New Benchmark Functions for Single-Objective Optimization Based on a Zigzag Pattern

JAKUB KUDELA AND RADOMIL MATOUSEK
Institute of Automation and Computer Science, Brno University of Technology, 601 90 Brno, Czech Republic
Corresponding authors: Jakub Kudela (jakub.kudela@vutbr.cz) and Radomil Matousek (matousek@fme.vutbr.cz)

This work was supported by IGA Brno University of Technology under Grant FSI-S-20-6538.

ABSTRACT Benchmarking plays a crucial role in both development of new optimization methods, and in conducting proper comparisons between already existing methods, particularly in the field of evolutionary computation. In this paper, we develop new benchmark functions for bound-constrained single-objective optimization that are based on a zigzag function. The proposed zigzag function has three parameters that control its behaviour and difficulty of the resulting problems. Utilizing the zigzag function, we introduce four new functions and conduct extensive computational experiments to evaluate their performance as benchmarks. The experiments comprise of using the newly proposed functions in 100 different parameter settings for the comparison of eight different optimization algorithms, which are a mix of canonical methods and best performing methods from the Congress on Evolutionary Computation competitions. Using the results from the computational comparison, we choose some of the parametrization of the newly proposed functions to devise an ambiguous benchmark set in which each of the problems introduces a statistically significant ranking among the algorithms, but the ranking for the entire set is ambiguous with no clear dominating relationship between the algorithms. We also conduct an exploratory landscape analysis of the newly proposed benchmark functions and compare the results with the benchmark functions used in the Black-Box-Optimization-Benchmarking suite. The results suggest that the new benchmark functions are well suited for algorithmic comparisons.

INDEX TERMS Numerical optimization, benchmarking, single objective problems, exploratory landscape analysis.

I. INTRODUCTION

Benchmarking plays a pivotal part in the development of new algorithms as well as in the comparison and assessment of contemporary algorithmic ideas [1]. For instance, one of the most powerful classes of algorithms for solving black-box optimization problems are Evolutionary Algorithms (EAs). An issue with EAs is that there are only a few theoretical performance results, which means that their performance comparisons and development rely heavily on benchmarking. These benchmarking experiments are constructed for performance comparisons on given classes of problems and should support the selection of appropriate algorithms for a given real-world application [2]. Another utilization of benchmarking is in the qualification of the theoretical predictions of the behaviour of algorithms [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Ran Cheng.

There are, currently, two main lines of development in benchmarking for EAs, the IEEE Congress on Evolutionary Computation (CEC) competitions [4], that have been running in the current form since 2013, and the Black-Box-Optimization-Benchmarking [5] workshops (BBOB workshops), which started in 2009. Both of these venues provide a variety of materials that describe the various benchmarks, and provide code for the best-performing algorithms in a given year. Also, the BBOB benchmarks are run on a platform called the Comparing Continuous Optimizer (COCO) benchmark suite [6]. The COCO suite serves as a place for comparing various algorithms for unconstrained continuous numerical optimization. On the other hand, the CEC competitions that are organized every year aim at comparing state-of-the-art stochastic search methods on test environments that are specifically crafted each year. These benchmark functions are developed from a set of commonly used benchmark functions, such as the Ackley’s function, Griewank’s function, Rastrigin’s function, Rosenbrock’s function, Schwefel’s...
function, and many others [7]. However, recent research has found that the BBOB and CEC benchmark suites have a poor coverage of the corresponding problem space [8]–[10], have a very similar distribution in certain landscape analysis measures [11], and are highly correlated from the perspective of the performance of different algorithms [12]–[14].

In this paper, we introduce four new benchmark functions for bound-constrained single-objective optimization that are based on a zigzag pattern, and are non-differentiable and highly multimodal. Inspired by recently proposed tunable benchmark functions for combinatorial problems [15], the newly proposed functions have three parameters that can change their behaviour and difficulty. We conduct extensive numerical experiments to investigate how the different parametrizations work as benchmarks and show that they can be successfully used for algorithmic comparisons.

The rest of the paper is organized as follows. Section II introduces the individual components of the zigzag function and the newly proposed benchmark functions, and provides insight into their construction. In Section III we report on extensive computational experiments where we compare 100 different parametrizations of the newly proposed benchmark functions on eight EAs that are a mix of canonical methods as well as the best performing methods from the CEC competitions. In Section IV we devise an ambiguous benchmark set that is based on selected parametrizations of the benchmark functions. The individual problems in this set displayed good capabilities in giving unambiguous rankings for the considered algorithms, but taken as a whole set produced no clear dominating relationship between the results for the considered algorithms. In Section V, we conduct exploratory landscape analysis of the proposed functions and show how they compare to the functions from the BBOB benchmark suite. The conclusions and future research directions are outlined in Section VI.

II. NEW BENCHMARK FUNCTIONS

The newly proposed benchmark functions are constructed in the following way. First, a so-called “zigzag” function $z(x, k, m, \lambda)$ (or triangular wave) function is constructed based on the formula in (1), as shown at the bottom of the page, where $\lfloor \cdot \rfloor$ is the floor operator. Apart from the point $x \in \mathbb{R}$ at which it should be evaluated, the zigzag function also contains three other parameters: $k > 0$ that controls its period, $m \in [0, 1]$ that controls its amplitude, and $\lambda \in (0, 1)$ that controls the location of its local minima. The effect of the individual parameters on the shape of the zigzag function can be seen in Fig. 1. For $m = 0$ the zigzag function is identically equal to 1 for any point $x$ (regardless of the values of the other two parameters).

In the following step, we construct four basic benchmark functions (F1-F4) that combined the zigzag function with different multimodal functions, and that inherit the parameters from the zigzag function. Their construction starts with four 1-D functions $\phi_1, \ldots , \phi_4$, which are formulated in (2), as shown at the bottom of the page.

The common theme of all of these functions is that they are bounded on the interval $[-200, 200]$ by a maximum value of 200 (which allows for composing the functions with themselves without running to numerical difficulties), and there is a single global minimum located at 0, with function value 0. Although the optimization will take place only on the interval $[-100, 100]$ we will use a shift vector to change the placement of the optimal point (hence, the need for the functions to be “well-behaved” on $[-200, 200]$).

To obtain the benchmark functions for a dimension $D$, we use a simple sum of the functions $\phi$ for the individual components, but modify the inputs by a shift vector $s \in [-100, 100]^D$ and a rotation/scaling matrix $M$:

$$f_j(x, k, m, \lambda) = \sum_{i=1}^{D} \phi_i(x_i, k, m, \lambda) \quad j = 1, \ldots , 4$$

$$F_j(x, k, m, \lambda) = f_j(M_j(x - s_j), k, m, \lambda) \quad j = 1, \ldots , 4$$

The matrices $M$ are constructed in the exact same manner as in [16]. Both of these components are an integral part of good benchmark functions [17], as they create some additional difficulty for the various optimization algorithms [4], [18]. The shapes of the functions F1-F4 in one dimension (without any shift or scaling) for selected

\[
\begin{align*}
\phi_1(x, k, m, \lambda) &= 3 \cdot 10^{-9} |x - 40)|x - 185|x(x + 50)(x + 180)|z(x, k, m, \lambda) + 10| \sin(0.1x)| \\
\phi_2(x, k, m, \lambda) &= \phi_1(x, k, m, \lambda), k, m, \lambda) \\
\phi_3(x, k, m, \lambda) &= 3|\ln(1000|x| + 1)|z(x, k, m, \lambda) + 30 - 30| \cos\left(\frac{x}{10\pi}\right)| \\
\phi_4(x, k, m, \lambda) &= \phi_3(x, k, m, \lambda), k, m, \lambda) 
\end{align*}
\]
values of the parameters can be seen in Fig. 2, while their corresponding contour and surface plots for 2-D (this time, with shift and rotation/scaling) can be found in Appendix A. The individual parameters of the zigzag function, apart for their primary role in defining the shape of the zigzag, also serve secondary roles. The parameter $k$ can be though of as a “ruggedness” parameter, that makes the zigzag more frequent. The parameter $\lambda$ can be seen as a “deception” parameter, as low values of $\lambda$ skew the function in such a way that its derivatives (if existing) point predominantly away from the global optimum (in the case of the 1-D examples in Fig. 2, away from $x = 0$). Higher values of the parameter $m$ result in a “deeper” local optima. By setting these parameters to different values, we should be able to increase/decrease the difficulty of the resulting optimization problems [15], and to bring out the advantages and the disadvantages of different optimization methods.

### III. COMPARISON ON ALGORITHMS

#### A. ALGORITHM SELECTION AND EXPERIMENTAL SETTING

In this section, we investigate the behaviour of selected EAs on the newly proposed benchmark functions for different values of the input parameters. We chose two main categories of algorithms for this investigation: a) generally known and used EAs, b) best performing algorithms from the CEC Competitions on Bound Constraint Single Objective Optimization [4]. The other unifying theme of the chosen algorithms was that they had their code publicly available for the MATLAB language. The eight selected algorithms were:

- Adaptive Gaining-Sharing Knowledge (AGSK) – the runner-up of the CEC’20 competition. The algorithm improves and extends upon original GSK [19] algorithm by adding adaptive settings to two main control parameters: the knowledge factor and the knowledge ratio, that control junior and senior gaining
3.2 GHz Core i5 processor, 16 GB RAM, and Windows 10.

- Covariance Matrix Adaptation Evolution Strategy (CMAES) – a canonical algorithm that adapts the covariance matrix of a mutation distribution [21].
- Differential Evolution (DE) – a canonical algorithm, one of the most widely used ones for continuous optimization [22]. The implementation and parameter settings for DE used in this paper was the one that was shipped alongside the benchmark suite for the CEC’21 Competition.
- Hybrid Sampling Evolution Strategy (HSES) – winner of the CEC’18 Competition. An evolution strategy optimization algorithm which combined CMAES and the univariate sampling method [23].
- Improved Multi-operator Differential Evolution (IMODE) – winner of the CEC’20 Competition. This algorithm employs multiple differential evolution operators and a sequential quadratic programming local search technique for accelerating its convergence [24].
- Linear Population Size Reduction SHADE (LSHADE) – one of the most popular variants of adaptive DE, that was used as a basis for many of the best performing algorithms in the CEC Competitions in past few years [25].
- Multiple Adaptation DE Strategy (MadDE) – one of the best performing algorithms in the CEC’21 competition. This is another modification of the DE algorithm that uses a multiple adaptation strategy for its search mechanisms and for adapting its control parameters at the same time [26].
- Particle Swarm Optimization (PSO) – another canonical algorithm that simulates swarm behavior of various social animals such as the fish schooling or bird flocking [27]. The implementation and parameter settings for PSO used in this paper was the one that was shipped alongside the benchmark suite for the CEC’20 Competition.

To investigate the impact of the different values of the parameters of the benchmark functions, we chose 100 parameter settings (for all functions F1-F4) for computational evaluations, which are reported in Table 1. For the individual computations, we used rules similar to the CEC’21 competition: the eight algorithms were evaluated on the four benchmark functions with 100 different parameter settings with \( D = [5, 10, 15] \) dimensions, and a search space of \([−100, 100]^D\). The maximum number of function evaluations were set to 50,000, 200,000, 500,000 fitness function evaluations for problems with \( D = [5, 10, 15] \), respectively. All algorithms were run 30 times to get representative results. For every run, if the objective function value of the resulting solution was less than or equal to 1E–8, it was considered as zero.

For all algorithms, we used the same parameter settings that was used in the corresponding CEC competition [28]. All algorithms were run in a MATLAB R2020b, on a PC with 3.2 GHz Core i5 processor, 16 GB RAM, and Windows 10.

The particular values of matrices \( M \) and shifts \( s \), as well as the implementation of the benchmark functions in MATLAB, can be found in the authors github.1

### B. RESULTS

First, we investigate the “difficulty” of the four benchmark functions (with the 100 different parameter setting) by counting the number of unsolved instances, i.e., the number of times out of the 30 runs that any of the eight algorithms was not able to reach function value of at least 1E–8 within the specified number of function evaluations. These results are summarized in Fig. 3. The first thing to notice is that the basic forms of the benchmark functions with no zigzag (with \( m = 0 \)) were not “impossible” to solve for at least some of the algorithms, regardless of the dimension. However, the instances were also not “too easy” so that the algorithms could reliably find the optimum – this, in our opinion, makes these benchmark functions worth investigating. Unexpectedly, increasing the dimension of the problem led to increase in the number of unsolved instances. The parameter \( m \) displayed the highest effect on the difficulty of the resulting problem – larger values of \( m \) (meaning a more pronounced zigzag) generated problems that were harder to solve. Similarly, lower values of the parameter \( \lambda \in [0.01, 0.1] \) generally produced more difficult instances. The effects of different values of the parameter \( k \) were more subtle and function-dependent. For F1 and F2, increasing the period of the zigzag together with low values of \( \lambda \) produced the most difficult instances. For F3, increasing \( k \) exhibited a decrease in difficulty. Finally, for F4, the effect of \( k \) was mixed, with some instances being more or less difficult with increasing \( k \).

More interesting result can be found in comparing the individual algorithms on the different benchmark functions. For this comparison, we used the IOHprofiler [29], which is a benchmarking and profiling tool for optimization heuristics. Within the IOHprofiler, we chose the comparison based on [29].

| ID | \( k \) | \( m \) | \( \lambda \) | ID | \( k \) | \( m \) | \( \lambda \) | ID | \( k \) | \( m \) | \( \lambda \) |
|----|------|------|------|----|------|------|------|----|------|------|------|
| 1  | 1   | 0   | 0.01 | 2  | 1   | 0.1  | 0.01 | 5  | 1   | 0.1  | 0.01 |
| 2  | 2   | 0.10| 0.01| 27 | 2   | 0.50| 0.01| 52 | 2   | 0.90| 0.01 |
| 3  | 4   | 0.10| 0.01| 28 | 4   | 0.50| 0.01| 53 | 4   | 0.90| 0.01 |
| 4  | 8   | 0.10| 0.01| 29 | 8   | 0.50| 0.01| 54 | 8   | 0.90| 0.01 |
| 5  | 16  | 0.10| 0.01| 30 | 16  | 0.50| 0.01| 55 | 16  | 0.90| 0.01 |
| 6  | 1   | 1.00| 0.01| 31 | 1   | 1.00| 0.01| 56 | 1   | 1.00| 0.01 |
| 7  | 7   | 0.10| 0.01| 32 | 2   | 0.50| 0.10| 57 | 2   | 0.90| 0.10 |
| 8  | 4   | 0.10| 0.10| 33 | 4   | 0.50| 0.10| 58 | 4   | 0.90| 0.10 |
| 9  | 8   | 0.10| 0.10| 34 | 8   | 0.50| 0.10| 59 | 8   | 0.90| 0.10 |
| 10 | 16  | 0.10| 0.10| 35 | 16  | 0.50| 0.10| 60 | 16  | 0.90| 0.10 |
| 11 | 1   | 0.50| 0.50| 36 | 1   | 1.00| 0.50| 61 | 1   | 1.00| 0.50 |
| 12 | 2   | 0.50| 0.50| 37 | 2   | 0.50| 0.50| 62 | 2   | 0.50| 0.50 |
| 13 | 4   | 0.50| 0.50| 38 | 4   | 0.50| 0.50| 63 | 4   | 0.50| 0.50 |
| 14 | 8   | 0.50| 0.50| 39 | 8   | 0.50| 0.50| 64 | 8   | 0.50| 0.50 |
| 15 | 16  | 0.50| 0.50| 40 | 16  | 0.50| 0.50| 65 | 16  | 0.50| 0.50 |
| 16 | 1   | 0.90| 0.90| 41 | 1   | 1.00| 0.90| 66 | 1   | 1.00| 0.90 |
| 17 | 2   | 0.90| 0.90| 42 | 2   | 0.90| 0.90| 67 | 2   | 0.90| 0.90 |
| 18 | 4   | 0.90| 0.90| 43 | 4   | 0.90| 0.90| 68 | 4   | 0.90| 0.90 |
| 19 | 8   | 0.90| 0.90| 44 | 8   | 0.90| 0.90| 69 | 8   | 0.90| 0.90 |
| 20 | 16  | 0.90| 0.90| 45 | 16  | 0.90| 0.90| 70 | 16  | 0.90| 0.90 |
| 21 | 1   | 0.99| 0.99| 46 | 1   | 0.99| 0.99| 71 | 1   | 0.99| 0.99 |
| 22 | 2   | 0.99| 0.99| 47 | 2   | 0.99| 0.99| 72 | 2   | 0.99| 0.99 |
| 23 | 4   | 0.99| 0.99| 48 | 4   | 0.99| 0.99| 73 | 4   | 0.99| 0.99 |
| 24 | 8   | 0.99| 0.99| 49 | 8   | 0.99| 0.99| 74 | 8   | 0.99| 0.99 |
| 25 | 16  | 0.99| 0.99| 50 | 16  | 0.99| 0.99| 75 | 16  | 0.99| 0.99 |

1https://github.com/JakubKudela89/Zigzag
on a fixed-budget (defined by the maximum number of function evaluations) and compared the algorithms on each benchmark function and each dimension separately for all the 100 parameter settings. The ranking of the algorithms was obtained by using a Deep Statistical Comparison (DSC) analysis [30], which works by comparing distributions of the obtained solutions instead of using descriptive statistics. For the visualization of the results, we employed the PerformViz approach [31], which shows the algorithms that are most suited for a given problem and similarities among both problems and algorithms, within a single plot. The threshold for statistical significance was set to 0.05 for all comparisons. The results of the comparisons are shown in Fig. 4 and Fig. 5. For a particular problem, if a rectangle, that represents a certain problem (horizontal dimension) and a certain algorithm (vertical dimension), is bluer then the algorithm is better for the given problem (and, the redder it is, the worse is the algorithm). In the horizontal dimension, if the color of the rectangles remains the same, it signals that the ranking of the given algorithm remains the same across different problems. On the other hand, if the rectangles have the same color in the vertical dimension, it means that there was no
significant difference between the performance of the individual algorithms (and, probably, that this particular parameter settings is not very well suited to serve as a benchmark function).

For F1 in $D = 5$, there is a large section of parameter values that produced problems on which there was no significant difference among most of the algorithms. These were the parameter settings with either low $m \leq 0.1$, or lower $m \leq 0.5$ and higher $\lambda \geq 0.9$. The rest of the problems produced relatively stable rankings, where LSHADE performed the best, followed by DE and AGSK. The worst performing methods were PSO and HSES. Interestingly, there was a group of parameter settings for which otherwise well-performing DE struggled (and was ranked as the worst) – these settings corresponded to high values of $m \geq 0.9$ and $k \geq 8$ and lower values of $\lambda \leq 0.5$. For $D = 10$, there were only a few instances on which the algorithms were not well comparable. There were, however, two large groups of parameter settings on which two seemingly similar algorithms, CMAES and HSES, performed very differently. For the first group, that can be characterized by lower values of $m \leq 0.5$, CMAES ranked as a third best, while HSES ranked as the worst. For the second group ($m \geq 0.9$), the ranking of HSES became much better (between second and fifth), while CMAES became the worst ranking one. Overall, LSHADE remained the best performing algorithm, followed by DE, while PSO again ranked among the worst. A very similar pattern can be observed also for $D = 15$, where rankings of HSES and MadDE depended heavily on the value of $m$. A notable difference is that MadDE became the best performing algorithm, followed by LSHADE and DE.

For F2 in $D = 5$, there were no notable parameter settings that would result in an ambiguous ranking of the algorithms. The DE method experienced the largest variability in performance, while it ranked among the worst for problems with high values of $k \geq 8$ and $m \geq 0.9$, it was the best ranked algorithm for problems with low $m = 0.1$, and mediocre for the other problem settings. This time, the best performing algorithm overall was IMODE, followed by LSHADE and MadDE, while PSO performed the worst. For $D = 10$, there were again instances for which the ranking was ambiguous, but this time they corresponded to the “difficult” instances.
FIGURE 5. Comparison of the algorithms on different parameters settings for benchmark functions F3-F4, $D = \{5, 10, 15\}$.

with larger values of $m > 0.9$ and $k \geq 8$, and low $\lambda \leq 0.1$. Similarly to the previous case, DE performed well on the instances with low values of $m = 0.1$, but its performance degraded for larger $m$. The best ranked algorithm was LSHADE, followed by MadDE, IMODE and HSES. This time, the worst performing methods were CMAES and AGSK. For $D = 15$, the situation changed quite drastically. The parameter settings resulting in ambiguous ranking were the ones with low values of $k \leq 4$. Once again, the ranking of DE depended heavily on $m$. By far the best performing method was the canonical CMAES, while the worst were AGSK and PSO.

For F3 there were no parameter settings that would result in an ambiguous ranking in any of the investigated dimensions. For $D = 5$, the methods that were the most impacted by changes in parameters were HSES and CMAES, which both benefited from low values of $k \leq 4$ and high $m \geq 0.9$, and DE, which struggled for higher values of $m \geq 0.9$ but otherwise ranked among the best. A peculiar behaviour can be seen for AGSK, that performed relatively well for problems with either low values of $m = 0.1$, or for $m = 0.5$ with larger values of $\lambda \geq 0.9$, or for $m \geq 0.9$ but only with $\lambda = 0.9$ and $k = 16$. Otherwise, AGSK performed consistently among the worst, along with CMAES and PSO, while the best ranking algorithm was LSHADE, with HSES and DE also ranking high for some settings. For $D = 10$, the issues that DE had with high values of $m$ remained. Meanwhile, HSES performed very well for problems with $m \geq 0.9$, while LSHADE and DE dominated the rest. Interestingly, CMAES also performed relatively well, apart from the instances with $m = 0.9$. The consistently worst performing algorithms were AGSK and PSO. A very similar pattern also appeared in $D = 15$, where HSES and CMAES dominated the instances with $m \geq 0.9$, while LSHADE and DE dominated the rest.

For F4, there were a few instances that resulted in an ambiguous ranking, but we did not notice any clear pattern. For $D = 5$, there was a large variability in the best performing algorithm, as all algorithms apart from PSO and AGSK were ranked as the best one for some parameter settings. HSES struggled with problems with both $m \geq 0.5$ and $\lambda \leq 0.1$, but otherwise performed the best. CMAES had a similar
dependence on the parameters, apart from a few instances with lower values of \( k \leq 2 \), where it performed very well. LSHADE, IMODE, and MadDE performed consistently well on all instances, while DE performed well on instances with either low values of \( m = 0.1 \), or on instances that had simultaneously high values of \( m \geq 0.9, \lambda \geq 0.9 \), and \( k = 16 \). For \( D = 10 \), HSES and CMAES ranked the best in the majority of instances. What separated them was that CMAES was better ranked for problems with either \( m = 0.1 \), or with \( m = 0.2 \) together with \( k \leq 2 \), while HSES dominated the rest. IMODE showed mediocre performance, apart from instances with \( m = 0.1 \), where it performed significantly worse, and where, coincidentally, DE performed much better than on the other instances. LSHADE ranked consistently at the top, while PSO, AGSK and DE (for \( m \geq 0.5 \)) ranked at the bottom. Finally, for \( D = 15 \), the best performing algorithm again depended mainly on \( m \). For \( m \geq 0.9 \), HSES was by far the best one (while DE was the worst), while for \( m \leq 0.5 \), DE and CMAES were ranked as the best. Both LSHADE and MadDE performed consistently well across all instances, while PSO and AGSK were the worst.

Generally speaking, the performance of some algorithms, such as DE, HSES, and CMAES, exhibited high dependence on the parametrization of the benchmark functions, while others showed only little dependence. Among the three parameters, \( m \) had the most pronounced effect on the rankings, while \( \lambda \) and \( k \) showed significant effect mainly for particular combinations off all three parameters (in a similar fashion to the investigation of the number of unsolved instances).
FIGURE 8. Parallel plots of the results of ELA for different parametrizations of F1–F4, $D = 10$, grouped by the value of the parameter $m$. 
On the other hand, most of the parametrizations displayed good ability to differentiate between the various algorithms. The algorithm that performed consistently well across the different benchmarks was LSHADE – it is, then, unsurprising that it was chosen as the basis of many of the best performing algorithms for the CEC competitions in recent years. Naturally, studying the performance of the algorithms on problems in even higher dimensions, or with modified number of function evaluations, could bring more varied results.

**IV. CREATING AMBIGUOUS BENCHMARK SET**

In this section, we select a few parametrizations of the proposed benchmark functions to create a benchmark set. This selection had several goals. The first was to choose parametrizations that result in a clear ranking (for the specific problem). The second was to choose parametrizations across the range of possible values of the parameters. And the third was to select the parametrizations in such a way that the rankings for the resulting benchmark set are ambiguous. The last goal corresponds to having a benchmark set that is comprised of functions on which different algorithms perform differently, without a clear favourite. This selection was done by carefully examining the results of the comparison of the algorithms and choosing, for each benchmark function, two parametrization, that together resulted in the ambiguous benchmark set. We also decided to include additional dimension $D = 20$, with a maximum of 1,000,000 function evaluations for these parametrizations (this was excluded from the previous comparisons because of excessive computational requirements). This resulted in the creation of a benchmark set with 32 instances (four functions F1-F4, two parametrizations each, four different dimension), that are summarized in Table 2. The 1-D plots of the chosen functions can be seen in Fig. 2, while their 2-D contour and surface plots can be seen in the Appendix A.

Detailed results of the computations, including convergence analysis and detailed statistics, can be found in the Appendix B. The results of the comparison of the eight chosen algorithms on the ambiguous benchmark set are presented in Fig. 6. From this comparison, it can be seen that most of
the algorithms, apart from PSO and AGSK, were best-ranked for at least one of the problems in the set. Also, for at least one problem, all algorithms placed in the bottom half of the rankings. This signals that the chosen parametrizations cover a wide range of the single-objective optimization problem space (measured by the performance of different algorithms). However, it is clear from the results that some of the selected algorithms performed consistently better than others. To quantify this relationship, we used the post-hoc test from the aforementioned DSC framework, the results of which are shown in Fig. 7. For a given row (one selected algorithm), if a column has a red color, it means that the selected algorithm performs significantly better (with a statistical significance threshold of 0.05) on the benchmark set that the algorithm in the column. If it has a blue color, it means that it performs significantly worse, and, if it has a grey color, the ranking is ambiguous. From Fig. 7, we can see that PSO and AGSK are ranked as the worst (but with unclear comparison between the two), and that LSHADE is ranked better that AGSK, CMAES, DE, IMODE, and PSO (but not better that HSES and MadDE). Other than that, the algorithms are mutually incomparable on our ambiguous benchmark set.

V. EXPLORATORY LANDSCAPE ANALYSIS

To better explore the problem space that is covered by the different parametrizations of the proposed benchmark functions, we used the method of exploratory landscape analysis (ELA) [32], within the flacco library [33]. We focus only on the landscape features that have been found to be invariant under shift and scale [10] and the ones that provide

FIGURE 10. Contour and surface plots of the parametrization of F1-F4 used for the ambiguous benchmark set.
expressive results [8]. The 20 selected ELA measures were the following:

- cm_angle.angle.mean
- ela_distr.skewness
- ela_distr.kurtosis
- ela_distr.number_of_peaks
- ela_meta.lin_simple.adj_r2
- ela_meta.lin_interact.adj_r2
- ela_meta.quad_w_interact.adj_r2
- ela_meta.quad_simple.adj_r2
- ela_meta.lin_w_interact.adj_r2
- disp.ratio_mean_02
- disp.ratio_median_25
- nbc.nb_fitness.cor
- pca.expl_var_PC1.cov_init
- pca.expl_var.cov_init
- pca.expl_var.cor_init
- ic.eps.ratio
- ic.eps.s
- ic.h.max
- ic.m0
We chose only a single dimension $D = 10$ as the representative and used 8,000 function evaluations for robust computation of the features [8]. We performed additional analysis to find the effect of the parameters on the resulting ELA features and found that the main difference stemmed from varying the parameter $m$, which was the same parameter that had the highest impact on the number of unsolved instances and on the ranking of the selected optimization algorithms. The results of the ELA are summarized in Fig. 8, where the plotted lines are grouped based on the value of the parameter $m$. Although the results show high diversity in the values of the individual measures, it is hard to judge how diverse these values are compared to the commonly used benchmark functions. This is why we provide a comparison of all the parametrizations of F1-F4 and the parametrizations chosen for the ambiguous benchmark set with the benchmark functions from the BBOB suite [5], that are shown in Fig. 9. In general, the values of the ELA measures covered by the different parametrizations of the F1-F4 functions were comparable to the ones covered by the BBOB benchmark.
functions. Notably, some F1-F4 parametrizations covered values that were outside the range of the BBOB functions for measures cm_angle.angle.mean, disp.ratio_mean_02, disp.ratio_median_25, pca.expl_var_PC1.cov_init, ic.eps.s, and ic.eps.ratio. When looking at the parametrizations of F1-F4 for the ambiguous benchmark set, we can see that particularly the ELA measures of the chosen F3 and F4 parametrizations fall into places where there are only a few BBOB counterparts. This suggests that either including these benchmark functions into the BBOB suite, or creating a new benchmark suite that includes them, could provide a better coverage of the problem space of bound-constrained single-objective optimization problems.

VI. CONCLUSION
In this paper, we introduced four new benchmark functions for bound constrained single objective optimization that are based on a highly parametrizable zigzag pattern. The construction of these benchmark functions was straightforward enough to allow for a broad range of extensions, variations,
and further study. The extensive computational experiments showed that different parametrizations of these functions can serve as good benchmarks, and we were able to create an ambiguous benchmark set on which the ranking of the selected algorithms was unclear, although the ranking on the individual instances was clear-cut. These results, together with the conducted exploratory landscape analysis, suggest that the new benchmark functions are well suited for algorithmic comparisons. Future research directions will encompass comparing even broader selection of algorithms, particularly ones that were not very well represented by the studied methods (such as swarm intelligence algorithms). Another interesting path will be in investigating the role of bias in the optimal function value (in our experiments, all optima had the function value of 0). Finally, we plan on developing new multimodal benchmark functions [34] by utilizing the presented framework.

APPENDIX A
CONTOUR AND SURFACE PLOTS OF SELECTED FUNCTIONS
See Figure 10.
TABLE 3. Detailed statistics of the 30 runs of the selected algorithms on the ambiguous benchmark set.

| ID | AGSK | CMS | DET | HMS | DSTE | LSHAD | MSEH | ModDE | PSO |
|----|------|-----|-----|-----|------|-------|------|-------|-----|
| min | 0.079 | 4.1E-05 | 0.148 | 0 | 0 | 0.018 |
| median | 0.039 | 1.9E-05 | 0.182 | 0.165 | 0.045 | 0.139 | 0.783 |
| max | 0.391 | 1.7E-04 | 2.087 | 0.377 | 0.227 | 0.377 | 1.822 |
| std | 0.173 | 1.8E-04 | 0.987 | 0.105 | 0.099 | 0.150 | 0.930 |
| mean | 0.247 | 0.158 | 0.307 | 0.407 | 0.258 | 0.158 | 0.287 |
| 1 mean | 2.389 | 0.695 | 0.344 | 0.640 | 0.305 | 0.237 | 0.375 |
| 2 mean | 3.384 | 0.305 | 0.277 | 0.242 | 0.150 | 0.452 | 0.377 |
| 3 mean | 5.332 | 0.281 | 0.172 | 0.184 | 0.158 | 0.277 | 0.347 |
| 4 mean | 0.486 | 0.398 | 0.361 | 0.119 | 0.407 | 0.087 | 0.163 | 0.606 |
| 5 mean | 2.394 | 0.158 | 0.079 | 0.621 | 0.238 | 0.158 | 0.306 |
| 6 mean | 4.643 | 0.349 | 0.582 | 0.233 | 0.128 | 0.482 | 0.380 |
| 7 mean | 0.302 | 0.384 | 0.110 | 1.221 | 3.515 | 0.527 | 1.871 | 2.679 |
| 8 mean | 4.374 | 0.983 | 0.745 | 2.437 | 6.379 |
| 9 mean | 1.144 | 0.187 | 0.482 | 0.156 | 0.104 | 0.122 | 0.277 | 1.340 |

APPENDIX B

CONVERGENCE PLOTS AND DETAILED STATISTICS OF THE SELECTED ALGORITHMS ON THE AMBIGUOUS BENCHMARK SET

See Figures 11–14, and Table 3.

REFERENCES

[1] M. Hellwig and H.-G. Beyer, "Benchmarking evolutionary algorithms for single objective real-valued constrained optimization—A critical review," Swarm Evol. Comput., vol. 44, pp. 925–944, Feb. 2019.

[2] O. Mersmann, P. Preuss, J. Rubsamen, and C. Weihs, "Analyzing the BBOB results by means of benchmarking concepts," Evol. Comput., vol. 23, no. 1, pp. 161–185, Mar. 2015.

[3] R. Rardin and R. Uzsoy, "Experimental evaluation of heuristic optimization algorithms: A tutorial," J. Heuristics, vol. 7, pp. 261–304, May 2001.

[4] A. W. Mohamed, A. A. Hadi, A. K. Mohamed, P. Agrawal, A. Kumar, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2021 special session and competition on single objective bound constrained numerical optimization," Cairo Univ., Giza, Egypt, Tech. Rep., 2020. [Online]. Available: https://github.com/P-N-Suganthan/2021-SOBICO

[5] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter blackbox optimization benchmarking 2009: Noiseless functions definitions," Ph.D. dissertation, INRIA, Saclay, France, 2009.
[6] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff, “COCO: A platform for comparing continuous optimizers in a black-box setting,” Optim. Methods Softw., vol. 36, no. 1, pp. 114–144, Jan. 2021.

[7] M. Jamil and X.-S. Yang, “A literature survey of benchmark functions for global optimisation problems,” Int. J. Math. Model. Numer. Optim., vol. 4, no. 2, pp. 150–194, 2013.

[8] R. D. Lang and A. P. Engelbrecht, “An exploratory landscape analysis-based benchmark suite,” Algorithm., vol. 14, no. 3, p. 78, Feb. 2021.

[9] M. A. Muñoz and K. Smith-Miles, “Generating new space-filling test instances for continuous black-box optimization,” Evol. Comput., vol. 28, no. 3, pp. 379–404, Sep. 2020.

[10] U. Škvorc, T. Eftimov, and P. Korošec, “Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis,” Appl. Soft Comput., vol. 90, May 2020, Art. no. 106138.

[11] A. W. Mohamed, B. Bischof, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph, “Exploratory landscape analysis,” in Proc. 13th Annu. Conf. Genetic Evol. Comput., New York, NY, USA: ACM, 2011, pp. 829–836.

[12] R. Storn and K. Price, “Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces,” J. Global Optim., vol. 11, no. 4, pp. 341–359, 1997.

[13] G. Zhang and Y. Shi, “Hybrid sampling evolution strategy for solving single objective bound constrained problems,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jul. 2018, pp. 1–7.

[14] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, “Improved multi-operator differential evolution algorithm for solving unconstrained problems,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jul. 2020, pp. 1–8.

[15] R. Tanabe and A. S. Fukunaga, “Improving the search performance of SHADE using linear population size reduction,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jul. 2014, pp. 1658–1665.

[16] S. Biswas, D. Saha, S. De, A. D. Cobb, S. Das, and B. A. Jalaian, “Improving differential evolution through Bayesian hyperparameter optimization,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2021, pp. 832–840.

[17] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proc. Int. Conf. Neural Netw. (ICNN), vol. 4, Nov./Dec. 1995, pp. 1942–1948.

[18] A. Kazikova, M. Pluhacek, and R. Senkerik, “Why tuning the control parameters of metaheuristic algorithms is so important for fair comparison?” MENDEL., vol. 26, no. 2, pp. 9–16, Dec. 2020.

[19] C. Doerr, H. Wang, F. Ye, S. van Rijn, and T. Bäck, “IOHprofiler: A benchmarking and profiling tool for iterative optimization heuristics,” 2018, arXiv:1810.05281.

[20] T. Eftimov, G. Petelin, and P. Korošec, “DSCTool: A web-service-based framework for statistical comparison of stochastic optimization algorithms,” Appl. Soft Comput., vol. 87, Feb. 2020, Art. no. 105977.

[21] T. Eftimov, R. Hribar, Ž. Škvorc, G. Popovski, G. Petelin, and P. Korošec, “PerformViz: A machine learning approach to visualize and understand the performance of single-objective optimization algorithms,” in Proc. Genetic Evol. Comput. Conf. Companion, Jul. 2020, pp. 7–8.

[22] O. Mersmann, B. Bischof, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph, “Exploratory landscape analysis,” in Proc. 13th Annu. Conf. Genetic Evol. Comput., New York, NY, USA: ACM, 2011, pp. 829–836.

[23] P. Kerschke and H. Trautmann, “Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the R-package flacco,” in Applications in Statistical Computing, Cham, Switzerland: Springer, 2019, pp. 93–123.

[24] B. Y. Qu, J. J. Liang, Z. Y. Wang, Q. Chen, and P. N. Suganthan, “Novel benchmark functions for continuous multimodal optimization with comparative results,” Swarm Comput., vol. 26, pp. 23–34, Feb. 2016.