A Study on Rotation Invariance in Differential Evolution

Fabio Caraffini[a], Ferrante Neri[a,*]

[a]*Institute of Artificial Intelligence, School of Computer Science and Informatics, De Montfort University, United Kingdom*

Abstract

Epistasis is the correlation between the variables of a function and is a challenge often posed by real-world optimisation problems. Synthetic benchmark problems simulate a highly epistatic problem by performing a so-called problem's rotation.

Mutation in Differential Evolution (DE) is inherently rotational invariant since it simultaneously perturbs all the variables. On the other hand, crossover, albeit fundamental for achieving a good performance, retains some of the variables, thus being inadequate to tackle highly epistatic problems.

This article proposes an extensive study on rotational invariant crossovers in DE. We propose an analysis of the literature, a taxonomy of the proposed method and an experimental setup where each problem is addressed in both its non-rotated and rotated version. Our experimental study includes 280 problems over five different levels of dimensionality and nine algorithms.

Numerical results show that 1) for a fixed quota of transferred design variables, the exponential crossover displays a better performance, on both rotated and non-rotated problems, in high dimensions while the binomial crossover seems to be preferable in low dimensions; 2) the rotational invariant mutation DE/current-to-rand is not competitive with standard DE implementations throughout the entire set of experiments we have presented; 3) DE crossovers that perform a change of coordinates to distribute the moves over the components of the offspring offer high-performance results on some problems. However, on average the standard DE/rand/1/exp appears to achieve the best performance on both rotated and non-rotated testbeds.

Keywords: Epistasis, Separability, Rotational Invariance, Differential Evolution

---

1. Introduction

Metaheuristics are high-level search procedures designed to find, generate, or select a search rule that may provide a good approximation of the solution of an optimisation problem [5]. The use of metaheuristics is especially popular for addressing real-world optimisation

*F. Neri is the corresponding author

Email addresses: fabio.caraffini@dmu.ac.uk (Fabio Caraffini), fneri@dmu.ac.uk (Ferrante Neri)

problems which often offer incomplete or imperfect information or limited computation capacity.

By applying an operational logic, when a problem possesses the hypotheses for applying an exact method, unless the execution time prohibits it, the exact method is often chosen since it provides the theoretical solution of the problem, see [42]. When the lack of information about the problem or the lack of hypotheses do not allow the application of an exact method, the application of a metaheuristic is the only way to detect a reasonably good sub-optimal solution.

Since metaheuristics are not theoretically guaranteed to work, rigorous experimentalism and data analysis play a crucial role when the performance is measured and compared to other existing metaheuristics, see [16, 17]. Since the capability of a metaheuristic to reach a good solution is not guaranteed multiple runs are performed and the statistical significance of the results is calculated.

In order to perform fair and directly comparable experiments, researchers in computer science proposed benchmarks of test problems. The latter is a set of artificially built mathematical functions that can be used for testing a newly proposed metaheuristic against existing ones. One precursor of the benchmark for optimisation is the list of objective functions proposed in [4]. Subsequently, this list of functions has been elaborated and the first popular benchmark of optimisation problems has been defined in [47]. The main idea was to have a set of problems where each function presented a different type of challenge, e.g. some functions are unimodal while others multimodal. In the subsequent years, this logic has been taken further enlarging the pool of functions and attempting to make the problems more challenging. Some examples of benchmarks for numerical optimisation are in [19] and [33]. The latter benchmark underwent progressive revisions in [32], [11] and [3].

Benchmarks setting specific challenges have also been proposed. For example, a set of multimodal problems is proposed in [40], scalable/large-scale problems in [34], and constrained optimisation problems [35].

One challenge that often appears in benchmarks is the rotation of the problems. Rotation is here referred to an algebraic linear transformation occurring on the candidate solution while performing a function call (objective function evaluation). Optimisation problems characterised by a central symmetry of the fitness landscape or problems whose fitness value can be easily improved by perturbing single variables (this is the case for functions composed as sums, or product of functions applied to each variable separately) can usually be tackled/nearly solved by a naive or relatively simple metaheuristic. The rotation makes the symmetries more difficult to exploit and imposes that the direction of the gradient is searched by simultaneously taking into account multiple variables, see [23]. Equivalently, we may state that rotation increases the interdependence among variables, a.k.a. *epistasis*, see [26]. Thus, the problem's rotation has been introduced with the intention of making the promising gradient directions hard to find and the overall problem harder to solve.

The rotation can also be seen as a transformation that makes artificial problems closer to real-world problems. From the perspective of the algorithm, rotation jeopardises the possible prior knowledge of the problem, thus making the problem unpredictable and similar to a black-box coming from a simulator or an actual experiment, see [8, 10, 43] where the fitness can be noisy or time-variant.

While attempting to design robust algorithms, computer scientists aimed at proposing metaheuristics whose performance does not deteriorate after the rotation. These metaheuristics are known as *rotational invariant*. Two classical examples of inherently rotational invariant algorithms are the Rosenbrock Algorithm [41] which changes the coordinates by taking as a reference the direction of the gradient and the Covariance Matrix Adaptive Evolution Strategy (CMAES), see e.g. [20, 21, 22, 23], which roughly estimates the Hessian matrix and adapts accordingly a multivariate probability distribution from which candidate solutions are sampled.

Unlike these metaheuristics, Differential Evolution (DE) [44], is not inherently rotation invariant. However, DE is a very versatile framework which is easy to improve by modifying some components, embedding an adaptive scheme or adding some further search moves (by altering the DE moving operators), see [6, 12, 13, 37, 39, 46, 1]. It is worthwhile mentioning the DE variant proposed in [29] which makes use of a crossover matrix to generate new candidate solutions.

By following this logic, several attempts have been made in the literature to make DE rotational invariant. As explained in [2] DE mutation is inherently rotational invariant since it consists of adding to a vector a weighted difference that is a move that may involve all the variables at the same time. In a black-box problem, a random movement over all the variables has an average the same result regardless of the problem's rotation. On the contrary, the crossover is an operator that generates a solution by combining some variables coming from the mutant and some variables coming from the parent. The offspring solution will thus retain some of the variables of the parent while some others will be perturbed. If this strategy is successful for a problem, the same strategy will not lead to the same improvements after the problem's rotation. On the other hand, the crossover is essential for the DE functioning since no crossover would correspond to an highly exploratory logic since the offspring would have no common genes with respect to their parents.

In order to overcome this limitation, several types of modifications have been proposed to alter DE frameworks. Some of them are here listed.

- **Linear combination crossover:** the crossover is integrated into the mutation so that all the variables are simultaneously taken involved, see [38]. This idea has been further developed in [7] where an adaptive DE with multiple mutation strategies including the one in [38] has been proposed.

- **Coordinate transformation by orthogonalisation of a basis:** the direction of the gradient is at first estimated and then used with the candidate solutions of the population to apply the Gram-Schmidt orthogonalisation to change the coordinate system [48]. The same orthogonalisation for performing a coordinate change has been also implemented in [2] alongside to an ensemble of mutation and crossover strategies within a DE framework named Mutation and Mixed Crossover Strategy based DE (MMCDE).

- **Coordinate transformation by diagonalisation of the covariance matrix:** the population of DE candidate solutions is used to build a covariance matrix from which the eigenvalues and eigenvectors are extracted. These eigenvectors are used to find a transformation matrix, that is coordinate transformation. This is then used to modify the crossover and have a rotational invariant behaviour, see [18].

Albeit not directly addressing rotations, another closely study based on DE has been proposed in [50] where a partition entropy given by fuzzy clustering is used for solving problems with a strong interdependence among variables. Furthermore in [49] a rotational invariant technique is indirectly proposed by performing local sampling around the parent solution in lieu of the crossover.

Furthermore, two studies about rotational invariant DE algorithms in multi-objective optimisation have been proposed in [26] and [27]. While paper [27] focuses on rotated problems, paper [26] shows how fairly simple DE implementations can lead to satisfactory results on rotated problems.

The present paper aims at analysing the concept of rotation and its effect on problems. This paper shortly describes and compares the main techniques integrated within DE based algorithms previously proposed in the literature. A novel experimental setup aiming at studying the effect of rotational invariant components is here proposed. Our experimental study, performed on multiple dimensionality values, addresses the following research question:

> How could the search operators of Differential Evolution be modified to handle rotated problems?

or equivalently

> How could the search operators of Differential Evolution be modified to handle black-box problems characterised by a high epistasis?

The remainder of this article is organised in the following way. Section 2 briefly introduces the notation and clarifies how rotated problems are designed. Section 3 introduces and compares the modified DE version incorporating the rotational invariant components categorised in the taxonomy proposed above. Section 4 describes the experimental results and illustrates the numerical results obtained in this study. Section 5 provides the conclusions of this work and addresses the research question posed above.

2. Notation and Rotation

Without a loss of generality, we will refer to the minimisation problem of an objective function (or fitness) $f(\mathbf{x})$ where the candidate solution $\mathbf{x}$ is a vector of n design variables (or genes) $x_1, x_2, \ldots, x_n$ in an n-dimensional decision space $\mathbf{D}$. Thus, the optimization problem considered in this paper consists of the detection of that solution $\mathbf{x}^* \in \mathbf{D}$ such that $f(\mathbf{x}^*) < f(\mathbf{x})$, and this is valid $\forall \mathbf{x} \in \mathbf{D}$. Array variables are highlighted in bold face throughout this paper.

The rotation is a geometric concept which has a clear meaning in two and three dimensions. Furthermore, while the rotation of an object in the space always has a meaning, the rotation of a function (a fitness landscape) may lead to something that is not a function. For example, a 45^o rotation of the single variable function $f(\mathbf{x}) = \sin(x)$ having the point $(0, \sin(0))$ as the rotation focus would not be a function.

Besides this fact, in the n-dimensional case, the rotation of an object (so is the rotation of a fitness landscape) does not have a geometric meaning. However, in algebra, an n-dimensional rotation is interpreted as an extension of the rotation operation of a point through a matrix, see [36].

In other words, if $\mathbf{x}$ is an n-dimensional vector, its rotated vector is

$$\mathbf{x^{rot}} = \mathbf{Qx}$$

where $\mathbf{Q}$ is a rotation matrix. The rotation matrix $\mathbf{Q}$ is an orthogonal matrix, i.e. $\mathbf{Q^T} = \mathbf{Q^{-1}}$, or

$$\mathbf{QQ^T} = \mathbf{Q^TQ} = \mathbf{I}$$

where $\mathbf{Q^T}$ and $\mathbf{Q^{-1}}$ are the transpose and the inverse of the matrix $\mathbf{Q}$, respectively and $\mathbf{I}$ is the identity matrix, see [36]. From simple linear algebra considerations, we can observe that the determinant $\det(\mathbf{Q})$ is either equal to 1 or to -1, thus, the rotation matrix $\mathbf{Q}$ is non-singular.

In this sense, the rotation of a point/vector in the n-dimensional space can be interpreted as a special case of change of basis. However, since the optimisation process occurs on a static domain $\mathbf{D}$ and thus on a static reference system, the operation $\mathbf{Qx}$ is in practice a systematic replacement of the candidate solution.

In order to clarify the functioning of the rotated problems in modern benchmarks, let us consider a test problem, for example, the popular Rastrigin function in n dimensions [47]:

$$f_{Rastr}(\mathbf{x}) = a \cdot n + \sum_{i=1}^{n} \left(x_i^2 + a \cdot \cos(2\pi x_i)\right)$$

with $a = 10$. The optimisation problem consists of minimising f_{Rastr} in $\mathbf{D} = [-5.12, 5.12]^n$.

The rotated Rastrigin function as in [47, 30, 31] does not mean that the fitness landscape is directly rotated. The rotated Rastrigin function is simulated by systematically perturb the position of a candidate solution by means of the linear transformation $\mathbf{Qx}$.

For the sake of clarity, every time the objective function of the rotated Rastrigin has to be calculated for the candidate solution $\mathbf{x}$, the two steps in Algorithm 1 are applied [30, 31].

Algorithm 1 Calculation of the rotated Rastrigin fitness value

1: input $\mathbf{x}$
2: $\mathbf{x^{rot}} = \mathbf{Qx}$
3: $y = f_{Rastr}(\mathbf{x^{rot}})$
4: output y

The same two-step procedure is applied for all the other rotated problems.

In order to give a practical example of the rotational logic and its effect on the moving operators (i.e. the operators that enable the search moves), let us focus on the Rastrigin function in two variables. Fig. 1 shows the contour plot of the Rastrigin function and depicts a point $\mathbf{x}$ and its rotated counterpart $\mathbf{x^{rot}}$. Let us consider the candidate solution

$$\mathbf{x} = \begin{pmatrix} 0 \\ -3 \end{pmatrix}.$$

The Rastrigin function in two variables has its global minimum in the origin $(0, 0)$. This from $\mathbf{x}$ the optimum can be reached by keeping the first variable and perturbing the second one. A metaheuristic that perturbs the variables separately (or maintains some of the variables constant while perturbs the others) would easily solve this problem.

Let us now consider the case of rotated Rastrigin. Let us introduce a rotation matrix

$$\mathbf{Q} = \begin{pmatrix} -0.70711 & -0.70711 \\ 0.70711 & -0.70711 \end{pmatrix}$$

which performs a clockwise rotation around the origin of an angle $\theta = \frac{3}{4}\pi = 135^o$. We can now calculate the rotated candidate solution as

$$\mathbf{x^{rot}} = \mathbf{Qx} = \begin{pmatrix} 2.1213 \\ 2.1213 \end{pmatrix}.$$

The global minimum is still in the origin $(0, 0)$. However, a moving operator that perturbs the variables separately would likely fail since both the variables would need to be perturbed. Thus, in this case, the metaheuristic would benefit from a moving operator that simultaneously perturbs both the variables.

In this sense, the rotation increases the difficulty of the optimisation problems. From the perspective of the algorithm, operators that perturb the design variables separately can work can work effectively in some cases but are rotational variant since a rotation could make them ineffective.

3. Rotational Invariant Differential Evolution Algorithms

Although in this paper we assume that the reader is familiar with the standard DE, we introduce in Section 3.1 the DE notation and give a brief description of the DE framework, see [12, 13, 37]. Subsequently, Section 3.2 describes rotational invariant DEs by linear combination, Section 3.3 describes rotational invariant DEs by considering the decision space as an inner product space and Section 3.4 describes rotational invariant DEs by considering the decision space as a vector space, respectively.

3.1. Differential Evolution: general framework and rotational variance

DE algorithms work with a population of N_p candidate solutions:

$$\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^i}, \ldots, \mathbf{x^{N_p}}.$$

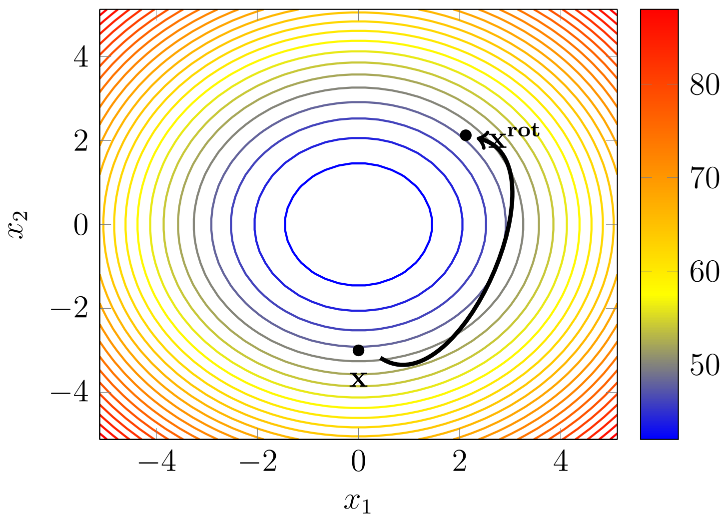

Figure 1: Contour plot of the Rastrigin function and rotation of a point

The algorithmic cyclically performs a set of operations. During each cycle, here indicated as generation, each candidate solution $\mathbf{x^i}$ is perturbed by means of a first mechanism, called *mutation* and a second perturbation mechanism called *crossover*. The perturbed individual, namely offspring, is indicated with $\mathbf{x^{off}}$. The fitness of the offspring is then calculated and compared with that of $\mathbf{x^i}$. The result of the comparison is recorded but no replacement occurs before the end of the generation. At the end of the generation, each $\mathbf{x^i}$ outperformed by its corresponding offspring $\mathbf{x^{off}}$ is replaced by the latter.

Algorithm 2 describes the general DE structure.

Algorithm 2 General Differential Evolution Framework

```
Generate an initial population of Np individuals
Evaluate fitness of each solution in population Np
while termination condition is not met do
    for each x^i in Np do
        Generate provisional offspring x^off' by mutation
        Generate offspring x^off by crossover
        Evaluate fitness of x^off
        Make a note whether x^i or x^off has a better performance
    end for
    for each x^i in Np do
        Perform all the replacements by choosing the best between parent offspring
    end for
end while
```

Thus, DE frameworks make use of two moving operators, mutation and crossover, respectively.

In the literature a wealth of mutations have been proposed, see e.g. [45]. The original mutation, so-called DE/rand/1, performs the perturbation of the individual $\mathbf{x^i}$ without using it directly. The DE/rand/1 mutation randomly samples three candidate solutions from the population: $\mathbf{x^r}$, $\mathbf{x^s}$, and $\mathbf{x^t}$, respectively. Then, the provisional offspring $\mathbf{x^{off'}}$ is calculated by means of the following formula

$$DE/rand/1 : \mathbf{x^{off'}} = \mathbf{x^t} + F\left(\mathbf{x^r} - \mathbf{x^s}\right)$$

where the scale factor F is a parameter to be set (usually $F > 0$ and $F < 1$ [53, 51]).

As observed in [2] the mutation is inherently rotational invariant since it is obtained as the weighted sum of one or more candidate solutions. Considering that the candidate solutions are vectors and the sum of vectors involves all their components, the mutation moves across all the variables at the same time, thus being rotational invariant.

The crossover in DE frameworks is of two kinds: binomial and exponential. Both of them generate the offspring solution $\mathbf{x^{off}}$ by combining the design variables of $\mathbf{x^{off'}}$ and those of $\mathbf{x^i}$. The two crossovers differ from the logic used to select which genes from provisional $\mathbf{x^{off'}}$ and which genes from the parent $\mathbf{x^i}$ are copied into the offspring $\mathbf{x^{off}}$

The binomial crossover consists of the following steps. A design variable index j_{rand} is randomly selected. The corresponding design variable in $\mathbf{x^{off'}}$ is selected and copied in $\mathbf{x^{off}}$. For all the other variables, a random number is generated by means of uniform distribution $\mathcal{U}(0, 1)$. If this number is equal or less than a parameter Cr (chosen between 0 and 1), the corresponding design variable is copied from $\mathbf{x^{off'}}$ to $\mathbf{x^{off}}$. If the generated number is greater than the parameter Cr, the corresponding design variable is copied from $\mathbf{x^i}$ to $\mathbf{x^{off}}$.

The pseudocode of the binomial crossover is shown in Algorithm 3.

Algorithm 3 Binomial Crossover Between $\mathbf{x^i}$ and $\mathbf{x^{off'}}$

```
1: generate an integer random index j_rand
2: x^off_{j_rand} = x^off'_{j_rand}
3: for j = 1 : n, j ≠ j_rand do
4:    generate a random value h from a uniform distribution U(0, 1)
5:    if h ≤ Cr then
6:       x^off_j = x^off'_j
7:    else
8:       x^off_j = x^i_j
9:    end if
10: end for
```

The exponential crossover consists of the following steps. Also in this case, the design variable index j_{rand} is randomly selected and the corresponding design variable in $\mathbf{x^{off'}}$ is selected and copied in $\mathbf{x^{off}}$. Then, contiguous design variables are copied, one by one, from the provisional offspring $\mathbf{x^{off'}}$ to the final offspring $\mathbf{x^{off}}$ until a random number is less than the crossover probability Cr.

The two version of DE employing the DE/rand/1 mutation as well as binomial and exponential crossover are indicated with DE/rand/1/bin and DE/rand/1/exp, respectively.

Algorithm 4 Exponential Crossover Between $\mathbf{x^i}$ and $\mathbf{x^{off'}}$

```
x^off = x^i
generate an integer random index j_rand
x^off_{j_rand} = x^{off'}_{j_rand}
generate a random value h from a uniform distribution U(0,1)
j = j_rand + 1
while h ≤ Cr AND j < n do
    x^off_j = x^{off'}_j
    if j == n then
        j = 1
    end if
    k = k + 1
    generate a random value h from a uniform distribution U(0,1)
end while
```

The main difference between the two crossover strategies is that while the binomial crossover copies scattered genes from the provisional offspring $\mathbf{x^{off'}}$ to the offspring $\mathbf{x^{off}}$ while the exponential crossover copies an entire sequence of contiguous genes.

Although this difference may have an impact on the DE performance both these crossovers have the same limitation in terms of rotational invariance: both the crossovers generate an offspring solution perturb only some design variables of $\mathbf{x^i}$ while the others are kept constant.

A naive solution would be to use only the mutation in DE. However, the resulting algorithm would not function well since it would behave almost like a random search. As shown in the theoretical study reported in [54], some form of crossover is essential to ensure a proper DE functioning.

3.2. Rotational invariant DE by linear combination

Since the core "defect" of DE crossover appears to be that it does not involve all the design variables, a straightforward way to obtain a rotational invariant DE is to combine crossover and mutation so that all the variables are simultaneously perturbed.

By following this logic, the current-to-rand mutation, see [28, 38], that is a moving operator which functions as a weighted sum that produces as the linear combination of multiple contributions has been proposed. The resulting algorithm, namely DE/current-to-rand/1 applies the following moving operator in order to generate $\mathbf{x^{off}}$ from a candidate solution $\mathbf{x^i}$

$$DE/current-to-rand/1 : \mathbf{x^{off}} = \mathbf{x^i} + K\left(\mathbf{x^t} - \mathbf{x^i}\right) + K \cdot F\left(\mathbf{x^r} - \mathbf{x^s}\right)$$

where $\mathbf{x^r}$, $\mathbf{x^s}$, and $\mathbf{x^t}$ are three candidate solutions randomly sampled from the population (exactly like in the case of DE/rand/1). The parameter K, namely combination coefficient, is randomly sampled from a uniform distribution $\mathcal{U}(0,1)$ at each offspring calculation. The parameter F is a constant to be set in algorithm design phase and plays the same role as the scale factor in DE/rand/1 mutation.

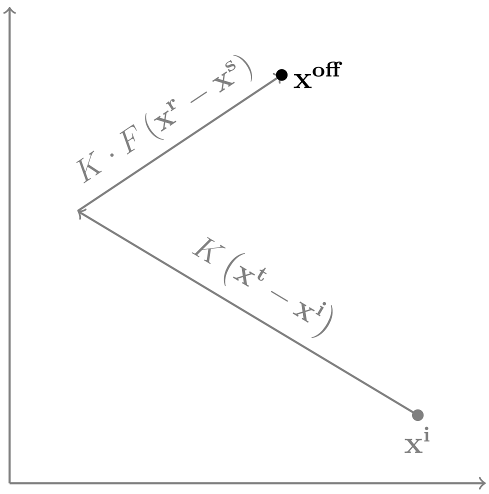

Figure 2: DE/current-to-rand/1 offspring generation (in a bi-dimensional decision space)

In order to better understand the functioning of DE/current-to-rand, a graphical presentation of its implementation is shown in Fig. 2.

Fig. 2 clearly shows that the moving operator of DE/current-to-rand/1 is a sum of vectors which perturbs $\mathbf{x^i}$ to generate $\mathbf{x^{off}}$.

In order to better understand the implementation of the rotational invariant mutation-crossover within the DE framework, see [9, 28, 38], Algorithm 5 shows a DE embedding the DE/current-to-rand/1 mutation.

Algorithm 5 Rotation-Invariant Differential Evolution: DE/current-to-rand [28, 38]

Generate an initial population of Np individuals
Evaluate fitness of each solution in population Np
while termination condition is not met **do**
  for each $\mathbf{x^i}$ in Np **do**
    Sample the random number K from $\mathcal{U}(0, 1)$
    Generate the offspring $\mathbf{x^{off}}$ by mutation
    DE/current-to-rand: $\mathbf{x^{off}} = \mathbf{x^i} + K\left(\mathbf{x^t} - \mathbf{x^i}\right) + K \cdot F\left(\mathbf{x^r} - \mathbf{x^s}\right)$
    Evaluate fitness of $\mathbf{x^{off}}$
    Make a note whether $\mathbf{x^i}$ or $\mathbf{x^{off}}$ has a better performance
  end for
  for each $\mathbf{x^i}$ in Np **do**
    Perform all the replacements by choosing the best between parent offspring
  end for
end while

3.3. Rotational invariant DE by orthogonalisation of a basis

In order to understand the spirit of this technique, let us remark that a candidate solution $\mathbf{x} \in D$ is an $n-$dimensional vector. Thus if we consider two vectors

$$\mathbf{x} = (x_1, x_2, \ldots, x_n)$$
$$\mathbf{y} = (y_1, y_2, \ldots, y_n)$$

we may introduce the scalar product

$$\mathbf{xy} = x_1y_1 + x_2y_2 + \ldots + x_ny_n.$$

In the case of $(\mathbf{xx})$ we would have

$$\mathbf{xx} = x_1^2 + x_2^2 + \ldots + x_n^2$$

and the norm $\|\mathbf{x}\|$ is

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}.$$

These notions are used to apply the Gram-Schmidt orthogonalisation, see [36], within the crossover.

The Rotation Invariant Differential Evolution (RIDE) proposed in [48, 49] contains a modified crossover which makes the metaheuristic rotational invariant. Two versions of this rotational invariant crossover have been defined according to the logic used to exchange the genes, binomial and exponential, respectively.

This rotational invariant crossover consists of the following steps. From the population of Np candidate solutions the centroid c is calculated (all the vectors are summed up element by element and each element is divided by Np):

$$\mathbf{c} = \frac{1}{Np}\sum_{j=1}^{Np}\mathbf{x^j}.$$

Then, for each element of the population $\mathbf{x^j}$ a new element $\mathbf{d^j}$ is calculated as

$$\mathbf{d^j} = \mathbf{x^j} - \mathbf{c}.$$

As shown in Fig. 3, we have thus a population of Np vectors all starting from the centroid $\mathbf{c}$:

$$\mathbf{d^1}, \mathbf{d^2}, \ldots, \mathbf{d^{Np}}.$$

Out of these Np vectors, n (like the dimensionality) of them are randomly selected. Clearly, this is possible only if $Np \geq n$. Let us indicate these n vectors selected from $\mathbf{d^1}, \mathbf{d^2}, \ldots, \mathbf{d^{Np}}$ as

$$\mathbf{v^1}, \mathbf{v^2}, \ldots, \mathbf{v^n}.$$

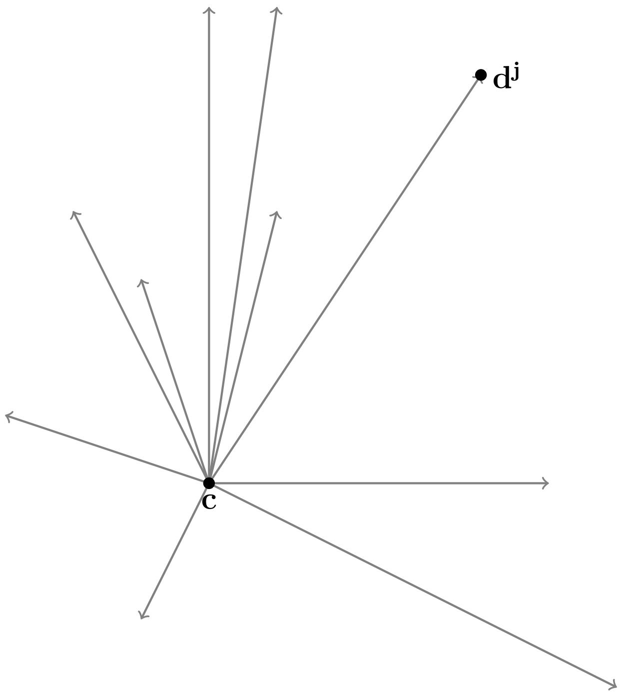

Figure 3: Population of Np difference vectors $\mathbf{d^j}$ with respect to a centroid $\mathbf{c}$ in RIDE

These n vectors are then processed by the Gram-Schmidt orthogonalisation to detect a basis of n orthonormal vectors, see [36]:

$$\begin{aligned}
\mathbf{e^1} &= \frac{\mathbf{v^1}}{\|\mathbf{v^1}\|} \\
\mathbf{e^2} &= \frac{\mathbf{v^2}-\left(\mathbf{e^1}\mathbf{v^2}\right)\mathbf{e^1}}{\|(\mathbf{e^1}\mathbf{v^2})\mathbf{e^1}\|} \\
&\dots \\
\mathbf{e^n} &= \frac{\mathbf{v^n}-\sum_{\mathbf{j=2}}^{\mathbf{n}}\left(\mathbf{e^{j-1}}\mathbf{v^j}\right)\mathbf{e^{j-1}}}{\left\|\mathbf{v^n}-\sum_{\mathbf{j=2}}^{\mathbf{n}}\left(\mathbf{e^{j-1}}\mathbf{v^j}\right)\mathbf{e^{j-1}}\right\|}
\end{aligned}$$

where $\left(\mathbf{e^{j-1}}\mathbf{v^j}\right)\mathbf{e^{j-1}}$ has been obtained by calculating at first the scalar product $\mathbf{e^{j-1}}\mathbf{v^j}$ and then multiplying this scalar by the vector e^{j-1}. Thus, $\mathbf{e^{j-1}}\mathbf{v^j}\mathbf{e^{j-1}}$ is a vector.

When this orthonormal basis $B = \{\mathbf{e^1}, \mathbf{e^2}, \dots, \mathbf{e^n}\}$ has been calculated, it feeds the crossover operation in the following way.

A crossover means that a gene in position j can be copied from the provisional offspring $\mathbf{x^{off'}}$ is copied into the offspring. Alternatively, the corresponding j^{th} gene of $\mathbf{x^i}$ is copied into the offspring. This concept can be expressed stating that the offspring is the parent solution $\mathbf{x^i}$ whose genes may be perturbed by adding $x_j^{off'} - x_j^i$ to it:

$$x_j^{off} = x_j^i + \left(x_j^{off'} - x_j^i\right).$$

This rotational invariant crossover decomposes the gene to be copied from $\mathbf{x^{off'}}$ over the directions of the orthonormal basis $B = \{\mathbf{e^1}, \mathbf{e^2}, \dots, \mathbf{e^n}\}$. If we pose

$$\mathbf{y} = \mathbf{x^{off'}} - \mathbf{x^i}$$

then we can decompose this vector over the directions of the basis B. The process of copying the j^{th} variable of the mutant into the offspring is expressed in its new basis by

$$\mathbf{x^{off}} = \mathbf{x^{i}} + \left(\mathbf{e^{j}}\mathbf{y}\right)\mathbf{e^{j}}.$$

Thus, also in this case, the crossover is an operator that involves a perturbation of all the variables.

Binomial and exponential rotational invariant crossovers by means of the basis $B = \{\mathbf{e^1}, \mathbf{e^2}, \ldots, \mathbf{e^n}\}$ are described in Algorithms 6 and 7, respectively.

Algorithm 6 Binomial Crossover Between $\mathbf{x^i}$ and $\mathbf{x^{off'}}$ by Orthogonalisation of a Basis

$\mathbf{y} = \mathbf{x^{off'}} - \mathbf{x^i}$
Calculate the orthonormal basis $B = \{\mathbf{e^1}, \mathbf{e^2}, \ldots, \mathbf{e^n}\}$ by Gram-Schmidt orthogonalisation
generate an integer random index j_{rand}
$x^{off}_{j_{rand}} = x^{off'}_{j_{rand}}$
for $j = 1 : n$, $j \neq j_{rand}$ **do**
  generate a random value h from a uniform distribution $\mathcal{U}(0,1)$
  if $h \leq Cr$ **then**
    $\mathbf{x^{off}} = \mathbf{x^{i}} + \left(\mathbf{e^{j}}\mathbf{y}\right)\mathbf{e^{j}}$
  end if
end for

Algorithm 7 Exponential Crossover Between $\mathbf{x^i}$ and $\mathbf{x^{off'}}$ by Orthogonalisation of a Basis Transformation

$\mathbf{y} = \mathbf{x^{off'}} - \mathbf{x^i}$
Calculate the orthonormal basis $B = \{\mathbf{e^1}, \mathbf{e^2}, \ldots, \mathbf{e^n}\}$ by Gram-Schmidt orthogonalisation
$\mathbf{x^{off}} = \mathbf{x^i}$
generate an integer random index j_{rand}
$x^{off}_{j_{rand}} = x^{off'}_{j_{rand}}$
generate a random value h from a uniform distribution $\mathcal{U}(0,1)$
$j = j_{rand} + 1$
while $h \leq Cr$ AND $j < n$ **do**
  $\mathbf{x^{off}} = \mathbf{x^{i}} + \left(\mathbf{e^{j}}\mathbf{y}\right)\mathbf{e^{j}}$
  if $j == n$ **then**
    $j = 1$
  end if
  $k = k + 1$
  generate a random value h from a uniform distribution $\mathcal{U}(0,1)$
end while

This approach, albeit ingenious, hides the following three limitations.

- In order to be applied this crossover requires a population size larger than the problem dimensionality. This crossover can be inconvenient (computationally expensive) for large-scale problems since it would impose the use of very large population size thus limiting the number of generations.

- In order to be applied this crossover requires that the vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}$ are linearly independent, see [36]. If this is not the case, the Gram Schmidt orthogonalisation cannot return an orthonormal basis and the entire crossover cannot be applied. Although in an $n-$dimensional continuous space it is likely that n vectors randomly sampled are linearly independent, the applicability of this crossover still depends on the randomly selected points.

- The basis is constructed around the difference vectors between candidate solutions and a centroid. Although this choice is indeed related to the evolution and can be seen as a guess of the direction of the gradient, there is no guarantee that the randomly chosen vectors are a sensible choice (or more sensible than a computationally cheaper linear combination approach).

3.4. Rotational invariant DE by Diagonalisation of the Covariance Matrix

Let us consider the population of NP candidate solutions

$$\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^{Np}}$$

and let us emphasise the components of the generic candidate solution $\mathbf{x^i}$:

$$\mathbf{x^i} = \left(x_1^i, x_2^i, \ldots, x_n^i\right).$$

Within a population let us calculate the average per component, i.e. the average first variable over the population, average second variable over the population etc. The average generic k^{th} variable over the population is given by

$$\bar{x_k} = \frac{1}{Np} \sum_{i=1}^{Np} x_k^i.$$

Thus, for each variable we have

$$\bar{x_1}, \bar{x_2}, \ldots, \bar{x_n}.$$

With these values we may construct a *covariance matrix* whose generic element is

$$Cov_{j,k} = \frac{\sum_{i=1}^{Np} \left(x_j^i - \bar{x_j}\right) \left(x_k^i - \bar{x_k}\right)}{Np - 1}.$$

As a remark to better understand this formula, each of the factors in the numerator is the deviation of a design variable from its population average.

Let us indicate with **Cov** the resulting $n \times n$ matrix, that is the covariance matrix. This matrix has been used in metaheuristic optimisation on several occasions, see e.g. [22, 23, 21].

The rotational invariant DE crossover technique proposed in [18] makes use of the **Cov** matrix. Like the method described in Section 3.3, this technique performs a change of coordinates but unlike the method in Section 3.3, derives the new vector basis (for changing the coordinates) without exploiting the properties of the scalar product.

More specifically, the crossover in [18] interprets the covariance matrix **Cov** as an endomorphism and diagonalise it, see [36]. In other words, the eigenvalues of the matrix **Cov** are determined. In [18] the Jacobi's method [15] has been used for numerically determining the eigenvalues. The determined eigenvalues are placed on the diagonal of the matrix **D** while the corresponding eigenvectors are placed as columns of a matrix **P**. These matrices would be such that

$$\mathbf{D} = \mathbf{P}^{-1}\mathbf{Cov}\mathbf{P}.$$

Under some hypotheses, the eigenvectors are linearly independent and compose a basis spanning the domain, see [36]. Thus, the vectors undergoing crossover can be decomposed along the directions of the vectors in the basis of eigenvectors. The change of variables can be easily performed by means of the transformation matrix **P**. According to [18], in order to perform the rotational invariant crossover between $\mathbf{x^i}$ and $\mathbf{x^{off'}}$ the two vectors are at first expressed on the basis of eigenvectors

$$\mathbf{x^i_{eig}} = \mathbf{P}\mathbf{x^i}$$

$$\mathbf{x^{off'}_{eig}} = \mathbf{P}\mathbf{x^{off'}}$$

The crossover according to Algorithm 3 or 4 is applied to $\mathbf{x^i_{eig}}$ and $\mathbf{x^{off'}_{eig}}$ to produce the offspring $\mathbf{x^{off}_{eig}}$. Finally, the offspring is generated by applying the following transformation

$$\mathbf{x^{off}} = \mathbf{P}^*\mathbf{x^{off}_{eig}}$$

where $\mathbf{P}^*$ is the conjugate transpose of **P**, i.e. a matrix obtained from **P** after transposing it and replacing all the complex entries with its conjugates.

The underpinning idea behind this method is that since the covariance matrix estimates the correlation between pairs of variables, a coordinate transformation along the directions of the eigenvectors would diagonalise the matrix thus making the correlation between pairs of variables null. In other words, the correlation between pairs of variables depends on the system of coordinates. A convenient system of coordinates allows the solution of an optimisation problem by performing perturbation along its axes.

The rotational invariant nature of this offspring generation lies in the fact that all the variables are perturbed. Furthermore, the reference system determined by the eigenvectors of the covariance matrix estimates the direction of the gradient, thus adapting the search towards the most convenient directions. This intuition is experimentally demonstrated in [18].

On the other hand, paper [18] states that the use of only a rotational invariant crossover can jeopardise the evolution. Thus, the authors propose the use of either a standard DE crossover or their rotational invariant proposal by means of a probability Pr.

The resulting crossover, here indicated as eigen-crossover, is described in Algorithm 8.

Algorithm 8 Eigen-Crossover Between $\mathbf{x^i}$ and $\mathbf{x^{off'}}$

```
generate a random value l from a uniform distribution
if l ≤ Pr then
    calculate the elements of the covariance matrix Cov
    apply the Jacobi's procedure [15] to calculate eigenvalues and eigenvectors (matrix P)
    calculate x^i_eig = P x^i
    calculate x^off'_eig = P x^off'
    apply the crossover between x^i_eig and x^off'_eig as in Algorithm 3 or Algorithm 4 to calculate x^off_eig
    calculate the transpose conjugate matrix P*
    calculate the offspring x^off = P* x^off_eig
else
    apply the crossover between x^i and x^off' as in Algorithm 3 or Algorithm 4 to calculate x^off
end if
```

The main advantage of this method with respect to that described in Section 3.3 is that the eigen-crossover does not impose a population size $Np \geq n$. Another important advantage of this approach is that since the covariance matrix is symmetric, the matrix is surely diagonalizable all its eigenvalues are all real and its eigenvectors are an orthogonal basis. In other words, the eigen-crossover can, in principle, be applied with no restrictions.

We have reported the description of the eigen-crossover as in [18]. However, it can be observed that the calculation of the transpose conjugate matrix $\mathbf{P}^*$ is not relevant in this case. Since the matrix **Cov** is inherently symmetric, its eigenvalues are all real numbers and its eigenvectors are all orthogonal, see [36]. Since the **Cov** matrix contain only real numbers, the matrix $\mathbf{P}$ is also a matrix of real numbers. Thus in the eigen-crossover description $\mathbf{P}^* = \mathbf{P^T}$, where $\mathbf{P^T}$ is the transpose of $\mathbf{P}$.

Although the eigen-crossover is an extremely promising technique, it presents some limitations.

- The method requires the handling of the matrix which leads to a computational overhead that has a quadratic complexity (with respect to the problem dimensionality).
- Since the covariance is based on a sampled population, there is no guarantee that the eigenvector directions are actually representative of the most convenient search directions. The eigen-crossover is likely to require some generations before being effective.
- The eigen-crossover requires the calculation of the eigenvalues and eigenvectors of a matrix. Since this task requires the calculation of several determinants, a theoretical

approach would not be computationally feasible even in moderate dimensions, e.g. $n = 20$. For this reason, an approximated method for calculating the eigenvalues [15] is proposed and used. This is another reason that could make the crossover not fully reliable.

Although the eigen-crossover is indeed a sophisticated, elegant and technically sound technique, the choice of the authors of [18] of using an auxiliary standard crossover is a sign of its imperfect robustness.

4. Numerical Results

In order to perform a thorough comparison on rotational invariance, we have designed a benchmark by modifying an existing testbed.

More specifically, we have considered a modern testbed composed of non-rotated and rotated problems, i.e. the CEC2014 testbed [32]. It is worth mentioning that the subsequent testbed, i.e. the CEC2015 [11], makes use only of some of the objective functions of the CEC2014, which is then more comprehensive.

For all the problems in CEC2014, two versions have been considered, the non-rotated and rotated versions, respectively. Two versions are considered to test each algorithm in this study twice, before and after rotation. This allows monitoring in each case, the possible performance deterioration due to the problem's rotation.

The non-rotated version can be easily obtained by removing the instruction $\mathbf{x^{rot}} = \mathbf{Qx}$, see Algorithm 1. Conversely, a rotated version has been generated by adding the instruction $\mathbf{x^{rot}} = \mathbf{Qx}$.

The original CEC2014 testbed albeit composed of 30 problems is based on 28 functions, since f_8 is the Rastrigin function and f_9 is its rotated version while f_{10} is the Schwefel function and f_{11} its rotated version.

Thus, out of the 30 test problems we have derived 56 test problems, 28 being non-rotated and 28 being rotated. For keeping consistency with the function enumeration of CEC2014 we omit problems f_8 and f_{10}. Problems f_9 and f_{11} are Rastrigin and Schwefel respectively that are non-rotated or rotated where appropriated and as indicated in the corresponding Table caption.

In order to report an extensive study which allows us to draw to convincing conclusions, the 56 problems under considerations have been repeated for multiple levels of dimensionality: 10, 20, 30, 50, 100 dimensions. In total, 280 test problems have been considered in this study. However, for the sake of brevity, we are reporting the detailed results in 10, 50 and 100 results. The total set of functions is used for the summary of results, see Section 4.3 below.

Furthermore we have included two real-world problems (which are often characterised by a high epistasis) from [14], that is problems p_1 and p_2, respectively. The problem p_1 is in six variables while the problem p_2 can be scaled. For the latter problem the 30-dimensional case (corresponding to 10 atoms) has been considered.

In the following sections, the standard DE algorithm in the DE/rand/1/bin and DE/rand/1/exp implementations have been compared against the rotational invariant DE versions. In order to perform a fair comparison, we have considered that the meaning of Cr in binomial and exponential crossovers is not the same. While Cr in the binomial crossover represents the quota of genes transferred from the provisional offspring to the offspring, in the exponential crossover Cr takes a different meaning according to the problem dimensionality.

For example, $Cr = 0.3$ for the binomial crossover means that approximately 30% of the genes will be transferred regardless of the dimensionality. On the other hand, for the exponential crossover, $Cr = 0.3$ means that one gene can be transferred with a 0.3 probability, the second with $0.3 \times 0.3 = 0.09$, the third with $0.3 \times 0.3 \times 0.3 = 0.027$. In practice, this means that in 10 variables the offspring will have a non-negligible quota of genes coming from the provisional offspring while in higher dimensions the offspring will be almost identical to its parent solution $\mathbf{x^i}$.

Since this study aims at understanding the rotational invariant properties of DE and since crossover plays an important role, we have set a comparable quota of genes to be transferred. Thus, for DE/rand/1/exp, if we indicate with Cr_b the crossover rate of DE/rand/1/bin, the crossover rate Cr_e of DE/rand/1/exp should be such that

$$Cr_e^{nCr_b} = 0.5.$$

This equation means that on average nCr_b genes will be transferred, see [25]. After nCr_b transfers the copy will become an unlikely event. By solving this equation we have

$$Cr_e = \frac{1}{\sqrt[nCr_b]{2}}.$$

For example, if $Cr_b = 0.3$ a Cr_e corresponding to the same quota of gene copies from the provisional offspring in 100 dimensions is $Cr_e = 0.977$. All the results are reported in Tables. In each Table, for each problem, the best algorithm is highlighted in bold. Furthermore, for each problem a statistical pair-wise comparison has been performed by applying the Wilcoxon signed-rank test, see [52]. Throughout the paper, we have taken DE/rand/1/exp as the reference algorithm. A + indicates that DE/rand/1/exp significantly outperforms the competitor, a − indicates that DE/rand/1/exp is outperformed, and a = is shown when the performance of the two algorithms is statistically indistinguishable.

The standard DE algorithms have been run with the following parameters selected in accordance with the indications provided in [54].

- DE/rand/1/bin $Np = n$, $F = 0.7$, $Cr = 0.3$
- DE/rand/1/exp $Np = n$, $F = 0.7$, $Cr = \frac{1}{\sqrt[n0.3]{2}}$

The results have been divided according to the design of the crossover. More specifically, we have grouped the results in the following way.

- Rotational invariant crossover integrated within the mutation
- Rotational invariant crossover running separately to the mutation

The detailed results are reported in the following two subsections. Then we have summarised the results and added more DE algorithms which address rotation with a statistical ranking.

Table 1: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/exp) for DE/rand/1/exp, DE/rand/1/bin, and DE/current-to-rand/1 on CEC2014 [32] non-rotated version in 10 dimensions.

	DE/rand/1/exp	DE/rand/1/bin		DE/current-to-rand/1	
f_1	**0.00e + 00 ± 0.00e + 00**	0.00e + 00 ± 0.00e + 00	+	2.75e + 08 ± 4.28e + 08	+
f_2	**0.00e + 00 ± 0.00e + 00**	0.00e + 00 ± 0.00e + 00	+	5.14e + 09 ± 2.99e + 09	+
f_3	**0.00e + 00 ± 0.00e + 00**	0.00e + 00 ± 0.00e + 00	+	1.31e + 04 ± 6.11e + 03	+
f_4	2.59e − 02 ± 5.13e − 02	**2.12e − 02 ± 6.39e − 02**	=	2.13e + 03 ± 1.41e + 03	+
f_5	**8.70e + 00 ± 7.30e + 00**	1.46e + 01 ± 5.70e + 00	+	2.02e + 01 ± 5.81e − 01	+
f_6	**0.00e + 00 ± 0.00e + 00**	2.41e − 04 ± 1.30e − 03	=	7.49e + 00 ± 1.40e + 00	+
f_7	1.46e − 02 ± 1.08e − 02	**1.06e − 02 ± 9.66e − 03**	=	1.02e + 02 ± 4.61e + 01	+
f_8	±	±		±	
f_9	**1.33e + 00 ± 1.04e + 00**	1.49e + 00 ± 9.53e − 01	=	5.01e + 01 ± 8.14e + 00	+
f_{10}	±	±		±	
f_{11}	8.44e + 00 ± 2.09e + 01	**7.43e + 00 ± 2.31e + 01**	=	9.86e + 02 ± 2.64e + 02	+
f_{12}	**3.37e − 03 ± 3.87e − 03**	4.43e − 03 ± 6.00e − 03	=	7.25e − 01 ± 2.26e − 01	+
f_{13}	**1.68e − 01 ± 5.83e − 02**	2.04e − 01 ± 3.96e − 02	+	3.47e + 00 ± 5.46e − 01	+
f_{14}	**1.47e − 01 ± 4.29e − 02**	1.50e − 01 ± 3.16e − 02	=	4.52e + 01 ± 1.62e + 01	+
f_{15}	**3.73e − 01 ± 9.32e − 02**	6.53e − 01 ± 9.56e − 02	+	3.19e + 03 ± 4.92e + 03	+
f_{16}	**9.95e − 02 ± 1.21e − 02**	1.65e − 01 ± 5.41e − 02	+	2.95e + 00 ± 4.34e − 01	+
f_{17}	7.92e + 00 ± 2.19e + 01	**5.01e + 00 ± 2.14e + 01**	-	1.30e + 06 ± 3.86e + 06	+
f_{18}	1.82e + 00 ± 6.39e + 00	**5.77e − 01 ± 5.63e − 01**	=	3.07e + 07 ± 1.65e + 08	+
f_{19}	4.54e − 02 ± 4.19e − 02	**1.53e − 02 ± 1.90e − 02**	-	1.70e + 01 ± 1.02e + 01	+
f_{20}	**2.34e − 01 ± 3.02e − 01**	4.15e − 01 ± 5.22e − 01	=	4.90e + 03 ± 1.29e + 04	+
f_{21}	8.82e − 01 ± 3.02e + 00	**7.75e − 01 ± 3.01e + 00**	=	4.28e + 05 ± 1.22e + 06	+
f_{22}	**4.57e − 01 ± 6.97e − 01**	4.25e + 00 ± 2.13e + 01	-	1.40e + 02 ± 9.35e + 01	+
f_{23}	3.16e + 02 ± 9.09e − 13	**3.16e + 02 ± 8.98e − 13**	=	4.30e + 02 ± 7.58e + 01	+
f_{24}	**1.06e + 02 ± 3.40e + 00**	1.08e + 02 ± 3.64e + 00	=	1.95e + 02 ± 2.25e + 01	+
f_{25}	**1.70e + 02 ± 2.31e + 01**	1.75e + 02 ± 2.47e + 01	=	2.01e + 02 ± 5.08e + 00	+
f_{26}	**1.00e + 02 ± 3.65e − 02**	1.00e + 02 ± 4.01e − 02	=	1.25e + 02 ± 4.05e + 01	+
f_{27}	2.54e + 02 ± 1.81e + 02	**1.43e + 02 ± 1.61e + 02**	=	4.49e + 02 ± 1.08e + 02	+
f_{28}	4.48e + 02 ± 2.56e + 01	**4.44e + 02 ± 1.31e + 01**	=	9.65e + 02 ± 2.22e + 02	+
f_{29}	**2.19e + 02 ± 5.26e + 00**	2.19e + 02 ± 4.65e + 00	=	2.09e + 04 ± 6.80e + 04	+
f_{30}	**3.76e + 02 ± 1.30e + 02**	4.07e + 02 ± 9.15e + 01	=	5.39e + 04 ± 1.06e + 05	+

4.1. Experimental results: Rotational invariant DE with integrated crossover

This section compares the standard DE algorithms against the DE/current-to-rand with the following parameters.

- DE/current-to-rand $Np = n$, $F = 0.7$

Tables 1, 2 , 3 show the results on non-rotated problems in 10, 50, and 100 dimensions respectively.

Results on non-rotated problems show that the rotational invariant DE/current-to-rand/1 is systematically outperformed by the standard DE implementations. Apart from isolated cases in low dimensions D/rand/1/exp appears to have the best performance for the non-rotated CEC 2014 test. This result could be explained by the fact that these non-rotated problems can be more effectively tackled by an exploitative strategy, i.e. a crossover, since some design variables are perturbed while the others are inherited from $\mathbf{x^i}$ to $\mathbf{x^{off}}$. This explanation is intuitive for the base problems which are mostly constructed as the sum of multiple contributions. The same straightforward explanation can be given for the hybrid functions but not for the composition functions where the weights are functions themselves, see [32].

The same algorithms have been run on the rotated problems. Numerical results are reported in Tables 4, 5, 6, respectively.

Numerical results on rotated problems are quite different from those on non-rotated problems. The rotation seems to be a clear challenge for the exponential crossover, especially

Table 2: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/exp) for DE/rand/1/exp, DE/rand/1/bin, and DE/current-to-rand/1 on CEC2014 [32] non-rotated version in 50 dimensions.

	DE/rand/1/exp	DE/rand/1/bin		DE/current-to-rand/1	
f_1	3.63e − 08 ± 1.36e − 08	**4.18e − 09 ± 1.92e − 09**	-	7.54e + 08 ± 1.81e + 08	+
f_2	1.64e − 05 ± 5.39e − 06	**1.09e − 05 ± 3.69e − 06**	-	6.77e + 10 ± 4.99e + 09	+
f_3	2.52e − 11 ± 1.15e − 11	**2.10e − 11 ± 7.21e − 12**	=	6.58e + 04 ± 7.87e + 03	+
f_4	**3.20e + 01 ± 6.56e + 00**	1.65e + 02 ± 2.70e + 01	+	1.46e + 04 ± 2.23e + 03	+
f_5	**2.01e + 01 ± 1.53e − 02**	2.06e + 01 ± 3.98e − 02	+	2.11e + 01 ± 3.13e − 02	+
f_6	2.80e − 03 ± 3.15e − 04	**9.02e − 04 ± 1.36e − 04**	-	5.29e + 01 ± 2.67e + 00	+
f_7	2.47e − 04 ± 1.33e − 03	**4.54e − 11 ± 3.41e − 11**	-	6.36e + 02 ± 5.85e + 01	+
f_8	±	±		±	
f_9	**5.87e + 01 ± 4.50e + 00**	2.39e + 02 ± 8.08e + 00	+	3.77e + 02 ± 2.23e + 01	+
f_{10}	±	±		±	
f_{11}	**1.35e + 03 ± 1.98e + 02**	7.25e + 03 ± 2.88e + 02	+	1.12e + 04 ± 5.07e + 02	+
f_{12}	**1.20e − 01 ± 1.37e − 02**	5.13e − 01 ± 4.31e − 02	+	2.72e + 00 ± 2.36e − 01	+
f_{13}	**5.32e − 01 ± 5.29e − 02**	5.96e − 01 ± 4.38e − 02	+	5.40e + 00 ± 2.68e − 01	+
f_{14}	**2.81e − 01 ± 2.69e − 02**	2.98e − 01 ± 2.68e − 02	+	2.10e + 02 ± 2.55e + 01	+
f_{15}	**1.02e + 01 ± 7.65e − 01**	2.98e + 01 ± 2.00e + 00	+	3.00e + 05 ± 1.46e + 05	+
f_{16}	**7.69e + 00 ± 4.79e − 01**	1.80e + 01 ± 3.13e − 01	+	2.09e + 01 ± 3.69e − 01	+
f_{17}	**1.74e + 01 ± 3.95e + 00**	3.27e + 02 ± 3.87e + 01	+	4.32e + 07 ± 2.84e + 07	+
f_{18}	**1.00e + 00 ± 2.69e − 01**	4.51e + 01 ± 3.48e + 00	+	7.00e + 09 ± 2.15e + 09	+
f_{19}	2.22e + 01 ± 9.17e − 01	**2.09e + 01 ± 6.90e + 00**	=	6.14e + 02 ± 1.65e + 02	+
f_{20}	**4.07e + 00 ± 6.00e − 01**	3.17e + 01 ± 3.22e + 00	+	2.11e + 04 ± 1.18e + 04	+
f_{21}	**1.01e + 01 ± 1.39e + 00**	1.43e + 01 ± 2.59e + 00	+	2.50e + 07 ± 1.88e + 07	+
f_{22}	**2.33e + 01 ± 2.73e − 01**	2.82e + 01 ± 8.94e − 01	+	2.31e + 03 ± 1.96e + 03	+
f_{23}	3.22e + 02 ± 1.54e − 12	**3.22e + 02 ± 1.42e − 12**	=	8.80e + 02 ± 6.59e + 01	+
f_{24}	2.67e + 02 ± 3.51e + 00	**2.64e + 02 ± 3.95e + 00**	-	3.32e + 02 ± 6.68e + 00	+
f_{25}	2.07e + 02 ± 9.13e − 03	**2.07e + 02 ± 7.06e − 04**	=	2.31e + 02 ± 4.83e + 00	+
f_{26}	**1.01e + 02 ± 4.22e − 02**	1.01e + 02 ± 5.12e − 02	+	2.01e + 02 ± 1.67e + 01	+
f_{27}	3.03e + 02 ± 1.79e + 01	**3.00e + 02 ± 1.31e − 02**	-	1.62e + 03 ± 7.04e + 01	+
f_{28}	**1.11e + 03 ± 9.59e + 01**	1.28e + 03 ± 3.44e + 01	+	4.85e + 03 ± 4.79e + 02	+
f_{29}	3.38e + 02 ± 1.96e − 01	**3.38e + 02 ± 2.05e − 01**	=	3.80e + 08 ± 1.72e + 08	+
f_{30}	**5.73e + 03 ± 7.31e + 00**	5.82e + 03 ± 4.29e + 01	+	1.26e + 06 ± 1.11e + 06	+

Table 3: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/exp) for DE/rand/1/exp, DE/rand/1/bin, and DE/current-to-rand/1 on CEC2014 [32] non-rotated version in 100 dimensions.

	DE/rand/1/exp	DE/rand/1/bin		DE/current-to-rand/1	
f_1	**6.85e + 03 ± 1.02e + 03**	4.13e + 04 ± 4.40e + 03	+	2.90e + 09 ± 8.00e + 08	+
f_2	**4.19e + 06 ± 5.37e + 05**	7.73e + 08 ± 1.07e + 08	+	1.48e + 11 ± 9.98e + 09	+
f_3	**5.34e + 00 ± 6.83e − 01**	8.40e + 02 ± 1.06e + 02	+	1.52e + 05 ± 1.02e + 04	+
f_4	**2.66e + 02 ± 9.32e + 00**	3.60e + 03 ± 2.68e + 02	+	3.27e + 04 ± 2.86e + 03	+
f_5	**2.04e + 01 ± 1.76e − 02**	2.09e + 01 ± 1.72e − 02	+	2.13e + 01 ± 2.07e − 02	+
f_6	**1.10e + 01 ± 5.71e − 01**	5.03e + 01 ± 2.51e + 00	+	1.21e + 02 ± 3.47e + 00	+
f_7	**1.03e + 00 ± 2.14e − 02**	1.20e + 01 ± 1.62e + 00	+	1.50e + 03 ± 9.06e + 01	+
f_8	±	±		±	
f_9	**3.95e + 02 ± 1.76e + 01**	8.53e + 02 ± 1.95e + 01	+	9.20e + 02 ± 3.14e + 01	+
f_{10}	±	±		±	
f_{11}	**1.16e + 04 ± 4.48e + 02**	2.19e + 04 ± 4.46e + 02	+	2.72e + 04 ± 4.73e + 02	+
f_{12}	**3.28e − 01 ± 2.84e − 02**	1.16e + 00 ± 6.98e − 02	+	3.44e + 00 ± 1.82e − 01	+
f_{13}	**6.46e − 01 ± 3.46e − 02**	8.43e − 01 ± 5.40e − 02	+	6.48e + 00 ± 1.69e − 01	+
f_{14}	**3.14e − 01 ± 2.72e − 02**	4.36e − 01 ± 7.70e − 02	+	4.60e + 02 ± 2.12e + 01	+
f_{15}	**5.44e + 01 ± 1.77e + 00**	8.11e + 04 ± 1.46e + 04	+	1.03e + 06 ± 3.06e + 05	+
f_{16}	**3.17e + 01 ± 6.29e − 01**	4.25e + 01 ± 3.28e − 01	+	4.48e + 01 ± 3.04e − 01	+
f_{17}	**1.34e + 03 ± 1.29e + 02**	4.95e + 03 ± 2.06e + 02	+	4.76e + 08 ± 2.05e + 08	+
f_{18}	**1.24e + 02 ± 7.94e + 00**	3.69e + 02 ± 1.93e + 01	+	1.63e + 10 ± 3.13e + 09	+
f_{19}	**9.40e + 01 ± 1.48e + 01**	1.09e + 02 ± 1.94e + 01	+	2.17e + 03 ± 5.16e + 02	+
f_{20}	**9.82e + 01 ± 5.39e + 00**	3.22e + 02 ± 1.44e + 01	+	8.86e + 04 ± 3.06e + 04	+
f_{21}	**5.46e + 02 ± 5.80e + 01**	2.81e + 03 ± 2.07e + 02	+	7.14e + 07 ± 3.07e + 07	+
f_{22}	**6.69e + 01 ± 6.97e + 00**	5.44e + 02 ± 9.06e + 01	+	4.48e + 03 ± 9.73e + 02	+
f_{23}	**3.42e + 02 ± 2.19e − 02**	3.44e + 02 ± 1.67e − 01	+	1.21e + 03 ± 5.06e + 01	+
f_{24}	**3.73e + 02 ± 2.45e + 00**	4.44e + 02 ± 2.95e + 00	+	4.78e + 02 ± 9.03e + 00	+
f_{25}	**2.15e + 02 ± 1.00e − 01**	2.24e + 02 ± 3.30e − 01	+	2.56e + 02 ± 6.82e + 00	+
f_{26}	**1.01e + 02 ± 3.93e − 02**	1.05e + 02 ± 3.11e − 01	+	2.07e + 02 ± 2.30e + 00	+
f_{27}	**5.48e + 02 ± 8.91e + 00**	1.20e + 03 ± 3.72e + 01	+	3.12e + 03 ± 1.24e + 02	+
f_{28}	**4.19e + 03 ± 1.64e + 02**	5.14e + 03 ± 1.09e + 02	+	1.10e + 04 ± 7.93e + 02	+
f_{29}	1.44e + 03 ± 5.68e + 01	**1.20e + 03 ± 4.49e + 01**	-	2.17e + 09 ± 3.43e + 08	+
f_{30}	**2.30e + 04 ± 2.49e + 02**	4.40e + 04 ± 1.57e + 03	+	9.46e + 06 ± 3.48e + 06	+

Table 4: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/exp) for DE/rand/1/exp, DE/rand/1/bin, and DE/current-to-rand/1 on CEC2014 [32] rotated version in 10 dimensions.

	DE/rand/1/exp	DE/rand/1/bin		DE/current-to-rand/1	
f_1	9.21e+03 ± 1.97e+04	**3.44e+02 ± 3.89e+02**	-	1.70e+08 ± 2.31e+08	+
f_2	0.00e+00 ± 2.32e−14	**0.00e+00 ± 0.00e+00**	-	6.05e+09 ± 3.36e+09	+
f_3	3.88e+00 ± 1.93e+01	**0.00e+00 ± 3.11e−14**	-	1.83e+04 ± 1.14e+04	+
f_4	1.44e+01 ± 1.67e+01	**1.16e+01 ± 1.53e+01**	=	2.11e+03 ± 1.79e+03	+
f_5	2.01e+01 ± 2.50e−02	**1.99e+01 ± 1.24e+00**	+	2.02e+01 ± 7.62e−01	+
f_6	1.17e−01 ± 3.04e−01	**2.98e−02 ± 1.61e−01**	-	8.03e+00 ± 1.27e+00	+
f_7	**4.58e−02 ± 2.34e−02**	9.18e−02 ± 4.62e−02	+	1.41e+02 ± 6.10e+01	+
f_8	±	±		±	
f_9	**7.20e+00 ± 2.39e+00**	8.45e+00 ± 2.24e+00	=	4.75e+01 ± 1.08e+01	+
f_{10}	±	±		±	
f_{11}	**3.73e+02 ± 1.20e+02**	4.84e+02 ± 1.51e+02	+	9.93e+02 ± 1.93e+02	+
f_{12}	**3.76e−01 ± 6.26e−02**	4.89e−01 ± 8.04e−02	+	7.91e−01 ± 1.85e−01	+
f_{13}	**1.77e−01 ± 5.47e−02**	1.84e−01 ± 4.10e−02	=	4.02e+00 ± 1.09e+00	+
f_{14}	1.58e−01 ± 5.24e−02	**1.57e−01 ± 3.81e−02**	=	3.06e+01 ± 1.09e+01	+
f_{15}	**1.06e+00 ± 2.45e−01**	1.36e+00 ± 2.95e−01	+	5.82e+03 ± 9.28e+03	+
f_{16}	**2.32e+00 ± 3.06e−01**	2.38e+00 ± 2.60e−01	=	3.08e+00 ± 5.19e−01	+
f_{17}	2.80e+02 ± 6.65e+02	**2.06e+02 ± 5.58e+02**	=	6.79e+05 ± 1.51e+06	+
f_{18}	5.44e+00 ± 8.62e+00	**2.03e+00 ± 8.77e−01**	-	7.26e+06 ± 3.73e+07	+
f_{19}	1.60e−01 ± 1.48e−01	**8.80e−02 ± 8.12e−02**	-	3.08e+01 ± 2.78e+01	+
f_{20}	1.10e+00 ± 1.10e+00	**5.56e−01 ± 5.90e−01**	-	4.46e+03 ± 5.18e+03	+
f_{21}	1.78e+01 ± 2.93e+01	**1.51e+00 ± 4.20e+00**	-	1.21e+06 ± 5.74e+06	+
f_{22}	9.93e+00 ± 2.94e+01	**3.05e+00 ± 6.20e+00**	=	1.58e+02 ± 1.18e+02	+
f_{23}	**3.29e+02 ± 8.98e−13**	3.29e+02 ± 9.09e−13	=	4.22e+02 ± 6.51e+01	+
f_{24}	**1.17e+02 ± 3.78e+00**	1.18e+02 ± 3.56e+00	=	1.95e+02 ± 2.33e+01	+
f_{25}	**1.59e+02 ± 3.47e+01**	1.61e+02 ± 2.51e+01	=	2.03e+02 ± 3.33e+00	+
f_{26}	**1.00e+02 ± 5.36e−02**	1.00e+02 ± 3.75e−02	=	1.13e+02 ± 2.94e+01	+
f_{27}	2.53e+02 ± 1.69e+02	**1.85e+02 ± 1.60e+02**	=	4.23e+02 ± 1.73e+02	+
f_{28}	**3.64e+02 ± 7.49e+01**	3.76e+02 ± 1.99e+01	=	9.44e+02 ± 1.64e+02	+
f_{29}	**2.51e+02 ± 4.32e+01**	3.37e+02 ± 6.01e+01	+	3.46e+06 ± 7.13e+06	+
f_{30}	5.68e+02 ± 9.53e+01	**5.65e+02 ± 4.72e+01**	=	1.34e+04 ± 3.77e+04	+

Table 5: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/exp) for DE/rand/1/exp, DE/rand/1/bin, and DE/current-to-rand/1 on CEC2014 [32] rotated version in 50 dimensions.

	DE/rand/1/exp	DE/rand/1/bin		DE/current-to-rand/1	
f_1	**7.90e+07 ± 1.61e+07**	4.23e+08 ± 6.49e+07	+	1.80e+09 ± 6.00e+08	+
f_2	**6.55e+04 ± 3.78e+04**	2.94e+06 ± 2.49e+06	+	1.05e+11 ± 1.38e+10	+
f_3	**3.66e+01 ± 2.25e+01**	3.29e+04 ± 5.73e+03	+	1.09e+05 ± 1.56e+04	+
f_4	**1.45e+02 ± 2.59e+01**	1.71e+02 ± 3.02e+01	+	2.32e+04 ± 4.18e+03	+
f_5	**2.07e+01 ± 3.87e−02**	2.11e+01 ± 4.12e−02	+	2.11e+01 ± 3.34e−02	+
f_6	**4.52e+01 ± 1.66e+00**	6.16e+01 ± 1.29e+00	+	5.31e+01 ± 2.21e+00	+
f_7	**8.07e−03 ± 5.23e−03**	1.18e−01 ± 5.36e−02	+	1.02e+03 ± 1.24e+02	+
f_8	±	±		±	
f_9	**2.89e+02 ± 1.58e+01**	4.18e+02 ± 1.19e+01	+	4.61e+02 ± 3.78e+01	+
f_{10}	±	±		±	
f_{11}	**9.05e+03 ± 3.60e+02**	1.25e+04 ± 3.68e+02	+	1.15e+04 ± 5.14e+02	+
f_{12}	**1.11e+00 ± 1.07e−01**	2.54e+00 ± 2.99e−01	+	2.88e+00 ± 2.90e−01	+
f_{13}	**5.63e−01 ± 5.54e−02**	6.42e−01 ± 6.01e−02	+	6.80e+00 ± 4.08e−01	+
f_{14}	**3.01e−01 ± 2.43e−02**	3.02e−01 ± 3.42e−02	=	2.38e+02 ± 2.64e+01	+
f_{15}	**3.05e+01 ± 1.91e+00**	3.98e+01 ± 1.97e+00	+	9.33e+05 ± 5.59e+05	+
f_{16}	**2.05e+01 ± 3.83e−01**	2.23e+01 ± 1.66e−01	+	2.09e+01 ± 4.75e−01	+
f_{17}	**4.00e+04 ± 3.76e+04**	1.62e+07 ± 3.02e+06	+	1.24e+08 ± 7.49e+07	+
f_{18}	**2.72e+02 ± 3.32e+01**	1.01e+04 ± 4.92e+03	+	7.04e+09 ± 2.34e+09	+
f_{19}	**2.05e+01 ± 4.46e+00**	4.76e+01 ± 6.66e+00	+	7.84e+02 ± 3.52e+02	+
f_{20}	**2.26e+02 ± 3.23e+01**	1.59e+04 ± 5.06e+03	+	3.43e+04 ± 1.13e+04	+
f_{21}	**5.35e+03 ± 2.01e+03**	5.06e+06 ± 1.31e+06	+	8.75e+06 ± 4.73e+06	+
f_{22}	9.32e+02 ± 1.78e+02	**9.13e+02 ± 1.30e+02**	=	7.95e+03 ± 1.13e+04	+
f_{23}	**3.41e+02 ± 1.75e−07**	3.41e+02 ± 2.49e−06	+	9.90e+02 ± 1.10e+02	+
f_{24}	2.63e+02 ± 2.76e+00	**2.59e+02 ± 1.17e+00**	-	3.35e+02 ± 9.16e+00	+
f_{25}	**2.32e+02 ± 3.27e+00**	2.80e+02 ± 6.10e+00	+	2.40e+02 ± 5.97e+00	+
f_{26}	**1.01e+02 ± 4.56e−02**	1.02e+02 ± 1.43e+00	+	2.05e+02 ± 2.56e+00	+
f_{27}	**1.34e+03 ± 2.02e+02**	1.76e+03 ± 4.13e+01	+	1.83e+03 ± 8.21e+01	+
f_{28}	**2.37e+03 ± 1.89e+02**	3.14e+03 ± 6.04e+02	+	5.68e+03 ± 6.78e+02	+
f_{29}	**1.67e+04 ± 4.31e+03**	4.43e+05 ± 1.48e+05	+	5.42e+08 ± 2.78e+08	+
f_{30}	**1.54e+04 ± 1.04e+03**	8.72e+04 ± 1.41e+04	+	6.38e+06 ± 2.87e+06	+

Table 6: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/exp) for DE/rand/1/exp, DE/rand/1/bin, and DE/current-to-rand/1 on CEC2014 [32] rotated version in 100 dimensions.

	DE/rand/1/exp	DE/rand/1/bin		DE/current-to-rand/1	
f_1	**6.51e + 08 ± 6.20e + 07**	3.25e + 09 ± 2.87e + 08	+	3.04e + 09 ± 4.73e + 08	+
f_2	**3.18e + 08 ± 6.91e + 07**	1.01e + 10 ± 1.54e + 09	+	2.22e + 11 ± 1.63e + 10	+
f_3	**4.30e + 04 ± 4.07e + 03**	3.17e + 05 ± 1.55e + 04	+	2.41e + 05 ± 1.58e + 04	+
f_4	**1.10e + 03 ± 5.43e + 01**	6.25e + 03 ± 6.81e + 02	+	4.93e + 04 ± 6.68e + 03	+
f_5	**2.10e + 01 ± 2.65e − 02**	2.13e + 01 ± 2.58e − 02	+	2.13e + 01 ± 2.38e − 02	+
f_6	**1.19e + 02 ± 1.96e + 00**	1.47e + 02 ± 2.08e + 00	+	1.22e + 02 ± 2.73e + 00	+
f_7	**1.37e + 00 ± 4.10e − 02**	5.03e + 01 ± 7.04e + 00	+	2.20e + 03 ± 1.78e + 02	+
f_8	±	±		±	
f_9	**9.65e + 02 ± 2.32e + 01**	1.22e + 03 ± 2.73e + 01	+	1.08e + 03 ± 4.43e + 01	+
f_{10}	±	±		±	
f_{11}	**2.42e + 04 ± 4.43e + 02**	3.00e + 04 ± 6.06e + 02	+	2.57e + 04 ± 8.93e + 02	+
f_{12}	**1.91e + 00 ± 1.06e − 01**	3.87e + 00 ± 2.92e − 01	+	3.48e + 00 ± 1.98e − 01	+
f_{13}	**6.74e − 01 ± 4.64e − 02**	9.00e − 01 ± 5.54e − 02	+	8.03e + 00 ± 2.88e − 01	+
f_{14}	**3.39e − 01 ± 2.70e − 02**	8.49e − 01 ± 5.43e − 01	+	6.71e + 02 ± 5.09e + 01	+
f_{15}	**2.73e + 03 ± 7.70e + 02**	9.00e + 05 ± 1.77e + 05	+	4.72e + 06 ± 1.55e + 06	+
f_{16}	**4.46e + 01 ± 3.29e − 01**	4.68e + 01 ± 1.70e − 01	+	4.51e + 01 ± 3.15e − 01	+
f_{17}	**8.15e + 07 ± 1.61e + 07**	2.25e + 08 ± 4.33e + 07	+	4.77e + 08 ± 1.43e + 08	+
f_{18}	**2.37e + 04 ± 1.87e + 04**	1.02e + 05 ± 6.31e + 04	+	1.78e + 10 ± 4.05e + 09	+
f_{19}	**1.32e + 02 ± 5.53e + 00**	1.37e + 02 ± 2.46e + 00	+	3.11e + 03 ± 6.43e + 02	+
f_{20}	**9.54e + 04 ± 1.80e + 04**	2.46e + 05 ± 5.41e + 04	+	1.78e + 05 ± 3.89e + 04	+
f_{21}	**1.32e + 07 ± 7.78e + 06**	9.12e + 07 ± 1.50e + 07	+	9.81e + 07 ± 3.07e + 07	+
f_{22}	**3.15e + 03 ± 2.05e + 02**	4.34e + 03 ± 1.62e + 02	+	6.05e + 03 ± 3.33e + 03	+
f_{23}	**3.45e + 02 ± 4.69e − 01**	3.64e + 02 ± 1.32e + 00	+	1.19e + 03 ± 5.21e + 01	+
f_{24}	**4.02e + 02 ± 2.12e + 00**	4.63e + 02 ± 4.45e + 00	+	4.94e + 02 ± 1.85e + 01	+
f_{25}	3.68e + 02 ± 1.08e + 01	6.38e + 02 ± 3.43e + 01	+	**2.68e + 02 ± 6.18e + 00**	-
f_{26}	2.12e + 02 ± 7.47e + 01	3.04e + 02 ± 1.92e + 02	+	**2.09e + 02 ± 3.34e + 00**	-
f_{27}	**3.26e + 03 ± 6.44e + 01**	4.01e + 03 ± 4.86e + 01	+	3.49e + 03 ± 1.13e + 02	+
f_{28}	**9.00e + 03 ± 4.79e + 02**	1.53e + 04 ± 6.52e + 02	+	1.39e + 04 ± 9.95e + 02	+
f_{29}	**2.02e + 05 ± 3.27e + 04**	4.07e + 06 ± 7.08e + 05	+	2.68e + 09 ± 5.00e + 08	+
f_{30}	**3.64e + 05 ± 7.46e + 04**	3.41e + 06 ± 6.77e + 05	+	3.05e + 07 ± 1.65e + 07	+

in higher dimensions. On the contrary, the binomial crossover seems to be able to display a very good performance on the majority of the rotated problems under consideration. This is in our view a very interesting result: non-rotated problems appear to benefit from a moving operator that keeps a contiguous section of the solution while rotated problems benefit from keeping some variables scattered over the solution (in these experiments the quota of transferred genes is constant).

The numerical experiments indicate that the DE/current-to-rand/1 does not seem to be competitive for the entire benchmark and does not appear to improve upon the standard DE performance for the test problems considered in this article. A convincing interpretation of these results is not straightforward and appears in contradiction with the explanation provided in Section 3.2 and in several papers in the literature, e.g. [28]. Nonetheless, albeit rotation-invariant, DE/current-to-rand/1 does not take the rotation of the problem into account. This rotational invariant integrated crossover is obtained by perturbing all the design variables of a randomised quantity. This strategy can be robust since it does not have the bias of the standard crossovers. The latter reward those problems that can be solved by keeping some of the variables while perturbing the others.

4.2. Experimental results: Rotational invariant DE with separate crossover

This section displays the numerical results of the comparison of the two standard DE implementations against the corresponding DE incorporating the Gram-Schmidt orthogonalisation [49] and the eigenvector procedure from [18]. More specifically the following algorithms with respective parameters have been compared.

- RIDE/rand/1/bin with the crossover in Algorithm 6 with $Np = n$, $F = 0.7$, $Cr = 0.3$
- RIDE/rand/1/exp with the crossover in Algorithm 7 with $Np = n$, $F = 0.7$, $Cr = \frac{1}{n0.\sqrt[3]{2}}$
- eigen-DE/rand/1/bin with the crossover in Algorithm 8 and Algorithm 3 in line 7 with $Np = n$, $F = 0.7$, $Cr = 0.3$, $Pr = 1$
- eigen-DE/exp/1/bin with the crossover in Algorithm 8 and Algorithm 4 in line 7 with $Np = n$, $F = 0.7$, $Cr = \frac{1}{n0.\sqrt[3]{2}}$, $Pr = 1$

In order to perform a fair comparison (and since the difference between bin and exp logic has been analysed above), we have grouped the results by comparing directly the crossover performance within the binomial and exponential strategy, respectively.

4.2.1. Comparison of binomial crossovers

Tables 7, 8, 9 show the comparison of DE/rand/1/bin, RIDE/rand/1/bin, eigen-DE/rand/1/bin for non-rotated problems in 10, 50 and 100 variables.

Table 7: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/bin) for DE/rand/1/bin, RIDE/rand/1/bin, and eigen-DE/rand/1/bin on CEC2014 [32] non-rotated version in 10 dimensions.

	DE/rand/1/bin	RIDErand/1/bin		eigen-DE/rand/1/bin	
f_1	**0.00e + 00 ± 0.00e + 00**	9.75e + 05 ± 1.98e + 06	+	7.49e + 03 ± 3.21e + 03	+
f_2	**0.00e + 00 ± 0.00e + 00**	8.25e + 08 ± 1.16e + 09	+	2.38e + 04 ± 2.52e + 04	+
f_3	**0.00e + 00 ± 0.00e + 00**	4.76e + 03 ± 2.34e + 03	+	6.71e + 03 ± 2.44e + 03	+
f_4	**2.12e − 02 ± 6.39e − 02**	8.43e + 01 ± 1.16e + 02	+	5.36e − 01 ± 8.23e − 01	+
f_5	**1.46e + 01 ± 5.70e + 00**	1.98e + 01 ± 1.88e + 00	+	1.94e + 01 ± 2.30e + 00	+
f_6	2.41e − 04 ± 1.30e − 03	3.66e + 00 ± 1.06e + 00	+	**1.46e − 04 ± 4.49e − 04**	=
f_7	**1.06e − 02 ± 9.66e − 03**	2.67e + 01 ± 2.81e + 01	+	1.66e − 01 ± 5.60e − 02	+
f_8	±	±		±	
f_9	**1.49e + 00 ± 9.53e − 01**	1.93e + 01 ± 5.76e + 00	+	1.70e + 01 ± 4.27e + 00	+
f_{10}	±	±		±	
f_{11}	**7.43e + 00 ± 2.31e + 01**	7.53e + 02 ± 1.76e + 02	+	8.43e + 02 ± 1.89e + 02	+
f_{12}	**4.43e − 03 ± 6.00e − 03**	1.10e + 00 ± 2.11e − 01	+	1.18e + 00 ± 2.42e − 01	+
f_{13}	2.04e − 01 ± 3.96e − 02	1.03e + 00 ± 9.34e − 01	+	**1.73e − 01 ± 2.95e − 02**	-
f_{14}	1.50e − 01 ± 3.16e − 02	9.49e + 00 ± 6.58e + 00	+	**1.32e − 01 ± 3.48e − 02**	-
f_{15}	**6.53e − 01 ± 9.56e − 02**	2.05e + 02 ± 8.39e + 02	+	1.88e + 00 ± 3.18e − 01	+
f_{16}	**1.65e − 01 ± 5.41e − 02**	2.58e + 00 ± 4.09e − 01	+	2.96e + 00 ± 2.21e − 01	+
f_{17}	**5.01e + 00 ± 2.14e + 01**	1.94e + 03 ± 1.09e + 03	+	1.02e + 04 ± 7.38e + 03	+
f_{18}	**5.77e − 01 ± 5.63e − 01**	1.36e + 03 ± 1.57e + 03	+	7.49e + 03 ± 5.81e + 03	+
f_{19}	**1.53e − 02 ± 1.90e − 02**	3.00e + 00 ± 1.08e + 00	+	3.07e + 00 ± 4.79e − 01	+
f_{20}	**4.15e − 01 ± 5.22e − 01**	3.56e + 02 ± 3.87e + 02	+	2.77e + 02 ± 1.33e + 02	+
f_{21}	**7.75e − 01 ± 3.01e + 00**	1.79e + 03 ± 9.11e + 02	+	3.09e + 03 ± 1.88e + 03	+
f_{22}	**4.25e + 00 ± 2.13e + 01**	3.16e + 01 ± 3.61e + 00	+	3.37e + 01 ± 3.72e + 00	+
f_{23}	3.16e + 02 ± 8.98e − 13	3.17e + 02 ± 4.12e + 01	+	**3.16e + 02 ± 8.63e − 13**	=
f_{24}	**1.08e + 02 ± 3.64e + 00**	1.43e + 02 ± 2.69e + 01	+	1.21e + 02 ± 3.59e + 00	+
f_{25}	1.75e + 02 ± 2.47e + 01	1.90e + 02 ± 1.43e + 01	=	**1.71e + 02 ± 1.58e + 01**	=
f_{26}	1.00e + 02 ± 4.01e − 02	1.01e + 02 ± 5.97e − 01	+	**1.00e + 02 ± 2.54e − 02**	=
f_{27}	1.43e + 02 ± 1.61e + 02	1.39e + 02 ± 1.44e + 02	=	**1.57e + 01 ± 7.14e + 01**	-
f_{28}	**4.44e + 02 ± 1.31e + 01**	4.82e + 02 ± 6.22e + 01	+	4.73e + 02 ± 2.26e + 01	+
f_{29}	**2.19e + 02 ± 4.65e + 00**	3.58e + 02 ± 6.90e + 01	+	1.22e + 03 ± 5.40e + 02	+
f_{30}	**4.07e + 02 ± 9.15e + 01**	9.54e + 02 ± 2.57e + 02	+	1.27e + 03 ± 2.59e + 02	+

Tables 10, 11, 12 show the comparison of DE/rand/1/bin, RIDE/rand/1/bin, eigen-DE/rand/1/bin for rotated problems in 10, 50 and 100 variables.

Table 8: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/bin) for DE/rand/1/bin, RIDE/rand/1/bin, and eigen-DE/rand/1/bin on CEC2014 [32] non-rotated version in 50 dimensions.

	DE/rand/1/bin	RIDE/rand/1/bin		eigen-DE/rand/1/bin	
f_1	**4.18e − 09 ± 1.92e − 09**	4.24e + 07 ± 1.87e + 07	+	1.60e + 08 ± 2.76e + 07	+
f_2	**1.09e − 05 ± 3.69e − 06**	1.34e + 10 ± 4.11e + 09	+	7.45e + 03 ± 1.91e + 04	+
f_3	**2.10e − 11 ± 7.21e − 12**	6.06e + 04 ± 7.53e + 03	+	9.27e + 04 ± 8.59e + 03	+
f_4	**1.65e + 02 ± 2.70e + 01**	9.30e + 02 ± 3.68e + 02	+	4.42e + 02 ± 5.37e + 01	+
f_5	**2.06e + 01 ± 3.98e − 02**	2.12e + 01 ± 3.54e − 02	+	2.12e + 01 ± 3.07e − 02	+
f_6	**9.02e − 04 ± 1.36e − 04**	4.14e + 01 ± 2.97e + 00	+	5.82e + 01 ± 1.21e + 00	+
f_7	**4.54e − 11 ± 3.41e − 11**	1.06e + 02 ± 2.94e + 01	+	1.46e − 01 ± 6.73e − 02	+
f_8	±	±		±	
f_9	**2.39e + 02 ± 8.08e + 00**	4.04e + 02 ± 2.09e + 01	+	4.52e + 02 ± 1.46e + 01	+
f_{10}	±	±		±	
f_{11}	**7.25e + 03 ± 2.88e + 02**	1.32e + 04 ± 3.99e + 02	+	1.32e + 04 ± 3.27e + 02	+
f_{12}	**5.13e − 01 ± 4.31e − 02**	3.55e + 00 ± 2.72e − 01	+	3.55e + 00 ± 2.79e − 01	+
f_{13}	**5.96e − 01 ± 4.38e − 02**	1.55e + 00 ± 9.41e − 01	+	6.19e − 01 ± 5.18e − 02	=
f_{14}	2.98e − 01 ± 2.68e − 02	3.59e + 01 ± 1.43e + 01	+	**2.86e − 01 ± 2.43e − 02**	=
f_{15}	**2.98e + 01 ± 2.00e + 00**	1.01e + 03 ± 1.30e + 03	+	4.03e + 01 ± 2.00e + 00	+
f_{16}	**1.80e + 01 ± 3.13e − 01**	2.22e + 01 ± 1.90e − 01	+	2.22e + 01 ± 2.42e − 01	+
f_{17}	**3.27e + 02 ± 3.87e + 01**	3.33e + 05 ± 1.20e + 05	+	3.41e + 07 ± 5.91e + 06	+
f_{18}	**4.51e + 01 ± 3.48e + 00**	2.46e + 06 ± 1.95e + 06	+	1.01e + 09 ± 1.72e + 08	+
f_{19}	**2.09e + 01 ± 6.90e + 00**	1.02e + 02 ± 3.08e + 01	+	2.59e + 02 ± 2.48e + 01	+
f_{20}	**3.17e + 01 ± 3.22e + 00**	3.94e + 02 ± 1.63e + 02	+	3.26e + 04 ± 7.85e + 03	+
f_{21}	**1.43e + 01 ± 2.59e + 00**	1.07e + 05 ± 2.61e + 04	+	1.15e + 07 ± 2.57e + 06	+
f_{22}	**2.82e + 01 ± 8.94e − 01**	1.45e + 03 ± 1.57e + 02	+	2.08e + 03 ± 1.81e + 02	+
f_{23}	**3.22e + 02 ± 1.42e − 12**	3.81e + 02 ± 2.28e + 01	+	4.48e + 02 ± 1.37e + 01	+
f_{24}	2.64e + 02 ± 3.95e + 00	**2.00e + 02 ± 4.62e − 03**	-	3.11e + 02 ± 2.32e + 00	+
f_{25}	2.07e + 02 ± 7.06e − 04	**2.00e + 02 ± 9.28e − 01**	-	2.44e + 02 ± 2.48e + 00	+
f_{26}	**1.01e + 02 ± 5.12e − 02**	1.03e + 02 ± 9.95e − 01	+	1.04e + 02 ± 9.44e − 01	+
f_{27}	**3.00e + 02 ± 1.31e − 02**	1.25e + 03 ± 1.61e + 02	+	1.67e + 03 ± 1.33e + 02	+
f_{28}	**1.28e + 03 ± 3.44e + 01**	3.70e + 03 ± 5.81e + 02	+	7.71e + 03 ± 4.95e + 02	+
f_{29}	**3.38e + 02 ± 2.05e − 01**	1.64e + 06 ± 4.07e + 06	+	6.81e + 07 ± 1.18e + 07	+
f_{30}	**5.82e + 03 ± 4.29e + 01**	1.11e + 05 ± 3.05e + 04	+	1.35e + 06 ± 3.17e + 05	+

Table 9: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/bin) for DE/rand/1/bin, RIDE/rand/1/bin, and eigen-DE/rand/1/bin on CEC2014 [32] non-rotated version in 100 dimensions.

	DE/rand/1/bin	RIDE/rand/1/bin		eigen-DE/rand/1/bin	
f_1	**4.13e + 04 ± 4.40e + 03**	2.59e + 08 ± 6.32e + 07	+	1.79e + 09 ± 1.63e + 08	+
f_2	**7.73e + 08 ± 1.07e + 08**	3.31e + 10 ± 5.84e + 09	+	2.53e + 10 ± 4.22e + 09	+
f_3	**8.40e + 02 ± 1.06e + 02**	1.90e + 05 ± 1.46e + 04	+	2.53e + 05 ± 1.27e + 04	+
f_4	**3.60e + 03 ± 2.68e + 02**	4.62e + 03 ± 8.24e + 02	+	1.55e + 04 ± 1.18e + 03	+
f_5	**2.09e + 01 ± 1.72e − 02**	2.13e + 01 ± 2.68e − 02	+	2.13e + 01 ± 1.67e − 02	+
f_6	**5.03e + 01 ± 2.51e + 00**	1.16e + 02 ± 3.31e + 00	+	1.41e + 02 ± 2.49e + 00	+
f_7	**1.20e + 01 ± 1.62e + 00**	3.50e + 02 ± 6.13e + 01	+	3.46e + 02 ± 4.09e + 01	+
f_8	±	±		±	
f_9	**8.53e + 02 ± 1.95e + 01**	9.89e + 02 ± 3.88e + 01	+	1.21e + 03 ± 3.04e + 01	+
f_{10}	±	±		±	
f_{11}	**2.19e + 04 ± 4.46e + 02**	3.02e + 04 ± 4.94e + 02	+	3.03e + 04 ± 4.20e + 02	+
f_{12}	**1.16e + 00 ± 6.98e − 02**	4.12e + 00 ± 2.54e − 01	+	4.12e + 00 ± 2.22e − 01	+
f_{13}	**8.43e − 01 ± 5.40e − 02**	2.98e + 00 ± 5.62e − 01	+	3.19e + 00 ± 2.76e − 01	+
f_{14}	**4.36e − 01 ± 7.70e − 02**	1.12e + 02 ± 2.11e + 01	+	1.68e + 02 ± 1.21e + 01	+
f_{15}	8.11e + 04 ± 1.46e + 04	**2.99e + 04 ± 1.56e + 04**	-	7.89e + 05 ± 1.61e + 05	+
f_{16}	**4.25e + 01 ± 3.28e − 01**	4.63e + 01 ± 2.92e − 01	+	4.63e + 01 ± 2.11e − 01	+
f_{17}	**4.95e + 03 ± 2.06e + 02**	7.18e + 06 ± 2.05e + 06	+	2.63e + 08 ± 5.75e + 07	+
f_{18}	**3.69e + 02 ± 1.93e + 01**	2.38e + 08 ± 1.10e + 08	+	9.49e + 09 ± 8.33e + 08	+
f_{19}	**1.09e + 02 ± 1.94e + 01**	3.20e + 02 ± 3.85e + 01	+	1.50e + 03 ± 1.32e + 02	+
f_{20}	**3.22e + 02 ± 1.44e + 01**	1.05e + 05 ± 2.24e + 04	+	2.33e + 05 ± 5.29e + 04	+
f_{21}	**2.81e + 03 ± 2.07e + 02**	1.83e + 06 ± 3.16e + 05	+	1.10e + 08 ± 1.51e + 07	+
f_{22}	**5.44e + 02 ± 9.06e + 01**	4.56e + 03 ± 2.43e + 02	+	6.74e + 03 ± 3.94e + 02	+
f_{23}	**3.44e + 02 ± 1.67e − 01**	6.67e + 02 ± 5.60e + 01	+	1.31e + 03 ± 6.67e + 01	+
f_{24}	4.44e + 02 ± 2.95e + 00	**3.99e + 02 ± 2.03e + 01**	-	4.92e + 02 ± 2.96e + 00	+
f_{25}	**2.24e + 02 ± 3.30e − 01**	2.62e + 02 ± 4.64e + 00	+	4.47e + 02 ± 1.27e + 01	+
f_{26}	**1.05e + 02 ± 3.11e − 01**	2.00e + 02 ± 9.64e − 02	+	3.27e + 02 ± 3.38e + 01	+
f_{27}	**1.20e + 03 ± 3.72e + 01**	3.38e + 03 ± 1.17e + 02	+	3.93e + 03 ± 8.41e + 01	+
f_{28}	**5.14e + 03 ± 1.09e + 02**	1.98e + 04 ± 8.74e + 02	+	2.19e + 04 ± 6.97e + 02	+
f_{29}	**1.20e + 03 ± 4.49e + 01**	1.48e + 08 ± 1.05e + 08	+	1.20e + 09 ± 1.36e + 08	+
f_{30}	**4.40e + 04 ± 1.57e + 03**	1.15e + 06 ± 1.81e + 05	+	1.09e + 07 ± 1.70e + 06	+

Table 10: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/bin) for DE/rand/1/bin, RIDE/rand/1/bin, and eigen-DE/rand/1/bin on CEC2014 [32] rotated version in 10 dimensions.

	DE/rand/1/bin	RIDE/rand/1/bin		eigen-DE/rand/1/bin	
f_1	**3.44e+02 ± 3.89e+02**	5.47e+06 ± 7.28e+06	+	1.36e+06 ± 8.35e+05	+
f_2	**0.00e+00 ± 0.00e+00**	1.55e+09 ± 1.70e+09	+	2.02e+04 ± 1.65e+04	+
f_3	**0.00e+00 ± 3.11e−14**	4.13e+03 ± 3.07e+03	+	8.06e+03 ± 2.53e+03	+
f_4	1.16e+01 ± 1.53e+01	1.12e+02 ± 9.01e+01	+	**1.15e+01 ± 1.56e+01**	=
f_5	**1.99e+01 ± 1.24e+00**	2.01e+01 ± 1.58e+00	+	1.99e+01 ± 1.66e+00	+
f_6	**2.98e−02 ± 1.61e−01**	3.19e+00 ± 8.12e−01	+	2.70e+00 ± 1.26e+00	+
f_7	**9.18e−02 ± 4.62e−02**	3.19e+01 ± 3.05e+01	+	1.63e−01 ± 4.41e−02	+
f_8	±	±		±	
f_9	**8.45e+00 ± 2.24e+00**	2.16e+01 ± 7.51e+00	+	1.65e+01 ± 2.90e+00	+
f_{10}	±	±		±	
f_{11}	**4.84e+02 ± 1.51e+02**	8.38e+02 ± 2.03e+02	+	9.16e+02 ± 1.76e+02	+
f_{12}	**4.89e−01 ± 8.04e−02**	1.20e+00 ± 2.18e−01	+	1.19e+00 ± 2.46e−01	+
f_{13}	1.84e−01 ± 4.10e−02	8.73e−01 ± 8.67e−01	+	**1.78e−01 ± 3.23e−02**	=
f_{14}	**1.57e−01 ± 3.81e−02**	7.41e+00 ± 6.14e+00	+	1.60e−01 ± 3.41e−02	=
f_{15}	**1.36e+00 ± 2.95e−01**	1.29e+02 ± 3.16e+02	+	1.93e+00 ± 4.07e−01	+
f_{16}	**2.38e+00 ± 2.60e−01**	2.61e+00 ± 3.36e−01	+	2.97e+00 ± 2.88e−01	+
f_{17}	**2.06e+02 ± 5.58e+02**	5.85e+02 ± 3.00e+02	+	1.16e+04 ± 6.02e+03	+
f_{18}	**2.03e+00 ± 8.77e−01**	2.23e+02 ± 1.12e+02	+	1.20e+04 ± 1.38e+04	+
f_{19}	**8.80e−02 ± 8.12e−02**	2.23e+00 ± 9.06e−01	+	3.24e+00 ± 5.59e−01	+
f_{20}	**5.56e−01 ± 5.90e−01**	7.16e+02 ± 9.76e+02	+	9.99e+02 ± 9.81e+02	+
f_{21}	**1.51e+00 ± 4.20e+00**	2.55e+03 ± 4.26e+03	+	3.29e+03 ± 1.44e+03	+
f_{22}	**3.05e+00 ± 6.20e+00**	4.21e+01 ± 2.30e+01	+	4.35e+01 ± 8.17e+00	+
f_{23}	3.29e+02 ± 9.09e−13	**3.26e+02 ± 4.27e+01**	+	3.29e+02 ± 9.03e−08	+
f_{24}	**1.18e+02 ± 3.56e+00**	1.47e+02 ± 3.02e+01	+	1.22e+02 ± 4.46e+00	+
f_{25}	**1.61e+02 ± 2.51e+01**	1.80e+02 ± 2.37e+01	+	1.68e+02 ± 2.24e+01	=
f_{26}	**1.00e+02 ± 3.75e−02**	1.00e+02 ± 7.29e−01	+	1.00e+02 ± 3.83e−02	=
f_{27}	1.85e+02 ± 1.60e+02	2.29e+02 ± 1.82e+02	+	**6.24e+01 ± 1.35e+02**	-
f_{28}	**3.76e+02 ± 1.99e+01**	4.75e+02 ± 9.32e+01	+	4.77e+02 ± 4.83e+01	+
f_{29}	**3.37e+02 ± 6.01e+01**	5.35e+02 ± 4.62e+02	+	1.57e+03 ± 1.04e+03	+
f_{30}	**5.65e+02 ± 4.72e+01**	1.30e+03 ± 4.13e+02	+	1.56e+03 ± 3.14e+02	+

Table 11: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/bin) for DE/rand/1/bin, RIDE/rand/1/bin, and eigen-DE/rand/1/bin on CEC2014 [32] rotated version in 50 dimensions.

	RIDE/rand/1/bin	RIDE/rand/1/bin		eigen-DE/rand/1/bin	
f_1	4.23e+08 ± 6.49e+07	**1.27e+08 ± 5.07e+07**	-	7.30e+08 ± 1.27e+08	+
f_2	2.94e+06 ± 2.49e+06	1.87e+10 ± 5.43e+09	+	**1.09e+06 ± 2.75e+06**	-
f_3	**3.29e+04 ± 5.73e+03**	9.59e+04 ± 1.12e+04	+	1.46e+05 ± 1.68e+04	+
f_4	**1.71e+02 ± 3.02e+01**	2.08e+03 ± 7.53e+02	+	4.64e+02 ± 5.88e+01	+
f_5	**2.11e+01 ± 4.12e−02**	2.11e+01 ± 3.41e−02	+	2.11e+01 ± 4.01e−02	+
f_6	6.16e+01 ± 1.29e+00	**4.23e+01 ± 4.05e+00**	-	6.61e+01 ± 1.53e+00	+
f_7	**1.18e−01 ± 5.36e−02**	2.08e+02 ± 4.01e+01	+	7.10e−01 ± 4.82e−02	+
f_8	±	±		±	
f_9	**4.18e+02 ± 1.19e+01**	4.29e+02 ± 2.37e+01	+	4.69e+02 ± 1.79e+01	+
f_{10}	±	±		±	
f_{11}	**1.25e+04 ± 3.68e+02**	1.35e+04 ± 3.66e+02	+	1.36e+04 ± 3.69e+02	+
f_{12}	**2.54e+00 ± 2.99e−01**	3.47e+00 ± 3.10e−01	+	3.53e+00 ± 3.22e−01	+
f_{13}	**6.42e−01 ± 6.01e−02**	2.59e+00 ± 8.19e−01	+	6.48e−01 ± 5.68e−02	=
f_{14}	**3.02e−01 ± 3.42e−02**	4.89e+01 ± 1.54e+01	+	3.27e−01 ± 3.44e−02	+
f_{15}	**3.98e+01 ± 1.97e+00**	3.88e+03 ± 3.32e+03	+	5.20e+01 ± 6.13e+00	+
f_{16}	**2.23e+01 ± 1.66e−01**	2.26e+01 ± 2.01e−01	+	2.26e+01 ± 1.42e−01	+
f_{17}	1.62e+07 ± 3.02e+06	**1.70e+06 ± 1.27e+06**	-	4.58e+07 ± 9.59e+06	+
f_{18}	**1.01e+04 ± 4.92e+03**	3.13e+06 ± 2.72e+06	+	9.93e+08 ± 2.79e+08	+
f_{19}	**4.76e+01 ± 6.66e+00**	8.94e+01 ± 2.32e+01	+	2.42e+02 ± 2.65e+01	+
f_{20}	**1.59e+04 ± 5.06e+03**	3.16e+04 ± 8.90e+03	+	6.42e+04 ± 1.31e+04	+
f_{21}	5.08e+06 ± 1.31e+06	**3.88e+05 ± 2.48e+05**	-	1.33e+07 ± 3.92e+06	+
f_{22}	**9.13e+02 ± 1.30e+02**	1.76e+03 ± 1.45e+02	+	2.43e+03 ± 2.05e+02	+
f_{23}	**3.41e+02 ± 2.49e−06**	4.51e+02 ± 3.26e+01	+	5.32e+02 ± 2.21e+01	+
f_{24}	2.59e+02 ± 1.17e+00	**2.00e+02 ± 1.38e−02**	-	2.99e+02 ± 1.69e+00	+
f_{25}	2.80e+02 ± 6.10e+00	**2.00e+02 ± 4.28e−01**	-	3.13e+02 ± 8.97e+00	+
f_{26}	**1.02e+02 ± 1.43e+00**	1.09e+02 ± 2.43e+01	=	1.05e+02 ± 1.20e+00	+
f_{27}	1.76e+03 ± 4.13e+01	**1.48e+03 ± 1.49e+02**	-	1.96e+03 ± 5.25e+01	+
f_{28}	**3.14e+03 ± 6.04e+02**	5.27e+03 ± 8.84e+02	+	8.98e+03 ± 5.32e+02	+
f_{29}	**4.43e+05 ± 1.48e+05**	3.21e+06 ± 3.83e+06	+	1.63e+08 ± 3.51e+07	+
f_{30}	**8.72e+04 ± 1.41e+04**	2.03e+05 ± 1.03e+05	+	1.97e+06 ± 4.38e+05	+

Table 12: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/bin) for DE/rand/1/bin, RIDE/rand/1/bin, and eigen-DE/rand/1/bin on CEC2014 [32] rotated version in 100 dimensions.

	DE/rand/1/bin	RIDE/rand/1/bin		eigen-DE/rand/1/bin	
f_1	3.25e + 09 ± 2.87e + 08	**1.51e + 09 ± 1.89e + 08**	-	4.20e + 09 ± 4.22e + 08	+
f_2	**1.01e + 10 ± 1.54e + 09**	6.40e + 10 ± 1.06e + 10	+	9.21e + 10 ± 6.52e + 09	+
f_3	3.17e + 05 ± 1.55e + 04	**2.87e + 05 ± 2.69e + 04**	-	3.83e + 05 ± 2.57e + 04	+
f_4	**6.25e + 03 ± 6.81e + 02**	8.21e + 03 ± 1.49e + 03	+	2.63e + 04 ± 2.33e + 03	+
f_5	**2.13e + 01 ± 2.58e − 02**	2.13e + 01 ± 2.57e − 02	=	2.13e + 01 ± 2.56e − 02	=
f_6	1.47e + 02 ± 2.08e + 00	**1.30e + 02 ± 4.76e + 00**	-	1.53e + 02 ± 1.81e + 00	+
f_7	**5.03e + 01 ± 7.04e + 00**	5.95e + 02 ± 8.19e + 01	+	9.29e + 02 ± 6.37e + 01	+
f_8	±	±		±	
f_9	1.22e + 03 ± 2.73e + 01	**1.08e + 03 ± 3.82e + 01**	-	1.33e + 03 ± 2.87e + 01	+
f_{10}	±	±		±	
f_{11}	**3.00e + 04 ± 6.06e + 02**	3.06e + 04 ± 5.20e + 02	+	3.08e + 04 ± 4.03e + 02	+
f_{12}	**3.87e + 00 ± 2.92e − 01**	4.13e + 00 ± 2.40e − 01	+	4.19e + 00 ± 2.18e − 01	+
f_{13}	**9.00e − 01 ± 5.54e − 02**	4.03e + 00 ± 3.61e − 01	+	5.24e + 00 ± 1.93e − 01	+
f_{14}	**8.49e − 01 ± 5.43e − 01**	1.82e + 02 ± 2.46e + 01	+	3.08e + 02 ± 1.58e + 01	+
f_{15}	9.00e + 05 ± 1.77e + 05	**2.73e + 05 ± 1.51e + 05**	-	4.35e + 06 ± 9.73e + 05	+
f_{16}	4.68e + 01 ± 1.70e − 01	4.69e + 01 ± 2.01e − 01	+	**4.67e + 01 ± 2.47e − 01**	=
f_{17}	2.25e + 08 ± 4.33e + 07	**4.71e + 07 ± 1.61e + 07**	-	3.92e + 08 ± 5.42e + 07	+
f_{18}	**1.02e + 05 ± 6.31e + 04**	2.26e + 08 ± 9.46e + 07	+	8.96e + 09 ± 1.06e + 09	+
f_{19}	**1.37e + 02 ± 2.46e + 00**	3.65e + 02 ± 4.23e + 01	+	1.67e + 03 ± 1.49e + 02	+
f_{20}	2.46e + 05 ± 5.41e + 04	**1.90e + 05 ± 3.98e + 04**	-	4.14e + 05 ± 8.38e + 04	+
f_{21}	9.12e + 07 ± 1.50e + 07	**7.29e + 06 ± 3.27e + 06**	-	1.56e + 08 ± 3.06e + 07	+
f_{22}	**4.34e + 03 ± 1.62e + 02**	4.89e + 03 ± 2.17e + 02	+	6.87e + 03 ± 5.16e + 02	+
f_{23}	**3.64e + 02 ± 1.32e + 00**	6.81e + 02 ± 5.30e + 01	+	1.40e + 03 ± 8.34e + 01	+
f_{24}	4.63e + 02 ± 4.45e + 00	**4.24e + 02 ± 2.54e + 01**	-	5.33e + 02 ± 5.54e + 00	+
f_{25}	6.38e + 02 ± 3.43e + 01	**3.62e + 02 ± 1.86e + 01**	-	6.85e + 02 ± 4.03e + 01	+
f_{26}	3.04e + 02 ± 1.92e + 02	**2.41e + 02 ± 8.40e + 00**	=	5.14e + 02 ± 2.57e + 01	+
f_{27}	4.01e + 03 ± 4.86e + 01	**3.56e + 03 ± 1.50e + 02**	-	4.36e + 03 ± 5.73e + 01	+
f_{28}	**1.53e + 04 ± 6.52e + 02**	2.28e + 04 ± 1.24e + 03	+	2.54e + 04 ± 8.80e + 02	+
f_{29}	**4.07e + 06 ± 7.08e + 05**	3.10e + 08 ± 1.54e + 08	+	1.71e + 09 ± 2.12e + 08	+
f_{30}	3.41e + 06 ± 6.77e + 05	**3.07e + 06 ± 7.61e + 05**	=	3.21e + 07 ± 5.82e + 06	+

Numerical results on the binomial crossover show that the standard DE/rand/bin displays on average a better performance than that if its competitors. Nevertheless, in some cases, especially for rotated problems in 100 dimensions, RIDE/rand/1/bin outperforms DE/rand/1/bin. However, in the remaining cases, RIDE/rand/1/bin displays a significantly poorer performance than that of its standard counterpart. The eigen-DE/rand/1/bin does not appear to be competitive to the standard DE/rand/1/bin for almost all problems under consideration.

4.2.2. Comparison of exponential crossovers

Tables 13, 14, 15 show the comparison of DE/rand/1/exp, RIDE/rand/1/exp, eigen-DE/rand/1/exp for non-rotated problems in 10, 50 and 100 variables.

Tables 16, 17, 18 show the comparison of DE/rand/1/exp, RIDE/rand/1/exp, eigen-DE/rand/1/exp for rotated problems in 10, 50 and 100 variables.

Numerical results on the exponential crossover appear to be substantially different from those on the binomial crossover. While still on the majority of problems the standard DE/rand/1/exp displays the best performance, RIDE/rand/1/exp and eigen-DE/rand/1/exp display quite a performance, especially for rotated high dimensional problems. It can be observed that RIDE/rand/1/exp outperforms its competitors on some problems, see Table 18, while for the remaining cases is outperformed even by orders of magnitude by the standard DE/rand/1/exp. Conversely, eigen-DE/ran/1/exp displays a robust behaviour by consistently achieving a good performance which is often only slightly worse than DE/rand/1/exp.

Table 13: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/exp) for DE/rand/1/exp, RIDE/rand/1/exp, and eigen-DE/rand/1/exp on CEC2014 [32] non-rotated version in 10 dimensions.

	DE/rand/1/exp	RIDE/rand/1/exp		eigen-DE/rand/1/exp	
f_{1}	**9.21e+03 ± 1.97e+04**	3.67e+06 ± 4.16e+06	+	1.34e+04 ± 1.32e+04	+
f_{2}	**0.00e+00 ± 2.32e−14**	9.50e+08 ± 1.54e+09	+	1.32e+03 ± 1.80e+03	+
f_{3}	**3.88e+00 ± 1.93e+01**	3.84e+03 ± 3.07e+03	+	5.88e+00 ± 6.72e+00	+
f_{4}	1.44e+01 ± 1.67e+01	7.78e+01 ± 3.78e+01	+	**1.28e+01 ± 1.58e+01**	=
f_{5}	2.01e+01 ± 2.50e−02	**1.94e+01 ± 3.16e+00**	+	2.01e+01 ± 7.62e−01	=
f_{6}	**1.17e−01 ± 3.04e−01**	3.75e+00 ± 1.06e+00	+	5.86e−01 ± 1.09e+00	=
f_{7}	**4.58e−02 ± 2.34e−02**	2.90e+01 ± 2.11e+01	+	7.33e−02 ± 3.62e−02	+
f_{8}	±	±		±	
f_{9}	**7.20e+00 ± 2.39e+00**	1.98e+01 ± 8.45e+00	+	1.23e+01 ± 3.54e+00	+
f_{10}	±	±		±	
f_{11}	**3.73e+02 ± 1.20e+02**	7.17e+02 ± 2.02e+02	+	6.54e+02 ± 2.22e+02	+
f_{12}	**3.76e−01 ± 6.26e−02**	9.18e−01 ± 2.22e−01	+	5.48e−01 ± 3.54e−01	+
f_{13}	**1.77e−01 ± 5.47e−02**	1.10e+00 ± 9.45e−01	+	1.80e−01 ± 5.39e−02	=
f_{14}	**1.58e−01 ± 5.24e−02**	7.80e+00 ± 8.41e+00	+	1.74e−01 ± 5.15e−02	=
f_{15}	**1.06e+00 ± 2.45e−01**	2.25e+01 ± 3.48e+01	+	1.39e+00 ± 3.21e−01	+
f_{16}	**2.32e+00 ± 3.06e−01**	2.70e+00 ± 3.09e−01	+	2.93e+00 ± 3.04e−01	+
f_{17}	**2.80e+02 ± 6.65e+02**	5.61e+02 ± 2.94e+02	+	2.89e+02 ± 1.97e+02	+
f_{18}	**5.44e+00 ± 8.62e+00**	3.01e+02 ± 2.96e+02	+	4.48e+01 ± 2.86e+01	+
f_{19}	**1.60e−01 ± 1.48e−01**	2.23e+00 ± 8.28e−01	+	1.57e+00 ± 5.30e−01	+
f_{20}	**1.10e+00 ± 1.10e+00**	7.11e+02 ± 1.09e+03	+	3.74e+01 ± 2.45e+01	+
f_{21}	**1.78e+01 ± 2.93e+01**	2.08e+03 ± 1.17e+03	+	1.48e+02 ± 2.28e+02	+
f_{22}	**9.93e+00 ± 2.94e+01**	3.63e+01 ± 1.23e+01	+	2.51e+01 ± 7.18e+00	+
f_{23}	3.29e+02 ± 8.98e−13	**2.96e+02 ± 6.80e+01**	+	3.29e+02 ± 1.14e−12	+
f_{24}	**1.17e+02 ± 3.78e+00**	1.44e+02 ± 2.66e+01	+	1.17e+02 ± 4.12e+00	=
f_{25}	**1.59e+02 ± 3.47e+01**	1.91e+02 ± 1.67e+01	+	1.66e+02 ± 3.54e+01	=
f_{26}	1.00e+02 ± 5.36e−02	1.00e+02 ± 4.89e−01	+	**1.00e+02 ± 6.11e−02**	=
f_{27}	2.53e+02 ± 1.69e+02	1.88e+02 ± 1.83e+02	=	**1.25e+02 ± 1.76e+02**	-
f_{28}	**3.64e+02 ± 7.49e+01**	4.18e+02 ± 1.60e+02	+	4.39e+02 ± 5.51e+01	+
f_{29}	**2.51e+02 ± 4.32e+01**	4.59e+02 ± 1.15e+02	+	3.13e+02 ± 6.99e+01	+
f_{30}	**5.68e+02 ± 9.53e+01**	1.26e+03 ± 3.79e+02	+	8.34e+02 ± 2.10e+02	+

Table 14: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/exp) for DE/rand/1/exp, RIDE/rand/1/exp, and eigen-DE/rand/1/exp on CEC2014 [32] non-rotated version in 50 dimensions.

	DE/rand/1/exp	RIDE/rand/1/exp		eigen-DE/rand/1/exp	
f_{1}	**3.63e−08 ± 1.36e−08**	3.26e+07 ± 1.57e+07	+	8.91e+05 ± 3.93e+05	+
f_{2}	**1.64e−05 ± 5.39e−06**	1.17e+10 ± 4.73e+09	+	2.08e+06 ± 1.85e+06	+
f_{3}	**2.52e−11 ± 1.15e−11**	5.35e+04 ± 6.06e+03	+	2.87e+03 ± 1.31e+03	+
f_{4}	**3.20e+01 ± 6.56e+00**	1.18e+03 ± 5.23e+02	+	2.42e+02 ± 2.14e+01	+
f_{5}	**2.01e+01 ± 1.53e−02**	2.11e+01 ± 4.06e−02	+	2.11e+01 ± 2.79e−01	+
f_{6}	**2.80e−03 ± 3.15e−04**	3.95e+01 ± 3.91e+00	+	4.79e+01 ± 2.54e+00	+
f_{7}	**2.47e−04 ± 1.33e−03**	1.12e+02 ± 3.82e+01	+	2.68e−03 ± 8.96e−04	+
f_{8}	±	±		±	
f_{9}	**5.87e+01 ± 4.50e+00**	3.15e+02 ± 1.68e+01	+	3.80e+02 ± 1.99e+01	+
f_{10}	±	±		±	
f_{11}	**1.35e+03 ± 1.98e+02**	1.12e+04 ± 4.26e+02	+	1.11e+04 ± 3.38e+02	+
f_{12}	**1.20e−01 ± 1.37e−02**	3.14e+00 ± 2.31e−01	+	2.83e+00 ± 5.50e−01	+
f_{13}	**5.32e−01 ± 5.29e−02**	1.14e+00 ± 8.83e−01	+	5.34e−01 ± 4.12e−02	=
f_{14}	2.81e−01 ± 2.69e−02	3.60e+01 ± 1.75e+01	+	**2.78e−01 ± 2.16e−02**	=
f_{15}	**1.02e+01 ± 7.65e−01**	1.15e+03 ± 1.01e+03	+	3.89e+01 ± 1.76e+00	+
f_{16}	**7.69e+00 ± 4.79e−01**	2.18e+01 ± 2.56e−01	+	2.16e+01 ± 5.86e−01	+
f_{17}	**1.74e+01 ± 3.95e+00**	3.91e+05 ± 1.59e+05	+	1.01e+04 ± 3.46e+03	+
f_{18}	**1.00e+00 ± 2.69e−01**	1.80e+06 ± 8.83e+05	+	2.33e+03 ± 3.07e+03	+
f_{19}	**2.22e+01 ± 9.17e−01**	8.27e+01 ± 2.45e+01	+	4.35e+01 ± 3.78e+00	+
f_{20}	**4.07e+00 ± 6.00e−01**	3.86e+02 ± 1.42e+02	+	2.67e+02 ± 7.09e+01	+
f_{21}	**1.01e+01 ± 1.39e+00**	1.28e+05 ± 4.24e+04	+	4.27e+03 ± 8.94e+02	+
f_{22}	**2.33e+01 ± 2.73e−01**	8.13e+02 ± 1.53e+02	+	8.95e+02 ± 1.72e+02	+
f_{23}	**3.22e+02 ± 1.54e−12**	3.49e+02 ± 5.02e+01	+	3.26e+02 ± 1.91e+00	+
f_{24}	2.67e+02 ± 3.51e+00	**2.00e+02 ± 5.69e−13**	-	3.07e+02 ± 2.21e+00	+
f_{25}	2.07e+02 ± 9.13e−03	**2.00e+02 ± 0.00e+00**	-	2.17e+02 ± 8.33e−01	+
f_{26}	1.01e+02 ± 4.22e−02	1.48e+02 ± 4.83e+01	+	**1.01e+02 ± 5.11e−02**	=
f_{27}	**3.03e+02 ± 1.79e+01**	1.19e+03 ± 8.51e+01	+	1.20e+03 ± 3.74e+02	+
f_{28}	**1.11e+03 ± 9.59e+01**	2.00e+03 ± 1.32e+03	+	5.03e+03 ± 3.58e+02	+
f_{29}	**3.38e+02 ± 1.96e−01**	2.71e+06 ± 1.28e+07	+	1.37e+05 ± 1.36e+05	+
f_{30}	**5.73e+03 ± 7.31e+00**	1.01e+05 ± 3.63e+04	+	3.70e+04 ± 5.79e+03	+

Table 15: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/exp) for DE/rand/1/exp, RIDE/rand/1/exp, and eigen-DE/rand/1/exp on CEC2014 [32] non-rotated version in 100 dimensions.

	DE/rand/1/exp	RIDE/rand/1/exp		eigen-DE/rand/1/exp	
f_1	**6.85e + 03 ± 1.02e + 03**	1.86e + 08 ± 4.40e + 07	+	1.25e + 08 ± 2.06e + 07	+
f_2	**4.19e + 06 ± 5.37e + 05**	2.95e + 10 ± 6.29e + 09	+	1.79e + 07 ± 4.59e + 06	+
f_3	**5.34e + 00 ± 6.83e − 01**	1.33e + 05 ± 7.89e + 03	+	1.37e + 05 ± 3.50e + 04	+
f_4	**2.66e + 02 ± 9.32e + 00**	3.08e + 03 ± 9.47e + 02	+	2.63e + 03 ± 2.04e + 02	+
f_5	**2.04e + 01 ± 1.76e − 02**	2.13e + 01 ± 2.72e − 02	+	2.13e + 01 ± 1.47e − 02	+
f_6	**1.10e + 01 ± 5.71e − 01**	9.90e + 01 ± 4.73e + 00	+	1.24e + 02 ± 2.75e + 00	+
f_7	**1.03e + 00 ± 2.14e − 02**	3.15e + 02 ± 6.37e + 01	+	1.09e + 00 ± 1.36e − 02	+
f_8	±	±		±	
f_9	**3.95e + 02 ± 1.76e + 01**	8.51e + 02 ± 3.48e + 01	+	1.09e + 03 ± 2.39e + 01	+
f_{10}	±	±		±	
f_{11}	**1.16e + 04 ± 4.48e + 02**	2.74e + 04 ± 7.51e + 02	+	2.76e + 04 ± 5.31e + 02	+
f_{12}	**3.28e − 01 ± 2.84e − 02**	3.74e + 00 ± 2.98e − 01	+	3.79e + 00 ± 3.24e − 01	+
f_{13}	6.46e − 01 ± 3.46e − 02	1.71e + 00 ± 1.09e + 00	+	**6.43e − 01 ± 4.37e − 02**	=
f_{14}	**3.14e − 01 ± 2.72e − 02**	8.37e + 01 ± 1.94e + 01	+	3.25e − 01 ± 3.01e − 02	=
f_{15}	**5.44e + 01 ± 1.77e + 00**	4.79e + 03 ± 2.65e + 03	+	2.38e + 04 ± 8.36e + 03	+
f_{16}	**3.17e + 01 ± 6.29e − 01**	4.59e + 01 ± 3.26e − 01	+	4.60e + 01 ± 3.24e − 01	+
f_{17}	**1.34e + 03 ± 1.29e + 02**	7.21e + 06 ± 1.95e + 06	+	3.50e + 06 ± 9.56e + 05	+
f_{18}	**1.24e + 02 ± 7.94e + 00**	6.98e + 07 ± 3.51e + 07	+	1.79e + 08 ± 6.49e + 07	+
f_{19}	**9.40e + 01 ± 1.48e + 01**	2.80e + 02 ± 3.76e + 01	+	4.77e + 02 ± 4.33e + 01	+
f_{20}	**9.82e + 01 ± 5.39e + 00**	6.16e + 04 ± 6.79e + 03	+	3.79e + 04 ± 1.26e + 04	+
f_{21}	**5.46e + 02 ± 5.80e + 01**	2.04e + 06 ± 5.35e + 05	+	6.25e + 05 ± 1.39e + 05	+
f_{22}	**6.69e + 01 ± 6.97e + 00**	3.22e + 03 ± 2.38e + 02	+	3.66e + 03 ± 3.48e + 02	+
f_{23}	**3.42e + 02 ± 2.19e − 02**	4.41e + 02 ± 1.12e + 02	+	5.04e + 02 ± 1.26e + 01	+
f_{24}	3.73e + 02 ± 2.45e + 00	**2.00e + 02 ± 2.34e − 08**	-	4.71e + 02 ± 3.61e + 00	+
f_{25}	2.15e + 02 ± 1.00e − 01	**2.00e + 02 ± 0.00e + 00**	-	2.86e + 02 ± 4.35e + 00	+
f_{26}	**1.01e + 02 ± 3.93e − 02**	2.00e + 02 ± 0.00e + 00	+	2.03e + 02 ± 2.23e + 01	+
f_{27}	**5.48e + 02 ± 8.91e + 00**	2.81e + 03 ± 1.94e + 02	+	3.36e + 03 ± 6.46e + 01	+
f_{28}	4.19e + 03 ± 1.64e + 02	**4.19e + 03 ± 3.59e + 03**	=	1.69e + 04 ± 9.70e + 02	+
f_{29}	**1.44e + 03 ± 5.68e + 01**	2.71e + 08 ± 2.34e + 08	+	1.21e + 08 ± 2.16e + 07	+
f_{30}	**2.30e + 04 ± 2.49e + 02**	6.12e + 05 ± 1.39e + 05	+	1.72e + 06 ± 3.33e + 05	+

Table 16: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/exp) for DE/rand/1/exp, RIDE/rand/1/exp, and eigen-DE/rand/1/exp on CEC2014 [32] rotated version in 10 dimensions.

	DE/rand/1/exp	RIDE/rand/1/exp		eigen-DE/rand/1/exp	
f_1	**9.21e + 03 ± 1.97e + 04**	3.67e + 06 ± 4.16e + 06	+	1.34e + 04 ± 1.32e + 04	+
f_2	**0.00e + 00 ± 2.32e − 14**	9.50e + 08 ± 1.54e + 09	+	1.32e + 03 ± 1.80e + 03	+
f_3	**3.88e + 00 ± 1.93e + 01**	3.84e + 03 ± 3.07e + 03	+	5.88e + 00 ± 6.72e + 00	+
f_4	1.44e + 01 ± 1.67e + 01	7.78e + 01 ± 3.78e + 01	+	**1.28e + 01 ± 1.58e + 01**	=
f_5	2.01e + 01 ± 2.50e − 02	**1.94e + 01 ± 3.16e + 00**	+	2.01e + 01 ± 7.62e − 01	=
f_6	**1.17e − 01 ± 3.04e − 01**	3.75e + 00 ± 1.06e + 00	+	5.86e − 01 ± 1.09e + 00	=
f_7	**4.58e − 02 ± 2.34e − 02**	2.90e + 01 ± 2.11e + 01	+	7.33e − 02 ± 3.62e − 02	+
f_8	±	±		±	
f_9	**7.20e + 00 ± 2.39e + 00**	1.98e + 01 ± 8.45e + 00	+	1.23e + 01 ± 3.54e + 00	+
f_{10}	±	±		±	
f_{11}	**3.73e + 02 ± 1.20e + 02**	7.17e + 02 ± 2.02e + 02	+	6.54e + 02 ± 2.22e + 02	+
f_{12}	**3.76e − 01 ± 6.26e − 02**	9.18e − 01 ± 2.22e − 01	+	5.48e − 01 ± 3.54e − 01	+
f_{13}	**1.77e − 01 ± 5.47e − 02**	1.10e + 00 ± 9.45e − 01	+	1.80e − 01 ± 5.39e − 02	=
f_{14}	**1.58e − 01 ± 5.24e − 02**	7.80e + 00 ± 8.41e + 00	+	1.74e − 01 ± 5.15e − 02	=
f_{15}	**1.06e + 00 ± 2.45e − 01**	2.25e + 01 ± 3.48e + 01	+	1.39e + 00 ± 3.21e − 01	+
f_{16}	**2.32e + 00 ± 3.06e − 01**	2.70e + 00 ± 3.09e − 01	+	2.93e + 00 ± 3.04e − 01	+
f_{17}	**2.80e + 02 ± 6.65e + 02**	5.61e + 02 ± 2.94e + 02	+	2.89e + 02 ± 1.97e + 02	+
f_{18}	**5.44e + 00 ± 8.62e + 00**	3.01e + 02 ± 2.96e + 02	+	4.48e + 01 ± 2.86e + 01	+
f_{19}	**1.60e − 01 ± 1.48e − 01**	2.23e + 00 ± 8.28e − 01	+	1.57e + 00 ± 5.30e − 01	+
f_{20}	**1.10e + 00 ± 1.10e + 00**	7.11e + 02 ± 1.09e + 03	+	3.74e + 01 ± 2.45e + 01	+
f_{21}	**1.78e + 01 ± 2.93e + 01**	2.08e + 03 ± 1.17e + 03	+	1.48e + 02 ± 2.28e + 02	+
f_{22}	**9.93e + 00 ± 2.94e + 01**	3.63e + 01 ± 1.23e + 01	+	2.51e + 01 ± 7.18e + 00	+
f_{23}	3.29e + 02 ± 8.98e − 13	**2.96e + 02 ± 6.80e + 01**	+	3.29e + 02 ± 1.14e − 12	+
f_{24}	**1.17e + 02 ± 3.78e + 00**	1.44e + 02 ± 2.66e + 01	+	1.17e + 02 ± 4.12e + 00	=
f_{25}	**1.59e + 02 ± 3.47e + 01**	1.91e + 02 ± 1.67e + 01	+	1.66e + 02 ± 3.54e + 01	=
f_{26}	1.00e + 02 ± 5.36e − 02	1.00e + 02 ± 4.89e − 01	+	**1.00e + 02 ± 6.11e − 02**	=
f_{27}	2.53e + 02 ± 1.69e + 02	1.88e + 02 ± 1.83e + 02	=	**1.25e + 02 ± 1.76e + 02**	-
f_{28}	**3.64e + 02 ± 7.49e + 01**	4.18e + 02 ± 1.60e + 02	+	4.39e + 02 ± 5.51e + 01	+
f_{29}	**2.51e + 02 ± 4.32e + 01**	4.59e + 02 ± 1.15e + 02	+	3.13e + 02 ± 6.99e + 01	+
f_{30}	**5.68e + 02 ± 9.53e + 01**	1.26e + 03 ± 3.79e + 02	+	8.34e + 02 ± 2.10e + 02	+

Table 17: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/exp) for DE/rand/1/exp, RIDE/rand/1/exp, and eigen-DE/rand/1/exp on CEC2014 [32] rotated version in 50 dimensions.

	DE/rand/1/exp	RIDE/rand/1/exp		eigen-DE/rand/1/exp	
f_1	7.90e+07 ± 1.61e+07	5.88e+07 ± 2.88e+07	-	**5.45e+07 ± 5.35e+07**	-
f_2	**6.55e+04 ± 3.78e+04**	2.06e+10 ± 5.51e+09	+	3.62e+05 ± 5.48e+05	+
f_3	**3.66e+01 ± 2.25e+01**	7.78e+04 ± 7.35e+03	+	4.98e+03 ± 2.70e+03	+
f_4	**1.45e+02 ± 2.59e+01**	2.18e+03 ± 8.29e+02	+	2.77e+02 ± 3.26e+01	+
f_5	**2.07e+01 ± 3.87e−02**	2.11e+01 ± 3.54e−02	+	2.11e+01 ± 2.46e−01	+
f_6	4.52e+01 ± 1.66e+00	**4.32e+01 ± 2.92e+00**	-	5.61e+01 ± 2.12e+00	+
f_7	**8.07e−03 ± 5.23e−03**	2.08e+02 ± 4.90e+01	+	1.78e−01 ± 5.73e−02	+
f_8	±	±		±	
f_9	**2.89e+02 ± 1.58e+01**	3.70e+02 ± 2.45e+01	+	4.16e+02 ± 2.30e+01	+
f_{10}	±	±		±	
f_{11}	**9.05e+03 ± 3.60e+02**	1.18e+04 ± 4.64e+02	+	1.19e+04 ± 4.90e+02	+
f_{12}	**1.11e+00 ± 1.07e−01**	3.02e+00 ± 3.23e−01	+	2.84e+00 ± 7.69e−01	+
f_{13}	**5.63e−01 ± 5.54e−02**	2.74e+00 ± 9.13e−01	+	5.64e−01 ± 4.50e−02	=
f_{14}	**3.01e−01 ± 2.43e−02**	5.96e+01 ± 1.44e+01	+	3.10e−01 ± 2.25e−02	=
f_{15}	**3.05e+01 ± 1.91e+00**	8.16e+03 ± 8.12e+03	+	4.56e+01 ± 3.08e+00	+
f_{16}	**2.05e+01 ± 3.83e−01**	2.21e+01 ± 1.95e−01	+	2.22e+01 ± 3.04e−01	+
f_{17}	**4.00e+04 ± 3.76e+04**	7.13e+05 ± 4.12e+05	+	8.40e+04 ± 7.33e+04	+
f_{18}	**2.72e+02 ± 3.32e+01**	1.76e+06 ± 1.23e+06	+	4.44e+03 ± 3.84e+03	+
f_{19}	**2.05e+01 ± 4.46e+00**	6.78e+01 ± 2.26e+01	+	3.88e+01 ± 1.36e+01	+
f_{20}	**2.26e+02 ± 3.23e+01**	2.62e+04 ± 5.73e+03	+	1.20e+03 ± 6.43e+02	+
f_{21}	**5.35e+03 ± 2.01e+03**	3.11e+05 ± 1.31e+05	+	1.48e+04 ± 1.44e+04	+
f_{22}	**9.32e+02 ± 1.78e+02**	1.15e+03 ± 1.77e+02	+	1.29e+03 ± 1.99e+02	+
f_{23}	**3.41e+02 ± 1.75e−07**	3.84e+02 ± 7.39e+01	+	3.54e+02 ± 5.55e+00	+
f_{24}	2.63e+02 ± 2.76e+00	**2.00e+02 ± 1.17e−12**	-	2.94e+02 ± 2.47e+00	+
f_{25}	2.32e+02 ± 3.27e+00	**2.00e+02 ± 0.00e+00**	-	2.24e+02 ± 4.11e+00	-
f_{26}	1.01e+02 ± 4.56e−02	1.42e+02 ± 4.73e+01	+	**1.01e+02 ± 5.98e−02**	=
f_{27}	**1.34e+03 ± 2.02e+02**	1.45e+03 ± 1.97e+02	+	1.69e+03 ± 1.23e+02	+
f_{28}	2.37e+03 ± 1.89e+02	**1.95e+03 ± 1.53e+03**	=	6.50e+03 ± 5.07e+02	+
f_{29}	**1.67e+04 ± 4.31e+03**	1.76e+06 ± 2.99e+06	+	1.68e+06 ± 2.01e+06	+
f_{30}	**1.54e+04 ± 1.04e+03**	1.73e+05 ± 7.32e+04	+	6.03e+04 ± 1.35e+04	+

Table 18: Average error ± standard deviation and Wilcoxon test (reference: DE/rand/1/exp) for DE/rand/1/exp, RIDE/rand/1/exp, and eigen-DE/rand/1/exp on CEC2014 [32] rotated version in 100 dimensions.

	DE/rand/1/exp	RIDE/rand/1/exp		eigen-DE/rand/1/exp	
f_1	6.51e+08 ± 6.20e+07	**2.84e+08 ± 6.86e+07**	-	1.26e+09 ± 1.38e+08	+
f_2	**3.18e+08 ± 6.91e+07**	5.18e+10 ± 1.38e+10	+	1.56e+09 ± 2.21e+08	+
f_3	**4.30e+04 ± 4.07e+03**	1.77e+05 ± 1.29e+04	+	2.62e+05 ± 7.38e+04	+
f_4	**1.10e+03 ± 5.43e+01**	4.30e+03 ± 1.28e+03	+	4.07e+03 ± 4.58e+02	+
f_5	**2.10e+01 ± 2.65e−02**	2.13e+01 ± 2.51e−02	+	2.13e+01 ± 2.31e−02	+
f_6	1.19e+02 ± 1.96e+00	**1.07e+02 ± 4.36e+00**	-	1.37e+02 ± 2.83e+00	+
f_7	**1.37e+00 ± 4.10e−02**	4.77e+02 ± 1.01e+02	+	9.69e+00 ± 1.17e+00	+
f_8	±	±		±	
f_9	9.65e+02 ± 2.32e+01	**9.52e+02 ± 4.79e+01**	=	1.19e+03 ± 3.89e+01	+
f_{10}	±	±		±	
f_{11}	**2.42e+04 ± 4.43e+02**	2.75e+04 ± 6.18e+02	+	2.85e+04 ± 5.00e+02	+
f_{12}	**1.91e+00 ± 1.06e−01**	3.86e+00 ± 2.27e−01	+	3.82e+00 ± 2.70e−01	+
f_{13}	6.74e−01 ± 4.64e−02	3.76e+00 ± 4.12e−01	+	**6.70e−01 ± 3.72e−02**	=
f_{14}	**3.39e−01 ± 2.70e−02**	1.42e+02 ± 2.48e+01	+	3.85e−01 ± 3.91e−02	+
f_{15}	**2.73e+03 ± 7.70e+02**	3.05e+04 ± 1.85e+04	+	1.33e+05 ± 4.08e+04	+
f_{16}	**4.46e+01 ± 3.29e−01**	4.64e+01 ± 2.57e−01	+	4.65e+01 ± 2.37e−01	+
f_{17}	8.15e+07 ± 1.61e+07	**8.17e+06 ± 3.68e+06**	-	5.41e+07 ± 2.05e+07	-
f_{18}	**2.37e+04 ± 1.87e+04**	6.34e+07 ± 2.78e+07	+	1.82e+08 ± 6.54e+07	+
f_{19}	**1.32e+02 ± 5.53e+00**	2.91e+02 ± 5.26e+01	+	5.17e+02 ± 4.32e+01	+
f_{20}	**9.54e+04 ± 1.80e+04**	9.99e+04 ± 1.70e+04	=	2.32e+05 ± 4.29e+04	+
f_{21}	1.32e+07 ± 7.78e+06	**2.29e+06 ± 6.91e+05**	-	5.09e+06 ± 2.02e+06	-
f_{22}	**3.15e+03 ± 2.05e+02**	3.46e+03 ± 2.34e+02	+	4.30e+03 ± 2.50e+02	+
f_{23}	**3.45e+02 ± 4.69e−01**	3.95e+02 ± 1.24e+02	=	5.51e+02 ± 1.76e+01	+
f_{24}	4.02e+02 ± 2.12e+00	**2.00e+02 ± 1.38e−08**	-	4.83e+02 ± 3.89e+00	+
f_{25}	3.68e+02 ± 1.08e+01	**2.00e+02 ± 0.00e+00**	-	4.25e+02 ± 1.28e+01	+
f_{26}	2.12e+02 ± 7.47e+01	**2.00e+02 ± 0.00e+00**	-	2.78e+02 ± 1.32e+01	+
f_{27}	3.26e+03 ± 6.44e+01	**3.04e+03 ± 1.54e+02**	-	3.80e+03 ± 9.57e+01	+
f_{28}	9.00e+03 ± 4.79e+02	**7.88e+03 ± 5.82e+03**	=	2.01e+04 ± 8.29e+02	+
f_{29}	**2.02e+05 ± 3.27e+04**	1.34e+08 ± 1.95e+08	+	2.21e+08 ± 3.92e+07	+
f_{30}	**3.64e+05 ± 7.46e+04**	1.39e+06 ± 4.83e+05	+	3.44e+06 ± 7.62e+05	+

According to our interpretation, both the schemes that change the system of coordinates have got a potential since they both attempt to follow the directions of the gradient, thus compensating a possible high epistasis situation. On the other hand, in their current form, they both present some limitations which affect their performance and robustness. The limitation is that they both integrate some stochastic mechanisms which are not grounded on a theoretical basis. Then, they both apply an exact approach in order to determine a new system of coordinates and thus the moving operator. The use of an incorrect assumption may mislead the functioning of RIDE and eigen-DE logics.

4.3. Summary of the results and extra benchmark algorithms

In order to further enhance the statistical significance of the numerical results the Holm-Bonferroni procedure, see [24] and [16], has been applied for the seven algorithms under study and the problems under consideration. Furthermore, inspired by the study in [2], a DE without crossover (here indicated as DE/rand/1/no-xo) with $F = 0.7$ and the MMCDE proposed in [2] have also been run for comparison. The results have been grouped into the two categories, i.e. non-rotated and rotated problems. Table 19 shows the summary results for the 140 non-rotated problems while Table 20 shows the results for the 140 rotated problems.

Table 19: Holm-Bonferroni procedure on non-rotated CEC2014 at 10, 50 and 100 dimension values (reference: DE/rand/1/exp, Rank $= 8.35e + 00$)

j	Optimizer	Rank	z_j	p_j	δ/j	Hypothesis
1	DE/rand/1/bin	7.68e+00	-1.76e+00	3.89e-02	5.00e-02	Rejected
2	eigen-DE/rand/1/exp	6.08e+00	-5.98e+00	1.09e-09	2.50e-02	Rejected
3	RIDE/rand/1/exp	5.11e+00	-8.57e+00	5.30e-18	1.67e-02	Rejected
4	MMCDE	4.76e+00	-9.48e+00	1.26e-21	1.25e-02	Rejected
5	RIDE/rand/1/bin	3.86e+00	-1.19e+01	8.04e-33	1.00e-02	Rejected
6	DE/rand/1/no-xo	3.79e+00	-1.21e+01	8.25e-34	8.33e-03	Rejected
7	eigen-DE/rand/1/bin	3.08e+00	-1.39e+01	2.34e-44	7.14e-03	Rejected
8	DE/current-to-rand/1	2.14e+00	-1.64e+01	8.12e-61	6.25e-03	Rejected

The summary of the results shows that for non-rotated problems both DE/rand/1/exp and DE/rand/1/bin display the best performance. Consistently with the literature, for rotated problems, eigen-DE/rand/1/exp and RIDE/rand/1/exp appear to be quite effective but still not as good as the simple DE/rand/1/exp which is significantly the best algorithm among those considered in this study on epistasis. The results on rotated problems partly justify the fact that rotated problems could be studied by designing rotation invariant components. Nonetheless, the standard DE/rand/1/exp is indeed a robust algorithm. If the crossover rate is set properly, by taking into account the average number of swapped genes with respect to the number of variables, DE/rand/1/exp performance is hard to be beaten when an extensive experimental testbed is taken into account.

Table 20: Holm-Bonferroni procedure (reference: DE/rand/1/exp, Rank = $8.00e+00$) on rotated CEC2014 at 10, 50 and 100 dimension values.

j	Optimizer	Rank	z_j	p_j	δ/j	Hypothesis
1	eigen-DE/rand/1/exp	6.20e+00	-4.76e+00	9.87e-07	5.00e-02	Rejected
2	DE/rand/1/bin	6.10e+00	-5.04e+00	2.33e-07	2.50e-02	Rejected
3	RIDE/rand/1/exp	5.93e+00	-5.48e+00	2.12e-08	1.67e-02	Rejected
4	MMCDE	5.27e+00	-7.21e+00	2.74e-13	1.25e-02	Rejected
5	DE/rand/1/no-xo	4.08e+00	-1.04e+01	1.84e-25	1.00e-02	Rejected
6	RIDE/rand/1/bin	4.07e+00	-1.04e+01	1.32e-25	8.33e-03	Rejected
7	eigen-DE/rand/1/bin	2.93e+00	-1.34e+01	2.38e-41	7.14e-03	Rejected
8	DE/current-to-rand/1	2.37e+00	-1.49e+01	1.70e-50	6.25e-03	Rejected

Extended results are made available online[1]

4.4. Results on real-world problems

Numerical results on p_1 and p_2 of [14] are reported in Table 21 for the algorithms under consideration. The best results are highlighted in the bold font.

Table 21: Average fitness ± standard deviation and statistic comparison (reference: algorithm on the left column) of DE/rand/1/bin, DE/rand/1/bin, DE/cuurent-to-rand/1, RIDE/rand/1/bin, RIDE/rand/1/bin, eigen-DE/rand/1/bin and eigen-DE/rand/1/bin on Problem 1 (6D) and Problem 2 (30D) of CEC2011[14].

	DE/rand/1/exp	DE/rand/1/bin		DE/current-to-rand/1	
p_1	$\mathbf{2.42e+00 \pm 2.25e+00}$	$3.21e+00 \pm 4.36e+00$	=	$1.76e+01 \pm 3.09e+00$	+
p_2	$-1.99e+01 \pm 9.89e-01$	$\mathbf{-2.51e+01 \pm 3.08e+00}$	-	$-1.56e+01 \pm 1.40e+00$	+
	DE/rand/1/bin	RIDE/rand/1/bin		eigen-DE/rand/1/bin	
p_1	$\mathbf{3.21e+00 \pm 4.36e+00}$	$1.99e+01 \pm 2.11e+00$	+	$1.41e+01 \pm 3.64e+00$	+
p_2	$\mathbf{-2.51e+01 \pm 3.08e+00}$	$-4.31e+00 \pm 4.38e-01$	+	$-7.21e+00 \pm 7.94e-01$	+
	DE/rand/1/exp	RIDE/rand/1/exp		eigen-DE/rand/1/bin	
p_1	$\mathbf{2.42e+00 \pm 2.25e+00}$	$1.62e+01 \pm 3.50e+00$	+	$1.08e+01 \pm 3.06e+00$	+
p_2	$\mathbf{-1.99e+01 \pm 9.89e-01}$	$-1.42e+01 \pm 1.05e+00$	+	$-8.54e+00 \pm 8.83e-01$	+

For p_1 DE/rand/1/exp achieves the best performance while for p_2 the best performed is achieved by DE/rand/1/best. It can be observed that also for the real-world problems the standard DE algorithms tend to consistently outperform their rotation-invariant counterparts.

The full list of algorithms under consideration are also listed and ranked in Table 22 which displays the result of the Holm-Bonferroni procedure.

5. Conclusion

This article proposes an experimental study aiming at understanding the functioning of DE crossover in the presence of epistatic problems. Since the epistasis is simulated by

[1] www.tech.dmu.ac.uk/~fcaraf00/NumericalResults/DE_RotInvStudy_Results.pdf

Table 22: Holm-Bonferroni procedure (reference: DE/rand/1/exp, Rank = $8.50e + 00$) for the CEC2011 real-worl problems

j	Optimizer	Rank	z_j	p_j	δ/j	Hypothesis
1	DE/rand/1/bin	8.50e+00	0.00e+00	5.00e-01	5.00e-02	Accepted
2	eigen-DE/rand/1/exp	5.50e+00	-1.22e+00	1.10e-01	2.50e-02	Accepted
3	RIDE/rand/1/exp	5.00e+00	-1.43e+00	7.65e-02	1.67e-02	Accepted
4	DE/current-to-rand/1	4.50e+00	-1.63e+00	5.12e-02	1.25e-02	Accepted
5	eigen-DE/rand/1/bin	4.00e+00	-1.84e+00	3.31e-02	1.00e-02	Accepted
5	DE/rand/1/no-xo	4.00e+00	-1.84e+00	3.31e-02	1.00e-02	Accepted
5	MMCDE	4.00e+00	-1.84e+00	3.31e-02	1.00e-02	Accepted
6	RIDE/rand/1/bin	1.00e+00	-3.06e+00	1.10e-03	8.33e-03	Rejected

means of a problem's rotation, in literature several rotational invariant algorithms have been proposed.

On the basis of an extensive experimental analysis, the following conclusions have been reached:

1. for a fixed ratio of transferred design variables from the parent to the offspring, the exp crossover seems to display a better performance for non-rotated problems while the bin crossover appears to achieve a better performance for rotated problems, especially in high dimensions;
2. the DE/current-to-rand/1 strategy does not seem to efficiently tackle rotated problems since, according to our experiments, it is consistently outperformed by the standard DE algorithms. The weak point of this methods is, according to us, the fact it corresponds to a random vector sum, thus making the algorithm excessively exploratory;
3. the crossover that changes the variables by means of the Gram-Schmidt orthogonalisation (as in an inner product space) also does not seem to be competitive with the standard DE. The weak point is that the directions of the new basis are randomly selected and not according to convenient directions;
4. eigen-DE is a very interesting and promising scheme that, however, does not appear on its own to robustly compete with a standard DE/rand/1/exp. Besides requiring more computational resources than a standard DE, the weak point is that eigen-DE builds a covariance matrix from a sample of points (candidate solutions). These points may or may not be representative of the fitness landscape. Although the diagonalisation of the covariance matrix is grounded on a rigorous theoretical basis, the matrix itself, in its current implementation, is an approximation which is not necessarily reliable.

In conclusion, problems characterised by a high epistasis can potentially benefit from a proper change of coordinates. However, the detection of a mechanism that can reliably detect the most convenient system of coordinates is not a straightforward task. Standard DE schemes already handle well high epistasis and, if the parameters are properly set, are hard to enhance. This finding is implicitly confirmed by the literature where modern algorithms addressing rotated problems incorporate an additional moving operator (which gets occasionally activated) to a standard DE framework.

Appendix: A summarising scheme of the algorithms in this study

Table 23 schematically summarises the algorithms involved in this study.

Table 23: Algorithms in this study with their main characterising features

Optimizer	Brief description	Parameters
DE/rand/1/no-xo	no crossover	$N_p = n$, $F = 0.7$
DE/rand/1/bin	binomial crossover	$N_p = n$, $F = 0.7$, $Cr = 0.3$
DE/rand/1/exp	exponential crossover	$N_p = n$, $F = 0.7$ $Cr = \frac{1}{n 0.\sqrt[3]{2}}$
RIDE/rand/1/bin	change of coordinates by orthogonalisation and binomial crossover in the new basis	$N_p = n$, $F = 0.7$, $Cr = 0.3$
RIDE/rand/1/exp	change of coordinates by orthogonalisation and exponential crossover in the new basis	$N_p = n$, $F = 0.7$, $Cr = \frac{1}{n 0.\sqrt[3]{2}}$
eigen-DE/rand/1/bin	change of coordinates by diagonalisation of the covariance matrix and binomial crossover	$N_p = n$, $F = 0.7$, $Cr = 0.3$
eigen-DE/rand/1/exp	change of coordinates by diagonalisation of the covariance matrix and exponential crossover	$N_p = n$, $F = 0.7$, $Cr = \frac{1}{n 0.\sqrt[3]{2}}$
MMCDE	pool of mutation strategies and change of coordinates by orthogonalisation and both binomial and exponential crossovers	$N_p = n$, $F = 0.7$, $Cr_{bin} = 0.3$, $Cr_{exp} = \frac{1}{n 0.\sqrt[3]{2}}$

References

[1] Al-Dabbagh, R. D., Neri, F., Idris, N., Baba, M. S., 2018. Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy. Swarm and Evolutionary ComputationTo appear.

[2] Alam Anik, M. T., Noman, A. S. M., Ahmed, S., 2013. Preserving rotation invariant properties in differential evolution algorithm. In: 2013 2nd International Conference on Advances in Electrical Engineering (ICAEE). pp. 235–240.

[3] Awad, N. H., Ali, M. Z., Suganthan, P. N., Liang, J. J., Qu, B. Y., 2016. Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. Tech. rep., Zhengzhou University, Jordan University and Nanyang Technological University, Zhengzhou China, Jordan and Singapore.

[4] Bäck, T., 1996. Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press.

[5] Bianchi, L., Dorigo, M., Gambardella, L. M., Gutjahr, W. J., Jun 2009. A survey on metaheuristics for stochastic combinatorial optimization. Natural Computing 8 (2), 239–287.

[6] Brest, J., Zamuda, A., Bošković, B., 2015. Adaptation in the differential evolution. In: Fister, I., Fister Jr., I. (Eds.), Adaptation and Hybridization in Computational Intelligence. Springer International Publishing, pp. 53–68.

[7] Bujok, P., Tvrdìk, J., Polàkovà, R., July 2014. Differential evolution with rotation-invariant mutation and competing-strategies adaptation. In: 2014 IEEE Congress on Evolutionary Computation (CEC). pp. 2253–2258.

[8] Caponio, A., Cascella, G. L., Neri, F., Salvatore, N., Sumner, M., 2007. A fast adaptive memetic algorithm for on-line and off-line control design of pmsm drives. IEEE Transactions on System Man and Cybernetics-part B 37 (1), 28–41.

[9] Caraffini, F., Neri, F., 2018. Rotation invariance and rotated problems: An experimental study on differential evolution. In: Applications of Evolutionary Computation. No. 10784 in Lecture Notes in Computer Science. Springer, pp. 597–614.
[10] Caraffini, F., Neri, F., Iacca, G., Mol, A., 2013. Parallel memetic structures. Information Sciences 227 (0), 60 – 82.
[11] Chen, Q., Liu, B., Zhang, Q., Liang, J. J., Suganthan, P. N., Qu, B. Y., 2014. Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Learning-based Real-Parameter Single Objective Optimization Session on Real-Parameter Optimization. Tech. rep., Zhengzhou University and Nanyang Technological University, Zhengzhou China and Singapore.
[12] Das, S., Mullick, S. S., Suganthan, P. N., 2016. Recent advances in differential evolution - an updated survey. Swarm and Evolutionary Computation 27, 1–30.
[13] Das, S., Suganthan, P., feb. 2011. Differential Evolution: A Survey of the State-of-the-Art. Evolutionary Computation, IEEE Transactions on 15 (1), 4–31.
[14] Das, S., Suganthan, P. N., 2010. Problem definitions and evaluation criteria for cec 2011 competition on testing evolutionary algorithms on real world optimization problems. Jadavpur University, Nanyang Technological University, Kolkata.
[15] Demmel, J., Veseli, K., 1992. Jacobis method is more accurate than qr. SIAM Journal on Matrix Analysis and Applications 13 (4), 1204–1245.
[16] Garcia, S., Fernandez, A., Luengo, J., Herrera, F., 2008. A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Computing 13 (10), 959–977.
[17] García, S., Molina, D., Lozano, M., Herrera, F., 2008. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 Special Session on Real Parameter Optimization. Journal of Heuristics 15 (6), 617–644.
[18] Guo, S. M., Yang, C. C., Feb 2015. Enhancing differential evolution utilizing eigenvector-based crossover operator. IEEE Transactions on Evolutionary Computation 19 (1), 31–49.
[19] Hansen, N., Auger, A., Finck, S., Ros, R., et al., 2010. Real-parameter black-box optimization benchmarking 2010: Noiseless functions definitions. Tech. Rep. RR-6829, INRIA.
[20] Hansen, N., Niederberger, A. S. P., Guzzella, L., Koumoutsakos, P., 2008. Evolutionary Optimization of Feedback Controllers for Thermoacoustic Instabilities. In: IUTAM Symposium on Flow Control and MEMS. Vol. 7. Springer, pp. 311–317.
[21] Hansen, N., Niederberger, A. S. P., Guzzella, L., Koumoutsakos, P., 2009. A Method for Handling Uncertainty in Evolutionary Optimization With an Application to Feedback Control of Combustion. IEEE Transactions on Evolutionary Computation 13 (1), 180–197.
[22] Hansen, N., Ostermeier, A., 1996. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In: Proceedings of the IEEE International Conference on Evolutionary Computation. pp. 312–317.
[23] Hansen, N., Ostermeier, A., 2001. Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9 (2), 159–195.
[24] Holm, S., 1979. A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6 (2), 65–70.
[25] Iacca, G., Neri, F., Mininno, E., Ong, Y. S., Lim, M. H., 2012. Ockham's Razor in Memetic Computing: Three Stage Optimal Memetic Exploration. Information Sciences 188, 17–43.
[26] Iorio, A. W., Li, X., 2005. Solving rotated multi-objective optimization problems using differential evolution. In: Webb, G. I., Yu, X. (Eds.), Proceedings of AI 2004: Advances in Artificial Intelligence. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 861–872.
[27] Iorio, A. W., Li, X., 2006. Rotated test problems for assessing the performance of multi-objective optimization algorithms. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation. ACM, New York, NY, USA, pp. 683–690.
[28] Lampinen, J., Zelinka, I., 1999. Mechanical engineering design optimization by differential evolution. In: Corne, D., Dorigo, M., Glover, F. (Eds.), New Ideas in Optimization. McGraw-Hill, pp. 127–146.

[29] Li, Y., Feng, J., Hu, J., 2016. Covariance and crossover matrix guided differential evolution for global numerical optimization. SpringerPlus 5 (1), 1176.

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

[31] Liang, J., Qu, B., Suganthan, P., Chen, Q., 2014. Problem definitions and evaluation criteria for the cec 2015 competition on learning-based real-parameter single objective optimization. Tech. rep., Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore.

[32] Liang, J. J., Qu, B. Y., Suganthan, 2013. Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session on Real-Parameter Optimization. Tech. Rep. 201311, Zhengzhou University and Nanyang Technological University, Zhengzhou China and Singapore.

[33] Liang, J. J., Qu, B. Y., Suganthan, P. N., Hernndez-Daz, A. G., 2013. Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization. Tech. Rep. 201212, Zhengzhou University and Nanyang Technological University, Zhengzhou China and Singapore.

[34] Lozano, M., Molina, D., Herrera, F., 2011. Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. Soft Computing 15 (11), 2085–2087.

[35] Mallipeddi, R., Suganthan, P. N., 2010. Problem Definitions and Evaluation Criteria for the CEC 2010 Competition on Constrained Real-Parameter Optimization Real-Parameter Numerical Optimization. Tech. rep., Nanyang Technological University, Singapore.

[36] Neri, F., 2016. Linear Algebra for Computational Sciences and Engineering. Springer.

[37] Neri, F., Tirronen, V., 2010. Recent Advances in Differential Evolution: A Review and Experimental Analysis. Artificial Intelligence Review 33 (1–2), 61–106.

[38] Price, K., 1999. An Introduction to Differential Evolution. In: Corne, D., Dorigo, M., Glover, F., Dasgupta, D., Moscato, P., Poli, R., Price, K. V. (Eds.), New Ideas in Optimization. McGraw-Hill, pp. 79–108.

[39] Qing, A., 2010. A literature survey on differential evolution. In: Qing, A., Lee, C. K. (Eds.), Differential Evolution in Electromagnetics. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–17.

[40] Qu, B. Y., Liang, J. J., Wang, Z. Y., Chen, Q., Suganthan, P. N., 2016. Novel benchmark functions for continuous multimodal optimization with comparative results. Swarm and Evolutionary Computation 26, 23–34.

[41] Rosenbrock, H. H., 1960. An automatic Method for finding the greatest or least Value of a Function. The Computer Journal 3 (3), 175–184.

[42] Rothlauf, F., 2011. Optimization methods. In: Design of Modern Heuristics: Principles and Application. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 45–102.

[43] Salvatore, N., Caponio, A., Neri, F., Stasi, S., Cascella, G. L., 2010. Optimization of Delayed-State Kalman Filter-based Algorithm via Differential Evolution for Sensorless Control of Induction Motors. IEEE Transactions on Industrial Electronics 57 (1), 385–394.

[44] Storn, R., 1996. Differential evolution design of an IIR-filter. In: Proceedings of IEEE International Conference on Evolutionary Computation. pp. 268–273.

[45] Suganthan, P. N., 2012. Differential evolution algorithm: Recent advances. In: Dediu, A.-H., Martín-Vide, C., Truthe, B. (Eds.), Theory and Practice of Natural Computing: First International Conference, TPNC 2012, Tarragona, Spain, October 2-4, 2012. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 30–46.

[46] Suganthan, P. N., Das, S., Mukherjee, S., Chatterjee, S., 01 2014. Adaptation methods in differential evolution: A review. In: 20th International Conference on Soft Computing MENDEL. Vol. 2014. pp. 131–140.

[47] Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger, A., Tiwari, S., 2005. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Tech. Rep. 2005005, Nanyang Technological University and KanGAL, Singapore and IIT Kanpur, India.

[48] Takahama, T., Sakai, S., 2010. Solving Nonlinear Optimization Problems by Differential Evolution with a Rotation-Invariant Crossover Operation using Gram-Schmidt process. In: Proceedings of the World Congress on Nature and Biologically Inspired Computing. pp. 533–540.
[49] Takahama, T., Sakai, S., 2011. Efficient Nonlinear Optimization by Differential Evolution with a Rotation-Invariant Local Sampling Operation. In: Proceedings of IEEE Congress on Evolutionary Computation. pp. 2215–2222.
[50] Takahama, T., Sakai, S., 2011. Fuzzy c-means clustering and partition entropy for species-best strategy and search mode selection in nonlinear optimization by differential evolution. In: 2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011). pp. 290–297.
[51] Weber, M., Neri, F., Tirronen, V., 2011. A Study on Scale Factor in Distributed Differential Evolution. Information Sciences 181 (12), 2488–2511.
[52] Wilcoxon, F., 1945. Individual comparisons by ranking methods. Biometrics Bulletin 1 (6), 80–83.
[53] Zaharie, D., 2002. Critical Values for Control Parameters of Differential Evolution Algorithm. In: Matušek, R., Ošmera, P. (Eds.), Proceedings of 8th International Mendel Conference on Soft Computing. pp. 62–67.
[54] Zaharie, D., Jun. 2009. Influence of crossover on the behavior of differential evolution algorithms. Appl. Soft Comput. 9 (3), 1126–1138.