRate | 0        | 0            | 1           |
|          | nfe      | 500,000  | 500,000      | 20,198.4    |
| G10 [7,049.2480] | feasRate | 1        | 0.96         | 1           |
|          | 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     |
| G11 [0.7499] | feasRate | 1        | 0.76         | 1           |
|          | 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       |
| G12 [-1.0000] | feasRate | 1        | 1            | 1           |
|          | best     | 0.05394  | 0.05394      | 0.05394     |
|          | average  | 0.3534   | 0.2930       | 0.05394     |
|          | worst    | 1.0000   | 1.0000       | 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    |
| G13 [0.05394] | feasRate | 1        | 0.96         | 1           |
|          | 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%) |
| G14 [-47.76] | feasRate | 0.88     | 0.56         | 1           |
| Function [best known] | nfe | feasRate | SADE-kNN | SADE-kNN MCR | SADE-kNNMCR + |
|----------------------|-----|----------|----------|--------------|---------------|
| G15 [961.7150]       |     |          | 108,558.52 | 276,038.16   | 54,356.1      |
|                      |     |          | 1        | 1            | 1             |
|                      | best | 961.7150 | 961.7150 | 961.7150     | 961.7150      |
|                      | average | 961.7150 | 961.7150 | 961.7150     | 961.7150      |
|                      | worst | 961.7150 | 961.7150 | 961.7150     | 961.7150      |
|                      | succ_nfe | 1,143.1  | 1,724.28  | 5,412.4 (−53%) |
|                      | succRate | 0.92     | 0.92      | 1            |
|                      | nfe | 54,909.56 | 17.245.28 | 5,412.4      |
|                      | feasRate | 0.92     | 0.92      | 1            |
| G16 [−1.9051]        |     |          | −1.9051  | −1.9051      | −1.9051       |
|                      |     |          | −1.9051  | −1.9051      | −1.9051       |
|                      | average | −1.9051  | −1.9051  | −1.9051      | −1.9051       |
|                      | worst | −1.9051  | −1.9051  | −1.9051      | −1.9051       |
|                      | succ_nfe | 4,633.36 | 4,330.66  | 1,447.8 (−67%) |
|                      | succRate | 0.96     | 0.96      | 1            |
|                      | nfe | 4,633.36 | 2,415.76  | 1,447.8      |
|                      | feasRate | 0.96     | 0.96      | 1            |
| G17 [8,853.53]       |     |          | 8,853.53 | 8,853.53     | 8,853.53      |
|                      |     |          | 8,868.96 | 8,910.50     | 8,860.07      |
|                      | average | 8,927.59 | 9,011.91 | 8,938.51     |
|                      | worst | 69,887.31 | 70,850.44 | 258,603.70 (+ 73%) |
|                      | succ_nfe | 0.76     | 0.40      | 0.68         |
|                      | nfe | 173,115.36 | 345,508.28 | 304,584     |
|                      | feasRate | 0.96     | 0.96      | 0.84         |
| G18 [−0.8660]        |     |          | −0.86   | −0.866       | −0.866        |
|                      |     |          | −0.8654 | −0.8649      | −0.866        |
|                      | average | −0.8536 | −0.8597  | −0.866       |
|                      | worst | −253,743.70 | 85,907.04 | 5,855.1 (−93%) |
|                      | succ_nfe | 0.40     | 0.92      | 1            |
|                      | nfe | 401,500.28 | 119,034.84 | 5,855.1     |
|                      | feasRate | 0.96     | 0.96      | 0.84         |
| G19 [32.6555]        |     |          | 32.6632 | 32.659       | 32.6569       |
|                      |     |          | 39.2811 | 38.080       | 32.6586       |
|                      | average | 81.044  | 70.502   | 32.6676      |
|                      | worst | 253,743.70 | 85,907.04 | 5,855.1      |
|                      | succ_nfe | 0.40     | 0.92      | 1            |
|                      | nfe | 500,000  | 500,000  | 0            |
|                      | feasRate | 0.64     | 0.92      | 0.92         |
| G21 [193.7245]       |     |          | 193.7546 | 193.7546     | 193.7245      |
|                      |     |          | 193.7546 | 193.7546     | 199.4284      |
|                      | average | 193.7546 | 193.7546 | 324.7028     |
|                      | worst | 193.7546 | 193.7546 | 122,454      |
|                      | succ_nfe | 0        | 0        | 0            |
|                      | succRate | 0.64     | 0.92      | 0.92         |
|                      | nfe | 500,000  | 500,000  | 258,371.4    |
|                      | feasRate | 0.64     | 0.92      | 0.92         |
| G23 [−400.055]       |     |          | −400.055 | −400.055     | −400.055      |
|                      |     |          | −400.055 | −154.191     | −400.055      |
|                      | average | −400.055 | −0.0024  | −400.055     |
|                      | worst | −400.055 | −0.0024  | −400.055     |
|                      | succ_nfe | 68,852.00 | 88,843.33 | 51,922.6 (−25%) |
bar chart indicating those areas. A detailed explanation including an illustrative example of the assembly of a PP may be found in Garcia et al. 2017b.

### 6 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 Garcia et al. 2017a. The performance is compared by their application to the sets of numerical experiments listed at the beginning of Sect. 2: the G-Suite benchmark functions proposed for the IEEE-CEC 2006 competition and the suite of five benchmark mechanical engineering problems.

#### 6.1 G-Suite functions

##### 6.1.1 Direct comparisons

Table 5 presents the direct comparisons between the methods, in terms of the performance measures defined in Sect. 2.3.1, 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.

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, G9, 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-
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%.

6.1.2 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 Sect. 2.3.2) 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.

6.1.3 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 Sect. 2.3.3. The PPs shown in Fig. 1 again indicate that SADE-kNN-MCR + is the best performer in terms of accuracy and
robustness: it is easily seen that its curve is above all others, reaching $\rho(\tau) = 1$ with the lowest $\tau$ value.

Regarding the performance of SADE-kNN-MCR compared with the original SADE-kNN method: the contribution of the MCR procedure has already been indicated at some of the direct comparisons of Table 5 (see, for instance, the remarks regarding the G10 and G21 problems in the second-to-last paragraph of Sect. 3.1.1). Also, Table 6 indicates that SADE-kNN-MCR wins for 6 of the benchmark problems (G5, G11, G13, G15, G18 and G23) and loses for 3 problems (G14, G16 and G17). Finally, this better performance of the SADE-kNN-MCR method can also be visualized when comparing the respective PP curves of Fig. 1: the curve corresponding to SADE-kNN-MCR is above the SADE-kNN curve for most of the range of $\tau$ values; also, it reaches $\rho(\tau) = 1$ with the lowest $\tau$ value.

### 6.2 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 among the algorithms.

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 Garcia et al. 2017b. 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 2 shows that SADE-kNN-MCR + finds best values for 80% of the problems with $\rho(1) = 0.8$. 

| Problem                  | SADE-kNN | SADE-kNN-MCR | SADE-kNN-MCR + |
|--------------------------|----------|--------------|----------------|
| Pressure Vessel          | Best     | 6,059.7180   | 6,059.7143     | 6,059.7143     |
|                          | Average  | 6,276.0322   | 6,577.5020     | 6,176.5486     |
|                          | Worst    | 7,332.8529   | 7,332.8415     | 6,820.4100     |
| Welded Beam              | Best     | 1.726247     | 1.724852       | 1.724852       |
|                          | Average  | 1.784099     | 1.802360       | 1.724852       |
|                          | Worst    | 2.319204     | 2.497832       | 1.724852       |
| Cantilever               | Best     | 65,036.880   | 64,578.194     | 64,578.194     |
|                          | Average  | 70,160.740   | 66,815.056     | 66,581.615     |
|                          | Worst    | 74,223.488   | 68,597.130     | 68,597.130     |
| Speed Reducer            | Best     | 2,996.3481   | 2,996.3481     | 2,996.3481     |
|                          | Average  | 2,999.8637   | 3,002.1815     | 2,996.3481     |
|                          | Worst    | 3,035.6255   | 73,035.625     | 2,996.3481     |
| Tension/Compr. Spring    | Best     | 0.012675     | 0.012665       | 0.012665       |
|                          | Average  | 0.012952     | 0.013095       | 0.013285       |
|                          | Worst    | 0.015754     | 0.017773       | 0.025651       |
Table 8 Comparison of results

| Method                  | G1 [-15.0000] | G2 [-0.8036] | G3 [-1.0005] | G4 [-30.665.5386] | G5 [5.126.49] | G6 [-9.618.138] | G7 [24.3062] | G8 [-0.99582] | G9 [680.630] | G10 [7.049.2480] | G11 [0.7499] | G12 [-1.0000] | G13 [0.05394] | G14 [-47.76] | G15 [961.7150] |
|-------------------------|---------------|--------------|--------------|-------------------|--------------|-----------------|--------------|--------------|-------------|----------------|-------------|--------------|-------------|-------------|----------------|
| succ_nfe                | 2,528.5       | -0.8036      | 169,089.4 (+80%) | 1,208.2 (-76%)  | 5,126.49     | 655.5 (-86%)    | 64,067.2 (-56%) | 182.8 (-66%) | 38,226.8 (+79%) | 20,198.4 (-56%) | 0.7499      | 211.5 (-68%) | 54,356.1 (-41%) | 5,412.4 (-36%) | 961.715     |
| SuccRate                | 7.972         | -0.713       | 5,113         | 7,175             | 5,126.49     | 7,335           | 24.09        | 676           | 9,831       | 7,260.041     | 1,334.2     | -1           | 0.453        | -47.24      | 961.715     |
| Best                    | -15           | -0.8036      | -1.0005       | -15               | 5,126.49     | -6,961.8138     | 24.3062      | -0.09582     | 680.630     | 7,049.2480    | 0.7499      | -1           | 0.05394     | -47.76      | 961.715     |
| succ_nfe                | 15,730.00     | -0.6252      | -0.3127       | 5,113             | 5,126.49     | -6,961.8138     | 24.3062      | -0.09582     | 30.618      | 7,049.2480    | 10.478      | -1           | 0.4873      | -47.76      | 961.715     |
| SuccRate                | 35,406        | -0.739       | -0.998        | 7,175             | 5,126.49     | 9,603           | 77.674       | -0.09582     | 29,440      | 7,049.2480    | 2,792       | -1           | 0.05394     | -47.76      | 8,519       |
| SuccRate                | 64,274        | 192,297      | 33,066        | 15,104            | 19,281       | 6,254           | 99,614       | 6,254         | 43,942      | 152,110       | 1,354       | 1           | 1           | 92,730      | 95,022      |

An enhanced surrogate-assisted differential evolution for constrained optimization problems 6407
Altogether, \textbf{SADE-kNN-MCR +} is the best overall performer.

7 Comparison with other recent methods

Since in the previous section \textbf{SADE-kNN-MCR +} had demonstrated its better performance among the proposed \textbf{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 Sect. 2.3: \textbf{ASRES}, \textbf{SA-DECV}, \textbf{SA-DECV\_SR} and \textbf{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 Sect. 2.

7.1 G-Suite functions

7.1.1 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 among the runs; the average number of function evaluations considering the successful runs \texttt{succ\_nfe}; and the ratio between successful and total runs \texttt{SuccRate}. The cells corresponding to \texttt{succ\_nfe} show also, in parenthesis, the percentage values corresponding to the reduction/increase of the \texttt{succ\_nfe} values provided by \textbf{SADE-kNN-MCR +} in relation to the smaller value observed for the other methods.

In Table 8, it is interesting to compare the performance between the \textbf{SA-DECV} and the \textbf{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 \textbf{SA-DECV}, the \textbf{SA-DECV\_SR} provides optimal solutions to problems G7, G10 and G19. It also improves the solution found by \textbf{SA-DECV} in problem G21. \textbf{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 \textbf{SADE-kNN-MCR +} and \textbf{e-DE}: \textbf{SA-DECV} methods failed in functions G13, G14, G17 and G23, while \textbf{SADE-kNN-MCR +} and \textbf{e-DE} found optimal solution for most problems. Moreover, they found good solutions in most executions, as indicated by their \texttt{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.

A more comprehensive comparison of the performance of all methods will be presented in terms of nonparametric statistical tests previously described in Sect. 2.3: the sign test and the performance profiles.

### 7.1.2 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.

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.

### 7.1.3 Performance profiles

The succ_nfe values were also used as the metric for the generation of the performance profiles shown in Fig. 3, according to the procedure described in Sect. 2.3.3. Differently from the PP curves compared in Sects. 3.1.3 and 3.2, for some curves, it is not quite obvious if they are above or below the other competitors. Therefore, to provide a better global indication of the relative performance of the methods, in this case it is useful to build the bar chart of Fig. 4 indicating the areas under the PP curves.

In Fig. 3 it is interesting to observe that, for smaller values of \( \tau \), the curve corresponding 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**,
SR and ASRES, it ends up with the second larger area in the bar chart of Fig. 4. 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 Fig. 4, reflecting the fact that it requires fewer evaluations for the greater number of problems.

7.2 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 (although with different recombination variants: SADE-kNN-MCR + uses DE/rand/1/bin, while SA-DECV-SR combines other DE variants); (2) some of the solutions are approximated by a similarity-based surrogate model (both methods use \( k \)-NN, but SA-DECV-SR uses the surrogate model also to approximate the sum of constraint violations); and 3) rank-based constraint-handling techniques are employed (SADE-kNN-MCR + uses MCR, while SA-DECV-SR employs the Stochastic Ranking (SR) method (Runarsson and Yao 2000)).

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 among all runs; and the average number of function evaluations considering the successful runs. Unlike Sect. 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.

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 lowest 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 (Miranda-Varela and Mezura-Montes 2018) 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 most of the measures shown in Table 11: average, worst, and number of function evaluations. SA-DECV-SR performed better only for best solution, where the value provided by SADE-kNN-MCR + (0.012732) reached the first termination criterion \( f(x^*) - f(x) < 0.0001 \), thus stopping the evolution process.

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. Figure 5 shows the performance using as metric the succ_nfe values presented in Table 11, indicating that SADE-kNN-MCR + is the best performer, since not only it gives better results for two problems (\( \rho(1) = 0.666 \), but

| Prob                       | Algorithm         | Best          | Average       | Worst         | Succ_nfe |
|---------------------------|-------------------|---------------|---------------|---------------|----------|
| Welded Beam               | 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           | 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| SADE-kNN-MCR +    | 0.012732      | 0.012765      | 0.012902      | 1,409.4  |
|                           | SA-DECV-SR        | 0.0127127     | 0.013047148   | 0.0143943     | 1,617    |
still finds optimal executions in all of them (there is $\tau$ such that $\rho(\tau) = 1$).

8 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 “rebirth” 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; the merit-based database management scheme of the $k$-NN model evaluates only part of the population; and even the population “rebirth” strategy does not require additional computational costs, since the new candidate solutions generated whenever the population converges to a local optimum are used to approximate their objective function values.

Two variants of the proposed method (SADE-$k$NN-MCR and SADE-$k$NN-MCR $+$) were applied to a set of 24 constrained benchmark problems and 5 mechanical engineering problems. The results from both variants were compared with those from the original SADE-$k$NN method (Garcia et al. 2017a). Then, the best performer among the SADE methods (SADE-$k$NN-MCR $+$) was compared with other algorithms presented in the literature that also employ surrogate models and constraint-handling techniques (SADE-$k$NN, ASRES, SA-DECV, SA-DECV$-$SR and e-DE). The performance of all methods is assessed based on direct comparisons, nonparametric statistical tests, and performance profile (PP) graphs. The direct comparisons consider statistical parameters collected from all runs, taking different measures (best, average, worst results, number of evaluations, success ratio, etc.). The nonparametric statistical tests consider the well-known sign test that indicates if the results of two methods are statistically different. Finally, the PP graphs allow a quick global evaluation of the accuracy and robustness of the methods. 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-$k$NN-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 Vieira et al. (2012); Lucena et al. 2014; Monteiro et al. 2016; Lima et al. 2005; Pina et al. 2011). 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-$k$NN-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 Vieira et al. (2012); Monteiro et al. 2016; Baioco and de Lima Jr. MHA, Albrecht CH, de Lima BSLP, Jacob BP, Rocha DM2018). 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.

Funding The authors acknowledge the support of the Brazilian funding agencies CNPq (Grant numbers 306069/2016, 306186–2017, 308873/2017–3); FAPEMIG (grant numbers E-26/200.314/2016, E-26/ 202.767/2017); FAPEMIG (Grant number TEC-PPM-174–18); and CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), finance code 001.
Data availability
Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest
R de PG declares that he has no conflict of interest. Beatriz Souza Leite Pires de Lima declares that she has no conflict of interest. ACde CL declares that he has no conflict of interest. Breno Pinheiro Jacob declares that he has no conflict of interest.

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

References

Anderson KS, Hsu Y (1999) Genetic crossover strategy using an approximation concept. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), IEEE, pp 527–533
Awad NH, Ali MZ, Mallipeddi R, Suganthan PN (2018) An improved differential evolution algorithm using efficient adapted surrogate model for numerical optimization. Inf Sci 451–452:326–347. https://doi.org/10.1016/j.ins.2018.04.024
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 pp. 1-21
Bergamaschi PR, Saramago SDFP, dos Santos CL (2008) Comparative study of SOP and metaheuristics for robotic manipulator design. Appl Numer Math 58(9):1396–1412
Corder GW, Foreman DI (2009) Nonparametric statistics: an introduction. Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach. John Wiley & Sons, Hoboken, NJ, USA, pp 101–111
de Lima BSLP, Jacob BP, Ebecken NFF (2005) A hybrid fuzzy/ge-netic algorithm for the design of offshore oil production risers. Int J Numer Meth Eng 64(11):1459–1482. https://doi.org/10.1002/nme.1416
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. Adv Eng Softw 76:110–124. https://doi.org/10.1016/j.advengsoft.2014.06.003
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. Optim Eng 12(1–2):215–235. https://doi.org/10.1007/s11081-009-9103-5
de Pina AC, Albrecht CH, de Lima BSLP, Jacob BP (2014a) Wavelet network meta-models for the analysis of slender offshore structures. Eng Struct 68:71–84. https://doi.org/10.1016/j.engstruct.2014.02.039
de Pina AC, Monteiro BdF, Albrecht CH, de Lima BSLP, Jacob BP (2014b) ANN and wavelet network meta-models for the coupled analysis of floating production systems. Appl Ocean Res 48:21–32. https://doi.org/10.1016/j.apor.2014.07.009
DNVGL (2017a) Recommended Practice DNVGL-RP-F109: On-bottom stability design of submarine pipelines.
DNVGL (2017b) Recommended Practice DNVGL-RP-F105: Free spanning pipelines.
Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. Paper presented at the IEEE transactions on evolutionary computation,
Deb K (1991) Optimal design of a welded beam via genetic algorithms. AIAA J 29(11):2013–2015
Deb K (2000) An efficient constraint handling method for genetic algorithms. Comput Methods Appl Mech Eng 186:311–338
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 Evol Comput 1(1):3–18. https://doi.org/10.1016/j.swevo.2011.02.002
Dolan ED, More JJ (2002) Benchmarking optimization software with performance profiles. Math Program 91(2):201–213. https://doi.org/10.1007/s101070100263
Elsayed SM, Ray T, Sarker RA (2014) A surrogate-assisted differential evolution algorithm with dynamic parameters selection for solving expensive optimization problems. In: Evolutionary Computation (CEC), 2014 IEEE Congress on, IEEE, pp 1062–1068
Emmerich MT, Giannakoglou KC, Naujoks B (2006) Single-and multiobjective evolutionary optimization assisted by gaussian random field metamodels. IEEE Trans on Evol Comput 10:421–439
Erbudur F, Hasańcebi O, Tutünük I, Kılıç H (2000) Optimal design of planar and space structures with genetic algorithms. Comput Struct 75(2):209–224
Ferrari S, Stengel RF (2005) Smooth function approximation using neural networks. IEEE Trans Neural Netw 16:24–38
Fisher RA (1925) Theory of statistical estimation. In: Mathematical Proceedings of the Cambridge Philosophical Society, Cambridge University Press.
Fonseca LG, Barbosa HJC, Lemonge ACC (2009) A similarity-based surrogate model for enhanced performance in genetic algorithms. Oapsearch 46(1):89–107
Gandomi AH, Yang X-S (2014b) Optimal design of submarine pipeline routes by genetic algorithm with different constraint handling techniques. Adv Eng Softw 76:110–124. https://doi.org/10.1016/j.advengsoft.2014.06.003
Garcia RdP, de Lima BSLP, Lemonge ACdC, Jacob BP (2017b) A rank-based constraint handling technique for engineering design optimization problems solved by genetic algorithms. Comput Struct 187:77–87. https://doi.org/10.1016/j.compstruct.2017.03.023
Garcia RdP, de Lima BSLP, Lemonge ACdC (2017a) 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.
Gibbons JD, Chakraborti S (2003) Nonparametric Statistical Inference. Statistics: a Series of Textbooks and Monographs, Fourth edn. Marcel Dekker, Inc., New York - Basel
Grefenstette JJ (1985) Fitzpatrick JM Genetic search with approximate fitness evaluations. In: Proc. Of the Intl. Conf. on Genetic Algorithms and Their Applications, pp 112–120
Hendrickx W, Gorissen D, Dhaene T Grid enabled sequential design algorithms. Comput Methods Appl Mech Eng 186:311–338
Hrstka O, Kuc ˇerova´ A (2004) Improvements of real coded genetic algorithms. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), IEEE, pp 527–533
Hong YS, Lee H, Tahk MJ (2003) Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks. Eng Optim 35(1):91–102
Hrstka O, Kučerová A (2004) Improvements of real coded genetic algorithms based on differential operators preventing premature convergence. Adv Eng Softw 35(3–4):237–246
An enhanced surrogate-assisted differential evolution for constrained optimization problems

Hu H, Lee DL (2006) Range nearest-neighbor query. IEEE Trans Knowl Data Eng 18(1):78–91. https://doi.org/10.1109/TKDE.2006.15

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

Kecman V (2005) Support vector machines—an introduction. Support vector machines: theory and applications pp. 605–605

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

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

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

Kybic J, Blu T, Unser M (2002a) Generalized sampling: a variational approach. I. Theory. IEEE Trans Signal Process 50:1965–1976

Kybic J, Blu T, Unser M (2002b) Generalized sampling: a variational approach. II. Appl IEEE Trans Signal Process 8:1977–1985

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

Liang JJ, Runarsson TP, Mezura-Montes E, Clerc M, Suganthan PN, CoelloCAC, Deb K (2006) Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. J Appl Mech 41(8):8–31

Liu Y, Sun F (2011) A fast differential evolution algorithm using k-Nearest Neighbour predictor. Exp Syst Appl 38(4):4245–4258

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.

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

Lynn N, Suganthan PN (2015) Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation. Swarm Evol Comput 24:11–24. https://doi.org/10.1016/j.swevo.2015.05.002

Mallipeddi R, Lee M (2015) An evolving surrogate model-based differential evolution algorithm. Appl Soft Comput 34:770–787. https://doi.org/10.1016/j.asoc.2015.06.010

Miranda-Varela M-E, Mezura-Montes E (2018) Constraint-handling techniques in surrogate-assisted evolutionary optimization. An Empirical Study Appl Soft Comput 73:215–229. https://doi.org/10.1016/j.asoc.2018.08.016

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,

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 Syst Ocean Technol 11:55–67. https://doi.org/10.1007/s40868-016-0017-8

Mullur AA, Messac A (2006) Metamodeling using extended radial basis functions: a comparative approach. Eng Comput 21:203–217

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

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

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

Queipo NV, Haftka RT, Shyy W, Goel T, Vaidyanathan R, Tucker PK (2005) Surrogate-based analysis and optimization. Paper Present Prog Aerospace Sci 41:1–28

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,

Raile A (1999) Optimal sampling strategies for learning a fitness model. In: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), IEEE, pp 2078–2085

Regis RG (2014) Evolutionary programming for high-dimensional constrained expensive black-box optimization using radial basis functions. IEEE Trans Evol Comput 18:326–347

Regis RG, Shoemaker CA (2007) A stochastic radial basis function method for the global optimization of expensive functions. INFORMS J Comput 19:497–509

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

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

Runarsson TP (2006) Approximate evolution strategy using stochastic ranking, paper presented at the 2006 IEEE International Conference on Evolutionary Computation, Vancouver, BC, Canada, pp. 16–21

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

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

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

Silva da 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 ACuC, Rio de Janeiro

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

Storn R, Price K (1997) Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11:341–359

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

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

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

Vieira IN, de Lima BSLP, Jacob BP (2012) Bio-inspired algorithms for the optimization of offshore oil production systems. Int J
Wang YS, Shi YJ, Yue BX, Teng HF (2010) An efficient differential evolution algorithm with approximate fitness functions using neural networks. In: International Conference on Artificial Intelligence and Computational Intelligence, Heidelberg, Berlin, Springer, pp 334–341

Wu G, Shen X, Li H, Chen H, Lin A, Suganthan PN (2018) Ensemble of differential evolution variants. Inf Sci 423:172–186

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

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

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.