Comparing Black-Box Differential Evolution and Classic Differential Evolution

Aljoša Vodopija
Jožef Stefan Institute and
Jožef Stefan International
Postgraduate School
Ljubljana, Slovenia
aljosa.vodopija@ijs.si

Tea Tušar
Jožef Stefan Institute
Ljubljana, Slovenia
tea.tusar@ijs.si

Bogdan Filipič
Jožef Stefan Institute and
Jožef Stefan International
Postgraduate School
Ljubljana, Slovenia
bogdan.filipic@ijs.si

ABSTRACT

Recently, black-box differential evolution (BBDE) has been proposed to overcome the search biases and sensitivity to rotation of the classic differential evolution (DE). To date, BBDE has been studied only for the 'rand' strategy and even for this strategy, no systematic experimental study has been published yet. In this paper we provide such a study and further examine whether the idea from BBDE can be extended to two other DE strategies, 'best' and 'target-to-best'. We compare in detail these DE and BBDE variants using the COCO (Comparing COntinuous Optimizers) platform to assess their overall performance and invariance to rotation. The results show that BBDE with the 'rand' strategy performs better than the original algorithm, but this is not true for the other two strategies. We also demonstrate that while the BBDE variants are less sensitive to rotation than the DE variants, some sensitivity to this transformation still persists and remains currently unexplained.

CCS CONCEPTS

• **Theory of computation → Mathematical optimization**; **Continuous optimization**; **Bio-inspired optimization**;

KEYWORDS

Benchmarking; Black-box optimization; Search bias; Differential evolution

ACM Reference Format:
Aljoša Vodopija, Tea Tušar, and Bogdan Filipič. 2018. Comparing Black-Box Differential Evolution and Classic Differential Evolution. In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3205651.3208309

1 INTRODUCTION

Differential Evolution is a powerful and versatile evolutionary algorithm for solving continuous optimization problems [1]. For this reason on the one hand, and its simplicity and ease of implementation on the other hand, DE and its variants have been successfully applied to solve numerous real-world problems from diverse domains of science and engineering [7].

However, it has been observed in [9] that DE does not satisfy the demands of black-box optimization—we say that an optimizer satisfies the demands of black-box optimization if its performance is invariant to translation, rotation, reflection, permutation of parameters, and order-preserving transformation of the objective function. More precisely, it has been shown that the performance of the classic DE (the DE variant often referred to as DE/rand/1/bin and denoted as 'rand' in this work) is not invariant to orthogonal rotation of the coordinate system [9], which is undoubtedly a serious drawback, since the coordinate system orientation is an arbitrary choice. Moreover, it is evident from experiments [9] that its exploration is quite biased, e.g., toward search in a non-justifiable direction, it favors one coordinate system orientation over another, etc. To overcome these biases, the black-box differential evolution has been proposed and it has been experimentally shown that BBDE satisfies the demands of black-box optimization [9].

Nevertheless, no systematic experimental study addressing the BBDE efficiency has been published yet. Therefore, it is not known whether BBDE outperforms the classic DE. In addition, this idea has been investigated only for the 'rand' strategy, while other variants of DE have not yet been explored in this regard. Taking into account the 'no free lunch' theorem and the extensive experimental study presented in [11], it is evident that the performance of different variants of DE varies over problems. For this reason, it is quite natural to ask what about other variants of DE. Can other variants of DE be modified to satisfy the demands of black-box optimization as well, and what is the performance of these algorithms in comparison to their original variants?

In this study, we perform a detailed comparison between BBDE and DE and examine whether other variants of DE can be modified to satisfy the demands of black-box optimization. As we will see in the following section, one of the crucial modifications in BBDE is that the crossover operator is completely disregarded. Taking into account this fact and the results of the experimental study from [11], we focus on two strategies determining how new solutions are created and mutated: 'best' and 'target-to-best'.

The experiments are carried out using the COCO platform [5] on the *bbob* suite [3], which contains 24 single-objective optimization problems with diverse characteristics. Among them, two problems (*ellipsoid* and *Rastrigin*) exist in separable and rotated variants enabling the study of algorithms' sensitivity to rotation. The rest of

ACM ISBN 978-1-4503-5764-7/18/07…$15.00
https://doi.org/10.1145/3205651.3208309

Algorithm 1 DE

```
Input: population size and stopping criterion;
Output: population $\mathbb{P}$ of solutions;
 1: create the initial population $\mathbb{P}$ of random solutions;
 2: evaluate the solutions in $\mathbb{P}$;
 3: while stopping criterion not met do
 4:     $\mathbb{P}_{\text{new}} \leftarrow \emptyset$;
 5:     for all $p \in \mathbb{P}$ do
 6:         create a trial vector $p_{\text{trial}}$;
 7:         ensure feasibility of $p_{\text{trial}}$;
 8:         evaluate $p_{\text{trial}}$;
 9:         if $p_{\text{trial}}$ is better than $p$ then
10:             $p \leftarrow p_{\text{trial}}$;
11:         end if
12:         $\mathbb{P}_{\text{new}} \leftarrow \mathbb{P}_{\text{new}} \cup \{p\}$;
13:     end for
14:     $\mathbb{P} \leftarrow \mathbb{P}_{\text{new}}$;
15: end while
16: return $\mathbb{P}$;
```

this paper is organized as follows. Section 2 introduces the algorithms used in our study. Section 3 is dedicated to the experimental setup, while the results are discussed in Section 4. The conclusions are summarized in Section 5.

2 METHODS

This section presents the algorithms used in this study. After introducing the general form of the DE algorithm, we discuss the three chosen DE variants together with their BBDE modifications.

2.1 The DE Algorithm

Like other population-based methods, DE begins with a randomly created population of solutions $\mathbb{P}$. In each iteration, for each solution $p \in \mathbb{P}$ named *target vector*, DE creates a *trial vector* p_{trial}. These two vectors are then compared and the better one is selected as a member for the new population. The DE pseudocode is shown in Algorithm 1. All the algorithms studied in this paper follow this structure and differ only in the creation of the trial vector (Line 6 in Algorithm 1).

In this study, we bound the decision space by predefined box constraints and it can happen that the trial vector violates them. For this reason, we include a simple constraint handling technique that ensures the feasibility of p_{trial} (Line 7 in Algorithm 1) as follows. Any component of the trial vector that violates the box constraints is replaced with a randomly selected feasible value.

2.2 The 'rand' Strategy

The DE strategy DE/rand/1/bin, also called classic DE and denoted with DE/rand in the remainder of this paper, is arguably the most popular DE variant. It creates the trial vector as follows. First, three pairwise different solutions p_0, p_1, p_2 are selected from $\mathbb{P} - \{p\}$, where p_0 is named *base vector*. Second, the trial vector is initialized as $p_{\text{trial}} = p_0 + F(p_1 - p_2)$, where F, named *scaling factor*, is an apriori selected positive real number fixed over the entire run. Finally, the trial vector is altered by crossover with the target vector. The pseudocode describing the creation of trial vectors is shown in Algorithm 2.

Algorithm 2 Trial vector creation by DE/rand

```
Input: population $\mathbb{P}$ and target vector $p \in \mathbb{P}$;
Output: trial vector $p_{\text{trial}}$;
 1: randomly select three pairwise different solutions
    $p_0, p_1, p_2 \in \mathbb{P} - \{p\}$;
 2: create a trial vector $p_{\text{trial}} = p_0 + F \cdot (p_1 - p_2)$;
 3: alter $p_{\text{trial}}$ by crossover with $p$;
 4: return $p_{\text{trial}}$;
```

Algorithm 3 Trial vector creation by BBDE

```
Input: population $\mathbb{P}$ and target vector $p \in \mathbb{P}$;
Output: trial vector $p_{\text{trial}}$;
 1: randomly select two different solutions $p_1, p_2 \in \mathbb{P}$;
 2: sample a scaling factor $F$ from $X_F \sim \exp(\mathcal{N}(0,1))$;
 3: create a trial vector $p_{\text{trial}} = p + F \cdot (p_1 - p_2)$;
 4: return $p_{\text{trial}}$;
```

Algorithm 4 Trial vector creation by BBDE-N

```
Input: population $\mathbb{P}$ and target vector $p \in \mathbb{P}$;
Output: trial vector $p_{\text{trial}}$;
 1: randomly select two different solutions $p_1, p_2 \in \mathbb{P}$;
 2: sample a scaling factor $F$ from $X_F \sim \exp(\mathcal{N}(0,1))/\sqrt{D}$;
 3: create a trial vector $p_{\text{trial}} = p + F \cdot (p_1 - p_2)$;
 4: return $p_{\text{trial}}$;
```

From the perspective of black-box optimization demands, there are two main concerns about DE/rand—its asymmetrical exploration and the bias for separable directions brought by crossover with the target vector. The first concern is that the exploration of DE/rand is not symmetrically distributed around the target vector for target vectors that are far away from the center of the population. To overcome this drawback, in BBDE the base vector is always equal to the target vector, and the two selected solutions p_1, p_2 do not have to be different from p. This ensures symmetrically distributed exploration around the target vector (for more information see [9]). The second concern is that the crossover operator in DE imposes a search bias for separable directions. For this reason, BBDE uses no crossover. In addition, in BBDE the scaling factor F is sampled for each solution in each iteration, thus it is not fixed over the entire run. The pseudocode describing how BBDE creates the trial vector is shown in Algorithm 3.

The distribution X_F in Line 2 of Algorithm 3 can have different forms. Originally, the author of BBDE proposed to use a log-normal distribution $X_F \sim \exp(\mathcal{N}(0,1))$. However, in [10] the same author observed that a normalization of the scaling factor based on the dimension of the decision space (denoted in this paper by D) $X_F \sim \exp(\mathcal{N}(0,1))/\sqrt{D}$ improved the algorithm performance. We will refer to this special case of BBDE as BBDE-N. Its pseudocode is shown in Algorithm 4.

Algorithm 5 Trial vector creation by DE/best

```
Input: population $\mathbb{P}$ and target vector $p \in \mathbb{P}$;
Output: trial vector $p_{\text{trial}}$;
1: randomly select two different solutions $p_1, p_2 \in \mathbb{P} - \{p\}$;
2: create a trial vector $p_{\text{trial}} = p_{\text{best}} + F \cdot (p_1 - p_2)$;
3: alter $p_{\text{trial}}$ by crossover with $p$;
4: return $p_{\text{trial}}$;
```

Algorithm 6 Trial vector creation by BBDE/best

```
Input: population $\mathbb{P}$ and target vector $p \in \mathbb{P}$;
Output: trial vector $p_{\text{trial}}$;
1: randomly select two different solutions $p_1, p_2 \in \mathbb{P}$;
2: sample a scaling factor $F$ from $X_F \sim \exp(\mathcal{N}(0, 1))$;
3: create a trial vector $p_{\text{trial}} = p_{\text{best}} + F \cdot (p_1 - p_2)$;
4: return $p_{\text{trial}}$;
```

Algorithm 7 Trial vector creation by DE/target-to-best

```
Input: population $\mathbb{P}$ and target vector $p \in \mathbb{P}$;
Output: trial vector $p_{\text{trial}}$;
1: create a trial vector $p_{\text{trial}} = p + F \cdot (p_{\text{best}} - p)$;
2: alter $p_{\text{trial}}$ by crossover with $p$;
3: return $p_{\text{trial}}$;
```

Algorithm 8 Trial vector creation by BBDE/target-to-best

```
Input: population $\mathbb{P}$ and target vector $p \in \mathbb{P}$;
Output: trial vector $p_{\text{trial}}$;
1: randomly select a solution $p_2 \in \mathbb{P} - \{p_{\text{best}}\}$;
2: sample a scaling factor $F$ from $X_F \sim \exp(\mathcal{N}(0, 1))$;
3: create a trial vector $p_{\text{trial}} = p + F \cdot (p_{\text{best}} - p_2)$;
4: return $p_{\text{trial}}$;
```

2.3 The 'best' Strategy

The 'best' strategy differs from the 'rand' strategy in the selection of the base vector. Unlike in the 'rand' strategy, where the base vector is selected at random, the 'best' strategy always selects the best-so-far solution p_{best} as the base vector. We will refer to this variant of DE as DE/best. The pseudocode describing the creation of the trial vector is shown in Algorithm 5.

The modification toward the demands of black-box optimization in this case is very similar to that in the case of the 'rand' strategy. We remove crossover, since it forces the algorithm to explore in separable directions. However, we do not wish to remove the preference for exploring around the best-so-far solution as this is a feature of the 'best' strategy. Given that other solutions are obviously worse than the best-so-far solution, this bias is justifiable. For example, experiments show [11] that it can improve the algorithm performance on many optimization problems. Next, in this modification the base vector is always equal to the best-so-far solution, but the two selected vectors p_1 and p_2 do not have to be different from p. Finally, the scaling factor is sampled from a given distribution $X_F \sim \exp(\mathcal{N}(0, 1))$. We will refer to this algorithm as BBDE/best. The pseudocode describing how BBDE/best creates the trial vector is shown in Algorithm 6.

2.4 The 'target-to-best' Strategy

Similarly to the 'best' strategy, the 'target-to-best' strategy takes into account the information of the best-so-far solution. This time, however, the trial vector is created so that it lies on a line between the target vector and the best-so-far solution. To achieve this, the base vector and p_2 are equal to the target vector, while p_1 is equal to the best-so-far solution. The pseudocode describing the creation of the trial vectors is shown in Algorithm 7.

The modification toward satisfying the black-box optimization demands is quite similar as before. The crossover is removed, and the scaling factor is sampled from $X_F \sim \exp(\mathcal{N}(0, 1))$. To make the algorithm even less biased, we remove the constraint that the trial vector lies on a line between the target vector and the best-so-far solution. Therefore, p_2 is not equal to the target vector, but to a randomly selected solution different from p_{best}. In this way, the exploration of the algorithm is more flexible. For example, if the best-so-far solution is close to the center of the population, the exploration is almost unbiased and very similar to that of BBDE. On the other hand, if the best-so-far solution is far from the population center, the algorithm pushes the whole population toward the best-so-far solution, but as soon as the population reaches this solution—which is now close to the center of the population—the search explores near the best-so-far solution in an 'unbiased' way. Thus, we can say that this variant of BBDE adapts its bias in a 'self-adaptive' fashion. We will refer to this variant of BBDE as BBDE/target-to-best. The pseudocode describing how BBDE/target-to-best creates the trial vector is shown in Algorithm 8.

3 EXPERIMENTAL SETUP

3.1 The COCO Platform

The experiments were performed using COCO, a platform for systematic and sound comparisons of real-parameter global optimizers [5]. COCO provides benchmark function testbeds, experimentation templates, and tools for processing and visualizing data generated by one or several optimizers. Its performance assessment is based on the runtime (measured in the number of objective function evaluations) needed to reach specific targets [4]. A target is the absolute difference in the objective function value between the known optimal solution and the best-so-far solution found by the algorithm. We use COCO's default target values of 10^i for $i \in \{-8, -7.8, \ldots, 1.8, 2\}$.

The suite of problems used in our study was the *bbob* suite from 2017, which contains twenty-four noiseless single-objective optimization problems without explicit constraints in fifteen instances and six dimensions of the decision space (2, 3, 5, 10, 20, 40). The *bbob* suite consists of five groups of problems: separable problems, problems with low or moderate conditioning, problems with high conditioning and unimodal, multi-modal problems with adequate global structure, and multi-modal problems with weak global structure [3].

To assess the overall performance of the algorithms presented in the previous section, all 24 problems from the *bbob* suite are used. In addition, to investigate the invariance to rotation and the importance of separability, we use ellipsoid and Rastrigin problems. There are two variants of ellipsoid problem. The first variant (f_2) is a separable ellipsoid problem where its principle axes are aligned with the coordinate axes. The second variant is a rotated ellipsoid problem (f_{10}), therefore, no longer aligned with the coordinate system orientation. Similarly to ellipsoid problem there are two variants of Rastrigin problem a highly multimodal optimization problem. The first one is a separable variant (f_3), which has a very regular and symmetric structure. The second variant (f_{15}) is transformed—rotation included—in order to alleviate this symmetry and regularity (see [3] for more details). Clearly, this variant of Rastrigin problem is no longer separable.

3.2 Parameter Settings

In all the experiments, the decision space was bounded with $[-5, 5]^D$ and the population size was equal to $15D$. The crossover probability *CR* was set to 0.7 and the scaling factor to 0.5 for all original algorithms: DE/rand, DE/best, and DE/target-to-best. Finally, each algorithm run was stopped when the following criterion was met: The final target (10^{-8}) was reached or $5 \cdot 10^5 D$ evaluations were performed and the last population was completed.

3.3 Implementations

All the algorithms were implemented in Python 2.7 [12]. The *SciPy* [6] implementation of DE was used for all variants of DE, while all variants of BBDE were implemented by ourselves mimicking the classic DE implementation from *SciPy*. In addition, the experiments and the plots were produced with COCO version 2.2.1.

4 RESULTS

In this section, we present the results obtained while experimenting with the presented algorithms. Like Section 2, this section is also divided based on the strategies used. The results are presented as empirical cumulative distribution function (ECDF) of simulated runtimes. The ECDF displays the proportion of problems solved within a specified budget aggregated over all problem instances [4].

For each strategy we first discuss the overall results where ECDFs are additionally aggregated over all problems of the same dimension. Then, we address the question about invariance to rotation and separability. This is done by comparing the ellipsoid problem and Rastrigin problem presented in Section 3.1. Here, we present the results only for dimensions 10 and 20, since the differences are not perceptible for other dimensions.

In addition, the results include a comparison with four DE variants that were tested on the *bbob* suite in the past: DEuniform [2], DE [8], DEAE [8], and R-DE-10e5 [13]. The performance of these algorithms is shown in the background of the ECDF plots in gray.

4.1 The 'rand' Strategy

In the initial set of experiments, we compared DE/rand, BBDE, and BBDE-N. The results are shown in Figure 1.

For small dimensions of the decision space (2, 3, 5), the DE/rand variant performs better than BBDE and there are almost no differences between BBDE and BBDE-N. This is probably because the normalization is proportional to the square of the dimension. Thus, the scaling factor does not change significantly for small dimensions, since the denominator ($\sqrt{D}$) in the normalization is close to one. In contrast, BBDE-N performs significantly better than DE/rand and BBDE for high dimensions (10, 20, 40).

Figure 2 shows that the performance of DE/rand is highly correlated with the chosen coordinate system orientation. For example, the algorithm cannot reach even the first target (10^2) on the rotated ellipsoid problem of dimension 20, while it solves the separable version of this problem. In contrast, BBDE and BBDE-N are *almost* invariant to rotation on the ellipsoid problem. While BBDE-N is able to solve both ellipsoid problems for dimensions 10 and 20, it needs around 17% more evaluations to do so for the rotated problems (this is not easily visible from the plots due to the logarithmic scale of the x axis). Similarly holds for BBDE, but because it's worse than BBDE-N, the differences in its performance on the separable and rotated ellipsoid are easier to see (especially for dimension 20). Analogous results were achieved also on the two Rastrigin problems. These findings do not corroborate the invariance to rotation of BBDE as shown in [9]. Given that the mechanisms used in BBDE and BBDE-N seem invariant to rotation, it is unclear what causes this mild but persisting sensitivity to this transformation.

4.2 The 'best' Strategy

Next, we performed the experiments comparing DE/best and BBDE/-best. The results are presented in the same form as before and shown in Figure 3.

We can observe that there are no significant differences between the two algorithms for dimensions 2 and 3, and that DE/best performs significantly better on high dimensions (10, 20, 40) for large number of evaluations.

From Figure 4 it is evident that BBDE/best is less sensitive to rotation than DE/best. However, as shown by the results on the ellipsoid problems, some sensitivity to rotation is still present.

4.3 The 'target-to-best' Strategy

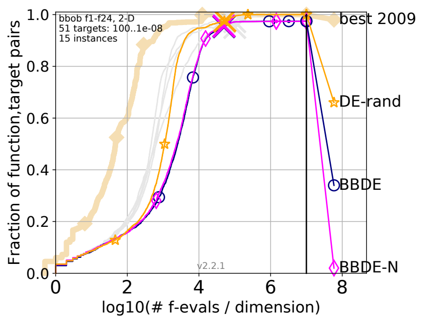
Finally, we compared the strategies DE/target-to-best and BBDE/-target-to-best. The results are presented in the same form as before and shown in Figure 5. As we can see, the strategy DE/target-to-best generally performs better than BBDE/target-to-best.

Figure 6 shows that while DE/target-to-best is highly sensitive to rotation, the performance of BBDE/target-to-best is very similar on separable and rotated problems (the differences are sometimes difficult to see from the plots).

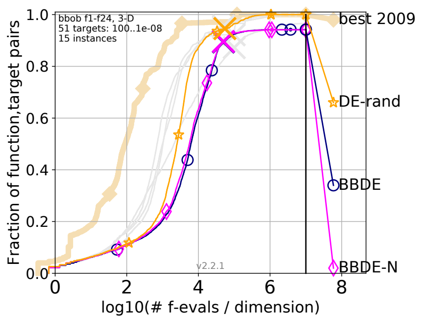
5 CONCLUSIONS

In this paper, we compared three variants of DE and their modifications to meet the demands of black-box optimization. To assess the overall performance and the invariance to rotation, we used the COCO platform and the *bbob* test suite.

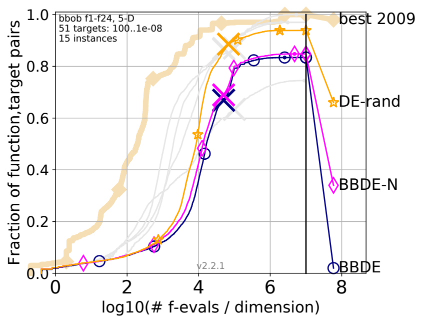
We have seen that modifying the 'rand' strategy can improve its general performance, while the modifications for the 'best' and 'target-to-best' strategies have shown worse results in general. However, the comparisons on separable and rotated variants of the same problems have demonstrated that the modifications are, for the most part, invariant to rotation. Further work is needed to explain the remaining sensibility to rotation.



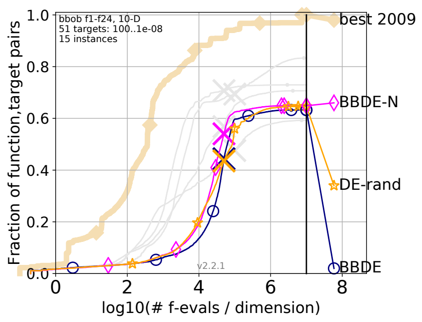
(a) Dimension 2



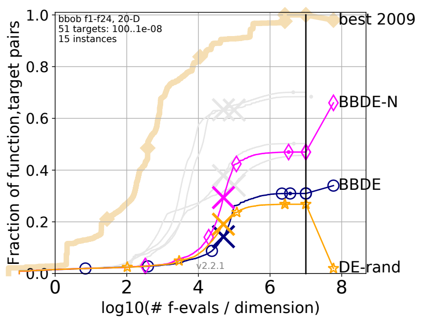
(b) Dimension 3



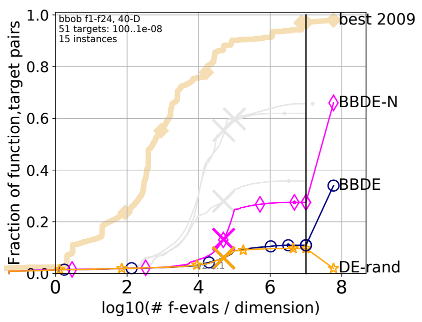
(c) Dimension 5



(d) Dimension 10

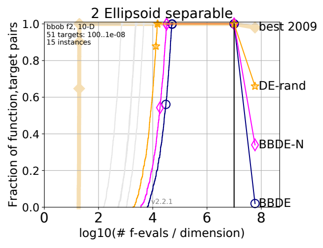


(e) Dimension 20

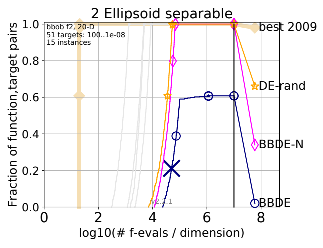


(f) Dimension 40

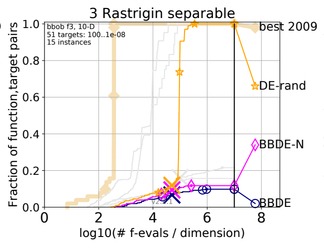
Figure 1: ECDFs of simulated (bootstrapped) runtimes of the 'rand' strategy for different problem dimensions (see Section 3.1 for more details), where the yellow line represents DE/rand, the purple line represents BBDE-N, and the blue line represents BBDE.



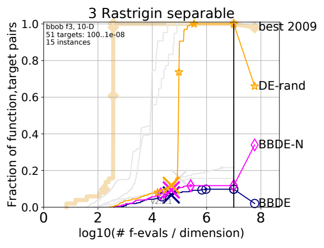
(a) Separable ellipsoid, f_2, dimension 10



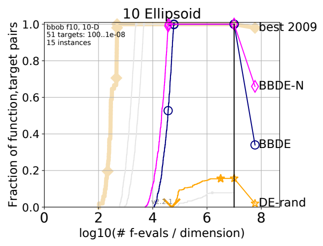
(b) Separable ellipsoid, f_2, dimension 20



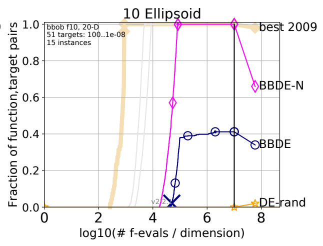
(c) Separable Rastrigin, f_3, dimension 10



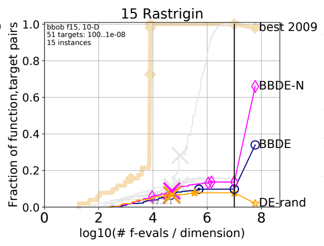
(d) Separable Rastrigin, f_3, dimension 20



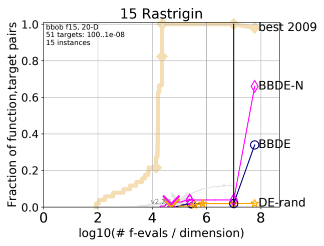
(e) Rotated ellipsoid, f_{10}, dimension 10



(f) Rotated ellipsoid, f_{10}, dimension 20

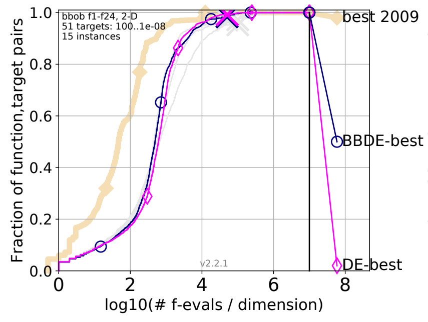


(g) Rotated Rastrigin, f_{15}, dimension 10

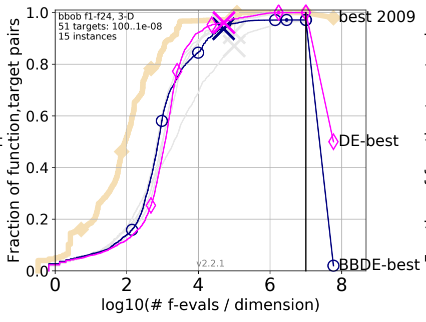


(h) Rotated Rastrigin, f_{15}, dimension 20

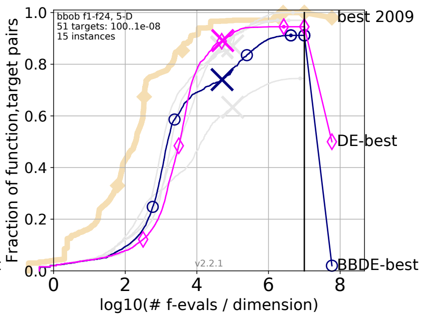
Figure 2: ECDFs of simulated (bootstrapped) runtimes of the 'rand' strategy for selected functions and dimensions 10 and 20 (see Section 3.1 for more details), where the yellow line represents DE/rand, the purple line represents BBDE-N, and the blue line represents BBDE. The top four plots show results on separable versions of the ellipsoid and Rastrigin problems (functions f_2 and f_3), while the bottom four plots present the results on the rotated versions of these problems (functions f_{10} and f_{15}).



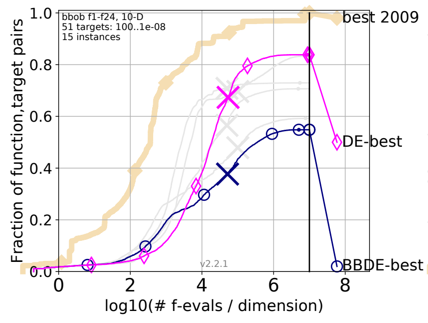
(a) Dimension 2



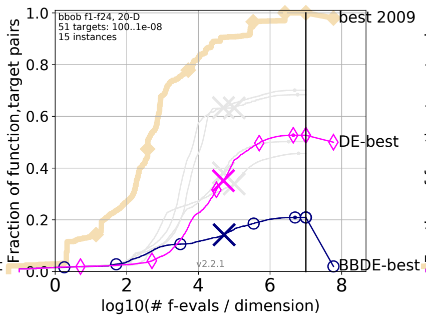
(b) Dimension 3



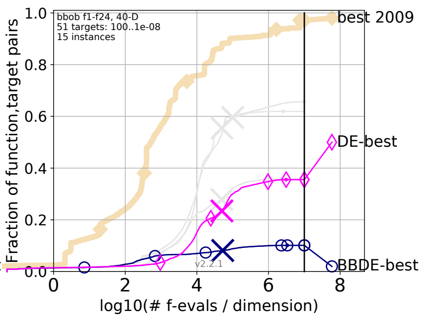
(c) Dimension 5



(d) Dimension 10

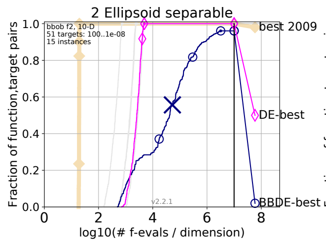


(e) Dimension 20

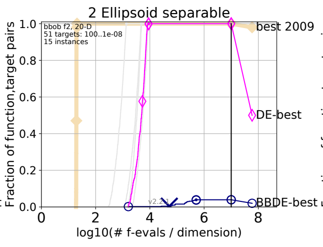


(f) Dimension 40

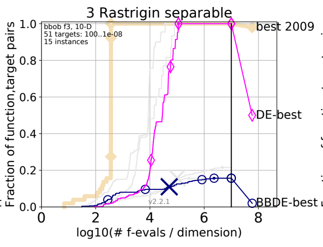
Figure 3: ECDFs of simulated (bootstrapped) runtimes of the 'best' strategy for different problem dimensions (see Section 3.1 for more details), where the purple line represents DE/best, and the blue line represents BBDE/best.



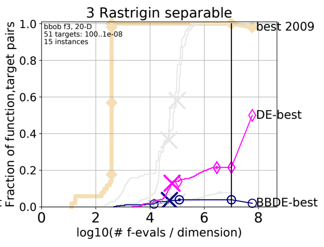
(a) Separable ellipsoid, f_2, dimension 10



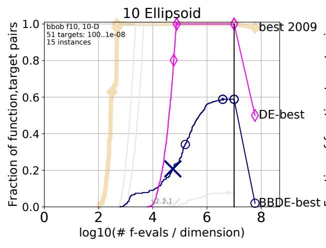
(b) Separable ellipsoid, f_2, dimension 20



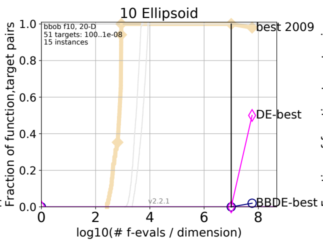
(c) Separable Rastrigin, f_3, dimension 10



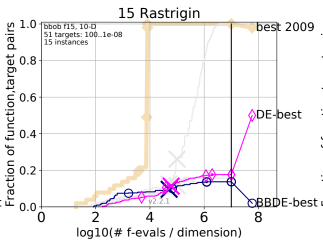
(d) Separable Rastrigin, f_3, dimension 20



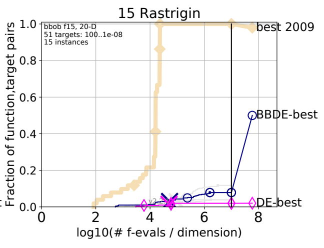
(e) Rotated ellipsoid, f_{10}, dimension 10



(f) Rotated ellipsoid, f_{10}, dimension 20

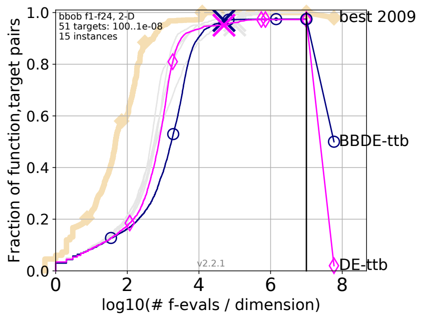


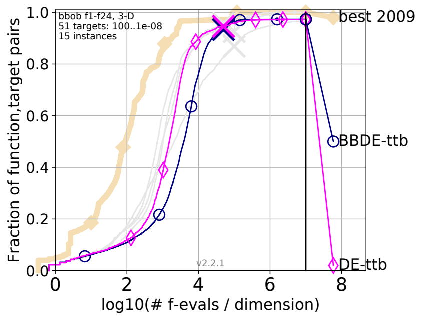
(g) Rotated Rastrigin, f_{15}, dimension 10

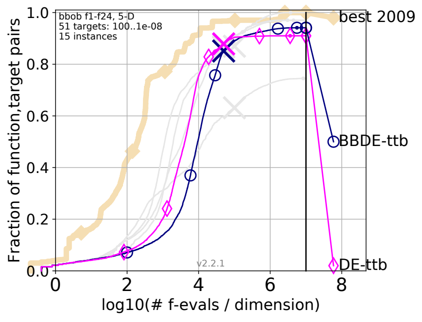


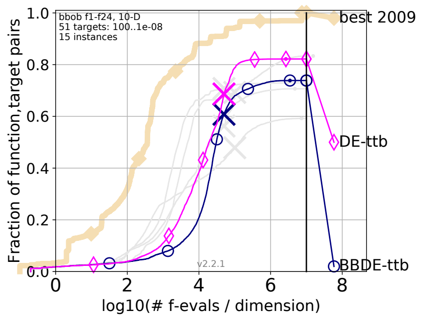
(h) Rotated Rastrigin, f_{15}, dimension 20

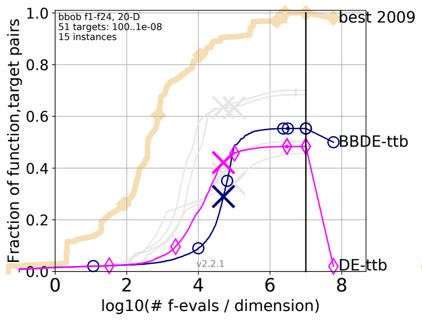
Figure 4: ECDFs of simulated (bootstrapped) runtimes of the 'best' strategy for selected functions and dimensions 10 and 20 (see Section 3.1 for more details), where the purple line represents DE/best and the blue line represents BBDE/best. The top four plots show results on separable versions of the ellipsoid and Rastrigin problems (functions f_2 and f_3), while the bottom four plots present the results on the rotated versions of these problems (functions f_{10} and f_{15}).


(a) Dimension 2


(b) Dimension 3


(c) Dimension 5


(d) Dimension 10


(e) Dimension 20

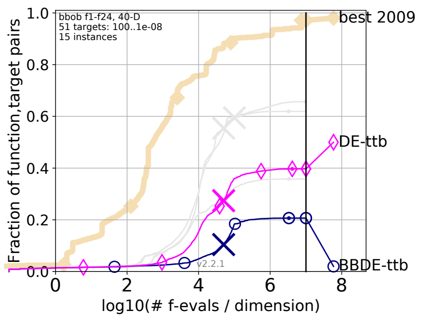
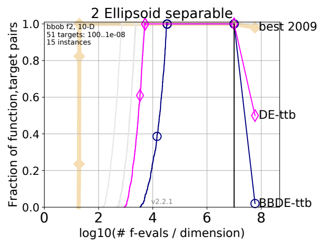
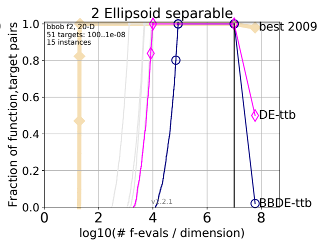
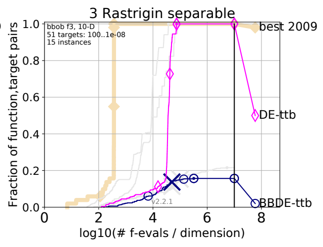
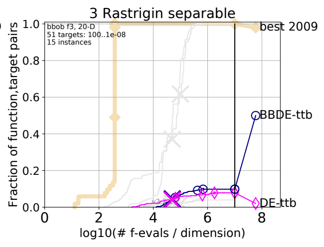

(f) Dimension 40

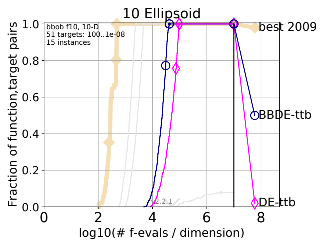
Figure 5: ECDFs of simulated (bootstrapped) runtimes of the 'target-to-best' strategy for different problem dimensions (see Section 3.1 for more details), where the purple line represents DE/target-to-best, and the blue line represents BBDE/target-to-best.

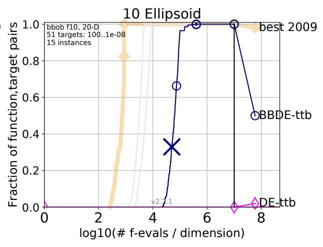

(a) Separable ellipsoid, f_2, dimension 10

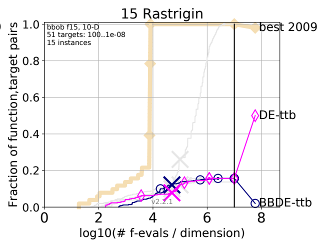

(b) Separable ellipsoid, f_2, dimension 20


(c) Separable Rastrigin, f_3, dimension 10


(d) Separable Rastrigin, f_3, dimension 20


(e) Rotated ellipsoid, f_{10}, dimension 10


(f) Rotated ellipsoid, f_{10}, dimension 20


(g) Rotated Rastrigin, f_{15}, dimension 10

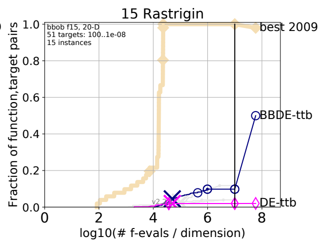

(h) Rotated Rastrigin, f_{15}, dimension 20

Figure 6: ECDFs of simulated (bootstrapped) runtimes of the 'target-to-best' strategy for selected functions and dimensions 10 and 20 (see Section 3.1 for more details), where the purple line represents DE/target-to-best and the blue line represents BBDE/target-to-best. The top four plots show results on separable versions of the ellipsoid and Rastrigin problems (functions f_2 and f_3), while the bottom four plots present the results on the rotated versions of these problems (functions f_{10} and f_{15}).

Further experiments are also necessary to establish whether the 'best' and 'target-to-best' strategies can (or cannot) be modified so that they satisfy the demands of black-box optimization and at the same time improve the performance. For example, one could investigate whether an approach similar to the normalization used in BBDE-N could be used for these variants as well. In addition, there are several differences between DE and BBDE. Thus, it is not evident which differences are important and which are not, since here we make a comparison of algorithms in a general sense. Although it seems that the omission of crossover is the most influential modification, a more detailed investigation would be needed to confirm this.

ACKNOWLEDGMENTS

We acknowledge financial support from the Slovenian Research Agency (young researcher program, project no. Z2-8177, and research core funding no. P2-0209). This work is also part of a project that has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 692286.

REFERENCES

[1] Swagatam Das, Sankha Subhra Mullick, and P.N. Suganthan. 2016. Recent advances in differential evolution – An updated survey. *Swarm and Evolutionary Computation* 27 (2016), 1–30. https://doi.org/10.1016/j.swevo.2016.01.004

[2] Á Fialho, W. Gong, and Z. Cai. 2010. Probability Matching-based Adaptive Strategy Selection vs. Uniform Strategy Selection within Differential Evolution: An Empirical Comparison on the BBOB–2010 Noiseless Testbed. In *Companion Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2010)*. 1527–1534.

[3] S. Finck, N. Hansen, R. Ros, and A. Auger. 2014. *Real-Parameter Black-Box Optimization Benchmarking 2010: Noiseless Functions Definitions*. Technical Report RR-6829. INRIA, Saclay, France.

[4] N. Hansen, A Auger, D. Brockhoff, D. Tušar, and T. Tušar. 2016. COCO: Performance Assessment. *ArXiv e-prints* https://arxiv.org/abs/1605.03560arXiv:1605.03560 (2016), 16.

[5] N. Hansen, A. Auger, O. Mersmann, T. Tušar, and D. Brockhoff. 2016. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *ArXiv e-prints* arXiv:1603.08785 (2016), 10.

[6] Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001. SciPy: Open source scientific tools for Python. (2001). http://www.scipy.org/ Accessed on 2018-03-21.

[7] Ferrante Neri and Ville Tirronen. 2010. Recent advances in differential evolution: A survey and experimental analysis. *Artificial Intelligence Review* 33, 1 (2010), 61–106. https://doi.org/10.1007/s10462-009-9137-2

[8] P. Pošík and V. Klemš. 2012. Benchmarking the Differential Evolution with Adaptive Encoding on Noiseless Functions. In *Companion Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2012)*. 189–196.

[9] K. V. Price. 2017. How symmetry constrains evolutionary optimizers. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2017)*. 1712–1719. https://doi.org/10.1109/CEC.2017.7969508

[10] K. V. Price. 2017. How Symmetry Constrains Evolutionary Optimizers: A Black Box Differential Evolution Case Study. (2017). Keynote at the IEEE Congress on Evolutionary Computation (CEC 2017), San Sebastian, Spain. https://origin.ieeetv.ieee.org/ Accessed on 2018-03-21.

[11] K. V. Price, R. M. Storn, and J. A. Lampinen. 2005. *Differential Evolution*. Springer, Chapter 3, 139–156.

[12] Guido Rossum. 1995. *Python Reference Manual*. Technical Report CS-R9526. Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands.

[13] R. Tanabe and A. Fukunaga. 2015. Tuning differential evolution for cheap, medium, and expensive computational budgets. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2015)*. 2018–2025.