Soft Computing manuscript No.
(will be inserted by the editor)

Cluster-centroid-based mutation strategies for Differential Evolution

Giovanni Iacca* · **Vinícius Veloso de Melo**

Received: date / Accepted: date

Abstract Mutation is the main operator of Differential Evolution (DE), as it is responsible for combining the information of distinct solutions to generate a donor vector. Aiming at improving the search effectivity of DE, previous research incorporated the calculation of centroids into the DE mutations. In some of the existing methods the centroids are simply calculated as the center of some selected solutions (or, the entire population); in other cases, one-step clustering is used to perform a local search, or the centroids themselves are used as actual solutions in the population. As opposed to these methods, in this paper we extend some traditional DE mutation strategies to incorporate centroids calculated by deterministic hierarchical clustering. Experimental results on two sets of well-known benchmark problems show that the proposed cluster-centroid-based mutation strategies outperform, in general, the traditional rand/1 strategy, as well as several metaheuristics from the literature. Therefore, the use of clustering is an effective way to improve the search performance that could be exploited also in other population-based metaheuristics.

Keywords Clustering · Mutation Strategy · Differential Evolution · Global Optimization · Machine Learning.

Vinícius Veloso de Melo
Department of Analytics
Wawanesa Insurance
Winnipeg, MB, Canada
E-mail: vvdemelo@wawanesa.com
ORCID: 0000-0002-7507-1685

Giovanni Iacca
Department of Information Engineering and Computer Science
University of Trento
Via Sommarive 9, 38123 Povo (TN), Italy
Tel: +39 0461 28 5220
E-mail: giovanni.iacca@unitn.it
ORCID: 0000-0001-9723-1830

1 Introduction

Metaheuristics, such as nature-inspired algorithms [1–9], are techniques employed to solve difficult optimization problems. However, it is known that these techniques do not guarantee that the optimal solution will be found even if the search space is reduced [10, 11]. Moreover, optimization processes based on metaheuristics are usually time-consuming. In an attempt to reduce these issues, previous research has investigated hybrid approaches combining metaheuristics and machine learning [11–14].

One important aspect in metaheuristics is a proper balance between *exploration* and *exploitation*. More specifically, the *exploration* phase searches, usually at the beginning of the search process, for promising regions of the search space, i.e., regions where high-quality solutions may be found [13, 15]. Then, the *exploitation* phase is meant to refine those high-quality solutions. A possible refining approach can be the use of local optimization methods, in order to reach the local optimum quickly and accurately [16–18].

One of the most popular metaheuristics is currently Differential Evolution (DE), originally introduced in [19] and later improved in various works, such as [20–26]. The main reason behind the success of DE is its inherent self-adaptation, which in turn balances exploration and exploitation. In particular, the key element of the algorithm is its *mutation strategy*: essentially, DE mutations create additive solution perturbations by means of linear combinations of relative distances between the solutions in the current population (in the form of *differential vectors*, hence the name of the algorithm). As pointed out by Feoktistov [27], this mechanism allows the algorithm to perform adaptive mutations: at the beginning of the evolutionary process, those distances are large, thus mutation is more explorative; at the end, distances tend to become smaller, thus mutation becomes more exploitative and it allows to refine the search.

Because of the importance of mutations in DE, several studies have focused on the development of more efficient mutation strategies, including ensembles [28, 29] and self-adaptation thereof [30, 31]; others evaluated how the mutation strategy affects the population diversity [32, 33]. In this context, a successful research direction concerns the use of *opposition-based* sampling, where solutions are generated by calculating opposite points w.r.t. the problem bounds [34], or *center-based* sampling, where solutions are generated w.r.t. the central point (i.e., the *centroid*) either of the entire population, as in [35, 36], or of the top individuals in the population. The latter mechanism, initially introduced in [37], has been used in [38, 39] for the DE population initialization phase, and in a number of other DE variants, either for single-objective optimization [40–43], including large-scale optimization [44, 45], or for multi-objective optimization [46]. Of note, the same center-based mechanism has been used also in algorithms different from DE, such as cooperative co-evolution [47], or Particle Swarm Optimization [48].

Different from the above center-based sampling mechanism, other works have proposed the use of centroids of groups of *random* solutions. In particular, Fan and Lampinen [49] suggested the stochastic application of the Trigonometric Mutation Operator (TMO), which resulted in the so-called Trigonometric DE (TDE). In TDE, the donor vector is the center point of a hyper-geometric triangle, i.e., the mean of three randomly selected solutions. As a matter of fact, this mean solution is a centroid, although the authors do not employ this term. The mutation strategy is then the sum of three weighted differential vectors. Because of the weights, TMO inherently acts as a local search procedure, especially when the solutions are in a small neighborhood. Similarly to TDE, Ali et al. [50] proposed another centroid-based DE, where a Centroid Mutation Operator calculates the mean among the *current best solution* and two other solutions randomly selected from the current population. The authors concluded that this simple modification achieved better mean results in fewer function evaluations than both the basic DE and TDE.

It can be seen then that several works so far have proposed using centroids in DE. However, it is important to note that those works do not perform any explicit form of *clustering* [51] to calculate those centroids. Instead, they calculate the centroid on the entire population, a subset of top individuals, or a subset of random individuals. On the contrary, a clustering method organizes (in an unsupervised way) a set of data samples into different groups (clusters), where the samples in the same cluster are similar to each other (according to a similarity measure) and distinct from those in the other clusters. To the best of our knowledge, the only two works that applied an explicit clustering technique into a DE framework are the ones by Cai et al. [52], and Wang et al. [53].

In the first work [52], a hybrid DE based on a one-step k-means clustering, called clustering-based DE (C-DE), is proposed. Every T generations, C-DE runs *one step* of the k-means clustering method to group the current population into k clusters. The cluster centroids are taken as new solutions and added to the population according to some criteria. Therefore, they are not employed in a new mutation strategy. The authors state that by acting as a k multi-parent crossover, the one-step k-means clustering utilizes the information available in the current population more efficiently, and allows a better balance between exploration and exploitation. The experimental analysis showed, in fact, that C-DE outperformed several more elaborated DE variants. However, running a single step of the k-means clustering method may not be considered as a "true" clustering method (i.e., one that groups the data samples by applying an iterative process aimed at optimizing a certain objective function).

In the second work [53], hierarchical clustering method is incorporated into DE to improve its performance. Initially, DE runs for $T - 1$ generations, using the traditional *rand/1* mutation strategy. Then, every T generations the method clusters the population (of size N) into k groups, and calculates the centroid of each group as well as a global centroid, which is the centroid of the k centroids. Finally, a new population is formed by the k centroids and $N - k$ solutions randomly generated around the global centroid. Experiments performed on a set of well-known benchmark functions showed that the proposed method, named W-CDE, was able to find good solutions more quickly than many compared algorithms.

Motivated by the two related works above, in this paper we propose an extension of several traditional mutation strategies, based on the incorporation of centroids calculated by a "true" clustering method. Our hypothesis is that, while clusters can be used to identify the concentration of solutions and detect promising regions of the search space, the centroids (i.e., the centers of the clusters) can condense information of the solutions within the clusters. Therefore, as high-quality solutions can be located in promising regions, a clustering method can be used to *learn* the characteristics of those solutions and compactly represent this knowledge into the corresponding centroid. Such information can be employed by mutation strategies to generate new, possibly better, solutions. Consequently, the acquired "knowledge" can be useful to guide the search to promising regions more efficiently. From this point of view, incorporating an explicit clustering mechanism into the DE scheme in order to extract useful information from the structure of the population may provide a benefit over the existing approaches discussed above, that as seen calculate the centroid either of the entire population or of a subset of individuals. While this research direction appears particularly intriguing, yet it

remains largely unexplored, with the exception of the two works mentioned earlier, i.e., [52] and [53].

Based the aforementioned hypothesis, we tackle the following fundamental research questions:

1. Is it beneficial to use a "true" clustering method during the DE process, in order to calculate centroids?
2. Is it possible to use the cluster centroids in the mutation strategies? If so, what kind of cluster-centroid-based mutation strategy is more efficient?
3. What is the effect of the parametrization of the clustering method (in particular, the number of clusters), as well as the parametrization of the other parts of DE when clustering is used?

The main contribution of this work consists in the proposal of eight cluster-centroid-based mutation strategies, which use the cluster centroids in different ways to obtain different balances between exploration and exploitation. The main differences w.r.t. the two cluster-centroid-based methods introduced in [52] and [53] can be summarized as follows:

1. we use a deterministic hierarchical clustering (UPGMC), differently from the stochastic k-means used in [52];
2. we apply clustering at every generation of DE, instead of periodically as in [52, 53];
3. we use centroids (calculated by means of clustering) *into the mutation strategy*, not as actual solutions to be evaluated, as in [52, 53].

To evaluate the effectiveness of our proposal, we perform a thorough comparison of our proposed cluster-centroid-based mutation strategies with the two related methods proposed in [52] and [53], as well as several metaheuristics from the state-of-the-art, on two sets of widely investigated benchmark functions. Our analysis allows us to discuss in detail the effect of the proposed cluster-centroid-based mutation strategies, their strengths, and their weaknesses. Of note, while the experiments carried out in this paper use a specific clustering method, namely the *Unweighted Pair-Group Method using Centroids* (UPGMC) [54], the proposed mutation strategies are not tied specifically to it. In other words, different clustering methods might provide different performances, being useful in different cases.

The rest of the paper is organized as follows: in Section 2, the background concepts on DE UPGMC are introduced; Section 3 describes our proposed cluster-centroid-based mutation strategies; the computational experiments are then presented in Section 4; finally, Section 5 concludes the paper.

2 Background

In this section, we provide background information on DE and clustering, particularly the UPGMC technique used in our experimentation.

2.1 Differential Evolution

Differential Evolution [19] is a population-based stochastic search metaheuristic based on evolutionary concepts. One of its main advantages is that it has few control parameters, which makes the algorithm easy to configure to obtain acceptable performance. Over the years, the research community has been using and improving DE to successfully solve many kinds of hard-to-solve optimization problems [12, 13, 16, 18, 20, 55–64].

Algorithm 1 Pseudo-code of Differential Evolution

```
1:  procedure DifferentialEvolution(N, F, CR)
2:      for each i ∈ {1, ..., N} do
3:          x_i ← initialize()        ▷ randomly init. in problem bounds
4:          F_{x_i} ← f(x_i)
5:      end for
6:      while stopping criterion is not satisfied do
7:          for each i ∈ {1, ..., N} do
8:              v_i ← mutation({x_1, ..., x_N}, F)        ▷ see Eq. (1-3)
9:              u_i ← crossover(v_i, x_i, CR)             ▷ see Eq. (4)
10:             F_{u_i} ← f(u_i)
11:             if F_{u_i} < F_{x_i} then                 ▷ assuming minimization
12:                 x_i ← u_i
13:                 F_{x_i} ← F_{u_i}
14:             end if
15:         end for
16:     end while
17: end procedure
```

The pseudo-code of the basic DE is presented in Algorithm 1. At the beginning of the algorithm, a population of N solutions (i.e., vectors of D decision variables) is randomly initialized inside the problem bounds (line 3) and evaluated using the fitness function $f()$ specific for the problem at hand (line 4). Then, the algorithm performs an iterative evolutionary process, where each iteration is a *generation* consisting of: mutation (line 8), crossover (line 9), evaluation (line 10) and selection (line 11). This iterative process is continued until a stopping criterion (usually set on the maximum number of generations, or the maximum number of function evaluations) is satisfied.

As it can be noticed in the pseudo-code, the crossover operation is applied *after* mutation. This makes the mutation strategy, in a certain sense, the main search operator in DE: in fact, mutation defines the way solutions are selected from the population and combined to direct the search. Thus, most of the research on DE has focused on improving this operator [60, 65–67]. The various existing DE configurations, differing on the kind of mutation and crossover strategy, are usually indicated using the so-called DE $base/num/cross$ notation. The first element, $base$, indicates the parent vector, which can be either a random or a specific solution from the current population, for instance the best one. The second element, num, indicates the number of difference vectors that

are added to the base vector, generating a *donor vector*. Finally, the type of crossover that combines the base and donor vectors to generate the *trial solution* (i.e., the new solution to be evaluated) is identified by $cross$. Using the $base/num$ notation (omitting the crossover strategy $cross$), some of the most common mutation strategies in DE are:

1. *rand/1*

$$\mathbf{v_i} = \mathbf{x_{r1}} + F \times (\mathbf{x_{r2}} - \mathbf{x_{r3}}) \tag{1}$$

2. *best/2*

$$\mathbf{v_i} = \mathbf{x_{best}} + F \times (\mathbf{x_{r1}} - \mathbf{x_{r2}}) + F \times (\mathbf{x_{r3}} - \mathbf{x_{r4}}) \tag{2}$$

3. *rand/2*

$$\mathbf{v_i} = \mathbf{x_{r1}} + F \times (\mathbf{x_{r2}} - \mathbf{x_{r3}}) + F \times (\mathbf{x_{r4}} - \mathbf{x_{r5}}) \tag{3}$$

where $\mathbf{v_i}$ is the donor vector being generated, $\mathbf{x_{r1}}$, $\mathbf{x_{r2}}$, $\mathbf{x_{r3}}$, $\mathbf{x_{r4}}$, and $\mathbf{x_{r5}}$ are five distinct solutions randomly chosen from the current population, $\mathbf{x_{best}}$ is the best solution in the population, and $F \in [0, 2]$ (usually less than 1.0), is the so-called *scale factor*.

After mutation generates the donor vector, the crossover operator is applied to further refine the search. Such operator combines every i-th solution $\mathbf{x_i}$ in the population with the corresponding newly generated donor vector, $\mathbf{v_i}$, to generate the trial solution, $\mathbf{u_i}$. The most common crossover operator is the binomial crossover (usually abbreviated as bin), which works as follows:

$$u_{i,j} = \begin{cases} v_{i,j}, \text{ if } rand(0,1) \leq CR \text{ or } j = j_{rand} \\ x_{i,j}, \text{ otherwise} \end{cases} \tag{4}$$

where $j = 1, \ldots, D$ is the index of each variable, $rand(0, 1)$ is a random number sampled from a uniform distribution in $[0, 1]$, the crossover rate $CR \in [0, 1]$ is a parameter which controls the fraction of variables copied from the donor vector, and j_{rand} is an index randomly chosen from the set $\{1, \ldots, D\}$. Applying this crossover with the *rand/1* mutation leads to the well-known DE *rand/1/bin*, but as said several other combinations have been proposed and more can be obtained by combining mutation/crossover strategies.

Finally, the selection step selects the best solution (in terms of fitness value) between $\mathbf{x_i}$ and $\mathbf{u_i}$, keeping the same population size. Therefore, since DE does not accept solutions with worse fitness values, it can be considered a sort of (parallel) hill-climbing method.

2.2 UPGMC

Clustering [51] is a process that organizes a dataset into a (not necessarily predefined) number of groups (clusters), such that data samples within a cluster are more similar to each other than samples belonging to different clusters, w.r.t. a pre-defined distance/similarity measure. To achieve such organization, clustering methods usually minimize a given loss function in an iterative process, which is continued until a stopping criterion is met, e.g. a check on the convergence (when a better clustering cannot be found after a given number of iterations), or simply until all samples are grouped. Clustering is a well-known technique for discovering the inherent structure in any given dataset, and as such it is largely used in statistics and machine learning.

In a cluster, the centroid is usually calculated as the mean of the samples of that cluster. More specifically, given a cluster $C = \{\mathbf{x_1}, \ldots, \mathbf{x_c}\}$, where c is the number of samples in the cluster, each j-th coordinate of the corresponding centroid $\mathbf{C_c}$ is given by:

$$C_{c,j} = \frac{1}{c} \sum_{i=1}^{c} x_{i,j} \tag{5}$$

where $j = 1, \ldots, D$ is the index of the corresponding variable of the samples.

As discussed earlier, cluster centroids are at the basis of the mutation strategies proposed in this work. Here, clustering is performed by means of UPGMC [54]. This method works by grouping the most similar samples, as well as any previously formed groups, and *replacing* all samples in every new group with the group's unweighted centroid (i.e., the *center of mass*), thus ignoring any internal structural subdivision. The most similar samples have the smallest distance among them. On the other hand, to calculate the distance $d()$ between a cluster formed by two existing clusters, C_i and C_j, to another existing cluster C_k, the method uses the following recursive formula:

$$\begin{aligned} d(C_k, C_i \cup C_j) = \frac{n_i}{n_i+n_j} d(C_k, C_i) + \frac{n_j}{n_i+n_j} d(C_k, C_j) + \\ - \frac{n_i n_j}{(n_i+n_j)^2} d(C_i, C_j) \end{aligned} \tag{6}$$

where n_i, n_j , and n_k are the number of samples in C_i, C_j, and C_k, respectively. When a sample is added to a cluster, the new centroid is calculated as the mean of the samples in that cluster, according to Eq. (5). After clustering all samples, a cut in the clustering tree is done by traversing it in a top-down manner until finding the desired number of branches (k). Then, all samples below each cut are assigned to a single cluster[1].

Clustering methods have been employed in metaheuristics [69] to detect if there is a concentration of solutions in a few regions (thus resulting in redundant search, premature or early convergence) and, if needed, run e.g. a local search or restart the algorithm. In this work, clustering is employed instead to guide the search performed by the metaheuristic to generate more solutions in the promising regions of the search space, as we discuss below.

[1] For more details on the UPGMC method, we refer the interested reader to [68]

3 The proposed cluster-centroid-based mutation strategies

In order to use the information extracted from clustering to guide the search towards the more promising regions of the search space, we extend the mutation strategies presented in Section 2.1 by incorporating centroids calculated by UPGMC. This extension results in the following cluster-centroid-based mutation strategies:

st1: *rand-centroid/1*

$$\mathbf{v_i} = \mathbf{C_{rand}} + F \times (\mathbf{x_{r1}} - \mathbf{x_{r2}}) \quad (7)$$

st2: *rand-to-random-centroids/2*

$$\mathbf{v_i} = \mathbf{x_{r1}} + F \times (\mathbf{C_{r2}} - \mathbf{x_{r2}}) + F \times (\mathbf{C_{r3}} - \mathbf{x_{r3}}) \quad (8)$$

st3: *rand-to-best-and-random-centroid/1*

$$\mathbf{v_i} = \mathbf{x_{r1}} + F \times (\mathbf{x_{best}} - \mathbf{C_{rand}}) \quad (9)$$

st4: *best-to-random-centroid/1*

$$\mathbf{v_i} = \mathbf{x_{best}} + F \times (\mathbf{C_{r1}} - \mathbf{x_{r2}}) \quad (10)$$

st5: *rand-to-best-centroid/1*

$$\mathbf{v_i} = \mathbf{x_{r1}} + F \times (\mathbf{C_{best}} - \mathbf{x_{r2}}) \quad (11)$$

st6: *best-centroid/1*

$$\mathbf{v_i} = \mathbf{C_{best}} + F \times (\mathbf{x_{r1}} - \mathbf{x_{r2}}) \quad (12)$$

st7: *local-to-best-centroid/2*

$$\mathbf{v_i} = \mathbf{x_i} + F \times (\mathbf{C_{best}} - \mathbf{x_i}) + F \times (\mathbf{x_{r1}} - \mathbf{x_{r2}}) \quad (13)$$

st8: *local-to-best-centroid-random-centroids/2*

$$\mathbf{v_i} = \mathbf{x_i} + F \times (\mathbf{C_{best}} - \mathbf{x_i}) + F \times (\mathbf{C_{r1}} - \mathbf{C_{r2}}) \quad (14)$$

where: $\mathbf{x_i}$ is the i-th solution in the population; $\mathbf{C_{rand}}$ is the centroid of a randomly selected cluster; $\mathbf{C_{ri}}$ indicates the centroid of the cluster where a randomly selected solution $\mathbf{x_{ri}}$ ($i \in \{1, 2, 3\}$) is located; $\mathbf{C_{best}}$ indicates the centroid of the cluster where $\mathbf{x_{best}}$ (the best solution in the population) is located. In all the strategies above, clusters are obtained by means of UPGMC (performed at every generation of DE) and centroids are calculated according to Eq. (5).

An illustrative example on how to obtain the components necessary for the proposed cluster-centroid-based mutation strategies can be seen in Figure 1. The figure shows three clusters (obtained by means of UPGMC), with their corresponding centroids, represented by red diamonds indicated by $\mathbf{C_1}$, $\mathbf{C_2}$ and $\mathbf{C_3}$ respectively. $\mathbf{C_{rand}}$ is the centroid of a randomly selected cluster; in the figure, we assume $\mathbf{C_{rand}}$ to be set to $\mathbf{C_3}$. In order to choose $\mathbf{C_{ri}}$, first $\mathbf{x_{ri}}$ is randomly selected from the current population. E.g. if $\mathbf{x_{r1}}$ is selected, $\mathbf{C_{r1}}$ is the cluster in which $\mathbf{x_{r1}}$ is located, which is cluster 3 in the example. Thus $\mathbf{C_{r1}}$ is set to $\mathbf{C_3}$. The same procedure is applied to $\mathbf{x_{r2}}$ and $\mathbf{x_{r3}}$ to obtain, respectively, $\mathbf{C_{r2}}$ (that in the example is equal to $\mathbf{C_3}$) and $\mathbf{C_{r3}}$ (that in the example is equal to $\mathbf{C_2}$). Once all these components have been identified, it is possible to apply one of the eight cluster-centroid-based strategies defined above in Eq. (7-14). If $\mathbf{x_{best}}$ and $\mathbf{C_{best}}$ are needed (not shown in the figure), it is necessary to identify also the best solution in the population and the centroid of the cluster where it is located.

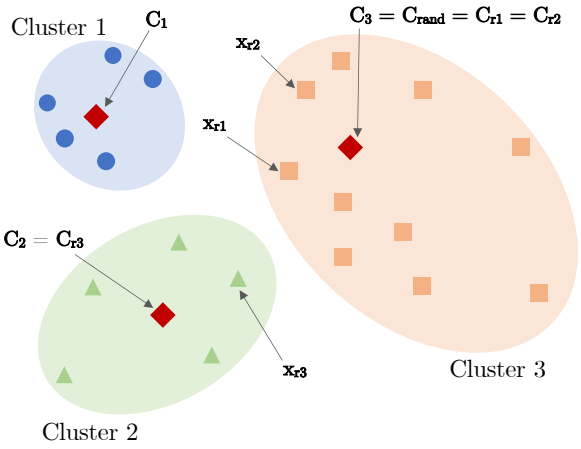

Fig. 1 Example representation on how to obtain cluster centroids ($\mathbf{C}$) and solutions ($\mathbf{x}$) for the proposed cluster-centroid-based mutation strategies, after applying the UPGMC clustering method (see the text for details).

Of note, two of the proposed strategies, *rand-centroid/1* and *best-centroid/1*, use a cluster centroid as the base vector, that is, they act as a local search around the selected centroid. This implies that if the centroid moves in small steps these strategies can get stuck easily. Furthermore, if the number of cluster centroids is small, they can result in low exploration and possibly premature convergence. On the other hand, a higher number of clusters may improve the performance. However, the cluster centroids are not guaranteed to be positioned in high-quality coordinates, even if the solutions in the cluster are good. The results presented in the next section allow, among the other things, the verification of these assumptions.

Another issue that must be noted is that, by using cluster centroids instead of actual solutions as base vectors, there is a loss of the information available in the single solutions. On the other hand, one may say that there is instead an aggregation of all the information available in a cluster into a single vector (i.e., the centroid). Notably, a similar idea is employed in Estimation of Distribution Algorithms [70, 71] (EDAs) and the so-called "compact" optimization algorithms [72, 73], which in some cases use Gaussian distributions to model the population and sample new solutions. In

our implementation, we suppose that cluster centroids can be used to identify promising regions of the search space, while the other difference vectors in the mutation strategy facilitate exploitation. The intuition behind this idea is that if such aggregated vectors are well-located into a promising region (which however, as we said earlier, might not always be the case), then the algorithm may be able to perform an efficient exploration, although without any guarantee on the fact that such region contains the global optimum. Nevertheless, while a cluster-centroid-based strategy may be useful to guide the search to promising regions, it may not be able to *refine* the solutions to the desired precision. This means that a local search method could be necessary to improve the algorithm performance. To this end, the other difference vectors based on single solutions are used in the proposed cluster-centroid-based mutation strategies.

As a final note, it is important to state that here we perform clustering in the genotype, rather than fitness, space. Therefore, solutions are clustered according to their variables only, meaning that a cluster may contain both high-quality and low-quality solutions in terms of fitness.

4 Computational experiments

To assess the performance of the proposed cluster-centroid-based mutations, we performed two separate experiments. The first experiment considered 13 box-constrained benchmark functions with different characteristics, taken from [74]. The second experiment was performed on the full BBOB benchmark available in the COCO framework (v15.03), used at the BBOB-2015 competition[2]. Of note, the experimentation is designed in such a way that the first experiment includes somehow simpler benchmark functions (e.g., without any shift on the optimum or rotational transformations), but it allows a direct comparison with the results obtained with other cluster-centroid-based DE variants reported in the literature. On the other hand, the second experiment includes the much harder BBOB benchmark functions, and it allows a direct comparison with some of the metaheuristics which are currently at the state-of-the-art in continuous global optimization. More details on this aspect are discussed below.

4.1 Experiment one

In the following, we first present the benchmark functions and the algorithm configurations used in the first experiment, and then we discuss the results.

[2] http://coco.gforge.inria.fr/doku.php?id=bbob-2015-downloads

4.1.1 Test benchmark functions

The definitions of the test problems are presented in Table 1. More details of the functions, which represent minimization problems, can be found in [74]. We should note that the whole set presented in [74] contains 23 functions, but for this work we selected only those that can be scaled to any dimensionality, thus ignoring the fixed-dimensional problems. Furthermore, these functions were selected also because they were employed in related works [52, 53], which as said earlier allows a direct comparison with the results presented in the literature. Of note, among the 13 selected functions the first five are unimodal, while the remaining ones are multimodal. Except f_8, whose optimum is $f_{opt} \approx -418.983 \times D$, the optimum of all the other functions is zero. Furthermore, the optimal solution corresponds to the center of the search space $\mathbf{x_{opt}} = (0, \ldots, 0)$ for all functions except f_5, f_{12} and f_{13}, for which $\mathbf{x_{opt}} = (1, \ldots, 1)$, and f_8, for which $\mathbf{x_{opt}} \approx (420.9687, \ldots, 420.9687)$. The fact that most of these functions are characterized by a search space symmetrical to the optimum makes these problems somehow less challenging from an optimization perspective, but on the other hand this allows us to reason on any possible bias that the cluster-centroid-based mutation might have towards the center of the search space. This is why, in the second experiment, we consider instead the more complex functions from the BBOB benchmark, which all have shifted optima and as such are less prone to potentially biased search logics that make use of cluster centroid information. We will come back to this aspect later.

4.1.2 Configurations of the proposed cluster-centroid-based DE

The parameter settings of the DE algorithm and the clustering methods are reported in Table 2 and 3 respectively. With reference to Table 2, we considered three different DE configurations, selected to provide valuable clues on the performance of the cluster-centroid-based mutation strategies in three different regimes of exploration/exploitation balances [63]. In particular, Configuration 1 is intended to balance exploration and exploitation, increasing DE's capability of escaping from local optima. As for Configuration 2, it is known that a large CR value makes the search more exploitative, possibly resulting in a premature convergence [75–77]. This problem is less present when the base vector is dynamic, such as in the *rand/1* strategy. On the other hand, when using either $\mathbf{x_{best}}$ or $\mathbf{C_{best}}$ as the $base$ vector, it is reasonable to expect stagnation, as a single solution is a reference for all mutation; thus, this would be a good strategy for a local search. Finally, Configuration 3 is intended to be more explorative, thus without the fast stagnation that happens whenever a high CR is used.

Table 1 Experiment one: Benchmark functions.

Function Definition	Bounds
$f_1(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$
$f_2(\mathbf{x}) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$
$f_3(\mathbf{x}) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	$[-100, 100]^D$
$f_4(\mathbf{x}) = \max_i \{\lvert x_i \rvert, 1 \le i \le D\}$	$[-100, 100]^D$
$f_5(\mathbf{x}) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]^D$
$f_6(\mathbf{x}) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^D$
$f_7(\mathbf{x}) = \sum_{i=1}^{D} i x_i^4 + U(0, 1)$	$[-1.28, 1.28]^D$
$f_8(\mathbf{x}) = \sum_{i=1}^{D} -x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500, 500]^D$
$f_9(\mathbf{x}) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^D$
$f_{10}(\mathbf{x}) = -20\exp\left(-0.2\sqrt{D^{-1}\sum_{i=1}^{D} x_i^2}\right) - \exp(\frac{1}{D}\sum_{i=1}^{D} \cos 2\pi x_i) + 20 + e$	$[-32, 32]^D$
$f_{11}(\mathbf{x}) = (1/4000)\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(x_i/\sqrt{i}\right) + 1$	$[-600, 600]^D$
$f_{12}(\mathbf{x}) = \frac{\pi}{D}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2\right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$ where : $y_i = 1 + (x_i + 1)/4$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i - a)^m, & x_i < -a, \end{cases}$	$[-50, 50]^D$
$f_{13}(\mathbf{x}) = \left\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})]\right\}/10 + (x_D - 1)[1 + \sin^2(2\pi x_D)]/10 + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	$[-50, 50]^D$

Table 2 Experiment one: DE settings.

Parameter	Configuration 1	Configuration 2	Configuration 3
Scale factor	$F = 0.5$	$F = 0.5$	$F = 0.9$
Crossover rate	$CR = 0.5$	$CR = 0.9$	$CR = 0.5$

Table 3 Experiment one: Clustering settings.

Parameter	Value
Algorithm	UPGMC
Distance measure	Euclidean
***k* (clusters)**	$k = 2, 5, 10$
Update	Every DE generation
Noise	1e-10

As for the clustering settings, shown in Table 3, we used the UPGMC method, based on the Euclidean distance (that is, the same measure employed by the related methods used in the comparison). However, it should be noted that other clustering methods and distance measures, possibly more efficient than UPGMC with Euclidean distance, could be used. We will leave this aspect for future investigations. We considered three different number of clusters, namely $k = 2, 5, 10$, and updated the clusters at every DE generation. Finally, the noise parameter was introduced to avoid that clustering failed because of the impossibility to find the desired number of clusters, e.g. due to duplicate solutions in the current population (which may happen especially when the algorithm converges). To overcome this problem without affecting the results, before performing clustering we added a uniform noise to the solutions, as follows:

$$x_{i,j_{rand}} = x_{i,j_{rand}} + rand(-noise,\ noise) \tag{15}$$

where $i = 1, \ldots, N$, j_{rand} is an index randomly chosen from the set $\{1, \ldots, D\}$, and $rand(-noise, noise)$ is a random number sampled uniformly in $[-noise, noise]$. In our experiments we used an arbitrary small value $noise = 1e-10$ to ensure that the solutions are actually different and allow clustering. It should be noted that while this process slightly jitters the solutions, it preserves their distribution in the search space. Furthermore, we should remark that this additional noise is not used during the solution evaluation phase, and as such it does not affect the search process and its exploration/exploitation capabilities.

As for the maximum number of function evaluations (in the following, referred to as NFEs), we set it as in [52, 53], to allow a fair comparison. Specifically, NFEs = 150,000 were used for f_1, f_6, f_{10}, f_{12}, and f_{13}; NFEs = 500,000 for f_3, f_4, and f_9; NFEs = 200,000 for f_2 and f_{11}; NFEs = 2,000,000 for f_5; and NFEs = 900,000 for f_8. All the problems were executed with $D = 30$ dimensions, with population size $N = 100$. The number of independent runs per problem/algorithmic configuration was 30.

4.1.3 Experimental analysis

The goal of the first experiment is to evaluate the performance of the proposed cluster-centroid-based mutation strategies versus: 1) the basic DE; 2) other cluster-centroid-based DE variants and 3) other metaheuristics from the literature.

Comparison with the basic DE This comparison aims at verifying if by introducing cluster centroid information in the mutation strategies leads to a performance improvement w.r.t. the basic DE, and then ranking the mutation strategies accordingly. Furthermore, we also assess the influence of k

on the search conducted by the proposed cluster-centroid-based mutation strategies. For this experiment, we ran DE with the traditional *rand/1* mutation strategy and the eight proposed cluster-centroid-based mutation strategies described in Section 3, all of them using the binomial crossover. see Eq. (4).

Figure 2 presents a multivariate plot comparing the error (w.r.t. the optimum), mean $\pm 95\%$ confidence interval across 30 independent runs, log-scale, obtained by all the nine strategies under comparison on the 13 selected benchmark functions, varying the number of clusters k and the DE configuration (i.e., the three settings listed in Table 2). The detailed results (in terms of best fitness values, mean and standard deviation across 30 independent runs), are provided in Tables 11-13 reported in the Appendix.

Let us consider first the comparison between the proposed cluster-centroid-based mutations and *rand/1*. With reference to the mean rank through the benchmark functions (see the last row in Tables 11-13, the lower the better), it can be noted that for Configuration 1 the best strategy was *st5*, followed by *st4*, while the last one was *st8*. For Configuration 2, there was a tie with *rand/1* and *st2* at the first place, followed by *st5*, while the last was, again, *st8*. With Configuration 3, the best strategy was *st3*, followed by *st7*, with *rand/1* and *st8* being the worst-performing strategies.

Another observation is that, compared to the cluster-centroid-based mutations, *rand/1* worked overall better on f_8 and f_9. More specifically, considering f_8, *rand/1* worked especially well with Configuration 1 and 2, while the only proposed strategies on par with (or slightly outperforming) it were *st4* and *st5* with $k = 10$ and Configuration 1. Concerning f_9, the only proposed strategy that was able to outperform *rand/1* was *st1* with $k = 5$ and Configuration 1. This observation is particularly important as it may indicate a potential bias of some (but not all) of the proposed cluster-centroid-based mutation strategies w.r.t. *rand/1*: considering in particular the case of f_8, as discussed earlier this function has its optimum shifted w.r.t. the center of the search space, differently from most of the other benchmark functions listed in Table 1, which have their optimum in zero. The lower performance of most cluster-centroid-based mutation strategies w.r.t. *rand/1* on f_8 might then be due to the fact that cluster centroid information might be more beneficial when the search space is symmetrical w.r.t. the optimum, i.e., the optimum is in the center of the search space. However, we should note that also f_5, f_{12} and f_{13} have their optimum shifted w.r.t. the center of the search space, yet the performance of *rand/1* appears in these cases overall inferior to that of the proposed cluster-centroid-based mutation strategies. On the contrary, f_9 does have its optimum in zero, yet *rand/1* appears overall superior on it. All in all, it seems that in general cluster-centroid-based mutation strategies tend to work better than *rand/1* especially (but, not only) when the optimum is in the center of the search space, although there are other properties of the fitness landscape that also affect the effectiveness of using cluster centroid information. As we will see in the second experiment, on the BBOB benchmark, this potential bias does not impede though the proposed cluster-centroid-based mutation strategies to perform well also on more complex problems, especially at low dimensionalities, even w.r.t. state-of-the-art techniques.

Let us now analyze Figure 2 w.r.t. the number of clusters k. Overall, a reduction of the mean error is observed in most cases as k increases. It is interesting to notice that such behavior is present mainly when using Configuration 1. Considering e.g. *st1*, a consistent performance improvement while increasing k was found on f_1, f_2, f_4, f_7, f_8, and f_{10} to f_{13}, making it the strategy that better used the knowledge provided by the clusters. Other strategies also largely benefited from having more clusters, with different DE configurations: for instance *st2* (see f_1, f_2, f_4, f_{10}, f_{12}, and f_{13}) and *st4* (see f_8 and f_{12}). Moreover, strategies *st3*, and *st6*-*st8* also obtained better performance with more clusters, although this is not clearly visible in the plot because of the log-scale[3]. The opposite behavior, that is a performance drop when k increases, also occurred, being clearly visible for strategies *st3* to *st7* when using Configuration 3. Nevertheless, for some problems, i.e., f_1, f_4, and f_5, this configuration showed the best overall performance across all configurations and strategies.

The main conclusions of this first analysis can then be summarized as follows: 1) using the difference between two cluster centroids negatively affects the search (*st8*); 2) using the cluster centroid corresponding to the best solution in the population to calculate the difference vector positively affects the search (*st5* and *st7*); 3) for Configuration 3, the larger the number of clusters, the worse the performance.

Comparison with other cluster-centroid-based DE variants
The previous comparison showed that cluster centroid information can improve the search when compared with the traditional *rand/1* mutation strategy. Hence, this second analysis compares the performance of the proposed cluster-centroid-based mutation strategies with other methods from the literature that also employ clustering to improve DE, i.e., C-DE [52] and W-CDE [53]. The results of this comparison are given in Table 4, where the "Summary" column shows the lowest mean error corresponding to the best fitness values reported in Tables 11-13, i.e., the best result for each problem found across the various combinations of cluster-centroid-based strategy, number of clusters, and DE configuration. This comparison is meant to verify if the overall best solutions found in this work are better than the solutions obtained by the existing cluster-centroid-based methods, given

[3] These improvements can be seen in more detail by looking at Tables 11-13.

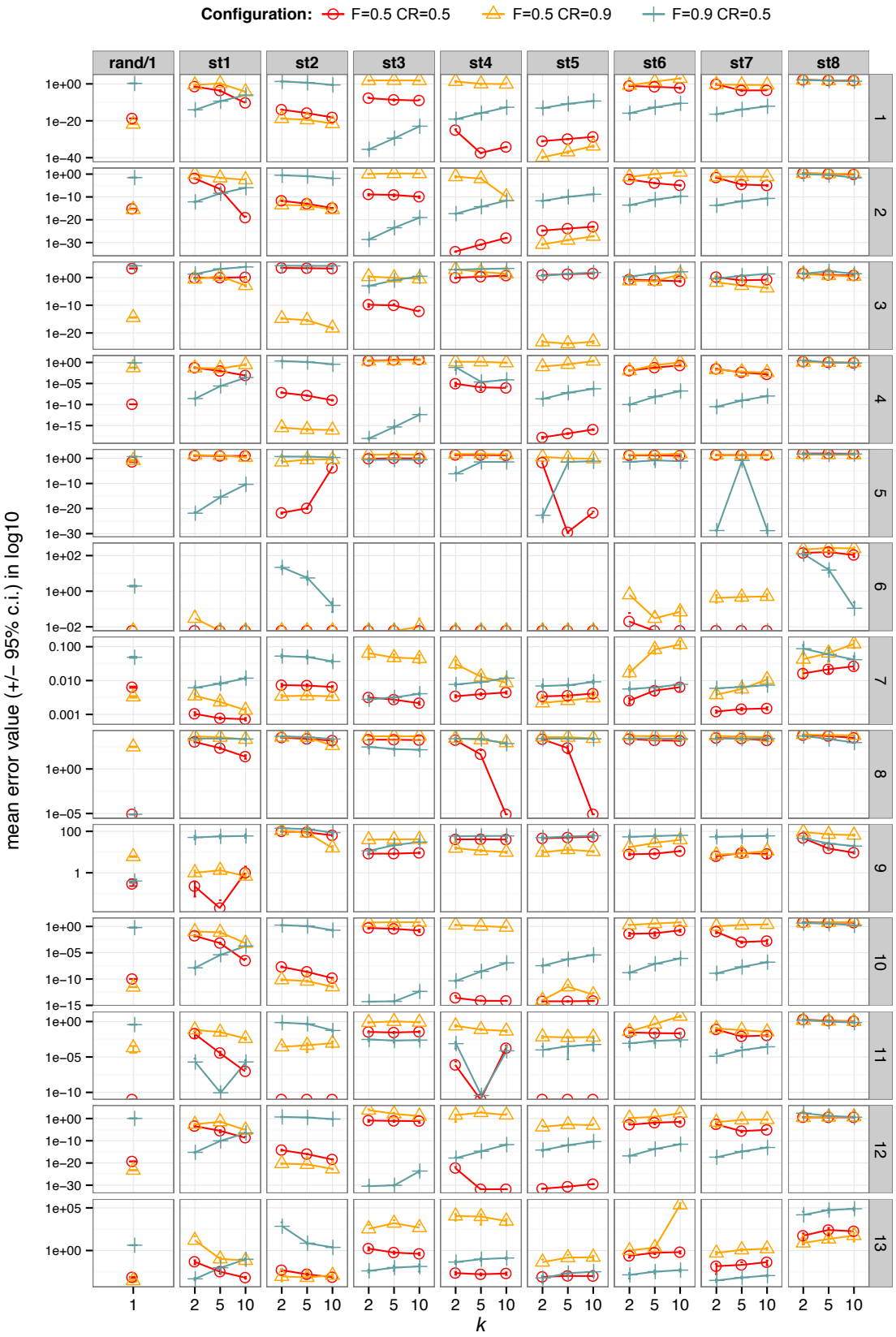

Fig. 2 Experiment one: Error (mean ±95% confidence interval across 30 independent runs, log-scale) obtained with *rand/1* and *st1* to *st8*, varying both the number of clusters k and the DE configuration (see Table 2).

that, as seen earlier, the quality of the results found in our work depends on the mutation strategy, the numbers of clusters, and the DE configuration. Then, for a more direct comparison we selected for each of the three configurations the best mutation strategy, and the intermediate number of clusters ($k = 5$). For the ranking process, mean errors below 1e-06 were converted to 0 in order to obtain a fair comparison with the results published in [53], while duplicate values were assigned the same rank.

From the table it can be observed that, even with the drawbacks identified in the strategies proposed in this work (see Section 3), it is possible to achieve results competitive with those of the other cluster-centroid-based DE variants from the literature. On the other hand, it is also important to remember that our approaches are based on the basic DE, with the addition of cluster centroid information in the mutation strategies, whereas the compared methods are even more improved versions of DE. Regarding the mean error, the best method was Configuration 3 *st3*. We also provide the median error, which shows that the best method was C-DE, followed by Configuration 1 *st5*. The last row has the mean rank, which shows that the two best methods were C-DE and Configuration 3 *st3*. Finally, we applied the multiple comparison Friedman test suggested in [78] to check if there is any difference among the methods, followed by the Nemenyi post hoc test to rank the methods if a significant difference is detected. For the analysis, we employed a significance level $\alpha = 0.05$. The p-value calculated by the Friedman test was 0.9583, thus, higher than α, suggesting no significant differences among the compared methods.

Comparison with other metaheuristics After having verified that the proposed cluster-centroid-based mutations improve, in general, on the traditional *rand/1* DE mutation strategy, and show competitive results w.r.t. previous cluster-centroid-based DE algorithms proposed in the literature, we conclude the analysis on the first experiment with a comparison of the proposed algorithm with some of the most successful methods from the literature.

Table 5 indicates that the results obtained by our best ranked approach (according to Table 4, Configuration 3 *st3*) are competitive with those of other DE-based approaches (namely: SaDE [79], DEahcSPX [80], and ODE [34]) and generally better than those obtained with Evolutionary Programming (in its "classical" and "fast" variants, respectively indicated as CEP and FEP, both from [81]), Particle Swarm Optimization (PSO) [82], two variants of Bacterial Foraging Algorithm (namely the original one, indicated as BFA, and its fast version, FBSA, both from [83]), Biogeography Based Optimization (BBO) [84], and two versions of Artificial Bee Colony (ABC and GABC), both taken from [85]. It should be remarked that the results shown in the table were taken from the original papers, to reduce any reproducibility issue, and were obtained with the same NFEs.

Concerning the mean error, the best method was SaDE, a DE variant with self-adaptation of the parameters, which was the best on 5 out of 13 functions, followed by our approach (Configuration 3 *st3*), which was the best on 4 functions, while the worst method was PSO. The p-value calculated by the Friedman test was 2.566e-10 (lower than α), meaning that there are significant differences among the compared methods. Therefore, we performed the Nemenyi post hoc test to calculate the pairwise comparisons, whose results are shown in Table 6. Taking our approach as the baseline, significant differences were detected for 3 out of 11 methods: CEP, BFA, and PSO. Of note, SaDE outperformed the same methods, as well as BBO.

4.2 Experiment two

In the following, we present the benchmark functions and the configurations of the algorithms used in the second experiment, and we discuss the results.

4.2.1 Test benchmark functions

For this experiment, we used the 24 benchmark functions available in the BBOB noiseless testbed, whose definitions can be found in [86]. These functions are separated into five groups: separable functions ($f_1 - f_5$); functions with low or moderate conditioning ($f_6 - f_9$); functions with high conditioning and unimodal ($f_{10} - f_{14}$); multimodal functions with adequate global structure ($f_{15} - f_{19}$); multimodal functions with weak global structure ($f_{20} - f_{24}$). The experiment was performed according to the BBOB-2015 competition rules, i.e., setting the maximum NFEs to $10,000 \times D$ (expensive scenario) for each run, and solving the problems in 2, 3, 5, 10, and 20 dimensions for a total of 15 independent runs per problem/problem size and algorithmic configuration.

4.2.2 Configurations of the proposed cluster-centroid-based DE

Similarly to the first experiment, we used UPGMC with Euclidean distance as clustering method, updating the clusters at every DE generation, and applying a noise of $1e-10$. Concerning the cluster-centroid-based mutation strategies, in this case we considered only the two strategies that achieved the best results in the first experiment, namely *st3* and *st5*. Considering also the other DE and clustering settings, we then performed an exhaustive parameter search of a total of 72 possible parameter configurations, i.e., combinations of scale factor F, crossover rate CR, number of clusters k, population size N and mutation strategies. The

Table 4 Experiment one: Error (mean and standard deviation across 30 independent runs) of three selected combinations of DE configurations (see Table 2) and mutation strategy, with k = 5, vs. C-DE and W-CDE. The "Summary" columns shows the lowest mean error obtained by any configuration for each problem, corresponding to the fitness values reported in Tables 11-13 (ignoring *rand/1*). The boldface indicates the lowest error for each problem.

Fun	Summary	Configuration 1 st5	Configuration 2 st2	Configuration 3 st3	C-DE [52]	W-CDE [53]
f_1	1.3510e-52	2.4660e-33	1.0730e-26	**8.3150e-41**	2.3500e-30	1.0000e-06
	(2.59e-52)	(1.58e-33)	(1.19e-26)	(1.06e-40)	(2.46e-30)	(2.00e-06)
f_2	2.2200e-40	1.2380e-26	5.2860e-19	**1.9440e-31**	4.3700e-22	0.0000e+00
	(5.06e-40)	(4.90e-27)	(3.39e-19)	(2.38e-31)	(2.38e-22)	(0.00e+00)
f_3	2.6830e-24	5.7180e+00	2.6610e-21	2.7140e-03	**1.1800e-24**	1.1000e-05
	(6.41e-24)	(4.07e+00)	(3.10e-21)	(2.96e-03)	(2.72e-24)	(1.40e-05)
f_4	2.1350e-28	6.3030e-20	2.2410e-17	**1.7430e-22**	1.6200e+00	7.5000e-05
	(4.60e-28)	(9.19e-20)	(6.04e-17)	(2.62e-22)	(1.19e+00)	(7.50e-05)
f_5	2.8530e-30	**5.9970e-30**	4.4420e-01	2.3920e-01	3.2500e-27	1.8300e+00
	(1.27e-29)	(1.18e-29)	(1.23e+00)	(9.56e-01)	(1.52e-26)	(6.83e+00)
f_6	0.0000e+00	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**	**0.0000e+00**
	(0.00e+00)	(0.00e+00)	(0.00e+00)	(0.00e+00)	(0.00e+00)	(0.00e+00)
f_7	6.1480e-04	3.2260e-03	3.1200e-03	2.5700e-03	1.9700e-03	**4.6000e-04**
	(2.54e-04)	(1.54e-03)	(1.23e-03)	(9.30e-04)	(6.49e-04)	(3.60e-04)
f_8	1.8190e-12	4.4950e+02	6.8930e+03	**8.9490e+01**	5.6600e+03	1.2500e+03
	(7.35e-13)	(1.19e+03)	(4.24e+02)	(1.20e+02)	(9.21e+02)	(5.03e+02)
f_9	1.7450e-05	8.6710e+01	1.5010e+02	3.8650e+01	5.5500e+01	**1.8000e-05**
	(7.22e-05)	(7.98e+00)	(9.20e+00)	(2.63e+01)	(2.45e+01)	(2.30e-05)
f_{10}	4.7780e-15	5.3470e-15	2.6590e-14	4.7780e-15	**3.7200e-15**	4.7000e-05
	(1.49e-15)	(1.74e-15)	(1.22e-14)	(1.49e-15)	(1.17e-15)	(4.50e-05)
f_{11}	0.0000e+00	**0.0000e+00**	**0.0000e+00**	2.1680e-03	**0.0000e+00**	**0.0000e+00**
	(0.00e+00)	(0.00e+00)	(0.00e+00)	(4.79e-03)	(0.00e+00)	(0.00e+00)
f_{12}	1.5710e-32	**1.5710e-32**	2.6030e-28	1.6710e-32	2.4700e-32	0.0000e+00
	(0.00e+00)	(0.00e+00)	(2.52e-28)	(7.12e-33)	(2.31e-32)	(0.00e+00)
f_{13}	2.4790e-05	9.5550e-04	6.3100e-04	8.1350e-03	**7.8300e-27**	0.0000e+00
	(1.75e-04)	(6.27e-04)	(7.34e-04)	(2.95e-02)	(3.89e-26)	(0.00e+00)
Mean Error	-	41.687	541.811	**9.877**	439.779	96.295
Median Error	-	6.30e-20	2.24e-17	2.17e-03	**1.18e-24**	1.10e-05
Mean Rank	-	3.077	3.154	3.0	**2.692**	3.077

parameter values considered in this search are shown in Table 7[4].

Consequently, we selected the best four CDE configurations in terms of aggregated results (considering all the BBOB functions in five dimensionalities) and the two configurations that achieved the closest performance curve (in terms of bootstrapped empirical cumulative distribution, see Figure 3) to the reference algorithm in COCO, namely *Best 2009* (obtained aggregating the best results from the BBOB-2009 competition). While the former are the overall top-performer configurations, the latter is able to achieve better results only when a lower NFEs is considered. These selected configurations are shown in Table 8.

It can be noticed that, apart from CDE-Conf-10, all the selected configurations are strongly exploitative, especially considering CR set to 0.9. Furthermore, all the configurations are characterized by small population sizes (up to N = 50, being the maximum possible value N = 200). Since these settings are likely to lead to a fast loss of population diversity, we added a simple restart mechanism that generates a new random population (without elitism) whenever the current population fitness variance is less than an empirically chosen threshold of 1e-16.

[4] In Table 7 and in the rest of the text, we use the acronym "CDE" to indicate our cluster-centroid-based DE, that should not be confused with the C-DE algorithm proposed by Cai et al. [52].

4.2.3 Experimental analysis

The goal of the second experiment is to investigate how the proposed cluster-centroid-based DE algorithm, configured with the six settings reported in Table 7, performs on a harder benchmark, i.e., the BBOB testbed, in comparison with the state-of-the-art algorithms that have been tested so far on the same benchmark. For this comparison, we considered the best algorithms that participated in the BBOB-2015 competition, namely:

1. The multivariate Brent STEP method and many of its variants (Sif, Sifeg, BSifeg, Srr, BSrr, and BSqi) [87];
2. IPOP-CMA-ES, i.e., Covariance Matrix Adaptation Evolution Strategy (CMA-ES) with two step-size adaptation mechanisms (CMA-MSR and CMA-TPA) and Cumulative Step-size Adaptation (CMA-CSA) [88];

Table 5 Experiment one: Error (mean and standard deviation across 30 independent runs) of the best ranked configuration from Table 4 (Configuration 3 *st3*) vs 11 metaheuristics from the literature. "NA" means "Not Available" (replaced by +Inf to calculate the ranks). The boldface indicates the lowest error for each problem.

Fun	Configuration 3 st3	FEP [81]	CEP [81]	SaDE [79]
f_1	8.315e-41(1.058e-40)	5.7e-04(1.3e-04)	2.2e-04(5.9e-04)	1.48e-18(9.28e-19)
f_2	**1.944e-31(2.384e-31)**	8.1e-03(7.7e-04)	2.6e-03(1.7e-04)	3.16e-15(1.34e-15)
f_3	2.714e-03(2.956e-03)	1.6e-02(1.4e-02)	5.0e-02(6.6e-02)	**4.02e-20(4.89e-20)**
f_4	1.743e-22(2.623e-22)	3.0e-01(5.0e-01)	2.00e+00(1.2e+00)	8.01e-10(3.49e-10)
f_5	2.392e-01(9.564e-01)	5.06e+00(5.87e+00)	6.17e+00(1.36e+01)	7.97e-02(5.64e-01)
f_6	**0.00e+00(0.00e+00)**	**0e+00(0e+00)**	5.78e+02 (1.13e+03)	**0e+00(0e+00)**
f_7	2.57e-03(9.3e-04)	7.6e-03(2.6e-03)	1.8e-02(6.4e-03)	6.21e-03(1.42e-03)
f_8	8.949e+01(1.203e+02)	1.45e+01(52.6)	4.6519e+03(634.5)	**0e+00(0e+00)**
f_9	3.865e+01(2.628e+01)	4.6e-02(1.6e-02)	8.90e+01 (2.31e+01)	**0e+00(0e+00)**
f_{10}	**4.778e-15(1.487e-15)**	1.8e-02(2.1e-03)	9.2e+00(2.8e+00)	3.08e-10(8.70e-11)
f_{11}	2.168e-03(4.788e-03)	1.6e-02(2.2e-02)	8.6e-02(12e-01)	**0e+00(0e+00)**
f_{12}	**1.671e-32(7.119e-33)**	9.2e-06(3.6e-06)	1.76e+00(2.4e+00)	4.48e-20(3.10e-20)
f_{13}	8.135e-03(2.95e-02)	1.6e-04(7.3e-05)	1.4e+00(3.7e+00)	4.83e-18(3.96e-18)
Mean Rank	3.769	6.615	8.923	**3.192**
Fun	**DEahcSPX [80]**	**ODE [34]**	**BFA [83]**	**PSO [82]**
f_1	6.82e-18(2.68e-18)	5.61e-24(5.24e-24)	7.8e-03(1e-03)	7.35e+00(1.95e+00)
f_2	2.59e-14(6.34e-15)	6.73e-13(2.17e-13)	4.72e-01(5.28e-02)	3.04e-01(1.87e-01)
f_3	2.65e-12(1.57e-12)	2.95e-08(2.19e-08)	3.00e-03(4.80e-03)	1.67e+03(4.69e+02)
f_4	2.19e-08(5.17e-09)	**2.08e-37(2.77e-37)**	2.80e-01(1.24e-02)	6.67e+01(1.56e+01)
f_5	**3.69e-22(8.56e-22)**	2.37e+01(1.50e+00)	6.33e+01(4.24e+01)	9.89e+02(6.40e+02)
f_6	3.24e+04(1.13e+03)	2.48e+04(8.83e+02)	3.86e+01(3.12e+00)	1.15e+01(3.15e+00)
f_7	5.84e-03(1.54e-03)	**2.04e-03(6.04e-04)**	1.16e-01(3.85e-02)	3.0e+01(1.34e-02)
f_8	**0e+00(0e+00)**	**0e+00(0e+00)**	4.12e+03(6.55e+02)	2.01e+03(3.86e+02)
f_9	2.56e+05(6.29e+03)	2.32e+05(1.17e+04)	1.49e+02(6.63e+00)	5.29e+01(1.08e+01)
f_{10}	7.16e-10(1.74e-10)	9.50e-13(3.34e-13)	4.70(8.21e-01)	1.54e+00(1.02e+00)
f_{11}	1.45e+05(3.28e+03)	1.20e+05(6.81e+03)	5.74e-02(7.30e-03)	9.92e-01(1.09e-01)
f_{12}	4.15e-19(2.60e-19)	8.14e-25(8.63e-25)	1.67e+01(9.84e+00)	9.92e-02(1.29e-01)
f_{13}	6.04e-17(2.78e-17)	5.99e-21(9.39e-21)	3.89e-01(9.09e-02)	3.88e+00(1.54e+00)
Mean Rank	5.808	4.962	9.308	10.308
Fun	**FBSA [83]**	**BBO [84]**	**ABC [85]**	**GABC [85]**
f_1	**2.50e-135(4.22e-13)**	1.529e-01(7.85e-02)	2.02e-13 (2.15e-13)	1.92e-22 (1.16e-22)
f_2	2.01e-29(3.31e-31)	1.178e-01(1.74e-02)	1.43e-07 (4.35e-08)	3.23e-12 (9.27e-13)
f_3	1.49e-04(2.93e-05)	1.045e+03(4.03e+02)	NA	NA
f_4	6.5e-03(2.3e-04)	4.345e-01(1.40e-01)	1.64e+01 (2.88e-00)	7.00e-00 (1.01e-00)
f_5	3.02e+01(1.28e+00)	8.552e+01(7.68e+00)	2.39e-01 (1.59e-01)	1.15e-00 (2.81e-00)
f_6	**0e+00(0e+00)**	**0e+00(0e+00)**	**0e+00 (0e+00)**	**0e+00(0e+00)**
f_7	5.63e-02(2.72e-02)	1.980e-02(8.99e-03)	1.93e-01 (5.29e-02)	8.94e-02 (2.42e-02)
f_8	5.53e+03(3.02e-02)	**0e+00(4.50e-02)**	3.78e+01 (1.40e-00)	2.09e-12 (6.66e-13)
f_9	3.54e+001(9.3e+00)	1.450e-02(6.20e-03)	1.55e-06 (2.87e-06)	1.15e-15 (3.22e-15)
f_{10}	7.99e-15(0e+00)	1.047e-01(2.18e-02)	2.05e-06 (5.86e-07)	2.15e-11 (1.06e-11)
f_{11}	**0e+00(0e+00)**	9.440e-02(3.95e-02)	3.39e-09 (1.00e-08)	2.08e-03 (6.54e-03)
f_{12}	**3.87e-31(8.47e-32)**	1.100e-03(1.70e-03)	1.43e-14 (1.60e-14)	3.07e-24 (4.71e-24)
f_{13}	1.35e-04(9.72e-06)	NA	2.02e-13 (1.67e-13)	**4.80e-23 (3.80e-23)**
Mean Rank	4.769	8.115	6.808	5.423

Table 6 Experiment one: Nemenyi post hoc test comparison of the algorithms compared in Table 5. The symbol "▲" indicates a statistical difference with $\alpha = 0.1$, while the symbol "●" indicates a statistical difference with $\alpha = 0.05$. A symbol in a cell indicates that the method in the row is statistically *worse* than the method in the column. Thus, empty cells indicate no statistical difference.

Method	Configuration 3 st3	FEP	CEP	SaDE	DEahcSPX	ODE	BFA	PSO	FBSA	BBO	ABC	GABC
Configuration 3 st3												
FEP												
CEP	●			●								
SaDE												
DEahcSPX												
ODE												
BFA	●			●		▲			▲			
PSO	●			▲	▲	●			●			●
FBSA												
BBO	▲			●								
ABC												
GABC												

Table 7 Experiment two: DE and clustering settings considered for the parameter search.

Parameter	Values
Scale factor	$F = 0.5, 0.9$
Crossover rate	$CR = 0.5, 0.9$
Clusters	$k = 2, 5, 10$
Population size	$N = k \times \{5, 10, 20\}$
Mutation strategy	*st3*, *st5*

Table 8 Experiment two: Selected CDE configurations after the parameter search.

Best aggregated results					
Name	***F***	***CR***	***k***	***N***	***st***
CDE-Conf-29	0.9	0.9	5	25	3
CDE-Conf-30	0.9	0.9	10	50	3
CDE-Conf-65	0.9	0.9	5	25	5
CDE-Conf-67	0.9	0.9	2	20	5
Closest to the *Best 2009* performance curve					
CDE-Conf-10	0.9	0.5	2	10	3
CDE-Conf-28	0.9	0.9	2	10	3

3. IPOPCMAv3p61, i.e., IPOP-CMA-ES with restart mechanism and population doubled at each restart [89];
4. CMA-ES with Gaussian Processes (GP5-CMA-ES) and Random Forests (RF5-CMA-ES) surrogate models, i.e., CMA-ES using respectively GP or RF as surrogate models to reduce the NFEs [89];
5. LHD-10xDefault, included in the MATLAB Surrogate Model Toolbox (MATSuMoTo), which uses multivariate Gaussian Processes as surrogate models and the expected improvement as infill criterion. Latin Hypercube is used to obtain the samples in the search space needed to build the surrogate model [90].

As it can be noticed, some of the methods above use surrogate models to improve the search, and as such they are supposed to achieve better results using a lower NFEs.

Furthermore, the comparison included the Bi-Population CMA-ES (BIPOP-CMA-ES) from the BBOB-2009 competition, to date one of the best methods for continuous global optimization. Finally, as a reference algorithm we included the *Best 2009* results available in COCO.

According to the indications given in the COCO framework [86, 91], we report the results of this comparison in terms of bootstrapped empirical cumulative distribution of the NFEs divided by problem dimensionality, see Figure 3. The figure shows the aggregated results (i.e., not divided by function group) of all the compared algorithms on the 24 benchmark functions on the five dimensionalities considered. The main performance measure used in COCO is the expected running time (ERT), which estimates the expected NFEs required to reach a particular target function value if an algorithm is continued until it finds that value. The ERT depends on a given target function value, $f_t = f_{opt} + \Delta f$, and is computed over all the available runs of an algorithm as the NFEs executed during each run until the best function value did not reach f_t, summed over all runs and divided by the number of runs that actually reached f_t [92, 93].

For the 2-D functions, it can be seen that our proposed CDE algorithm performed quite well, being CDE-Conf-28 the second-best method. Almost all our CDE configurations outperformed most of the CMA-ES variants, which is an important result as CMA-ES is the basis for some of the most effective methods for continuous global optimization. CMA-TPA obtained overall the best position (proportion of functions solved), but CDE-Conf-28 was faster (closer to the *Best 2009* performance curve, solving 80% of the problems in only $100 \times D$ NFEs). Furthermore, it can be noted that while both CMA-TPA and CDE-Conf-28 were slower than the Brent ones, they solved more problems to the desired precision (1e-8).

Regarding the 3-D functions, the best CDE configuration (CDE-Conf-65) uses *st5*. It is more exploratory than *st3* because it employs two *random* vectors, but not at the

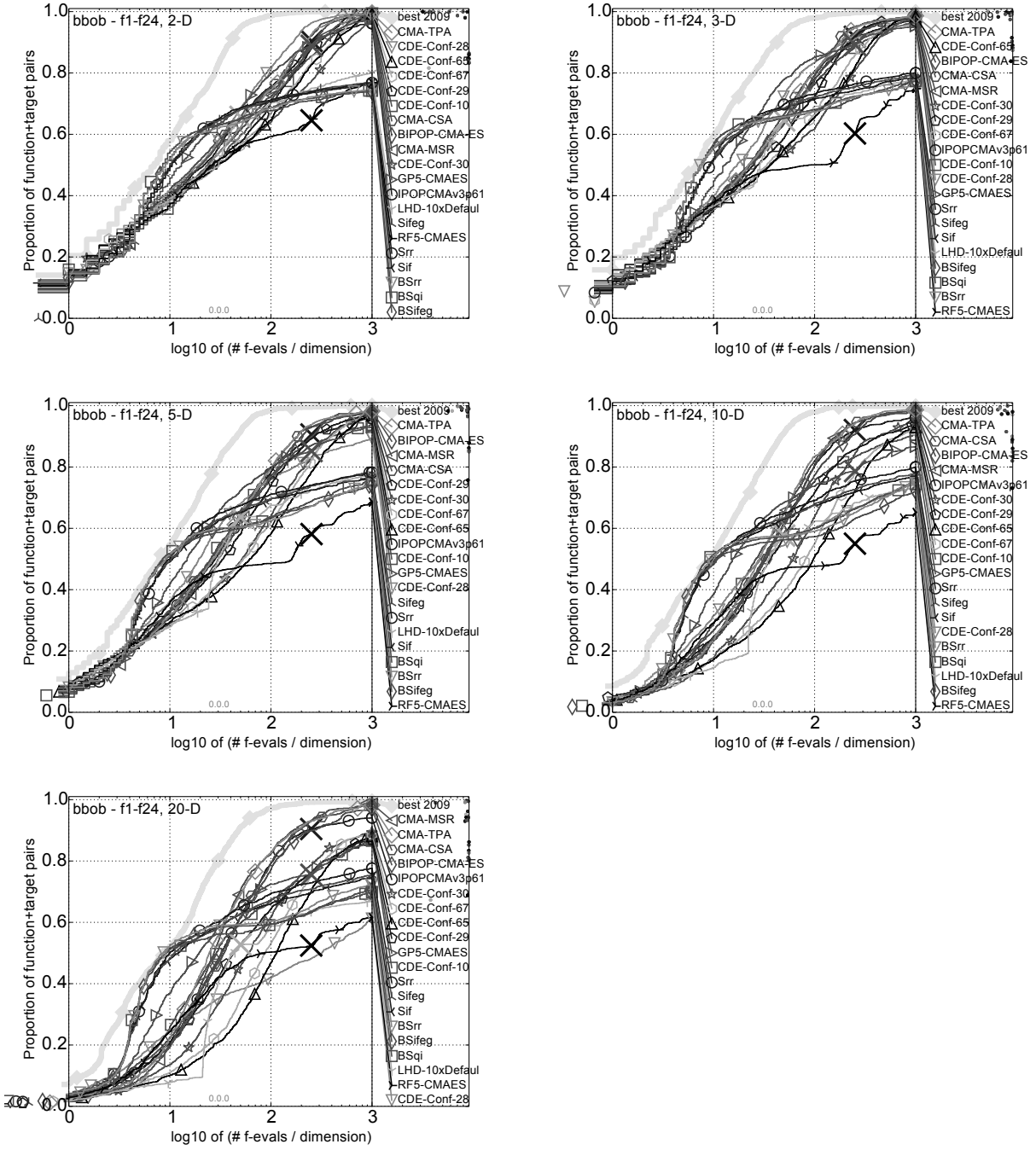

Fig. 3 Experiment two: Bootstrapped empirical cumulative distribution of the NFEs divided by problem dimensionality (NFEs/D) for 50 targets in $10^{[-8..2]}$ for all functions and dimensions. The *Best 2009* curve corresponds to the best expected running time (ERT) observed at the BBOB-2009 competition for each target.

maximum allowed capability (5 clusters instead of 10, and population size of 25 instead of 200). This configuration was the second-best method, also in this case being CMA-TPA the first one, and it outperformed six CMA-ES variants. However, CDE-Conf-65 was slow, taking $500 \times D$ NFEs to achieve 80% of the targets. Although faster, the other CDE configurations could only outperform IPOP-CMA-ES and the surrogate versions. In particular, CDE-Conf-28 was the fastest CDE configuration, getting close to the *Best 2009* performance curve for $10 \times D <$ NFEs $< 100 \times D$, but was unable to refine the solutions and adequately solve enough problems. For that reason, it had the worst aggregated result. Nevertheless, all the CDE configurations found high-quality solutions, while the Brent methods, LHD-10xDefault, and

RF5-CMA-ES failed. GP5-CMA-ES, on the other hand, was the fastest method until $100 \times D$ NFEs.

Scaling the problems to 5 dimensions, the CDE performance started to decrease especially w.r.t. the CMA-ES variants. Nevertheless, most CDE configurations could still beat IPOP-CMA-ES and the surrogate methods. The best CDE result was achieved by CDE-Conf-29, which being based on *st3* considers the current best solution to calculate the difference vector. After $100 \times D$ NFEs, this configuration solved approximately 75% of the problems. CDE-Conf-28 was once again faster than the other CDE configurations (see the curve in the range $10 \times D <$ NFEs $< 100 \times D$). Interestingly, CDE-Conf-65 was in this case the slowest configuration; nevertheless, it solved a large proportion of functions.

For 10-D, all the CDE configurations were outperformed by the CMA-ES variants, except those ones based on surrogate models. In this case CDE was still able to solve a large proportion of functions (see CDE-Conf-30) faster than the CMA-ES variants at low budgets, but it became slower and less precise as the budget increased. A local search method could be necessary from this point to solve more problems.

Finally, for 20-D the results show that the majority of the CDE configurations solved approximately 90% of the problems (except for CDE-Conf-28) after $1,000 \times D$ NFEs. They became much slower than CMA-ES after $10 \times D$ NFEs, but the overall results are promising and encourage further research to improve the performance of the CDE algorithm.

Overall, there seems to be a relationship between the problem size (i.e., the number of dimensions) and the number of clusters used by CDE, see Table 8. In fact, the best configurations were CDE-Conf-28 (k = 2), CDE-Conf-65 (k = 5), CDE-Conf-29 (k = 5), CDE-Conf-30 (k = 10), and CDE-Conf-30 (k = 10) for 2, 3, 5, 10, and 20 dimensions respectively. The opposite is also true, as the worst results were achieved by CDE-Conf-28 (k = 2) for $D > 2$. Therefore, such information could be helpful to adapt the number of clusters to be used by the algorithm depending on the problem size.

To conclude our analysis of the second experiment, we compared the algorithms above w.r.t. their success rate per function, as it is reported by COCO. This additional comparison is aimed at identifying on which problems exactly the proposed CDE algorithm performs better than the other algorithms from the state-of-the-art. An important aspect to note here is the success rate is defined in terms percentage of runs that reached the targets, regardless of the NFEs needed to do that. Therefore, this comparison provides different information w.r.t. the bootstrapped empirical cumulative distribution shown in Figure 3.

In Tables 9 and 10, we show the success rates on the problems for 5-D and 10-D, respectively[5]. In 5 dimensions, CDE solved most problems with a performance similar to that of the best algorithms (i.e., the CMA-ES variants). It even outperformed CMA-ES on f_3 and f_4 (separable functions: Rastrigin function and Büche-Rastrigin function, respectively), as well as f_{21} and f_{22} (weakly structured multimodal functions: Gallagher Gaussian functions). On the other hand, CDE failed almost completely on functions f_{15} and f_{19} (multimodal functions: Rastrigin function and Composite Griewank-Rosenbrock function F8F2, respectively), f_{23} and f_{24} (weakly structured multimodal functions: Katsuura function and Lunacek bi-Rastrigin function, respectively). IPOP-CMA-ES showed a very poor overall performance, while BIPOP-CMA-ES showed overall the highest success rates. Of note, the surrogate methods failed on most benchmark functions, although one may assume that a larger budget could improve their performance. Finally, the Brent methods were the best for the separable functions, but failed for the remaining problems.

In 10 dimensions, CDE failed completely on more problems and outperformed CMA-ES only on functions f_{21} and f_{22}. CDE-Conf-28 did not solve any problem to the desired precision, while CDE-Conf-30 solved nine of them. After checking the results, we believe that a better refinement process would improve the success rate, as CDE did find high-quality solutions yet with a slightly larger error than the desired precision. Once again BIPOP-CMA-ES was the overall best method, followed by the three CMA-ES variants (CSA, MSR, and TPA). Also in this case the Brent methods perfectly solved the separable functions, but failed on the remaining problems. The surrogate methods failed again on most benchmark functions. As they did not solve the 5-D problems, probably due to the small budget, it is understandable that they could not solve the larger instances either.

5 Conclusions

In this paper, we proposed eight novel cluster-centroid-based mutations strategies for Differential Evolution (DE), in the attempt to improve the performance of the traditional *rand/1* mutation strategy. In particular, we used a hierarchical clustering method (UPGMC) to cluster the solutions at each generation of the DE, and calculate the related centroids. We hypothesized that the information obtained by cluster centroids could provide the location of promising regions of the search space, thus improving the search efficiency.

In order to verify our hypothesis, we performed two experiments on two sets of well-known box-constrained global

[5] For brevity, we omit the results for 2-D and 3-D, where most of the compared methods showed 100% success rate. Likewise, we omit the results for 20-D, where the proposed CDE algorithm did not solve any of the problems to the desired precision (1e-8), thus having 0% success rate. We did note however that the algorithm could solve some problems if the budget is increased to $10e + 06 \times D$, as used in some works in the literature. However, this analysis is out of the scope of the comparison reported in this paper.

Table 9 Experiment two: Success rates across 15 independent runs with precision 1e-8 on 5-D problems. The symbol "-" indicates 0% success rate.

Algorithm	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12
Best 2009	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
IPOPCMAv3p61	1.00	-	-	-	1.00	-	0.20	-	-	-	-	-
BIPOP-CMA-ES	1.00	1.00	0.93	-	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CDE-Conf-10	1.00	1.00	1.00	0.40	0.67	1.00	0.87	1.00	0.93	-	-	0.07
CDE-Conf-28	1.00	0.67	-	-	0.07	0.07	0.73	-	-	-	-	-
CDE-Conf-30	1.00	1.00	0.67	-	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CDE-Conf-67	1.00	1.00	-	-	-	0.80	1.00	1.00	1.00	1.00	1.00	0.33
CDE-Conf-65	1.00	1.00	-	-	0.07	0.87	1.00	1.00	1.00	1.00	1.00	0.87
CDE-Conf-29	1.00	1.00	0.33	0.20	0.67	1.00	1.00	1.00	1.00	1.00	0.13	0.27
CMA-CSA	1.00	1.00	0.33	-	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CMA-MSR	1.00	1.00	0.93	-	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CMA-TPA	1.00	1.00	0.33	-	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
GP5-CMA-ES	0.73	0.13	-	-	1.00	-	-	-	-	0.07	0.07	-
LHD-10xDefault	-	-	-	-	1.00	-	-	-	-	-	-	-
RF5-CMA-ES	-	-	-	-	0.67	-	-	-	-	-	-	-
Sif	1.00	1.00	1.00	1.00	1.00	-	-	-	-	-	-	-
Sifeg	1.00	1.00	1.00	1.00	1.00	-	-	-	-	-	-	-
Srr	1.00	1.00	1.00	1.00	1.00	-	-	-	-	-	-	-
BSifeg	1.00	1.00	1.00	1.00	1.00	-	-	-	-	-	-	-
BSqi	1.00	1.00	1.00	1.00	1.00	-	-	-	-	-	-	-
BSrr	1.00	1.00	1.00	1.00	1.00	-	-	-	-	-	-	-

Algorithm	f13	f14	f15	f16	f17	f18	f19	f20	f21	f22	f23	f24
Best 2009	1.00	1.00	0.93	1.00	1.00	1.00	1.00	0.93	0.93	0.93	1.00	0.20
IPOPCMAv3p61	-	-	-	-	-	-	-	-	-	0.07	-	-
BIPOP-CMA-ES	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.20
CDE-Conf-10	-	-	-	-	-	-	-	0.67	0.67	0.80	-	-
CDE-Conf-28	-	-	-	-	-	-	-	-	0.40	0.20	-	-
CDE-Conf-30	1.00	1.00	-	0.07	0.07	-	-	0.60	1.00	1.00	-	-
CDE-Conf-67	0.40	1.00	-	-	0.87	0.67	-	0.67	1.00	1.00	-	-
CDE-Conf-65	0.33	1.00	-	0.07	0.80	0.40	-	0.93	1.00	1.00	-	-
CDE-Conf-29	0.07	0.67	0.13	-	-	-	-	0.73	1.00	1.00	-	-
CMA-CSA	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.60	0.40	1.00	-
CMA-MSR	1.00	1.00	1.00	1.00	1.00	1.00	-	-	0.40	0.47	1.00	-
CMA-TPA	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.67	0.53	0.87	-
GP5-CMA-ES	-	-	-	-	-	-	-	-	-	-	-	-
LHD-10xDefault	-	-	-	-	-	-	-	-	-	-	-	-
RF5-CMA-ES	-	-	-	-	-	-	-	-	-	-	-	-
Sif	-	-	-	-	-	-	-	0.13	0.07	-	-	-
Sifeg	-	-	-	-	-	-	-	0.20	0.13	-	-	-
Srr	-	-	-	-	-	-	-	0.07	0.20	-	-	-
BSifeg	-	-	-	-	-	-	-	0.07	0.13	-	-	-
BSqi	-	-	-	-	-	-	-	-	-	-	-	-
BSrr	-	-	-	-	-	-	-	-	-	-	-	-

Table 10 Experiment two: Success rates across 15 independent runs with precision 1e-8 on 10-D problems. The symbol "-" indicates 0% success rate.

Algorithm	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12
Best 2009	1.00	1.00	1.00	0.80	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
IPOPCMAv3p61	1.00	-	-	-	1.00	-	-	-	-	-	-	-
BIPOP-CMA-ES	1.00	1.00	0.13	-	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CDE-Conf-10	1.00	1.00	-	-	-	-	-	-	-	-	-	-
CDE-Conf-28	-	-	-	-	-	-	-	-	-	-	-	-
CDE-Conf-30	1.00	1.00	-	-	-	1.00	0.20	0.93	1.00	-	-	0.07
CDE-Conf-67	1.00	1.00	-	-	-	-	0.87	0.33	0.07	-	-	-
CDE-Conf-65	1.00	1.00	-	-	-	-	1.00	0.73	0.27	-	-	-
CDE-Conf-29	1.00	1.00	-	-	-	0.07	-	0.33	0.27	-	-	-
CMA-CSA	1.00	1.00	-	-	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CMA-MSR	1.00	1.00	1.00	-	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
CMA-TPA	1.00	1.00	0.07	-	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
GP5-CMA-ES	0.27	-	-	-	1.00	-	-	-	-	-	-	-
LHD-10xDefault	-	-	-	-	1.00	-	-	-	-	-	-	-
RF5-CMA-ES	-	-	-	-	0.67	-	-	-	-	-	-	-
Sif	1.00	1.00	1.00	1.00	1.00	-	-	-	-	-	-	-
Sifeg	1.00	1.00	1.00	1.00	1.00	-	-	-	-	-	-	-
Srr	1.00	1.00	1.00	1.00	1.00	-	-	-	-	-	-	-
BSifeg	1.00	1.00	1.00	1.00	1.00	-	-	-	-	-	-	-
BSqi	1.00	1.00	1.00	1.00	1.00	-	-	-	-	-	-	-
BSrr	1.00	1.00	1.00	1.00	1.00	-	-	-	-	-	-	-

Algorithm	f13	f14	f15	f16	f17	f18	f19	f20	f21	f22	f23	f24
Best 2009	1.00	1.00	0.80	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.07
IPOPCMAv3p61	-	-	-	-	-	-	-	-	0.07	-	-	-
BIPOP-CMA-ES	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.60	1.00	0.07
CDE-Conf-10	-	-	-	-	-	-	-	-	0.13	-	-	-
CDE-Conf-28	-	-	-	-	-	-	-	-	-	-	-	-
CDE-Conf-30	-	-	-	-	-	-	-	-	0.93	0.80	-	-
CDE-Conf-67	-	-	-	-	-	-	-	-	0.80	0.20	-	-
CDE-Conf-65	-	-	-	-	-	-	-	-	0.93	0.40	-	-
CDE-Conf-29	-	-	-	-	-	-	-	-	0.80	0.27	-	-
CMA-CSA	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.47	0.07	0.40	-
CMA-MSR	1.00	1.00	1.00	1.00	1.00	1.00	-	-	0.47	0.07	1.00	-
CMA-TPA	1.00	1.00	1.00	1.00	1.00	1.00	0.87	0.07	0.53	0.07	0.67	-
GP5-CMA-ES	-	-	-	-	-	-	-	-	-	-	-	-
LHD-10xDefault	-	-	-	-	-	-	-	-	-	-	-	-
RF5-CMA-ES	-	-	-	-	-	-	-	-	-	-	-	-
Sif	-	-	-	-	-	-	-	-	-	-	-	-
Sifeg	-	-	-	-	-	-	-	-	-	-	-	-
Srr	-	-	-	-	-	-	-	-	-	-	-	-
BSifeg	-	-	-	-	-	-	-	-	-	-	-	-
BSqi	-	-	-	-	-	-	-	-	-	-	-	-
BSrr	-	-	-	-	-	-	-	-	0.07	-	-	-

optimization benchmark functions. In the first experiment, we tested the proposed cluster-centroid-based mutation strategies with three different DE configurations on 13 selected benchmark functions, and used the results in three distinct comparisons. The first comparison considered the basic DE with *rand/1* mutation strategy: this result showed that the proposed cluster-centroid-based mutations statistically outperformed the traditional one, thus proving that using cluster centroid information to guide DE does provide a benefit on the search efficiency. Another interesting finding is that this positive effect does not correlate with the number of clusters on the contrary, the performance seems to decrease as more clusters are used. The solution might be to increase the population size in order to have a better sample of the search space leading to better clusters, at the cost of increasing the runtime.

The second comparison was with two other cluster-centroid-based versions of DE from the literature, i.e., W-CDE [53] and C-DE [52]. Differently from our approach, these two algorithms do not employ clustering information in the mutation strategies, but rather use cluster centroids to replace, periodically, some of the solutions in the population. Our results show that our cluster-centroid-based approach is competitive against both W-CDE and C-DE in terms of solution quality.

The third comparison included results from recent metaheuristics of different kinds, most of which nature-inspired. Also in this case, our algorithm compared favorably against the other methods, outperforming well-established methods such as PSO and ABC.

We then performed a second experiment on a harder benchmark (BBOB), and compared the proposed approach with the best algorithms from the BBOB-2015 competition, as well as BIPOP-CMA-ES from the BBOB-2009 competition. The results, in terms of aggregated bootstrapped empirical cumulative distribution, showed that in the case of low-dimensional problems the proposed cluster-centroid-based mutation strategies could achieve a performance similar to, and in some cases better than, that of various CMA-ES variants. On the other hand, on the higher-dimensional problems the performance of the proposed approach degraded, mostly because of a lack of better exploitation capabilities.

A possible reason for this lack of exploitation is the fact that in our proposal clustering is done in the solution space (i.e., clusters can contain mixed quality solutions), thus ignoring any fitness information. Therefore, we intend to investigate the inclusion of the solutions' quality (i.e., their fitness values) in the clustering process. Moreover, it will be interesting to consider alternative scale factor schemes (e.g. based on randomized scale factors) to help the clustering process and avoid adding explicit noise as we did here.

Another possible explanation for the degradation of performance as the problem dimensionality increases is the fact

that in our current proposal the number of clusters (k) was fixed during the entire DE optimization process. However, as the solutions change over time, it might be advantageous to adjust k dynamically during the evolutionary process. This intuition is supported by the observation that, for the harder high-dimensional problems, the best configurations of our method used a number of clusters $k \approx D$. Thus, larger problems were solved more efficiently by configurations using more clusters. Even though this observation seems to contradict the results obtained in the first experiment, where we noted that better results were obtained with less clusters, we believe that studying alternative clustering methods, capable to either automatically select k depending on the problem size and characteristics, or adjust k over time, might be an interesting research direction.

Regarding computational cost and speed, the technique presented in this paper is certainly slower than the other methods used in the comparisons: this is because the clustering process is executed at every DE generation, calculating the pairwise distances among all the solutions to be clustered (this process has $\mathcal{O}(N^3)$ time complexity [68], with N being the population size). Future investigations might be devoted to reduce this overhead, e.g. by introducing a dynamic scheme that automatically adapts the frequency of the clustering updates, or using a different clustering algorithm.

To conclude, the results presented herein contribute to justifying the hybridization of metaheuristics and machine-learning methods to improve the search. In particular, the use of cluster centroid information is a promising research topic that could be exploited not only in advanced DE variants, but also in other population-based metaheuristics.

Declarations

Funding: Not applicable.
Conflicts of interest/Competing interests: Not applicable.
Availability of data and material: The raw data are available upon request.
Code availability: The code is available upon request.
Authors' contributions: Both authors contributed to the design of this study. Experiments and data collection were performed by V. V. de Melo. The analysis was performed by V. V. de Melo and G. Iacca. The first draft of the manuscript was written by both authors. Both authors read and approved the final manuscript.

Compliance with ethical standards

Conflict of Interest: The authors declare that they have no conflict of interest.

References

1. Davis, Lawrence and others: Handbook of genetic algorithms, vol. 115. Van Nostrand Reinhold (1991)
2. Shi, Yuhui and Eberhart, Russell: A modified particle swarm optimizer. In: IEEE Congress on Evolutionary Computation, pp. 69–73 (1998)
3. Geem, Zong Woo and Kim, Joong Hoon and Loganathan, GV: A new heuristic optimization algorithm: harmony search. Simulation **76**(2), 60–68 (2001)
4. Karaboga, Dervis and Basturk, Bahriye: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization **39**(3), 459–471 (2007)
5. Tuba, Milan and Brajevic, Ivona and Jovanovic, Raka: Hybrid Seeker Optimization Algorithm for Global Optimization. Applied Mathematics & Information Sciences **7**(3), 867–875 (2013)
6. Hamed Soleimani and Govindan Kannan: A hybrid particle swarm optimization and genetic algorithm for closed-loop supply chain network design in large-scale networks. Applied Mathematical Modelling **39**(14), 3990–4012 (2015)
7. Ligang Cui and Jie Deng and Lin Wang and Maozeng Xu and Yajun Zhang: A novel locust swarm algorithm for the joint replenishment problem considering multiple discounts simultaneously. Knowledge-Based Systems **111**, 51–62 (2016)
8. Zhenyu Meng and Jeng-Shyang Pan: Monkey King Evolution: A new memetic evolutionary algorithm and its application in vehicle fuel consumption optimization. Knowledge-Based Systems **97**, 144–157 (2016)
9. Marko Mitić and Najdan Vuković and Milica Petrović and Zoran Miljković: Chaotic fruit fly optimization algorithm. Knowledge-Based Systems **89**, 446–458 (2015)
10. Wolpert, David H. and Macready, William G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1), 67–82 (1997)
11. de Melo, Vinicius Veloso and Delbem, Alexandre Claudio Botazzo and Pinto Junior, Dorival Leao and Federson, Fernando Marques: Improving global numerical optimization using a search-space reduction algorithm. In: Genetic and Evolutionary Computation Conference, pp. 1195–1202 (2007)
12. Vinícius Veloso de Melo and Alexandre Cláudio Botazzo Delbem: Investigating Smart Sampling as a population initialization method for Differential Evolution in continuous problems. Information Sciences **193**, 36–53 (2012)
13. de Melo, Vinicius V. and Delbem, Alexandre C. B.: Using Smart Sampling to Discover Promising Regions and Increase the Efficiency of Differential Evolution. In:

International Conference on Intelligent Systems Design and Applications, pp. 1394–1399 (2009)
14. Laetitia Jourdan and Clarisse Dhaenens and El-Ghazali Talbi: Using Datamining Techniques to Help Metaheuristics: A Short Survey. In: Hybrid Metaheuristics, pp. 57–69 (2006)
15. de Melo, Vinicius Veloso and Delbem, Alexandre C. B. and Pinto Junior, Dorival Leao and Federson, Fernando Marques: Discovering Promising Regions to Help Global Numerical Optimization Algorithms. In: Mexican International Conference on Artificial Intelligence, pp. 72–82 (2007)
16. Noman, N. and Iba, H.: Accelerating Differential Evolution Using an Adaptive Local Search. IEEE Transactions on Evolutionary Computation **12**(1), 107–125 (2008)
17. Chandra Sekhar Pedamallu and Linet Ozdamar: Investigating a hybrid simulated annealing and local search algorithm for constrained optimization. European Journal of Operational Research **185**(3), 1230–1245 (2008)
18. Noman, Nasimul and Bollegala, Danushka and Iba, Hitoshi: Differential evolution with self adaptive local search. In: Genetic and Evolutionary Computation Conference, pp. 1099–1106 (2011)
19. Storn, Rainer and Price, Kenneth: Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization **11**(4), 341–359 (1997)
20. Vinícius V. de Melo and Grazieli L. C. Carosio: Investigating Multi-View Differential Evolution for solving constrained engineering design problems. Expert Systems with Applications **40**(9), 3370–3377 (2013)
21. Boussaïd, Ilhem and Chatterjee, Amitava and Siarry, Patrick and Ahmed-Nacer, Mohamed: Biogeography-based optimization for constrained optimization problems. Computers & Operations Research **39**(12), 3293–3304 (2012)
22. Kovačević, Darko and Mladenović, Nenad and Petrović, Bratislav and Milošević, Pavle: DE-VNS: Self-adaptive Differential Evolution with crossover neighborhood search for continuous global optimization. Computers & Operations Research **52, Part B**(0), 157–169 (2014)
23. Marco Locatelli and Mirko Maischberger and Fabio Schoen: Differential evolution methods based on local searches. Computers & Operations Research **43**(0), 169–180 (2014)
24. dos Santos Coelho, Leandro and Souza, Rodrigo Clemente Thom and Mariani, Viviana Cocco: Improved differential evolution approach based on cultural algorithm and diversity measure applied to solve economic load dispatch problems. Mathematics and Computers in Simulation **79**(10), 3136–3147 (2009)
25. Pedrosa Silva, Rodrigo César and Lopes, Rodolfo Ayala and Guimarães, Frederico Gadelha: Self-adaptive mutation in the differential evolution. In: Genetic and Evolutionary Computation Conference, pp. 1939–1946 (2011)
26. Quan-Ke Pan and P.N. Suganthan and Ling Wang and Liang Gao and R. Mallipeddi: A differential evolution algorithm with self-adapting strategy and control parameters. Computers & Operations Research **38**(1), 394–408 (2011)
27. Feoktistov, V.: Differential evolution. Springer (2006)
28. Iacca, Giovanni and Caraffini, Fabio and Neri, Ferrante: Multi-Strategy Coevolving Aging Particle Optimization. International Journal of Neural Systems **24**(01) (2013)
29. Iacca, Giovanni and Neri, Ferrante and Caraffini, Fabio and Suganthan, Ponnuthurai Nagaratnam: A differential evolution framework with ensemble of parameters and strategies and pool of local search algorithms. In: European Conference on the Applications of Evolutionary Computation, pp. 615–626 (2014)
30. Iacca, Giovanni and Caraffini, Fabio and Neri, Ferrante: Continuous Parameter Pools in Ensemble Self-Adaptive Differential Evolution. In: IEEE Symposium Series on Computational Intelligence, pp. 1529–1536 (2015)
31. Yaman, Anil and Iacca, Giovanni and Coler, Matt and Fletcher, George and Pechenizkiy, Mykola: Multi-strategy differential evolution. In: International Conference on the Applications of Evolutionary Computation, pp. 617–633 (2018)
32. Karol Opara and Jarosław Arabas: Comparison of mutation strategies in Differential Evolution - A probabilistic perspective. Swarm and Evolutionary Computation **39**, 53–69 (2018)
33. Yaman, Anil and Iacca, Giovanni and Caraffini, Fabio: A comparison of three differential evolution strategies in terms of early convergence with different population sizes. In: AIP Conference Proceedings, vol. 2070-1, p. 020002 (2019)
34. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.: Opposition-based differential evolution. IEEE Transactions on Evolutionary computation **12**(1), 64–79 (2008)
35. Rahnamayan, S., Jesuthasan, J., Bourennani, F., Naterer, G.F., Salehinejad, H.: Centroid opposition-based differential evolution. International Journal of Applied Metaheuristic Computing (IJAMC) **5**(4), 1–25 (2014)
36. Rahnamayan, S., Jesuthasan, J., Bourennani, F., Salehinejad, H., Naterer, G.F.: Computing opposition by involving entire population. In: 2014 IEEE congress on evolutionary computation (CEC), pp. 1800–1807. IEEE (2014)
37. Rahnamayan, S., Wang, G.G.: Center-based sampling for population-based algorithms. In: 2009 IEEE

Congress on Evolutionary Computation, pp. 933–938. IEEE (2009)
38. Salehinejad, H., Rahnamayan, S.: Effects of centralized population initialization in differential evolution. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8. IEEE (2016)
39. Khanum, R.A., Jan, M.A.: Centroid-based initialized jade for global optimization. In: 2011 3rd Computer Science and Electronic Engineering Conference (CEEC), pp. 115–120. IEEE (2011)
40. Salehinejad, H., Rahnamayan, S., Tizhoosh, H.R.: CenDE: Centroid-based differential evolution. In: 2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE), pp. 1–4. IEEE (2018)
41. Esmailzadeh, A., Rahnamayan, S.: Enhanced differential evolution using center-based sampling. In: 2011 IEEE Congress of Evolutionary Computation (CEC), pp. 2641–2648. IEEE (2011)
42. Hiba, H., El-Abd, M., Rahnamayan, S.: Improving SHADE with center-based mutation for large-scale optimization. In: 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 1533–1540. IEEE (2019)
43. Mousavirad, S.J., Rahnamayan, S.: A novel center-based differential evolution algorithm. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2020)
44. Hiba, H., Mahdavi, S., Rahnamayan, S.: Differential evolution with center-based mutation for large-scale optimization. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8. IEEE (2017)
45. Hiba, H., Ibrahim, A., Rahnamayan, S.: Large-scale optimization using center-based differential evolution with dynamic mutation scheme. In: 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 3189–3196. IEEE (2019)
46. Hiba, H., Bidgoli, A.A., Ibrahim, A., Rahnamayan, S.: CGDE3: An efficient center-based algorithm for solving large-scale multi-objective optimization problems. In: 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 350–358. IEEE (2019)
47. Mahdavi, S., Rahnamayan, S., Deb, K.: Center-based initialization of cooperative co-evolutionary algorithm for large-scale optimization. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 3557–3565. IEEE (2016)
48. Mousavirad, S.J., Rahnamayan, S.: CenPSO: A novel center-based particle swarm optimization algorithm for large-scale optimization. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 2066–2071. IEEE (2020)
49. Fan, Hui-Yuan and Lampinen, Jouni: A Trigonometric Mutation Operation to Differential Evolution. Journal of Global Optimization **27**(1), 105–129 (2003)
50. Ali, Musrrat and Pant, Millie and Nagar, Atulya: Two New Approach Incorporating Centroid Based Mutation Operators For Differential Evolution. World Journal of Modelling and Simulation **7**(1), 16–28 (2011)
51. Tan, Pang-Ning and Steinbach, Michael and Kumar, Vipin: Introduction to Data Mining. Addison Wesley (2005)
52. Zhihua Cai and Wenyin Gong and Charles X. Ling and Harry Zhang: A clustering-based differential evolution for global optimization. Applied Soft Computing **11**(1), 1363–1379 (2011)
53. Yong-Jun Wang and Jiang-She Zhang and Gai-Ying Zhang: A dynamic clustering based differential evolution algorithm for global optimization. European Journal of Operational Research **183**(1), 56–73 (2007)
54. Sneath, Peter HA and Sokal, Robert R: Numerical taxonomy. The principles and practice of numerical classification. W.H. Freeman (1973)
55. Vinícius Veloso de Melo and Grazieli Luiza Costa Carosio: Evaluating differential evolution with penalty function to solve constrained engineering problems. Expert Systems with Applications **39**(9), 7860–7863 (2012)
56. Ferrante Neri and Giovanni Iacca and Ernesto Mininno: Disturbed Exploitation compact Differential Evolution for limited memory optimization problems. Information Sciences **181**(12), 2469–2487 (2011)
57. Pan, Quan-Ke and Wang, Ling and Gao, Liang and Li, W. D.: An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers. Information Sciences **181**, 668–685 (2011)
58. Wang, Yu and Li, Bin and Weise, Thomas: Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems. Information Sciences **180**, 2405–2420 (2010)
59. Chakraborty,, Uday K.: Advances in Differential Evolution. Springer (2008)
60. Bhowmik, Pavel and Das, Sauvik and Konar, Amit and Das, Swagatam and Nagar, Atulya K.: A new differential evolution with improved mutation strategy. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2010)
61. Rui Mendes and Arvind S. Mohais: DynDE: a differential evolution for dynamic optimization problems. In: IEEE Congress on Evolutionary Computation, pp. 2808–2815 (2005)
62. Min Zhang and Wenjian Luo and Xufa Wang: Differential evolution with dynamic stochastic selection for constrained optimization. Information Sciences **178**(15), 3043–3074 (2008)
63. Swagatam Das and Ponnuthurai Nagaratnam Suganthan: Differential Evolution: A Survey of the State-of-the-Art. IEEE Transactions on Evolutionary Computa-

tion **15**(1), 4–31 (2011)
64. Jingqiao Zhang and Arthur C. Sanderson: JADE: Adaptive Differential Evolution With Optional External Archive. IEEE Transactions on Evolutionary Computation **13**(5), 945–958 (2009)
65. Bi, Shujun and Zhou, Jianjun: Adaptive Differential Evolution Based on New Mutation Strategy. In: International Conference on Computational and Information Sciences, pp. 1103–1106 (2011)
66. Vinicius Veloso de Melo and Grazieli Luiza Costa Carosio: Automatic generation of evolutionary operators: a study with mutation strategies for the differential evolution. In: Symposium on Applied Computing, pp. 188–193 (2013)
67. Pant, M. and Ali, M. and Abraham, A.: Mixed Mutation Strategy Embedded Differential Evolution. In: IEEE Congress on Evolutionary Computation, pp. 1240–1246 (2009)
68. Müllner, Daniel: Modern hierarchical, agglomerative clustering algorithms. arXiv preprint arXiv:1109.2378 (2011)
69. Hallawa, Ahmed and Yaman, Anil and Iacca, Giovanni and Ascheid, Gerd: A framework for knowledge integrated evolutionary algorithms. In: European Conference on the Applications of Evolutionary Computation, pp. 653–669 (2017)
70. Larrañaga, Pedro and Lozano, Jose A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers (2001)
71. Lozano, Jose A and Larrañaga, Pedro and Inza, Iñaki and Bengoetxea, Endika: Towards a new Evolutionary Computation: advances on estimation of distribution algorithms, vol. 192. Springer (2006)
72. Neri, Ferrante and Iacca, Giovanni and Mininno, Ernesto: Compact Optimization. In: Handbook of Optimization: From Classical to Modern Approach, pp. 337–364. Springer (2013)
73. Giovanni Iacca and Fabio Caraffini: Re-sampled inheritance compact optimization. Knowledge-Based Systems **208**, 106416 (2020)
74. Yao, Xin and Liu, Yong and Lin, Guangming: Evolutionary Programming Made Faster. IEEE Transactions on Evolutionary Computation **3**(2), 82–102 (1999)
75. Zaharie, D.: Critical values for the control parameters of differential evolution algorithms. In: International Conference on Soft Computing, pp. 62–67 (2002)
76. Zaharie, D.: Control of population diversity and adaptation in differential evolution algorithms. In: International Conference on Soft Computing, vol. 9, pp. 41–46 (2003)
77. Zaharie, D.: Influence of crossover on the behavior of differential evolution algorithms. Applied soft computing **9**(3), 1126–1138 (2009)
78. Demšar, Janez: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research **7**, 1–30 (2006)
79. Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. In: 2005 IEEE congress on evolutionary computation, vol. 2, pp. 1785–1791. IEEE (2005)
80. Noman, N., Iba, H.: Accelerating differential evolution using an adaptive local search. IEEE Transactions on evolutionary Computation **12**(1), 107–125 (2008)
81. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Transactions on Evolutionary computation **3**(2), 82–102 (1999)
82. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-international conference on neural networks, vol. 4, pp. 1942–1948. IEEE (1995)
83. Chu, Y., Mi, H., Liao, H., Ji, Z., Wu, Q.: A fast bacterial swarming algorithm for high-dimensional function optimization. In: 2008 IEEE congress on evolutionary computation (ieee world congress on computational intelligence), pp. 3135–3140. IEEE (2008)
84. Xiangtao Li and Jinyan Wang and Junping Zhou and Minghao Yin: A perturb biogeography based optimization with mutation for global numerical optimization. Applied Mathematics and Computation **218**(2), 598–609 (2011)
85. Gao, Wei-Feng and Liu, San-Yang and Huang, Ling-Ling: A novel artificial bee colony algorithm with Powell's method. Applied Soft Computing **13**(9), 3763–3775 (2013)
86. Finck, Steffen and Hansen, Nikolaus and Ros, Raymond and Auger, Anne: Real-Parameter Black-Box Optimization Benchmarking 2009: Presentation of the Noiseless Functions. Tech. rep., INRIA (2010)
87. Pošík, Petr and Baudiš, Petr: Dimension selection in axis-parallel Brent-step method for black-box optimization of separable continuous functions. In: Companion of the Genetic and Evolutionary Computation Conference, pp. 1151–1158 (2015)
88. Atamna, Asma: Benchmarking IPOP-CMA-ES-TPA and IPOP-CMA-ES-MSR on the BBOB noiseless testbed. In: Companion of the Genetic and Evolutionary Computation Conference, pp. 1135–1142 (2015)
89. Bajer, Lukáš and Pitra, Zbyněk and Holeňa, Martin: Benchmarking Gaussian processes and random forests surrogate models on the BBOB noiseless testbed. In: Companion of the Genetic and Evolutionary Computation Conference, pp. 1143–1150 (2015)
90. Brockhoff, Dimo and Bischl, Bernd and Wagner, Tobias: The impact of initial designs on the performance of MATSuMoTo on the noiseless BBOB-2015 testbed: A preliminary study. In: Companion of the Genetic and Evolutionary Computation Conference, pp. 1159–1166

(2015)
91. N. Hansen and A. Auger and S. Finck and R. Ros.: Real-Parameter Black-Box Optimization Benchmarking 2012: Experimental Setup. Tech. rep., INRIA (2012)
92. Kenneth Price: Differential evolution vs. the functions of the second ICEO. In: IEEE Congress on Evolutionary Computation, pp. 153–157 (1997)
93. Hansen, N. and Auger, A. and Finck, S. and Ros, R.: Real-Parameter Black-Box Optimization Benchmarking 2010: Experimental Setup. Tech. rep., INRIA (2010)

A Extended results for experiment one

Single-sided paired Welch test ($\alpha = 0.05$) was employed to perform pairwise comparisons on the mean fitness, provided that the distribution of the fitness of each pair of compared algorithm configurations presented unequal variances. The symbol "*" indicates that the mean fitness obtained with the focal mutation strategy is statistically better than that obtained with *rand/1* (the baseline). Paired test was performed because all methods were run using the same 30 distinct initial populations (one for each of the 30 independent runs).

Table 11 Experiment one: Fitness (mean and standard deviation across 30 independent runs), using Configuration 1, varying both the mutation strategy (*rand/1* and *st1* to *st8*) and the number of clusters (k = 2, 5, 10). Results for *rand/1* are repeated because it does not use clustering.

Fun	k	*rand/1*	*st1*	*st2*	*st3*	*st4*	*st5*	*st6*	*st7*	*st8*
f_1	2	1.833e-19(1.064e-19)	6.807e-02(8.972e-02)	5.804e-21(3e-21)*	3.809e-08(2.118e-07)	4.153e-41(3.756e-41)*	1.225e-35(7.072e-36)*	1.45e-01(1.353e-01)	1.23e+00(1.834e+00)	2.82e+02(1.021e+02)
	5	1.833e-19(1.064e-19)	5.068e-04(1.297e-03)	1.008e-21(5.348e-22)*	4.785e-09(1.102e-08)	1.368e-38(9.579e-39)*	2.466e-33(1.575e-33)*	5.368e-02(6.245e-02)	5.585e-04(1.056e-03)	1.567e+02(5.452e+01)
	10	1.833e-19(1.064e-19)	1.048e-10(7.379e-10)	3.074e-23(1.788e-23)*	1.955e-09(1.216e-08)	6.325e-36(4.768e-36)*	2.614e-31(1.805e-31)*	1.38e-02(3.239e-02)	6.268e-04(2.916e-03)	1.147e+02(4.749e+01)
f_2	2	5.943e-16(1.715e-16)	2.7e-02(2.404e-02)	9.333e-17(2.81e-17)*	2.517e-09(5.532e-09)	2.124e-34(9.383e-35)*	2.882e-28(1.413e-28)*	9.178e-03(6.669e-03)	6.265e-02(5.915e-02)	4.823e+00(7.402e-01)
	5	5.943e-16(1.715e-16)	6.624e-07(1.308e-06)	1.93e-17(5.71e-18)*	1.504e-09(3.921e-09)	7.489e-32(4.349e-32)*	1.238e-26(4.901e-27)*	2.295e-04(1.988e-04)	5.52e-05(1.398e-04)	2.428e+00(5.029e-01)
	10	5.943e-16(1.715e-16)	1.677e-19(1.186e-18)*	1.079e-18(2.826e-19)*	2.42e-10(5.902e-10)	2.13e-29(1.157e-29)*	4.094e-25(1.409e-25)*	2.398e-05(2.816e-05)	2.264e-05(4.246e-05)	1.433e+00(3.037e-01)
f_3	2	1.877e+03(1.099e+03)	1.525e+00(1.167e+00)*	1.186e+03(1.282e+03)*	3.094e-10(9.051e-10)*	9.925e-01(1.472e+00)*	1.961e+00(1.012e+00)*	3.88e-01(2.845e-01)*	2.979e+00(7.989e+00)*	5.483e+01(2.669e+01)*
	5	1.877e+03(1.099e+03)	1.785e+00(2.252e+00)*	1.219e+03(1.031e+03)*	1.836e-10(1.157e-09)*	3.526e+00(3.183e+00)*	5.718e+00(4.074e+00)*	1.859e-01(1.919e-01)*	2.708e-02(2.617e-02)*	1.773e+01(5.64e+00)*
	10	1.877e+03(1.099e+03)	2.546e+00(3.146e+00)*	9.731e+02(6.981e+02)*	1.161e-12(4.975e-12)*	7.107e+00(6.28e+00)*	1.72e+01(1.117e+01)*	9.756e-02(1.476e-01)*	2.812e-02(4.953e-02)*	1.11e+01(4.356e+00)*
f_4	2	1.159e-10(3.614e-11)	1.165e-01(5.502e-02)	2.887e-12(1.07e-12)*	3.375e+00(1.403e+00)	3.325e-06(2.187e-05)	6.084e-19(4.277e-18)*	1.631e-02(8.147e-03)	5.71e-02(2.31e-02)	3.086e+00(4.549e-01)
	5	1.159e-10(3.614e-11)	1.872e-02(1.114e-02)	3.972e-12(1.304e-12)*	4.126e+00(1.532e+00)	9.797e-07(5.844e-06)	6.303e-20(9.186e-20)*	6.678e-03(3.979e-03)	8.638e-03(3.773e-03)	2e+00(3.531e-01)
	10	1.159e-10(3.614e-11)	1.606e-03(1.174e-03)	1.421e-12(4.237e-13)*	5.54e+00(2.64e+00)	9.963e-08(4.44e-07)	8.11e-18(3.837e-17)*	8.401e-03(6.12e-03)	1.416e-03(1.66e-03)	1.539e+00(2.753e-01)
f_5	2	3.86e-02(5.201e-02)	2.15e+01(9.625e-01)	2.853e-30(1.269e-29)*	7.193e-01(1.546e+00)	2.288e+01(1.335e+01)	4.093e-02(2.894e-01)	2.318e+01(1.841e+00)	2.722e+01(7.018e+00)	1.271e+02(1.146e+02)
	5	3.86e-02(5.201e-02)	1.797e+01(9.376e-01)	1.118e-28(3.33e-28)*	1.116e+00(1.808e+00)	2.603e+01(1.778e+01)	5.997e-30(1.183e-29)*	2.698e+01(1.687e+01)	2.369e+01(2.169e+00)	1.205e+02(2.292e+02)
	10	3.86e-02(5.201e-02)	1.876e+01(1.082e+00)	3.304e-04(1.267e-03)*	9.727e-01(1.715e+00)	1.648e+01(6.243e+00)	4.162e-22(2.938e-21)*	2.031e+01(1.772e+01)	2.317e+01(1.149e+01)	9.325e+01(8.696e+01)
f_6	2	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	4e-02(2.828e-01)	0e+00(0e+00)	2.494e+02(8.066e+01)
	5	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	3.008e+02(1.302e+02)
	10	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	2.047e+02(1.137e+02)
f_7	2	6.378e-03(2.555e-03)	1.272e-03(7.272e-04)*	5.795e-03(2.799e-03)	3.583e-03(2.281e-03)*	3.589e-03(1.536e-03)*	3.192e-03(1.224e-03)*	3.669e-03(3.138e-03)*	1.086e-03(4.426e-04)*	7.814e-03(5.012e-03)
	5	6.378e-03(2.555e-03)	7.499e-04(2.708e-04)*	5.692e-03(2.787e-03)	3.487e-03(2.493e-03)*	3.979e-03(1.466e-03)*	3.226e-03(1.535e-03)*	8.57e-03(9.459e-03)	1.348e-03(4.826e-04)*	1.702e-02(1.437e-02)
	10	6.378e-03(2.555e-03)	6.148e-04(2.539e-04)*	5.796e-03(2.231e-03)	2.578e-03(1.408e-03)*	4.443e-03(1.784e-03)*	3.879e-03(1.923e-03)*	1.102e-02(8.586e-03)	1.523e-03(6.907e-04)*	2.709e-02(2.298e-02)
f_8	2	-1.257e+04(6.365e-13)	-1.088e+04(4e+02)	-7.903e+03(2.941e+02)	-1.042e+04(5.913e+02)	-9.135e+03(1.714e+03)	-8.253e+03(1.09e+03)	-9.664e+03(7.724e+02)	-7.299e+03(4.936e+02)	-3.968e+03(3.23e+02)
	5	-1.257e+04(6.365e-13)	-1.215e+04(2.389e+02)	-8.113e+03(2.867e+02)	-1.056e+04(6.044e+02)	-1.248e+04(5.964e+02)	-1.212e+04(1.187e+03)	-1.076e+04(3.798e+02)	-8.188e+03(1.214e+03)	-5.921e+03(9.329e+02)
	10	-1.257e+04(6.365e-13)	-1.252e+04(6.862e+01)	-9.505e+03(9.819e+02)	-1.068e+04(5.46e+02)	-1.257e+04(7.35e-13)	-1.257e+04(1.299e-12)*	-1.111e+04(3.374e+02)	-1.005e+04(1.583e+03)	-9.014e+03(1.073e+03)
f_9	2	2.878e-01(5.293e-01)	3.924e-01(1.057e+00)	1.015e+02(9.516e+00)	1.027e+01(3.041e+00)	8.161e+01(1.432e+01)	7.735e+01(1.09e+01)	6.357e+00(4.801e+00)	9.327e+00(1.055e+01)	7.952e+01(2.386e+01)
	5	2.878e-01(5.293e-01)	1.745e-05(7.217e-05)*	1.026e+02(7.553e+00)	1.047e+01(2.504e+00)	8.296e+01(9.533e+00)	8.671e+01(7.979e+00)	6.041e+00(2.196e+00)	1.429e+01(2.151e+01)	2.124e+01(1.09e+01)
	10	2.878e-01(5.293e-01)	1.873e+00(7.044e+00)	9.35e+01(8.202e+00)	1.133e+01(3.023e+00)	8.034e+01(9.255e+00)	9.392e+01(6.727e+00)	1.071e+01(1.903e+00)	1.286e+01(2.199e+01)	1.169e+01(4.367e+00)
f_{10}	2	1.018e-10(2.473e-11)	3.322e-02(1.819e-02)	1.858e-11(5.068e-12)*	9.902e-01(7.991e-01)	7.407e-15(7.033e-16)*	5.276e-15(1.723e-15)*	5.386e-02(3.111e-02)	2.012e-01(3.353e-01)	6.427e+00(1.03e+00)
	5	1.018e-10(2.473e-11)	1.4e-03(1.277e-03)	8.06e-12(1.907e-12)*	6.999e-01(7.497e-01)	7.336e-15(8.523e-16)*	5.347e-15(1.742e-15)*	9.777e-02(2.191e-01)	2.025e-03(2.113e-03)	7.011e+00(1.407e+00)
	10	1.018e-10(2.473e-11)	6.327e-07(2.655e-06)	1.589e-12(3.954e-13)*	3.345e-01(5.461e-01)	7.194e-15(1.077e-15)*	5.986e-15(1.781e-15)*	3.443e-01(4.991e-01)	3.028e-03(8.172e-03)	6.425e+00(1.326e+00)
f_{11}	2	0e+00(0e+00)	3.874e-02(1.051e-01)	0e+00(0e+00)	2.876e-02(3.978e-02)	1.523e-06(1.077e-05)	0e+00(0e+00)	3.3e-02(2.625e-02)	1.499e-01(2.4e-01)	2.826e+00(6.051e-01)
	5	0e+00(0e+00)	7.223e-05(1.262e-04)	0e+00(0e+00)	3.839e-02(4.421e-02)	0e+00(0e+00)	0e+00(0e+00)	1.988e-02(1.734e-02)	8.74e-03(1.401e-02)	1.894e+00(3.305e-01)
	10	0e+00(0e+00)	1.878e-07(9.222e-07)	0e+00(0e+00)	4.561e-02(4.435e-02)	3.451e-04(1.726e-03)	0e+00(0e+00)	1.752e-02(1.826e-02)	1.131e-02(1.828e-02)	1.494e+00(2.008e-01)
f_{12}	2	4.981e-20(2.87e-20)	8.302e-04(1.164e-03)	1.081e-21(6.29e-22)*	2.391e-01(6.834e-01)	1.125e-22(7.957e-22)*	1.571e-32(0e+00)*	1.041e-03(1.194e-03)	5.178e-03(2.136e-02)	4.104e+00(1.956e+00)
	5	4.981e-20(2.87e-20)	6.553e-06(1.899e-05)	2.842e-22(1.921e-22)*	1.741e-01(3.728e-01)	1.571e-32(0e+00)*	1.571e-32(0e+00)*	8.176e-03(2.263e-02)	4.488e-06(1.068e-05)	4.635e+00(3.006e+00)
	10	4.981e-20(2.87e-20)	4.779e-09(3.365e-08)	9.661e-24(4.946e-24)*	1.475e-01(4.389e-01)	1.571e-32(0e+00)*	4.348e-32(2.895e-32)*	5.965e-02(1.565e-01)	1.574e-05(9.5e-05)	3.718e+00(2.023e+00)
f_{13}	2	7.214e-04(7.579e-04)	9.48e-02(1.609e-01)	1.742e-04(4.361e-04)*	3.001e+00(4.53e+00)	1.365e-03(3.764e-04)	7.684e-04(7.465e-04)	4.1e-01(5.46e-01)	2.669e-02(9.057e-02)	3.473e+01(8.993e+01)
	5	7.214e-04(7.579e-04)	5.941e-03(9.667e-03)	2.544e-04(4.946e-04)*	8.799e-01(1.096e+00)	1.385e-03(3.962e-04)	9.555e-04(6.272e-04)	9.998e-01(1.155e+00)	2.888e-02(1.387e-01)	1.308e+02(4.79e+02)
	10	7.214e-04(7.579e-04)	1.085e-03(9.413e-04)	6.125e-04(6.201e-04)	6.337e-01(8.131e-01)	1.648e-03(1.938e-03)	9.478e-04(5.948e-04)	1.081e+00(1.294e+00)	7.149e-02(1.518e-01)	1.194e+02(5.526e+02)
Mean Rank		3.436	4.205	3.821	5.282	3.333	**2.667**	5.923	5.436	8.513

Table 12 Experiment one: Fitness (mean and standard deviation across 30 independent runs), using Configuration 2, varying both the mutation strategy (*rand/1* and *st1* to *st8*) and the number of clusters (k = 2, 5, 10). Results for *rand/1* are repeated because it does not use clustering.

Fun	k	*rand/1*	*st1*	*st2*	*st3*	*st4*	*st5*	*st6*	*st7*	*st8*
f_1	2	1.505e-22(1.184e-22)	5.846e-01(5.49e-01)	1.064e-27(1.342e-27)*	2.592e+01(3.408e+01)	1.556e-02(3.813e-02)	1.306e-40(5.575e-40)*	3.254e-01(3.43e-01)	9.618e-01(5.224e-01)	2.201e+02(8.291e+01)
	5	1.505e-22(1.184e-22)	1.888e-02(1.49e-02)	1.073e-26(1.194e-26)*	5.783e+01(9.276e+01)	4.293e-01(1.407e+00)	2.817e-37(8.551e-37)*	6.5e+00(7.267e+00)	6.19e-01(3.986e-01)	1.14e+02(3.869e+01)
	10	1.505e-22(1.184e-22)	1.046e-04(2.174e-04)	4.069e-27(2.859e-27)*	7.391e+01(9.354e+01)	1.543e+00(1.007e+01)	4.194e-34(1.06e-33)*	1.018e+03(1.242e+03)	4.569e-01(4.367e-01)	7.676e+01(2.859e+01)
f_2	2	3.202e-16(1.535e-16)	1.119e+00(9.277e-01)	1.611e-19(8.854e-20)*	1.061e+00(8.922e-01)	1.004e-02(3.244e-02)	2.834e-33(2.419e-33)*	9.383e-02(4.747e-02)	1.232e-01(5.816e-02)	4.149e+00(9.068e-01)
	5	3.202e-16(1.535e-16)	4.447e-02(3.649e-02)	5.286e-19(3.388e-19)*	1.83e+00(1.502e+00)	2.397e-03(1.171e-02)	2.887e-30(2.815e-30)*	7.48e-01(3.911e-01)	1.045e-01(6.564e-02)	2.766e+00(4.912e-01)
	10	3.202e-16(1.535e-16)	6.68e-03(3.356e-02)	9.515e-20(4.303e-20)*	1.854e+00(1.429e+00)	2.904e-15(2.054e-14)	7.886e-28(4.861e-28)*	8.861e+00(5.811e+00)	7.641e-02(6.64e-02)	2.06e+00(4.534e-01)
f_3	2	3.831e-15(7.488e-15)	5.792e-01(4.085e-01)	3.473e-22(5.74e-22)*	7.8e-01(1.612e+00)	3.053e+02(3.863e+02)	1.226e-23(4.108e-23)*	1.38e-01(8.971e-02)	3.73e-02(2.416e-02)	3.913e+01(1.529e+01)
	5	3.831e-15(7.488e-15)	1.477e-02(1.423e-02)	2.661e-21(3.095e-21)*	7.04e-01(1.301e+00)	8.894e+01(1.411e+02)	2.683e-24(6.411e-24)*	4.886e-02(5.481e-02)	2.916e-03(2.732e-03)	1.133e+01(3.795e+00)
	10	3.831e-15(7.488e-15)	2.736e-03(5.393e-03)	9.205e-23(2.205e-22)*	5.019e-01(1.323e+00)	1.136e+01(3.894e+01)	1.497e-23(3.79e-23)*	2.282e+01(4.379e+01)	3.633e-04(5.048e-04)	5.293e+00(2.507e+00)
f_4	2	5.601e-02(6.121e-02)	9.845e-02(4.445e-02)	3.764e-19(6.234e-19)*	1.477e+00(1.032e+00)	1.436e+00(5.906e-01)	2.037e-01(1.785e-01)	1.485e-02(8.021e-03)*	4.113e-02(2.024e-02)*	2.227e+00(3.921e-01)
	5	5.601e-02(6.121e-02)	2.481e-02(1.104e-02)*	2.241e-17(6.037e-17)*	3.32e+00(1.778e+00)	1.522e+00(7.521e-01)	6.986e-01(5.089e-01)	8.63e-02(8.548e-02)	1.202e-02(8.452e-03)*	2.033e+00(4.164e-01)
	10	5.601e-02(6.121e-02)	8.845e-03(4.925e-03)*	1.505e-16(2.598e-16)*	4.855e+00(2.52e+00)	7.776e-01(3.63e-01)	4.247e+00(2.416e+00)	5.776e-01(3.586e-01)	7.041e-03(4.783e-03)*	1.653e+00(3.222e-01)
f_5	2	1.595e-01(7.891e-01)	2.488e+01(1.215e+00)	2.76e-24(1.951e-23)	3.139e+01(1.909e+01)	3.76e+01(3.137e+01)	6.618e+00(4.411e+00)	2.974e+01(1.772e+01)	2.63e+01(1.714e+00)	4.332e+01(1.521e+01)
	5	1.595e-01(7.891e-01)	1.39e+01(3.005e+00)	4.442e-01(1.229e+00)	3.49e+01(2.157e+01)	3.18e+01(2.967e+01)	1.116e+00(1.808e+00)	2.826e+01(1.724e+01)	2.707e+01(1.061e+01)	2.868e+01(4.624e+00)
	10	1.595e-01(7.891e-01)	2.47e-01(5.104e-01)	4.956e-01(1.252e+00)	2.897e+01(1.54e+01)	2.225e+01(2.623e+01)	7.973e-01(1.611e+00)	8.003e+01(6.109e+01)	2.889e+01(1.81e+01)	2.888e+01(1.175e+01)
f_6	2	0e+00(0e+00)	6e-02(3.136e-01)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	1.26e+00(6.21e+00)	8.2e-01(1.289e+00)	3.613e+02(1.18e+02)
	5	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	4e-02(1.979e-01)	8.6e-01(1.278e+00)	4.334e+02(1.33e+02)
	10	0e+00(0e+00)	0e+00(0e+00)	0e+00(0e+00)	2e-02(1.414e-01)	0e+00(0e+00)	0e+00(0e+00)	1e-01(3.03e-01)	7.6e-01(1.098e+00)	3.909e+02(1.678e+02)
f_7	2	3.296e-03(1.526e-03)	3.074e-03(2.936e-03)	2.517e-03(1.062e-03)*	8.475e-02(1.257e-01)	1.699e-02(2.809e-02)	2.043e-03(8.529e-04)*	2.378e-02(2.836e-02)	1.891e-03(2.046e-03)*	5.491e-03(2.961e-03)
	5	3.296e-03(1.526e-03)	2.653e-03(3.023e-03)	3.12e-03(1.23e-03)	5.744e-02(5.87e-02)	8.801e-03(7.961e-03)	2.394e-03(8.373e-04)*	1.539e-01(1.205e-01)	5.574e-03(5.911e-03)	1.239e-02(8.854e-03)
	10	3.296e-03(1.526e-03)	1.472e-03(1.09e-03)*	2.876e-03(1.133e-03)	6.869e-02(7.913e-02)	8.192e-03(5.898e-03)	3.271e-03(1.998e-03)	2.208e-01(1.621e-01)	1.461e-02(1.159e-02)	4.884e-02(5.971e-02)
f_8	2	-1.227e+04(2.009e+02)	-6.208e+03(1.415e+03)	-5.482e+03(2.128e+02)	-7.857e+03(5.418e+02)	-8.136e+03(2.268e+03)	-5.357e+03(2.275e+02)	-6.736e+03(7.671e+02)	-5.616e+03(1.181e+03)	-3.972e+03(6.432e+02)
	5	-1.227e+04(2.009e+02)	-7.167e+03(2.041e+03)	-5.676e+03(4.238e+02)	-7.737e+03(7.002e+02)	-9.754e+03(2.399e+03)	-5.601e+03(2.571e+02)	-7.343e+03(5.016e+02)	-7.776e+03(1.566e+03)	-5.707e+03(7.001e+02)
	10	-1.227e+04(2.009e+02)	-9.341e+03(2.28e+03)	-1.198e+04(9.921e+02)	-7.896e+03(5.811e+02)	-1.163e+04(1.239e+03)	-7.659e+03(2.446e+03)	-7.287e+03(5.088e+02)	-8.827e+03(8.539e+02)	-6.307e+03(5.595e+02)
f_9	2	6.041e+00(2.405e+00)	5.399e-01(1.556e+00)*	1.525e+02(7.382e+00)	4.243e+01(1.092e+01)	1.732e+01(5.567e+00)	1.11e+01(1.553e+01)	1.439e+01(6.148e+00)	4.405e+00(3.661e+00)*	1.124e+02(2.172e+01)
	5	6.041e+00(2.405e+00)	8.305e-05(1.301e-04)*	1.501e+02(9.195e+00)	4.601e+01(1.105e+01)	1.247e+01(4.301e+00)	1.707e+01(2.204e+01)	2.751e+01(7.165e+00)	5.442e+00(2.282e+00)	9.268e+01(1.773e+01)
	10	6.041e+00(2.405e+00)	4.242e-01(1.201e+00)*	2.136e+01(2.489e+01)	4.515e+01(7.736e+00)	1.219e+01(4.583e+00)	1.133e+01(4.059e+00)	3.769e+01(9.564e+00)	8.218e+00(2.488e+00)	8.186e+01(1.75e+01)
f_{10}	2	2.967e-12(1.193e-12)	2.015e-01(1.257e-01)	8.615e-15(2.407e-15)*	5.464e+00(1.659e+00)	1.139e+00(8.195e-01)	9.113e-15(3.302e-15)*	1.504e+00(7.4e-01)	4.976e-01(3.801e-01)	6.051e+00(1.075e+00)
	5	2.967e-12(1.193e-12)	1.913e-02(9.459e-03)	2.659e-14(1.224e-14)*	6.934e+00(1.612e+00)	7.985e-01(5.572e-01)	1.163e-13(7.518e-13)*	2.931e+00(4.361e-01)	1.616e+00(7.01e-01)	6.836e+00(1.438e+00)
	10	2.967e-12(1.193e-12)	1.304e-03(1.143e-03)	1.814e-14(9.036e-15)*	6.661e+00(1.783e+00)	6.487e-01(6.442e-01)	7.434e-14(2.127e-13)*	5.243e+00(1.521e+00)	2.123e+00(6.65e-01)	8.381e+00(1.99e+00)
f_{11}	2	1.971e-04(1.394e-03)	1.295e-01(1.794e-01)	0e+00(0e+00)	5.961e-01(3.188e-01)	6.976e-02(9.737e-02)	7.911e-03(1.424e-02)	6.874e-02(4.825e-02)	1.957e-01(1.082e-01)	2.16e+00(4.362e-01)
	5	1.971e-04(1.394e-03)	7.509e-03(1.062e-02)	0e+00(0e+00)	8.86e-01(3.982e-01)	5.605e-02(3.731e-02)	5.171e-03(7.43e-03)	4.424e-01(2.585e-01)	1.096e-01(6.367e-02)	1.522e+00(1.788e-01)
	10	1.971e-04(1.394e-03)	2.88e-03(6.588e-03)	7.886e-04(2.791e-03)	1.003e+00(6.417e-01)	4.884e-02(9.119e-02)	7.276e-03(1.134e-02)	6.683e+00(8.259e+00)	5.294e-02(4.143e-02)	1.337e+00(1.713e-01)
f_{12}	2	5.745e-24(6.553e-24)	4.938e-03(8.662e-03)	1.721e-29(1.366e-29)*	1.045e+01(5.303e+00)	8.049e+00(2.475e+01)	3.365e-04(2.38e-03)	5.077e-02(8.787e-02)	4.004e-03(4.789e-03)	3.97e+00(1.342e+00)
	5	5.745e-24(6.553e-24)	9.762e-05(1.157e-04)	2.603e-28(2.518e-28)*	1.35e+01(7.329e+00)	9.28e+00(4.424e+01)	2.095e-14(1.282e-13)	3.139e+00(1.652e+00)	9.366e-03(1.433e-02)	5.548e+00(2.924e+00)
	10	5.745e-24(6.553e-24)	1.218e-06(1.898e-06)	1.416e-28(1.391e-28)*	1.48e+01(7.325e+00)	5.434e+01(3.776e+02)	2.073e-03(1.466e-02)	1.154e+01(7.083e+00)	1.142e-01(1.938e-01)	7.506e+00(5.868e+00)
f_{13}	2	2.976e-04(6.406e-04)	4.049e-01(5.775e-01)	2.231e-04(4.81e-04)	1.338e+01(2.512e+01)	7.808e+03(3.112e+04)	4.112e-02(1.339e-01)	1.218e+00(1.127e+00)	2.458e-01(4.264e-01)	1.255e+01(1.327e+01)
	5	2.976e-04(6.406e-04)	1.849e-01(3.175e-01)	6.31e-04(7.335e-04)	1.028e+02(6.432e+02)	5.86e+02(1.67e+03)	6.843e-02(1.713e-01)	1.34e+00(9.949e-01)	9.742e-01(1.201e+00)	4.025e+01(8.505e+01)
	10	2.976e-04(6.406e-04)	1.303e-01(3.426e-01)	1.341e-03(3.458e-03)	5.848e+01(1.902e+02)	2.055e+03(7.643e+03)	2.173e-01(4.941e-01)	5.125e+04(2.815e+05)	1.952e+00(2.71e+00)	1.089e+02(3.281e+02)
Mean Rank		**2.410**	3.872	**2.410**	7.154	5.615	3.103	6.641	4.923	7.974

Table 13 Experiment one: Fitness (mean and standard deviation across 30 independent runs), using Configuration 3, varying both the mutation strategy (*rand/1* and *st1* to *st8*) and the number of clusters (k = 2, 5, 10). Results for *rand/1* are repeated because it does not use clustering.

Fun	k	*rand/1*	*st1*	*st2*	*st3*	*st4*	*st5*	*st6*	*st7*	*st8*
f_1	2	1.743e+00(4.458e-01)	5.489e-20(4.768e-20)*	5.354e-01(1.252e-01)*	1.351e-52(2.592e-52)*	1.654e-19(2.915e-19)*	4.064e-17(2.856e-17)*	2.06e-21(2.062e-21)*	5.616e-21(6.331e-21)*	2.797e+02(1.182e+02)
	5	1.743e+00(4.458e-01)	4.11e-14(2.065e-14)*	1.781e-01(5.612e-02)*	8.315e-41(1.058e-40)*	3.229e-16(1.859e-16)*	1.15e-13(7.192e-14)*	7.522e-17(4.202e-17)*	1.975e-17(1.497e-17)*	1.049e+02(4.245e+01)
	10	1.743e+00(4.458e-01)	2.092e-09(8.239e-10)*	1.424e-02(4.223e-03)*	6.298e-31(7.717e-31)*	2.874e-13(1.923e-13)*	2.393e-11(1.825e-11)*	1.086e-13(1.064e-13)*	8.88e-15(5.681e-15)*	6.168e+00(4.132e+00)
f_2	2	2.961e-02(4.631e-03)	2.529e-16(1.399e-16)*	2.725e-02(5.295e-03)*	2.22e-40(5.063e-40)*	9.447e-18(5.457e-18)*	1.226e-14(6.147e-15)*	1.576e-17(8.309e-18)*	3.249e-17(1.803e-17)*	1.995e+00(5.498e-01)
	5	2.961e-02(4.631e-03)	1.066e-11(4.249e-12)*	1.477e-02(2.132e-03)*	1.944e-31(2.384e-31)*	1.059e-14(5.501e-15)*	5.354e-12(2.387e-12)*	5.022e-14(2.193e-14)*	2.444e-14(1.053e-14)*	7.867e-01(4.055e-01)
	10	2.961e-02(4.631e-03)	2.453e-08(7.463e-09)*	2.577e-03(4.298e-04)*	3.215e-24(2.93e-24)*	4.546e-12(1.86e-12)*	2.181e-10(8.261e-11)*	6.988e-12(2.965e-12)*	1.546e-12(6.06e-13)*	5.403e-02(6.217e-02)
f_3	2	1.768e+04(2.601e+03)	1.895e+00(1.049e+00)*	1.739e+04(2.838e+03)	6.284e-07(1.843e-06)*	1.435e+03(1.147e+03)*	9.323e-01(5.426e-01)*	3.202e-01(2.163e-01)*	6.022e-02(3.744e-02)*	5.218e+01(1.861e+01)*
	5	1.768e+04(2.601e+03)	6.321e+02(1.315e+03)*	1.695e+04(2.501e+03)	2.714e-03(2.956e-03)*	2.219e+03(1.156e+03)*	1.073e+01(6.086e+00)*	7.276e+00(3.863e+00)*	1.283e+00(8.673e-01)*	3.322e+01(1.562e+01)*
	10	1.768e+04(2.601e+03)	4.773e+03(2.652e+03)*	1.549e+04(2.665e+03)*	4.912e-01(7.439e-01)*	3.231e+03(1.504e+03)*	3.611e+01(2.17e+01)*	3.905e+01(3.217e+01)*	7.117e+00(5.444e+00)*	1.428e+01(7.289e+00)*
f_4	2	8.011e-01(1.148e-01)	9.957e-13(4.997e-13)*	3.796e-01(6.079e-02)*	2.135e-28(4.6e-28)*	4.966e-06(3.591e-06)*	9.903e-12(5.099e-12)*	3.352e-14(1.978e-14)*	6.402e-14(5.711e-14)*	5.883e+00(9.876e-01)
	5	8.011e-01(1.148e-01)	7.225e-09(2.659e-09)*	2.913e-01(4.356e-02)*	1.743e-22(2.623e-22)*	3.453e-05(2.525e-05)*	1.188e-09(7.715e-10)*	1.161e-11(5.78e-12)*	6.353e-12(4.378e-12)*	2.058e+00(5.566e-01)
	10	8.011e-01(1.148e-01)	6.824e-06(1.569e-06)*	1.258e-01(2.477e-02)*	2.399e-17(2.495e-17)*	1.398e-04(9.027e-05)*	3.885e-08(1.665e-08)*	1.281e-09(6.706e-10)*	2.271e-10(1.008e-10)*	1.427e+00(4.57e-01)
f_5	2	4.923e+00(3.532e-01)	1.807e-28(1.068e-27)*	3.872e-01(8.273e-02)*	1.595e-01(7.891e-01)*	1.527e-06(1.08e-05)*	4.667e-29(7.265e-29)*	3.354e-29(3.857e-29)*	2.234e-29(2.05e-29)*	7.99e+01(1.002e+02)
	5	4.923e+00(3.532e-01)	7.823e-23(1.104e-22)*	7.943e-01(1.546e-01)*	2.392e-01(9.564e-01)*	1.044e-27(7.061e-27)*	1.938e-25(7.587e-25)*	7.973e-02(5.638e-01)*	7.973e-02(5.638e-01)*	4.699e+01(3.213e+01)
	10	4.923e+00(3.532e-01)	6.15e-17(7.965e-17)*	2.322e-01(5.322e-02)*	3.987e-01(1.208e+00)*	7.973e-02(5.638e-01)*	7.973e-02(5.638e-01)*	1.595e-01(7.891e-01)*	1.034e-29(2.671e-29)*	5.811e+01(1.719e+02)
f_6	2	1.94e+00(9.564e-01)	0e+00(0e+00)*	2e-01(4.041e-01)*	0e+00(0e+00)*	0e+00(0e+00)*	0e+00(0e+00)*	0e+00(0e+00)*	0e+00(0e+00)*	2.408e+02(1.129e+02)
	5	1.94e+00(9.564e-01)	0e+00(0e+00)*	0e+00(0e+00)*	0e+00(0e+00)*	0e+00(0e+00)*	0e+00(0e+00)*	0e+00(0e+00)*	0e+00(0e+00)*	3.09e+01(2.542e+01)
	10	1.94e+00(9.564e-01)	0e+00(0e+00)*	0e+00(0e+00)*	0e+00(0e+00)*	0e+00(0e+00)*	0e+00(0e+00)*	0e+00(0e+00)*	0e+00(0e+00)*	2.2e-01(8.873e-01)*
f_7	2	4.879e-02(1.766e-02)	4.885e-03(2.027e-03)*	3.642e-02(1.296e-02)*	2.364e-03(7.21e-04)*	1.028e-02(3.599e-03)*	6.277e-03(3.394e-03)*	4.678e-03(2.54e-03)*	4.916e-03(2.565e-03)*	3.1e-02(2.448e-02)*
	5	4.879e-02(1.766e-02)	6.718e-03(2.871e-03)*	3.753e-02(1.192e-02)*	2.57e-03(9.3e-04)*	1.057e-02(4.311e-03)*	5.846e-03(2.882e-03)*	5.109e-03(2.404e-03)*	5.788e-03(2.329e-03)*	8.906e-02(1.341e-01)
	10	4.879e-02(1.766e-02)	9.568e-03(4.282e-03)*	2.92e-02(1.159e-02)*	3.356e-03(9.791e-04)*	1.282e-02(4.971e-03)*	7.77e-03(3.733e-03)*	6.043e-03(2.675e-03)*	6.453e-03(2.372e-03)*	7.375e-02(6.336e-02)
f_8	2	-1.257e+04(0e+00)	-7.392e+03(2.661e+02)	-7.574e+03(2.675e+02)	-1.229e+04(6.857e+02)	-7.622e+03(3.57e+02)	-7.61e+03(2.731e+02)	-7.436e+03(2.152e+02)	-7.453e+03(2.194e+02)	-4.765e+03(4.91e+02)
	5	-1.257e+04(0e+00)	-7.495e+03(2.518e+02)	-7.529e+03(2.433e+02)	-1.248e+04(1.203e+02)	-7.896e+03(3.306e+02)	-7.506e+03(2.796e+02)	-7.443e+03(2.835e+02)	-7.508e+03(2.317e+02)	-9.704e+03(6.243e+02)
	10	-1.257e+04(0e+00)	-8.003e+03(4.03e+02)	-8.027e+03(4.167e+02)	-1.253e+04(7.849e+01)	-1.118e+04(1.431e+03)	-7.774e+03(2.985e+02)	-7.855e+03(3.746e+02)	-7.564e+03(2.748e+02)	-1.157e+04(3.85e+02)
f_9	2	4.01e-01(6.968e-01)	9.956e+01(8.431e+00)	1.398e+02(9.179e+00)	1.786e+01(1.485e+01)	1.163e+02(1.07e+01)	9.918e+01(1.261e+01)	1.041e+02(1.016e+01)	1.048e+02(8.198e+00)	6.856e+01(1.915e+01)
	5	4.01e-01(6.968e-01)	1.116e+02(7.753e+00)	1.411e+02(1.024e+01)	3.865e+01(2.628e+01)	1.201e+02(7.647e+00)	1.126e+02(8.724e+00)	1.124e+02(9.053e+00)	1.104e+02(8.54e+00)	3.24e+01(1.573e+01)
	10	4.01e-01(6.968e-01)	1.172e+02(8.244e+00)	1.275e+02(1.204e+01)	5.574e+01(2.53e+01)	1.203e+02(8.097e+00)	1.22e+02(8.787e+00)	1.206e+02(9.262e+00)	1.165e+02(8.88e+00)	1.939e+01(9.491e+00)
f_{10}	2	6.01e-01(1.467e-01)	5.14e-11(2.086e-11)*	3.353e-01(7.058e-02)*	5.134e-15(1.674e-15)*	9.047e-11(3.77e-11)*	1.215e-09(5.475e-10)*	1.016e-11(5.572e-12)*	1.425e-11(6.41e-12)*	5.657e+00(1.443e+00)
	5	6.01e-01(1.467e-01)	5.226e-08(1.613e-08)*	1.644e-01(3.253e-02)*	4.778e-15(1.487e-15)*	5.247e-09(2.416e-09)*	6.721e-08(2.564e-08)*	2.195e-09(7.759e-10)*	1.06e-09(4.127e-10)*	3.595e+00(1.945e+00)
	10	6.01e-01(1.467e-01)	1.196e-05(3.01e-06)*	3.802e-02(5.878e-03)*	6.484e-15(1.645e-15)*	1.606e-07(6.21e-08)*	1.005e-06(3.115e-07)*	8.147e-08(2.749e-08)*	2.311e-08(7.017e-09)*	8.94e-01(8.631e-01)
f_{11}	2	3.678e-01(8.603e-02)	3.878e-06(2.742e-05)*	3.566e-01(1.291e-01)	4.369e-03(7.184e-03)*	1.504e-03(4.206e-03)*	3.489e-07(2.467e-06)*	1.577e-05(1.115e-04)*	2.346e-05(1.659e-04)*	2.871e+00(7.821e-01)
	5	3.678e-01(8.603e-02)	1.81e-15(1.221e-14)*	8.754e-02(9.671e-02)*	2.168e-03(4.788e-03)*	7.293e-11(5.073e-10)*	1.479e-04(1.046e-03)*	5.914e-04(2.416e-03)*	2.862e-06(2.023e-05)*	1.38e+00(1.961e-01)
	10	3.678e-01(8.603e-02)	7.343e-09(3.531e-08)*	1.351e-03(1.151e-03)*	2.169e-03(4.357e-03)*	1.479e-04(1.046e-03)*	1.479e-04(1.046e-03)*	9.367e-04(2.983e-03)*	3.206e-07(2.265e-06)*	7.165e-01(2.014e-01)
f_{12}	2	1.159e+00(3.509e-01)	2.899e-21(3.318e-21)*	7.053e-01(2.424e-01)*	1.571e-32(0e+00)*	3.375e-18(1.353e-17)*	2.077e-18(1.862e-18)*	1.184e-22(1.102e-22)*	1.707e-22(1.141e-22)*	4.347e+00(3.887e+00)
	5	1.159e+00(3.509e-01)	4.644e-15(2.453e-15)*	4.6e-01(1.813e-01)*	1.671e-32(7.119e-33)*	3.826e-15(3.983e-15)*	5.352e-15(3.94e-15)*	4.636e-18(3.217e-18)*	1.139e-18(1.064e-18)*	2.152e+01(1.205e+02)
	10	1.159e+00(3.509e-01)	5.42e-10(4.764e-10)*	6.918e-02(3.902e-02)*	2.063e-24(1.459e-23)*	3.414e-12(4.911e-12)*	8.95e-13(6.614e-13)*	6.629e-15(4.935e-15)*	5.454e-16(4.129e-16)*	4.789e+00(1.122e+01)
f_{13}	2	4.217e+00(1.046e+00)	1.553e-04(4.958e-04)*	2.751e+00(5.583e-01)*	3.181e-03(6.226e-03)*	8.408e-02(1.073e-01)*	2.479e-05(1.753e-04)*	1.073e-04(3.402e-04)*	4.947e-05(2.448e-04)*	1.551e+03(5.157e+03)
	5	4.217e+00(1.046e+00)	5.697e-04(7.994e-04)*	2.294e+00(4.873e-01)*	8.135e-03(2.95e-02)*	1.812e-01(2.383e-01)*	3.986e-04(5.842e-04)*	4.529e-04(6.499e-04)*	1.825e-04(4.349e-04)*	8.95e+04(1.96e+05)
	10	4.217e+00(1.046e+00)	3.697e-02(1.909e-02)*	1.49e+00(3.075e-01)*	2.857e-03(5.805e-03)*	2.495e-01(1.769e-01)*	7.284e-04(7.64e-04)*	1.75e-03(4.332e-03)*	5.871e-04(7.103e-04)*	1.453e+05(1.83e+05)
Mean Rank		7.051	4.436	6.744	**2.308**	4.538	4.231	3.538	2.846	7.769