An extension for differential evolution based on centroids with self-adaptive noise and preselection

Original Scientific Paper

Dražen Bajer
J. J. Strossmayer University of Osijek,
Faculty of Electrical Engineering, Computer Science and Information Technology Osijek
Kneza Trpimira 2B
drazen.bajer@ferit.hr

Bruno Zorić
J. J. Strossmayer University of Osijek,
Faculty of Electrical Engineering, Computer Science and Information Technology Osijek
Kneza Trpimira 2B
bruno.zoric@ferit.hr

Mario Dudjak*
J. J. Strossmayer University of Osijek,
Faculty of Electrical Engineering, Computer Science and Information Technology Osijek
Kneza Trpimira 2B
mario.dudjak@ferit.hr

*Corresponding author

Abstract *– Continuous optimisation in different technical systems is an ever ongoing, but in many cases a challenging task. Evolutionary algorithms and, in particular, differential evolution (DE) have shown their potential as effective means of optimisation in numerous such instances, especially when the systems are "black boxes". Yet, in an endeavour to increase the performance of these algorithms even further, various modifications and extensions are steadily introduced. This paper is just another step in the pursuit of gaining additional performance. Correspondingly, a DE extension, one that might benefit not only the canonical algorithm, but also already modified or extended varieties is proposed to this end. The proposal features a straightforward incorporation into the algorithm and is based on the preselection, via a simple surrogate model, of multiple candidate centroids of the entire population that are infused with self-adaptive normal noise. The soundness and viability of its design have been shown experimentally by incorporating it into different DE variants and evaluating the performance gains on a diverse set of challenging benchmark functions.*

Keywords: *differential evolution, k-NN regression, noise-infused centroids, plug-in extension, surrogate model*

Received: July 10, 2026; Received in revised form: August 23, 2026; Accepted: September 1, 2026

1. INTRODUCTION

Optimisation in various technical systems is an everlasting endeavour that attempts to get the most out of them. A significant part of that concerns continuous or numerical optimisation which brings many challenges. This becomes most evident when the systems are essentially "black boxes". When only responses to the system inputs are available, it is difficult to find effective or efficient means of optimisation. Yet, evolutionary algorithms (EAs) [1] have shown their worth in numerous such instances (see, e.g., [2, 3, 4, 5]), since they generally do not require any knowledge about the intrinsics of the system at hand to perform an effective search for (near) optimal solutions. Amongst the multitude of EAs, differential evolution (DE) [6] stands out for its simplicity (at least when it comes to the canonical algorithm) and effectiveness. Unsurprisingly, a variety of modifications and extensions of the algorithm have been proposed with the goal of performance enhancement. Approaches like those in [7, 8, 9, 10, 11, 12, 13] are just a few of the pursuits to this end, but which differ significantly in their design and sophistication, nonetheless. This observation, however, can be generalised to most of the available modifications and extensions of DE. Whilst some are more sophisticated than others, their intricacy is not necessarily directly proportional to

performance gains. Simpler or less involved extensions are, therefore, interesting in their own right [14]. Many of these are regarded, in some respects, as complementary or auxiliary search mechanisms [10] since they are easily incorporated into DE without requiring notable alterations to the structure and main elements of the algorithm (in some sense, loosely coupled to the algorithm). Furthermore, they typically introduce completely different search patterns (compared to that of the base algorithm), and are, in many instances, of relatively low time and space complexity. Finally, and most importantly, they may yield considerable performance gains, at least on some problem classes, despite their relative simplicity.

Extensions that are loosely coupled to the algorithm presumably hold the following promise: They could be easily incorporated into already modified or extended DE variants to achieve additional performance enhancements. Unfortunately, the desired positive effect of such supplementary extensions cannot be assumed a priori [9, 10]. Whether the effect will be positive or negative depends on the unpredictable interactions between the extension to be introduced and already present modifications/extensions. Designing DE extensions that interact well with existing modifications or extensions is, therefore, an interesting but challenging undertaking. The challenge is inherently multifaceted.

This paper tackles the challenge by proposing a DE extension that features a minimal interference with the structure of the algorithm and combines the following:

1. Calculation of the centroid of the entire population
2. Component-wise computation of the standard deviation
3. Preselection through similarity-based modelling

By infusing the centroid with normal noise defined by the standard deviations, multiple candidate solutions harnessing the information of the entire population are generated. The most promising is sought through preselection via a simple and lightweight surrogate model. The potential of the proposal to enhance the performance of not only the canonical DE but also different already modified or extended varieties of the algorithm was experimentally shown on a diverse set of difficult benchmark functions.

The rest of the paper is organised as follows. Section 2 provides a concise outline of the DE algorithm and a literature overview of related works. The proposed DE extension is outlined in dail in Section 3. The results of the evaluation of the proposal are reported and discussed in Section 4. The drawn conclusions and possibilities for future work are presented in Section 5.

2. BACKGROUND

A continuous optimisation problem may be defined by the pair (f, S), where $f : S \rightarrow R$ is the objective function and $S=[s^{min}, s^{max}] \subseteq R^{\wedge d}$ the search space. Presuming minimisation, a point $x^* \in S$ such that $\forall x \in S: f(x^*) \leq f(x)$ is considered an optimal solution. However, for many systems the analytical form of the objective function is highly complex (e.g., discontinued, non-linear, and/or non-derivable) or even unknown, which renders them "black boxes" in all practical aspects. Accordingly, optimisation methods requiring only responses to given inputs are effectively applicable. Various EAs are amongst these, with DE being a prime example that was also subject to a plethora of modifications and extensions.

2.1. DIFFERENTIAL EVOLUTION

The canonical DE and its many variants have demonstrated their compelling performance on numerous problem types [15, 16]. The canonical algorithm offers a strong foundation for the introduction of different modifications and extensions aimed at gaining additional performance, at least on some problem classes. An outline of the structure of the canonical DE is shown in Fig. 1. As in common EAs, a population of candidate solutions is subject to variation operators and selection to drive the optimisation process.

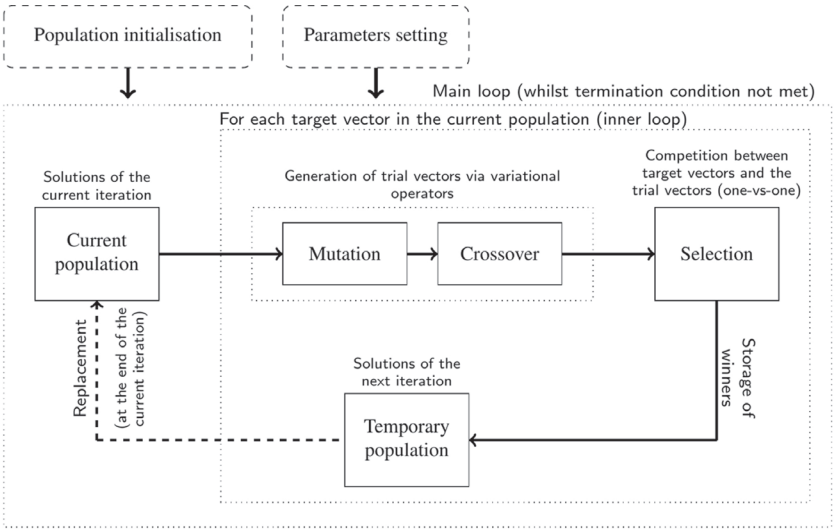


Fig. 1. Graphical outline of the DE algorithm structure

The population $P^t = (\boldsymbol{x}^{1,t},\ldots,\boldsymbol{x}^{NP,t})$, where $\boldsymbol{x}^{i,t} = (x_1^{i,t},\ldots,x_d^{i,t}) \in R^d$ for $i=1,\ldots,NP$ at the start, $t=0$, is typically initialised uniformly at random inside the search space [17]. In each of the subsequent iterations, $t>0$, a donor (or mutant) vector is generated for every target vector $\boldsymbol{x}^{i,t}$ using the linear combination

$$\mathbf{v}^{i,t} = \mathbf{x}^{r1,t} + F \cdot (\mathbf{x}^{r2,t} - \mathbf{x}^{r3,t}), \quad (1)$$

where $F \in \langle 0,1]$ is the scaling factor, and $r1$, $r2$, and $r3$ are mutually distinct indices selected at random such that, also, none equals i.

Immediately following mutation, (binomial) crossover is applied to the donor and target vectors to produce a trial vector

$$y_j^{i,t} = \begin{cases} v_j^{i,t} & \text{if } \mathcal{U}_j^{i,t}[0,1] \leq CR \vee j = r_i \\ x_j^{i,t} & \text{otherwise} \end{cases}, \; j = 1,\ldots,d, \quad (2)$$

where $U_j^{i,t}$ [0,1] is a uniform variate, $CR \in [0,1]$ is the crossover-rate, and r_i is drawn at random from $\{1,\ldots,d\}$.

Finally, the target and trial vectors are compared based on their qualities, and the better candidate moves to the next iteration $t+1$,

$$\mathbf{x}^{i,t+1} = \begin{cases} \mathbf{y}^{i,t} & \text{if } f(\mathbf{y}^{i,t}) \leq f(\mathbf{x}^{i,t}) \\ \mathbf{x}^{i,t} & \text{otherwise} \end{cases}. \quad (3)$$

Different termination conditions can be employed, but a run is usually stopped after a set maximum number of function evaluations ($NFEs_{max}$). The mutation strategy in Eq. (1) represents only one of several alternatives [15], although it is amongst the most frequently adopted. Rarely, crossover operators different from that in Eq. (2) are utilised (see, e.g., exponential crossover [6]). Virtually no other selection procedures are employed than the one in Eq. (3) (only in some cases a strict inequality is used). Furthermore, selecting a suitable configuration of the parameters NP, F, and CR is essential for achieving satisfactory performance on different problems [11]. Ultimately, the given description corresponds to the standard or canonical DE, commonly denoted as DE/rand/1/bin.

2.2. LITERATURE OVERVIEW OF RELATED WORKS

Much of the efforts to improve the performance of DE are aimed at incorporating parameter control schemes and/or altering of the mutation strategy (see, e.g., [18, 19, 20, 9, 13]), and with good reason as they are typically fruitful. However, such modifications to the algorithm do not introduce any major changes to the search pattern. Therefore, a variety of extensions and modifications, based on different concepts and designs, have been proposed in an attempt to augment the search pattern of the DE algorithm in more substantial ways. Naturally, not all are created equal, and some are more successful in that regard than others.

The richness of the available extensions or modifications for DE in terms of diversity is evident even from a glance at the literature. However, there are not many extensions or modifications that are clearly separated from the main algorithmic elements, i.e., ones that may be denoted as loosely coupled to the algorithm. Probably the best-known such extension for DE utilises opposition-based learning [7, 21]. Candidate solutions are generated by essentially "mirroring" the population (i.e., by computing opposite points), which provides a simple way of exploring complementary regions of the search space and facilitating, thereby, the exploration capabilities of the algorithm. The modifications in [22, 23] also attempt to improve the exploration capabilities of the algorithm by introducing uniform variates for some solution components in a probabilistic manner. Yet, neither of the two can be denoted as a loosely coupled extension or modification since they are intertwined with the variation operators of the underlying DE algorithm.

The trend of utilising concepts from machine learning (ML) to augment the DE algorithm is gaining traction. Unsurprisingly, even some of the earlier available extensions are based on ML concepts. For example, data clustering is leveraged in [8, 17], where the k-means algorithm is applied to create k-partitions of the population. The cluster centroids are exploited to generate candidate solutions, either in that "raw" form [8] or infused with noise [17]. Further, a well-known data-based oversampling technique (SMOTE) was used as the foundation for the extension in [10], where candidates are generated through convex combinations (with random weights) of select solutions and randomly chosen ones from their k-neighbourhood. Convex combinations are also used in the extension in [14], but of the entire population, instead of only pairs. Specifically, the weights are computed in different fashions to enable the creation of multiple and different candidates. Altogether, it is not difficult to conclude that the above-mentioned extensions focus on improving the exploitation capabilities of the underlying algorithm (which increases the convergence speed but might make the algorithm more prone to getting trapped in local optima—there is always the trade-off between exploration and exploitation at play).

In general terms, the overviewed extensions show that beneficial search patterns can be introduced without (significantly) interfering with the main algorithmic elements, but also that the way population information is transformed into candidate solutions is crucial as it determines, to a significant degree, whether focus is put on improving exploration or exploitation. The presented extension harnesses the information contained in the entire population and uses a simple regression algorithm to model the fitness landscape. Two-level extraction of the population information is used to produce multiple candidate solutions. This not only improves on [14], but also on [17] since it enables the generation of noise that is self-adaptive. Only the most promising candidate is evaluated and considered for introduction into the population. Choosing or preselecting it via a surrogate model is in line with current research trends of leveraging ML concepts in DE or EAs, in general.

3. PROPOSED EXTENSION FOR DIFFERENTIAL EVOLUTION

The proposal may be regarded as a self-contained auxiliary search mechanism that makes use of the entire current population to generate a specific candidate solution. As such, it is aimed at complementing the search pattern of the underlying algorithm and augmenting its exploitation capabilities. The incorporation of the proposal into DE does not require any alterations to the main algorithmic elements. Thus, it can be regarded as a "plug-in extension" as well.

3.1. OUTLINE OF THE DESIGN

The proposed extension essentially attempts to update the population of the current iteration, P^t, by replacing its worst solution. The whole process can be described in a couple of key steps. (Note, iteration indices are omitted from the equations for the sake of simplicity.)

Extraction of the information contained in the population Is achieved through the calculation of the population centroid (or centre of mass),

$$\mathbf{c} = \frac{1}{NP}\sum_{i=1}^{NP}\mathbf{x}^i, \tag{4}$$

and the computation of the standard deviations of the population at the component-level (or dimension-level),

$$\sigma_j = \sqrt{\frac{1}{NP}\sum_{i=1}^{NP}\left(x_j^i - c_j\right)^2},\ j = 1, \ldots, d\ . \tag{5}$$

In contrast to the approach in [14], the extraction of population information in the presented extension is twofold. In that regard, the component-wise standard deviations are assumed to be supplementary to the information represented by the population centroid.

Generation of multiple candidates Is realised by infusing the population centroid with normal (or Gaussian) noise defined by zero mean and the component-wise standard deviations,

$$\hat{c}_j^r = c_j + \mathcal{N}_j^r\left(0, \lambda\sigma_j\right),\ j = 1, \ldots, d,\ r = 1, \ldots, n_c, \tag{6}$$

where $n_c > 1$ is the number of candidates and $\lambda \in \langle 0,1]$ is a scaling parameter used to control the noise intensity. As the population (structure) changes throughout the optimisation process, so does the noise. This makes it self-adaptive in nature, unlike in [17], where the noise is defined by a constant (which may require careful tuning depending on the bounds of the search space). The generated candidates are thus more or less concentrated in proximity of the population centroid.

Preselection of the most promising candidate solution Is performed by choosing the best, $\hat{\boldsymbol{c}}^\star$, amongst $\hat{\boldsymbol{c}}^1, \ldots, \hat{\boldsymbol{c}}^{n_c}$ for real evaluation based on their predicted qualities, $\tilde{f}(\hat{\boldsymbol{c}}^1), \ldots, \tilde{f}(\hat{\boldsymbol{c}}^{n_c})$. Similarity-based modelling [24] via the weighted k-nearest neighbours algorithm for regression (k-NN-R) is utilised to this end to obtain a lightweight surrogate model. By adjusting the neighbourhood size, k, the localisation of its responses can be controlled, which corresponds to the smoothening of the fitness landscape.

Updating the population of the current iteration Is simply conducted by attempting to replace the worst solution contained in the population, $\boldsymbol{x}^{worst}$. The population is updated only if $f(\hat{\boldsymbol{c}}^\star) < f(\boldsymbol{x}^{worst})$. By solely considering one particular solution in P^t for replacement, the impact on the population structure is kept small and consistent, despite the high selection pressure.

Evidently, given the design of the proposal, it is primarily aimed at augmenting the exploitation capabilities of the underlying algorithm, especially as the optimisation process progresses and the population shrinks. The degree to which this is pronounced may be controlled by the scaling parameter λ in Eq. (6).

3.2. IMPLEMENTATION DETAILS

Despite the relatively straightforward design, there are still a few details that need to be mentioned and elaborated upon. This pertain to the generation of candidates by Eq. (6) and their preselection via the weighted k-NN-R surrogate.

Although unlikely, candidate solutions outside the search space, S, may be generated, necessitating the handling of bound constraints, nonetheless. Projection [25], as one of the simplest techniques, is used to repair out-of-bound components,

$$\hat{c}_j = \begin{cases} s_j^{\min} & \text{if } \hat{c}_j < s_j^{\min} \\ s_j^{\max} & \text{if } \hat{c}_j > s_j^{\max} \\ \hat{c}_j & \text{otherwise} \end{cases},\ j = 1, \ldots, d\ . \tag{7}$$

Typically, surrogates depend on special solution sets (databases) for training, which require additional space and careful management [24]. Yet, to keep things as simple as possible in terms of preselection, the population of the current iteration serves as the training set for the surrogate model. The choice of the weighted version of k-NN-R is not accidental. Namely, it may happen (for small values of λ) that all candidates have the same k neighbours, but due to the weighted neighbour-contributions it is highly unlikely that they will have the same predicted qualities.

3.3. INCORPORATION INTO DE

The proposal, summarised in Alg. 1, is simply introduced or grafted at the beginning of the main loop of the algorithm (just before the inner loop in Fig. 1). Its separation from the rest of the algorithm is thus clear-cut. This minimal interference (or loose coupling) with the algorithm in general is assumed beneficial in the sense of attenuating possible ill-behaved interactions with other algorithmic elements (including already present modifications or extensions).

Alg. 1. Outline of the proposed DE extension

1: Set n_c, k, and λ [only once]

2: Calculate centroid of P^t [by Eq. (4)]

3: Compute component-wise standard deviations of P^t [by Eq. (5)]

4: Generate candidates [by Eq. (6) and Eq. (7)]

5: Select the best amongst the candidates via weighted k-NN-R

6: Evaluate the selected solution and try to replace the worst in P^t

Since the proposal is only applied once at the beginning of each iteration, the total number of generated candidate solutions is around $\mathrm{NFE}_{\mathrm{smax}}/NP$. This fraction of candidates is assumed to be low enough to consider the proposal an auxiliary search mechanism that only complements the search pattern exhibited by the underlying algorithm without incurring any major disruptions to it. Furthermore, this also keeps the computational overhead resulting from its incorporation inside reasonable bounds.

4. EXPERIMENTAL EVALUATION AND ANALYSIS

The well-established CEC2014 benchmark suite [26] was employed to evaluate the potential of the proposal to yield additional performance in different DE varieties. The types of the 30 functions comprising the suite are enumerated in Table 1. Furthermore, the nine considered DE varieties are concisely presented in Table 2 alongside the parameter configurations associated with the present modifications/extensions (these are in line with the recommendations in their original papers).

The experimental analysis is divided into four parts. The first and main part concerns the analysis of the impact the proposal has on different DE varieties. The second part relates to the assessment of its sensitivity to the choice of control parameters. The third and fourth parts comprise the ablation study of the proposal, in terms of the preselection and population update steps.

Table 1. Types of functions comprising the CEC2014 benchmark suite

Functions	type	search space (S)
f_1-f_3	unimodal	$[-100,100]^d$
f_4-f_{16}	simple multimodal	
f_{17}-f_{22}	hybrid	
f_{23}-f_{30}	composition	

Table 2. DE algorithm variants used to evaluate the proposed extension

Algorithm	modification/ extension parameters	note
DE	—	DE/rand/1/bin (standard/canonical DE algorithm)
BBDE-N [27]	—	Black-box DE; Normalised log-normal distribution sampling of F
QODE [7]	J_r=0.05	DE/rand/1/bin; QOBL employed for population initialisation and the occasional creation of new solutions
DEMM [23]	p_c=0.5→0.005, p_{MM}=0.05→0.9	DE/rand/1/bin; Macromutations are occasionally performed instead of the typical mutation and crossover
CDE [8]	m=10, $k\sim U[2,\sqrt{NP}]$	DE/rand/1/exp; Periodical clustering of the population using one-step of the k-means algorithm to create new solutions
JADE [19]	p=0.05	DE/current-to-$p$best/1/bin; Adaptation of F and CR; augmented current-to-best/1 mutation strategy
jDE [18]	$\tau 1$=$\tau 2$=0.1	DE/rand/1/bin; Self-adaptation of F and CR
RNDE [20]	N_{lb}=3, N_{ub}=10	Asynchronous DE/neighbour/1/bin; adaptive neighbour-based mutation scheme; Adaptation of CR
aDE [28]	—	DE/rand/1/exp; Self-adaptation of F and CR

4.1. SETUP

The benchmark functions were considered for $d\in\{10,30\}$. In accord with the CEC2014 competition rules (see [26]), every algorithm instance was run independently 51 times, each run terminating after $d\cdot 10^4$ function evaluations (FEs) or, earlier, if an optimisation error of $\Delta f<10^{-8}$ was reached (which was taken as $\Delta f=0$ and counted as a successful run). At the beginning of each algorithm run, the population was initialised uniformly at random inside S. The same parameter configuration, represented by NP=100, F=0.5, and CR=0.9 (one of the most common configurations [11]), was employed for each DE variant in the experiments. Furthermore, the proposal was empirically parametrised with n_c=3, λ=0.1, and k=7 (this choice is vindicated in the second part of the experimental analysis).

4.2. ANALYSIS OF THE IMPACT ON DIFFERENT DE VARIANTS

The results obtained in the conducted experiments are concisely presented in Tables 3 and 4. Reported are the differences in average optimisation errors, for

each function type, achieved by the DE variants listed in Table 2 when compared to their counterparts incorporating the proposed extension. Specifically, count summaries of Wilcoxon signed-rank test results [29] at a 95% confidence level are showcased, with ↓ denoting a statistically significant advantage in favour of the variant incorporating the proposal, ↑ an advantage for the variant without it, and ∘ indicating no significant difference. To make the interpretation easier, cases where the incorporation of the proposal improved upon a particular DE variant are highlighted in grey. Additionally, total counts are provided per algorithm variant and per benchmark function type, along with the results of the Wilcoxon test for pair-wise comparisons [29] (R^+/R^-).

Table 3. Comparison of DE variants with and without the proposed extension on different function types for d=10

Algorithm	f_1-f_3 ↓/∘/↑	f_4-f_{16} ↓/∘/↑	f_{17}-f_{22} ↓/∘/↑	f_{23}-f_{30} ↓/∘/↑	Total ↓/∘/↑	R^+/R^-
DE	0 / 3 / 0	6 / 6 / 1	0 / 6 / 0	3 / 5 / 0	9 / 20 / 1	291.0 / 144.0
BBDE-N	3 / 0 / 0	7 / 5 / 1	4 / 2 / 0	5 / 2 / 1	19 / 9 / 2	340.0 / 95.00
QODE	0 / 1 / 2	5 / 7 / 1	0 / 6 / 0	1 / 6 / 1	6 / 20 / 4	289.0 / 176.0
DEMM	2 / 1 / 0	5 / 7 / 1	0 / 6 / 0	2 / 6 / 0	9 / 20 / 1	323.5 / 141.5
CDE	0 / 3 / 0	3 / 8 / 2	0 / 3 / 3	2 / 6 / 0	5 / 20 / 5	246.0 / 189.0
JADE	0 / 3 / 0	4 / 8 / 1	1 / 5 / 0	2 / 6 / 0	7 / 22 / 1	254.0 / 181.0
jDE	0 / 3 / 0	8 / 4 / 1	0 / 6 / 0	4 / 4 / 0	12 / 17 / 1	303.0 / 132.0
RNDE	0 / 3 / 0	5 / 6 / 2	0 / 6 / 0	2 / 6 / 0	7 / 21 / 2	282.5 / 182.5
aDE	0 / 2 / 1	6 / 6 / 1	2 / 4 / 0	2 / 6 / 0	10 / 18 / 2	246.0 / 219.0
Total	5 / 19 / 3	49 / 57 / 11	7 / 44 / 3	23 / 47 / 2	84 / 167 / 19	—

Table 4. Comparison of DE variants with and without the proposed extension on different function types for d=30

Algorithm	f_1-f_3 ↓/∘/↑	f_4-f_{16} ↓/∘/↑	f_{17}-f_{22} ↓/∘/↑	f_{23}-f_{30} ↓/∘/↑	Total ↓/∘/↑	R^+/R^-
DE	0 / 3 / 0	7 / 3 / 3	1 / 5 / 0	1 / 3 / 4	9 / 14 / 7	292.5 / 142.5
BBDE-N	2 / 0 / 1	6 / 5 / 2	3 / 3 / 0	6 / 2 / 0	17 / 10 / 3	324.0 / 111.0
QODE	0 / 0 / 3	4 / 6 / 3	0 / 4 / 2	2 / 0 / 6	6 / 10 / 14	109.0 / 356.0
DEMM	2 / 1 / 0	4 / 5 / 4	1 / 3 / 2	2 / 3 / 3	9 / 12 / 9	254.5 / 210.5
CDE	1 / 1 / 1	2 / 8 / 3	1 / 2 / 3	2 / 2 / 4	6 / 13 / 11	202.5 / 232.5
JADE	1 / 1 / 1	4 / 7 / 2	1 / 5 / 0	3 / 3 / 2	9 / 16 / 5	324.0 / 141.0
jDE	1 / 2 / 0	6 / 5 / 2	2 / 3 / 1	3 / 3 / 2	12 / 13 / 5	283.0 / 182.0
RNDE	1 / 2 / 0	5 / 7 / 1	1 / 5 / 0	0 / 4 / 4	7 / 18 / 5	236.0 / 229.0
aDE	0 / 3 / 0	4 / 7 / 2	0 / 3 / 3	3 / 3 / 2	7 / 16 / 7	172.0 / 263.0
Total	8 / 13 / 6	42 / 53 / 22	10 / 33 / 11	22 / 23 / 27	82 / 122 / 66	—

The overall outlook for the proposal is promising, as shown by the highlighted totals and the sums of ranks (R^+/R^-). This is especially the case of d=10, where it brings performance gains to the corresponding baseline variants in almost all cases. However, for d=30, although still able to yield additional performance for most of the considered DE variants, including the canonical DE, the benefits it brings are not as clear-cut as before. This goes to show that the intricate interactions between the proposed extension and already present extensions/modifications are indeed difficult to anticipate. Moreover, they only seem to be emphasised by increasing problem dimensionality. Such results provide further support for the earlier mentioned findings in [9, 10]. However, expecting that the presented extension, or any other extension for that matter, would bring exclusively positive effects is certainly unreasonable—there cannot be one size that fits all.

Delving deeper into comparisons involving particular algorithm variants, it can be noted that some are better suited and more receptive to the proposed extension, such as BBDE-N, jDE, and JADE, whilst some are less so, such as QODE and CDE. Little additional insight into these differences is provided by the convergence graphs in Fig. 2 (showcasing median runs) and the plots of average population diversity (APD) [30] in Fig. 3. Interestingly, for both the most and least receptive variants (BBDE-N and QODE, respectively), the incorporation of the proposal led, in many instances, to a similar reduction in population diversity throughout the optimisation process. On the other hand, in the majority of instances it had little effect on population

change for the canonical algorithm. Perhaps, the reasons for lacklustre receptiveness, observed in some variants, should be sought after in the modifications/extensions they already incorporate. However, this would require a dedicated and intricate analysis since e.g. CDE and QODE (and, in fact, all the other considered algorithm varieties) feature extensions/modifications based on completely different concepts. (Such an analysis is out-of-scope of the presented work.) Regardless, DE variants incorporating different parameter control schemes, such as jDE, JADE, and RNDE, seem to offer more fruitful ground in this regard. As a side note, the incorporation of the proposed extension preserves the mutual rankings of the considered DE variants (obtained through the Friedman test [29]) to a large degree, as indicated by both Pearson's and Spearman's-rank correlation coefficients (well-above 0.8 in all cases). This suggests that the impact of the proposal across the different algorithm varieties is relatively uniform. A similar note can be made about the computational overhead resulting from incorporating the proposal, which is reasonable as expected and ranges from around 4.5% to 6.1% (at least when estimated across the DE variants shown in the graphs/plots in Fig. 2 and Fig. 3).

An interesting perspective can be gained by turning attention to the performance across individual types of functions. The proposed extension seems highly effective for the simple multimodal functions (f_4 - f_{16}), as demonstrated by the total volume of statistically significant differences in its favour across all considered DE variants. It is also overall beneficial for the other types of functions when tackling lower-dimensional problem instances. However, its benefits diminish more or less with the increase in problem dimensionality on the composition (f_{23}-f_{30}) and the hybrid functions (f_{17}-f_{22}) depending on the DE variant at hand. However, by disregarding the less receptive varieties of the algorithm, those benefits are rendered more apparent. Inverting the narrative and taking a practical stance, to get the most out of the proposed extension, an appropriate DE variant needs to be found in which to incorporate it. Certainly, this is easier said than done. Yet, given the ease of incorporating it into virtually any algorithm variant and the potential performance gains it might bring, this should be worth investigating when tackling certain problem classes. To supplement the merit of such a statement, an additional experiment was conducted utilising the CEC2022 benchmark suite [31]. The 12 functions comprising the suite are listed in Table 5. As per the competition rules, problem dimensions of $d \in \{10,20\}$ were considered, and every algorithm instance was run independently 30 times, each run terminating after $2 \cdot 10^5$ or 10^6 FEs for d=10 and d=20, respectively, or earlier if an optimisation error of $\Delta f < 10^{-8}$ was reached. The population initialisation procedure and the parameter configurations for the DE variants and the proposal were kept the same as in the case of the CEC2014 benchmark. The obtained results are summarised in Table 6 in terms of the Wilcoxon test for pairwise comparisons (R^+/R^-).

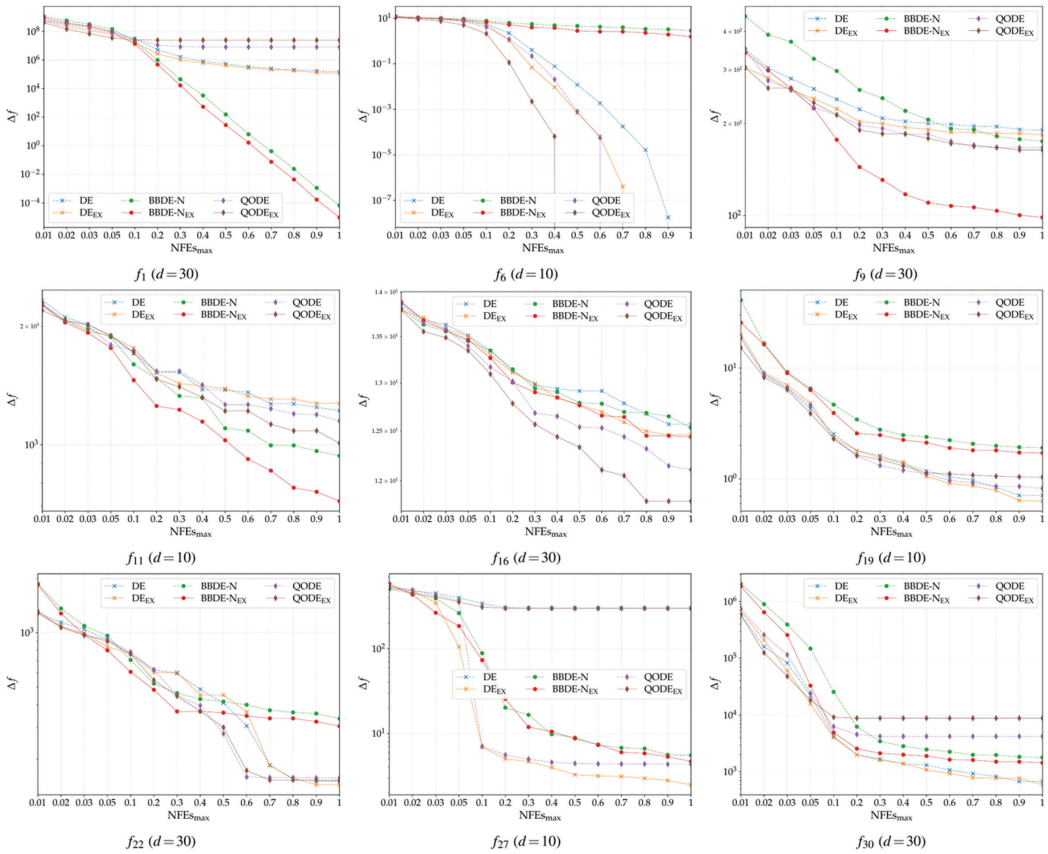
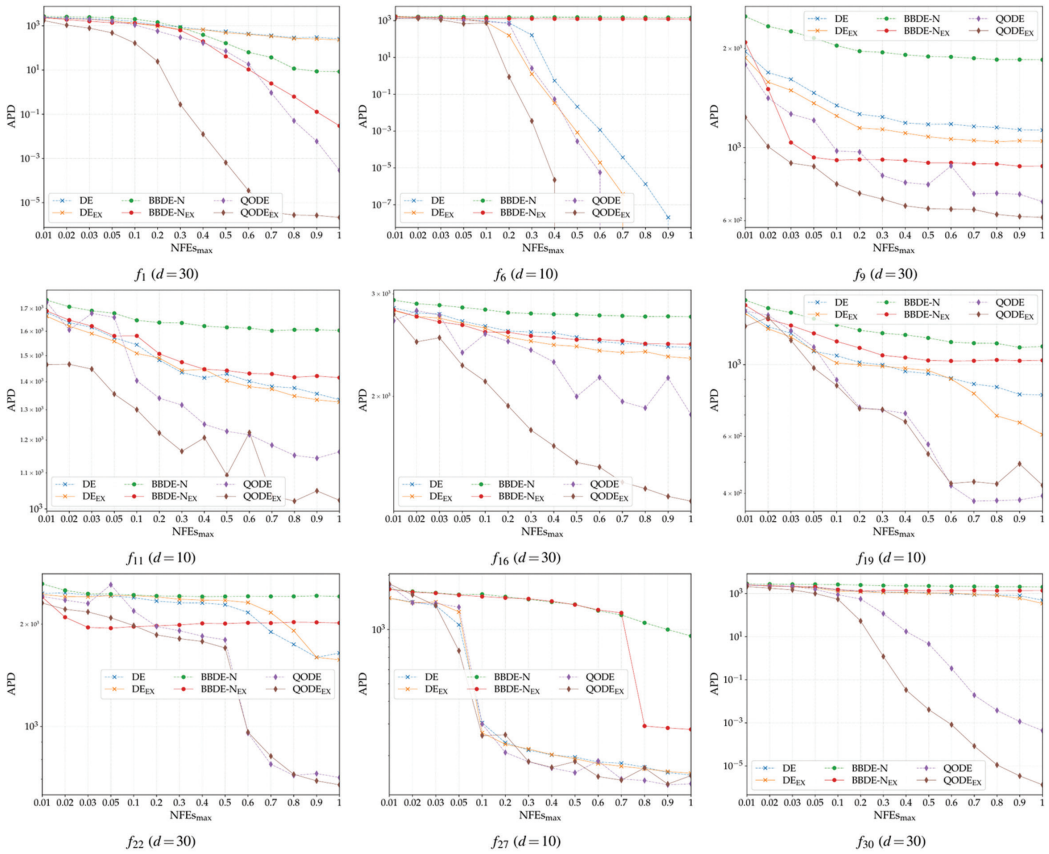


Fig. 2. Convergence graphs for a few select DE variants on several functions



The overall perspective observed on the first benchmark suite remains largely unaltered. The conclusion that the proposal is beneficial in most cases is supported by the results on the CEC2022 benchmark as well, whilst the increase in problem dimensionality certainly exacerbates the issues related to incorporating additional search mechanisms into existing algorithms. As for individual DE variants, those incorporating parameter control, like JADE, jDE, and RNDE, indeed seem like a more suitable starting point for practitioners aiming to incorporate the proposed extension, as the benefits are unimpeded even by the dimenensionality increase, whilst other ones (e.g. QODE) should perhaps be avoided. Interestingly, a turn can be observed with BBDE-N for d=20 on the CEC2022 suite, somewhat skewing the narrative of it being overwhelmingly receptive, as it did not previously show any notable overall performance decline with increased problem dimensionality. This corroborates the premise that there cannot be any one extension that is beneficial in all situations, either in the sense of underlying algorithms or problem instances.

Table 5. Types of functions comprising the CEC2022 benchmark suite

Functions	type	search space (S)
f_1	unimodal	$[-100,100]^d$
f_2-f_4	basic multimodal	
f_5-f_8	hybrid	
f_9-f_{12}	composition	

Table 6. Comparison of DE variants with and without the proposed extension on the CEC2022 suite for d=10 and d=30

Algorithm	d=10 R^+/R^-	d=30 R^+/R^-
DE	39.0 / 27.0	45.5 / 32.5
BBDE-N	42.5 / 23.5	28.5 / 49.5
QODE	17.0 / 49.0	0.0 / 78.0
DEMM	37.0 / 29.0	48.5 / 29.5
CDE	53.0 / 13.0	31.5 / 34.5
JADE	34.0 / 32.0	45.5 / 20.5
jDE	35.0 / 31.0	61.5 / 4.5
RNDE	47.0 / 19.0	41.5 / 24.5
aDE	43.5 / 34.5	38.5 / 27.5

The proposal surely has its merits, and besides the already discussed, it lead to higher average success rates (representing the ratio of successful and the total number of algorithm runs) in most of the considered DE variants. Naturally, it has its limitations as well. The noticeable overall performance drop with increased problem dimensionality, as observed with some algorithm varieties, is a valid concern that warrants further investigation. The reason(s) for this might be sought after in a couple of design choices made, like the employed surrogate model and the distribution used for sampling the noise.

4.3. ANALYSIS OF THE SENSITIVITY TO THE CHOICE OF CONTROL PARAMETERS

The proposal introduces three control parameters, n_c, λ, and k, that may necessitate tuning. In order to assess its sensitivity to the configuration of these parameters, an analysis was conducted, where a few different values for each parameter have been evaluated separately. Only the canonical DE algorithm and QODE (as the least receptive algorithm) were considered in the experiments, with the previously set parameter configuration serving as the basis (i.e., the one-factor-at-a-time method was employed). The obtained results are concisely presented in Tables 7 and 8, where the average ranks ($\bar{R}$) obtained from the Friedman test are shown (lower values are better). The best performing configurations per problem dimensionality are given in bold, whilst the best totals are highlighted in grey.

Table 7. Impact of the control parameters on overall performance of extended DE across all functions for d=10 / d=30

λ=0.1 and k=7				
n_c	3	5	7	10
$\bar{R}$	2.63 / **2.28**	2.67 / 2.50	**2.10** / 2.67	2.60 / 2.55
n_c=3 and k=7				
λ	0.1	0.3	0.5	1
$\bar{R}$	2.40 / **2.27**	2.33 / 2.52	**2.27** / 2.50	3.00 / 2.72
n_c=3 and λ=0.1				
k	3	5	7	10
$\bar{R}$	2.42 / 2.58	**2.32** / 2.79	2.55 / **2.20**	2.72 / 2.43

Table 8. Impact of the control parameters on overall performance of extended QODE across all functions for d=10 / d=30

λ=0.1 and k=7				
n_c	3	5	7	10
$\bar{R}$	**2.23** / 2.40	2.52 / 2.60	2.87 / 2.72	2.38 / **2.28**
n_c=3 and k=7				
λ	0.1	0.3	0.5	1
$\bar{R}$	**2.22** / 3.08	2.37 / 2.53	2.62 / 2.23	2.80 / **2.15**
n_c=3 and λ=0.1				
k	3	5	7	10
$\bar{R}$	2.70 / 2.25	2.37 / **2.17**	**2.18** / 2.27	2.75 / 2.67

At a glance, the differences in terms of average ranks are somewhat minor, suggesting that the differences in overall performance cannot be substantial. Correspondingly, no single parameter stands out in terms of influence on behaviour of the proposal (obviously, larger values of n_c implicate an increase in execution time of the proposal). Nevertheless, there are some differences between the extended canonical DE and extended QODE algorithms depending on the parameter configuration used. For the canonical DE algorithm, the base configuration (n_c=3, λ=0.1, and k=7) is never ranked best for d=10, but always is for d=30. Conversely, for QODE, the base configuration is ranked best for d=10, but never for d=30. Interestingly, the extended QODE profited notably from an increase in the scaling parameter λ when d=30, whilst the other two parameters did not impact its performance to nearly such an extent. This suggests that some performance gains can be had with parameter tuning, even with somewhat unreceptive algorithm varieties, and that λ is probably the parameter that should be tuned first.

From what can be gathered, it seems that the proposed extension is rather robust with respect to the choice of its control parameters, which is a desirable feature, as it presumes that no extensive tuning is necessary to potentially achieve satisfactory benefits from its incorporation (of course, having a receptive algorithm in mind). Yet, these reported results should be taken with a grain of salt, since they are preliminary and not comprehensive in terms of considered DE variants (however, such an analysis would have been computationally prohibitive).

4.4. ANALYSIS OF THE IMPACT OF THE PRESELECTION METHOD

As earlier elaborated upon, the choice of weighted k-NN-R as the surrogate model in the proposal is not accidental. To justify this, an experimental analysis has been conducted. Two versions of the proposal (as incorporated into the canonical DE algorithm) have been devised, where the weighted k-NN-R surrogate was replaced by random preselection of a candidate for real evaluation and by the (vanilla) k-NN-R algorithm. The obtained results are concisely reported in Table 9. Again, the Friedman test was employed to assess the differences in overall performance. Correspondingly, the presentation of the results is the same as previously.

Table 9. Impact of the preselection method on overall performance across all functions for d=10 / d=30

Preselection	random	k-NN-R	weighted k-NN-R
$\bar{R}$	2.07 / 1.92	2.07 / 2.20	**1.87 / 1.88**

Although the differences in average ranks are not substantial, it is obvious that consistently better overall performance is achieved by utilising the weighted k-NN-R algorithm compared to the other considered options. Surprisingly, random preselection led to even slightly better performance on the functions for d=30 than the plain k-NN-R algorithm. This can be interpreted to support the earlier concerns raised that there may be situations when all generated candidates have the same k neighbours and therefore the same predicted quality. As the optimisation process progresses and the population shrinks, this may occur more frequently, especially if the parameter λ is set to a relatively small value.

4.5. ANALYSIS OF THE IMPACT OF THE POPULATION UPDATE PROCEDURE

Only the worst solution in the current population of the underlying algorithm is considered for replace-

ment by the proposal. This design choice implies a high selection pressure, as noted earlier. To justify it and to show that this high selection pressure is not detrimental, an experimental analysis has been conducted. To this end, three versions of the proposal (as incorporated into the canonical DE algorithm) have been devised, which differ in the population update procedure. The first two, attempt to replace the best and the closest (in terms of Euclidean distance), respectively, whereas the third version adopts the procedure from [14] (the best, closest, and worst are sequentially considered for replacement). The results obtained for these versions are reported in Table 10, where once again the average ranks from the Friedman test are shown.

Table 10. Impact of the population update procedure on overall performance across all functions for d=10 / d=30

Replacement	best	*closest*	worst	sequential
$\tilde{R}$	2.72 / 2.80	2.35 / 2.57	**2.12 / 2.18**	2.82 / 2.45

The differences in average ranks are obvious, but not major. Nevertheless, they are clearly in favour of replacing the worst solution in the population. Conversely, the procedure exerting the lowest selection pressure yielded the poorest overall performance figures. Referring to the earlier discussion on the total number of candidate solutions generated by the proposal, this might not be surprising, since it is relatively small and a high selection pressure might be necessary for the proposal to make an effect on the search behaviour of the underlying algorithm.

5. CONCLUSION

Differential evolution has been successfully applied to ("black-box") optimisation tasks in numerous technical or engineering systems. The trend of introducing new modifications or extensions for the algorithm is apparent in the literature. This paper follows the trend by proposing an extension that is easily grafted into not only the canonical algorithm but also its modified/extended variants. Infusing the population centroid at the beginning of each iteration with normal noise defined by the component-wise standard deviations of the population, multiple candidates are generated. The most promising, as determined by the weighted k-NN-R surrogate, is evaluated and trialled against the worst solution in the current population for replacement. Promising results have been achieved considering both a challenging benchmark suite and a diverse set of DE variants. Yet, they provide also further support for the findings in [9, 10] that it is unreasonable to expect exclusively beneficial interactions with different already present extensions/modifications of the underlying algorithm.

Although the proposal was tested with a significant number of DE variants, considering additional ones would be beneficial, of course. However, testing the proposed extension with other types of algorithms, such as those comprising the rich set of swarm intelligence algorithms, is another avenue for future work that would be a major step in establishing a wider picture of it. Further, the proposal introduces three control parameters, which is not a negligible number, to say the least. Although the limited sensitivity analysis suggests it is robust to the choice of parameter configuration, a more comprehensive analysis would be necessary to provide better-grounded evidence for this. Additionally, the k-NN-R surrogate might be a limiting factor in applications on higher-dimensional problems due to ill-defined distance metrics. Therefore, replacing it with a more sensible but more complex model, like a radial basis function network, might be worthwhile in such cases. Finally, although some insights into the receptiveness of individual DE variants to the proposed extension were provided, an in-depth investigation into the intricacies and interactions of individual extensions along with a methodological framework enabling it for different extensions would represent a useful tool for practitioners and would be a major endeavour and an avenue for future work.

ACKNOWLEDGEMENT

This research was funded by the European union–NextGenerationEU through the project "Advanced Machine Learning Methods for Optimization and Robotic Vision" – SUPORT (581-UNIOS-36).

DECLARATION OF THE USE OF AI TOOLS

The authors declare that no AI tools were used for content generation.

REFERENCES

[1] A. E. Eiben, J. E. Smith, "Introduction to Evolutionary Computing", 2nd Ed., Springer-Verlag Berlin Heidelberg, 2015.

[2] L. Zheng, S. Chen, "A learning and sampling strategies collaborative aided differential evolution algorithm and application in hydrodynamic optimization", Results in Engineering, Vol. 29, 2026, p. 109261. doi: 10.1016/j.rineng.2026.109261

[3] F. S. Rohman, S. R. W. Alwi, S. N. A. A. Termizi, H. A. Er, D. Muhammad, A. Azmi, "Adaptive differential evolution approaches in real-time optimization of co-generation systems for enhanced energy minimization", Thermal Science and Engineering Progress, Vol. 71, 2026, p. 104534. doi: 10.1016/j.tsep.2026.104534

[4] T. Zhao, Z. Cui, Z. Wu, H. Duan, J. Chen, "Many-objective virtual power plants resource scheduling based on evolutionary multifactorial optimiza-

tion", Expert Systems with Applications, Vol. 323, 2026, p. 132186. doi: 10.1016/j.eswa.2026.132186

[5] R. Laorboot, N. Sombattheera, P. Akrawattananupong, P. Sungkaeo, C. Thaijiam, "Optimizing fractal geometries for multifrequency antennas in industrial, scientific, and medical bands using evolutionary particle swarm optimization: Modeling and design considerations", Engineering Applications of Artificial Intelligence, Vol. 177, 2026, p. 114732. doi: 10.1016/j.engappai.2026.114732

[6] K. V. Price, R. M. Storn, J. A. Lampinen, "Differential Evolution: A Practical Approach to Global Optimization", 1st Ed., Springer-Verlag Berlin Heidelberg, 2005. doi: 10.1007/3-540-31306-0

[7] S. Rahnamayan, H. R. Tizhoosh, M. M. A. Salama, "Opposition-based differential evolution", IEEE Transactions on Evolutionary Computation, Vol. 12, No. 1, 2008, pp. 64-79. doi: 10.1109/TEVC.2007.894200

[8] Z. Cai, W. Gong, C. X. Ling, H. Zhang, "A clustering-based differential evolution for global optimization", Applied Soft Computing, Vol. 11, No. 1, 2011, pp. 1363-1379. doi: 10.1016/j.asoc.2010.04.008

[9] D. Bajer, "Adaptive k-tournament mutation scheme for differential evolution", Applied Soft Computing, Vol. 85, 2019, p. 105776. doi: 10.1016/j.asoc.2019.105776

[10] D. Bajer, B. Zorić, M. Dudjak, "SMOTE inspired extension for differential evolution", Proceedings of the 10th International Conference on Bioinspired Optimization Methods and Their Applications, Maribor, Slovenia, 17-18 November 2022, pp. 216-231.

[11] D. Bajer, "Parameter control for differential evolution by storage of successful values at an individual level", Journal of Computational Science, Vol. 68, 2023, p. 101985. doi: 10.1016/j.jocs.2023.101985

[12] W. F. Mbasso, H. M. H. Farh, A. Harrison, A. A. Al-Shamma, "An adaptive multi-stage clustering-based differential evolution algorithm with dynamic niching for robust solving of high-dimensional nonlinear equation systems", Applied Soft Computing, Vol. 191, 2026, p. 114666. doi: 10.1016/j.asoc.2026.114666

[13] J. Gou, L. Sun, J. Liu, Z. Song, G. Guo, S.-G. Gao, K. Liu, N. Matsveichuk, Y. Sotskov, "An adaptive differential evolution with deeply informed mutation strategy and historical information for numerical optimization", Information Sciences, Vol. 739, 2026, p. 123146. doi: 10.1016/j.ins.2026.123146

[14] D. Bajer, B. Zorić, M. Dudjak, "Enhancing differential evolution through convex combinations of the whole population", Proceedings of the IEEE Zooming Innovation in Consumer Technologies International Conference, Novi Sad, Serbia, 3-4 June 2026, pp. 87-92. doi: 10.1109/ZINC69910.2026.11655688

[15] S. Das, S. S. Mullick, P. N. Suganthan, "Recent advances in differential evolution–an updated survey", Swarm and Evolutionary Computation, Vol. 27, 2016, pp. 1-30. doi: 10.1016/j.swevo.2016.01.004

[16] P. Verma, R. Chaudhary, H. Gupta, R. Salgotra, "Three decades of differential evolution: A bibliometric analysis (1995-2025)", Expert Systems with Applications, Vol. 313, 2026, p. 131451. doi: 10.1016/j.eswa.2026.131451

[17] D. Bajer, G. Martinović, J. Brest, "A population initialization method for evolutionary algorithms based on clustering and Cauchy deviates", Expert Systems with Applications, Vol. 60, 2016, pp. 294-310. doi: 10.1016/j.eswa.2016.05.009

[18] J. Brest, S. Greiner, B. Bošković, M. Mernik, V. Žumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems", IEEE Transactions on Evolutionary Computation, Vol. 10, No. 6, 2006, pp. 646-657. doi: 10.1109/TEVC.2006.872133

[19] J. Zhang, A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive", IEEE Transactions on Evolutionary Computation, Vol. 13, No. 5, 2009, pp. 945-958. doi: 10.1109/TEVC.2009.2014613

[20] H. Peng, Z. Guo, C. Deng, Z. Wu, "Enhancing differential evolution with random neighbors based strategy", Journal of Computational Science, Vol. 26, 2018, pp. 501-511. doi: 10.1016/j.jocs.2017.07.010

[21] S. Rahnamayan, H. R. Tizhoosh, M. M. A. Salama, "Quasi-oppositional differential evolution", Proceedings of the IEEE Congress on Evolutionary Computation, Singapore, 25-28 September 2007, pp. 2229-2236. doi: 10.1109/CEC.2007.4424748

[22] Z. H. Zhan, J. Zhang, "Enhance differential evolution with random walk", Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation, Philadelphia, PA, USA, 2012, pp. 1513-1514. doi: 10.1145/2330784.2331020

[23] G. Martinović, D. Bajer, "Data clustering with differential evolution incorporating macromutations", Proceedings of the 4th International Conference on Swarm, Evolutionary, and Memetic Computing, Chennai, India, 19-21 December 2013, pp. 158-169. doi: 10.1007/978-3-319-03753-0_15

[24] H. Tong, C. Huang, L. L. Minku, X. Yao, "Surrogate models in evolutionary single-objective optimization: A new taxonomy and experimental study", Information Sciences, Vol. 562, 2021, pp. 414-437. doi: 10.1016/j.ins.2021.03.002

[25] R. Biedrzycki, J. Arabas, D. Jagodziński, "Bound constraints handling in differential evolution: An experimental study", Swarm and Evolutionary Computation, Vol. 50, 2019, p. 100453. doi: 10.1016/j.swevo.2018.10.004

[26] J. J. Liang, B. Y. Qu, P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization", Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China, Technical Report 201311, and Nanyang Technological University, Singapore, Technical Report, 2013.

[27] A. Vodopija, T. Tušar, B. Filipič, "Comparing black-box differential evolution and classic differential evolution", Proceedings of the Genetic and Evolutionary Computation Conference Companion, Kyoto, Japan, 15-19 July 2018, pp. 1537-1544. doi: 10.1145/3205651.3208309

[28] N. Noman, D. Bollegala, H. Iba, "An adaptive differential evolution algorithm", Proceedings of the IEEE Congress on Evolutionary Computation, New Orleans, LA, USA, 5-8 June 2011, pp. 2229-2236. doi: 10.1109/CEC.2011.5949891

[29] J. Derrac, S. García, D. Molina, F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms", Swarm and Evolutionary Computation, Vol. 1, No. 1, 2011, pp. 3-18. doi: 10.1016/j.swevo.2011.02.002

[30] E. Rodríguez-Esparza, B. Morales-Castañeda, I. Aranguren, A. Valdivia, M. Navarro, D. Oliva, "Addressing limitations and inaccuracy of diversity metrics in evolutionary algorithms with an accurate dimension-wise diversity", Proceedings of the IEEE Congress on Evolutionary Computation, Hangzhou, China, 8-12 June 2025, pp. 1-8. doi: 10.1109/CEC65147.2025.11043022

[31] A. Kumar, K. V. Price, A. W. Mohamed, A. A. Hadi, P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2022 special session and competition on single objective bound constrained numerical optimization", Nanyang Technological University, Singapore, Technical Report, 2021.