IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 8, 2022, accepted January 14, 2022, date of publication January 18, 2022, date of current version January 21, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3144067

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 performances of all 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 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" $z(x, k, m, \lambda)$ (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 $x \in \mathcal{R}$ 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).

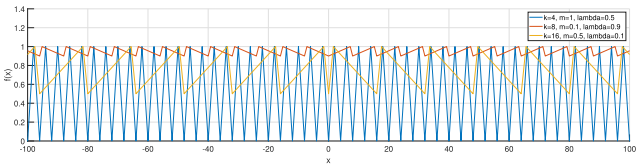

FIGURE 1. Zigzag function for different values of the parameters.

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 ϕ for the individual components, but modify the inputs by a shift vector $\mathbf{s} \in [-100, 100]^D$ and a rotation/scaling matrix $\mathbf{M}$:

$$f_j(\mathbf{x}, k, m, \lambda) = \sum_{i=1}^{D} \phi_j(x_i, k, m, \lambda) \quad j = 1, \ldots, 4$$
$$\mathrm{F}j(\mathbf{x}, k, m, \lambda) = f_j(\mathbf{M}_j(\mathbf{x} - \mathbf{s}_j), k, m, \lambda) \quad j = 1, \ldots, 4$$

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

$$z(x, k, m, \lambda) = \begin{cases} 1 - m + \dfrac{m}{\lambda}(|x|/k - \lfloor |x|/k \rfloor), & \text{if } |x|/k - \lfloor |x|/k \rfloor \le \lambda, \\ 1 - m + \dfrac{m}{1-\lambda}(1 - |x|/k + \lfloor |x|/k \rfloor), & \text{otherwise,} \end{cases} \tag{1}$$

$$\begin{aligned} \phi_1(x, k, m, \lambda) &= 3{\cdot}10^{-9}|(x-40)(x-185)x(x+50)(x+180)|z(x, k, m, \lambda) + 10|\sin(0.1x)| \\ \phi_2(x, k, m, \lambda) &= \phi_1(\phi_1(x, k, m, \lambda), k, m, \lambda) \\ \phi_3(x, k, m, \lambda) &= 3|\ln(1000|x| + 1)|z(x, k, m, \lambda) + 30 - 30|\cos(\frac{x}{10\pi})| \\ \phi_4(x, k, m, \lambda) &= \phi_3(\phi_3(x, k, m, \lambda), k, m, \lambda) \end{aligned} \tag{2}$$

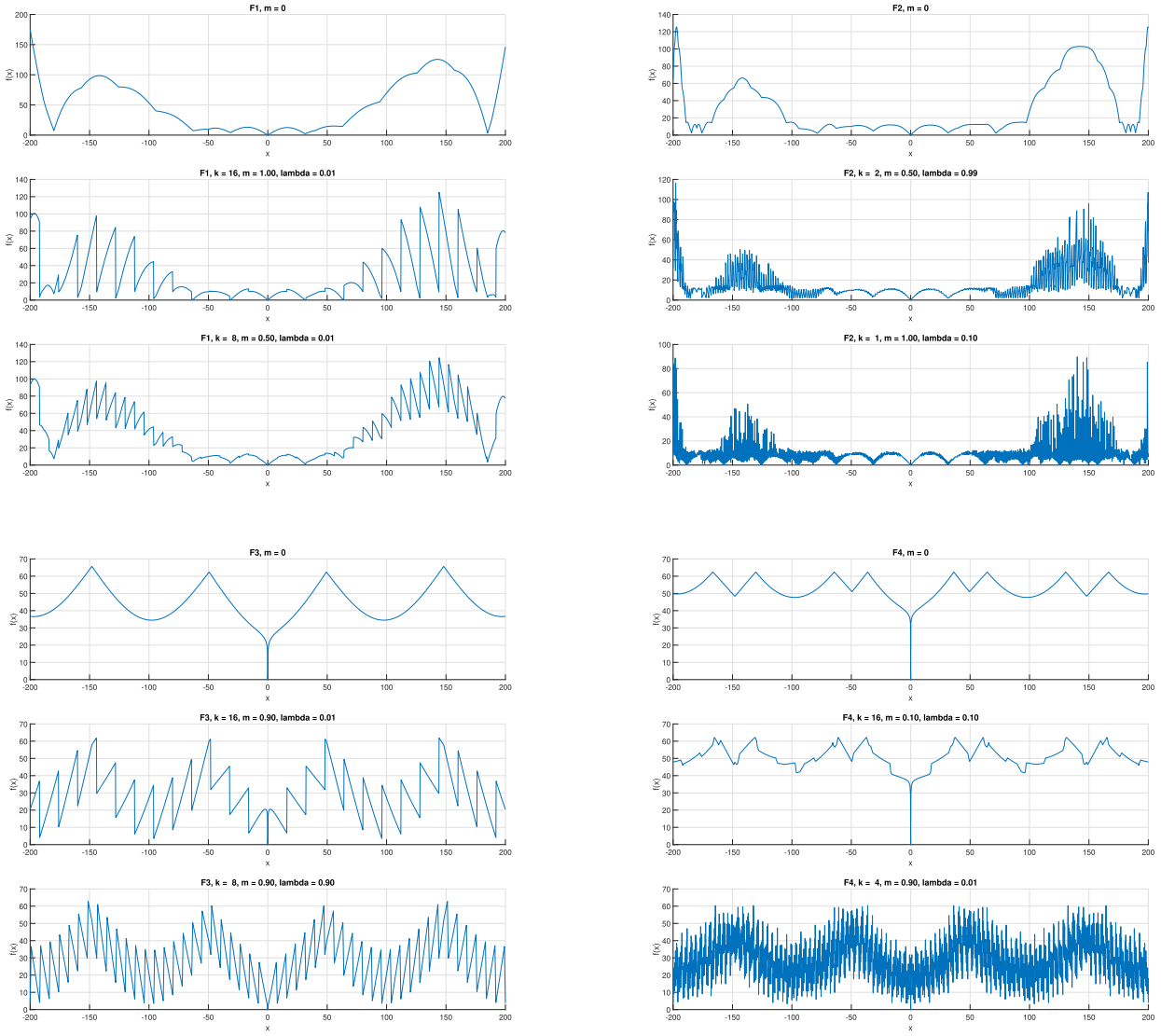

FIGURE 2. Shape of the benchmark functions F1-F4 for different values of the parameters.

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 λ can be seen as a ''deception'' parameter, as low values of λ 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

and sharing phases among the solutions during the optimization process [20].

- 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.

TABLE 1. Different parameter settings used in the computations.

ID	k	m	λ	ID	k	m	λ	ID	k	m	λ	ID	k	m	λ
1	1	0.10	0.01	26	1	0.50	0.01	51	1	0.90	0.01	76	1	1.00	0.01
2	2	0.10	0.01	27	2	0.50	0.01	52	2	0.90	0.01	77	2	1.00	0.01
3	4	0.10	0.01	28	4	0.50	0.01	53	4	0.90	0.01	78	4	1.00	0.01
4	8	0.10	0.01	29	8	0.50	0.01	54	8	0.90	0.01	79	8	1.00	0.01
5	16	0.10	0.01	30	16	0.50	0.01	55	16	0.90	0.01	80	16	1.00	0.01
6	1	0.10	0.10	31	1	0.50	0.10	56	1	0.90	0.10	81	1	1.00	0.10
7	2	0.10	0.10	32	2	0.50	0.10	57	2	0.90	0.10	82	2	1.00	0.10
8	4	0.10	0.10	33	4	0.50	0.10	58	4	0.90	0.10	83	4	1.00	0.10
9	8	0.10	0.10	34	8	0.50	0.10	59	8	0.90	0.10	84	8	1.00	0.10
10	16	0.10	0.10	35	16	0.50	0.10	60	16	0.90	0.10	85	16	1.00	0.10
11	1	0.10	0.50	36	1	0.50	0.50	61	1	0.90	0.50	86	1	1.00	0.50
12	2	0.10	0.50	37	2	0.50	0.50	62	2	0.90	0.50	87	2	1.00	0.50
13	4	0.10	0.50	38	4	0.50	0.50	63	4	0.90	0.50	88	4	1.00	0.50
14	8	0.10	0.50	39	8	0.50	0.50	64	8	0.90	0.50	89	8	1.00	0.50
15	16	0.10	0.50	40	16	0.50	0.50	65	16	0.90	0.50	90	16	1.00	0.50
16	1	0.10	0.90	41	1	0.50	0.90	66	1	0.90	0.90	91	1	1.00	0.90
17	2	0.10	0.90	42	2	0.50	0.90	67	2	0.90	0.90	92	2	1.00	0.90
18	4	0.10	0.90	43	4	0.50	0.90	68	4	0.90	0.90	93	4	1.00	0.90
19	8	0.10	0.90	44	8	0.50	0.90	69	8	0.90	0.90	94	8	1.00	0.90
20	16	0.10	0.90	45	16	0.50	0.90	70	16	0.90	0.90	95	16	1.00	0.90
21	1	0.10	0.99	46	1	0.50	0.99	71	1	0.90	0.99	96	1	1.00	0.99
22	2	0.10	0.99	47	2	0.50	0.99	72	2	0.90	0.99	97	2	1.00	0.99
23	4	0.10	0.99	48	4	0.50	0.99	73	4	0.90	0.99	98	4	1.00	0.99
24	8	0.10	0.99	49	8	0.50	0.99	74	8	0.90	0.99	99	8	1.00	0.99
25	16	0.10	0.99	50	16	0.50	0.99	75	16	0.90	0.99	100	16	1.00	0.99

The particular values of matrices $\mathbf{M}$ and shifts $\mathbf{s}$, 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 λ 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

[1] https://github.com/JakubKudela89/Zigzag

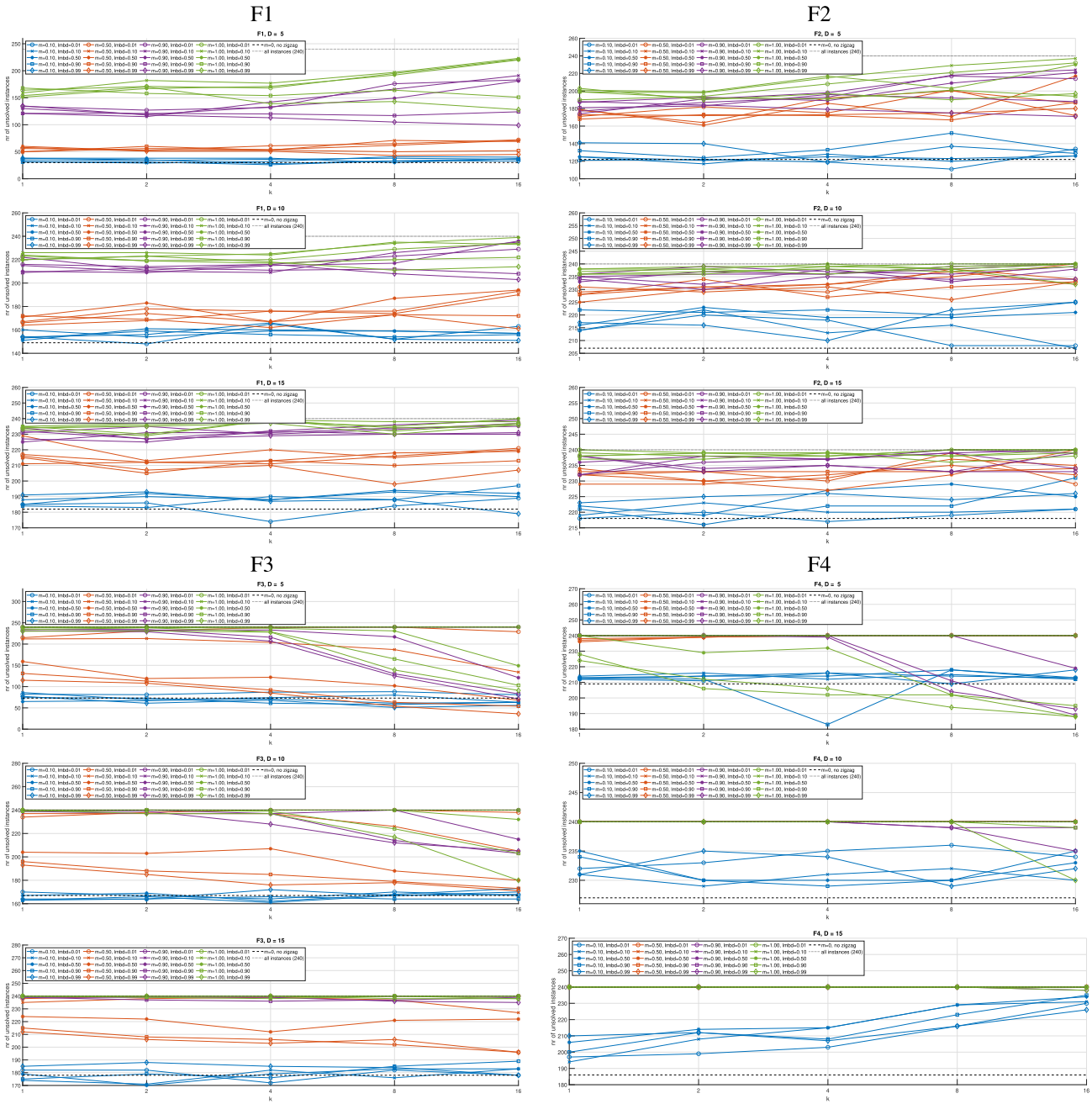

FIGURE 3. Number of unsolved instances for benchmark function F1-F4 for different values of parameters and dimensions.

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

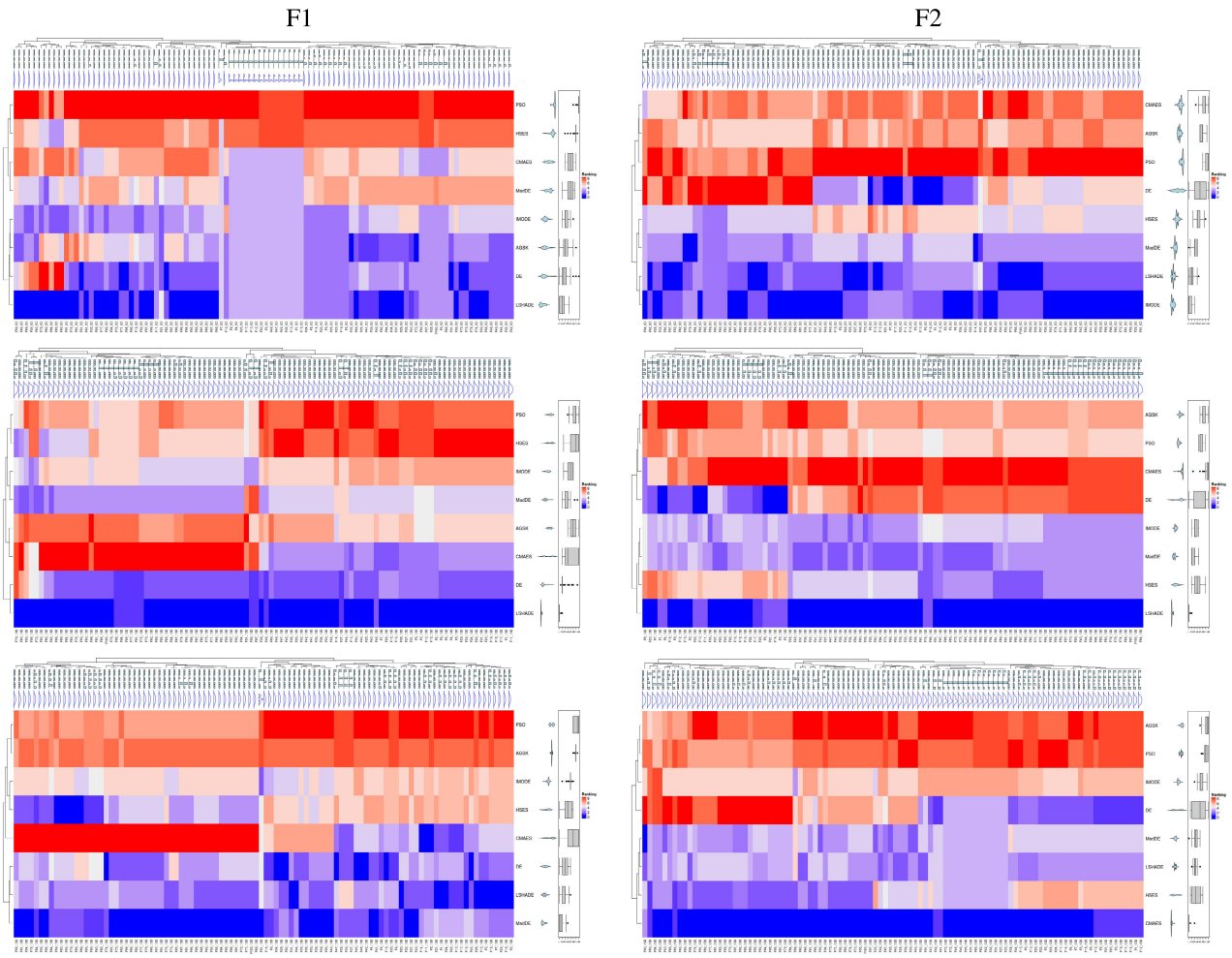

FIGURE 4. Comparison of the algorithms on different parameters settings for benchmark functions F1-F2, $D = \{5, 10, 15\}$.

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 $m = 0.1$, 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 CMAES 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

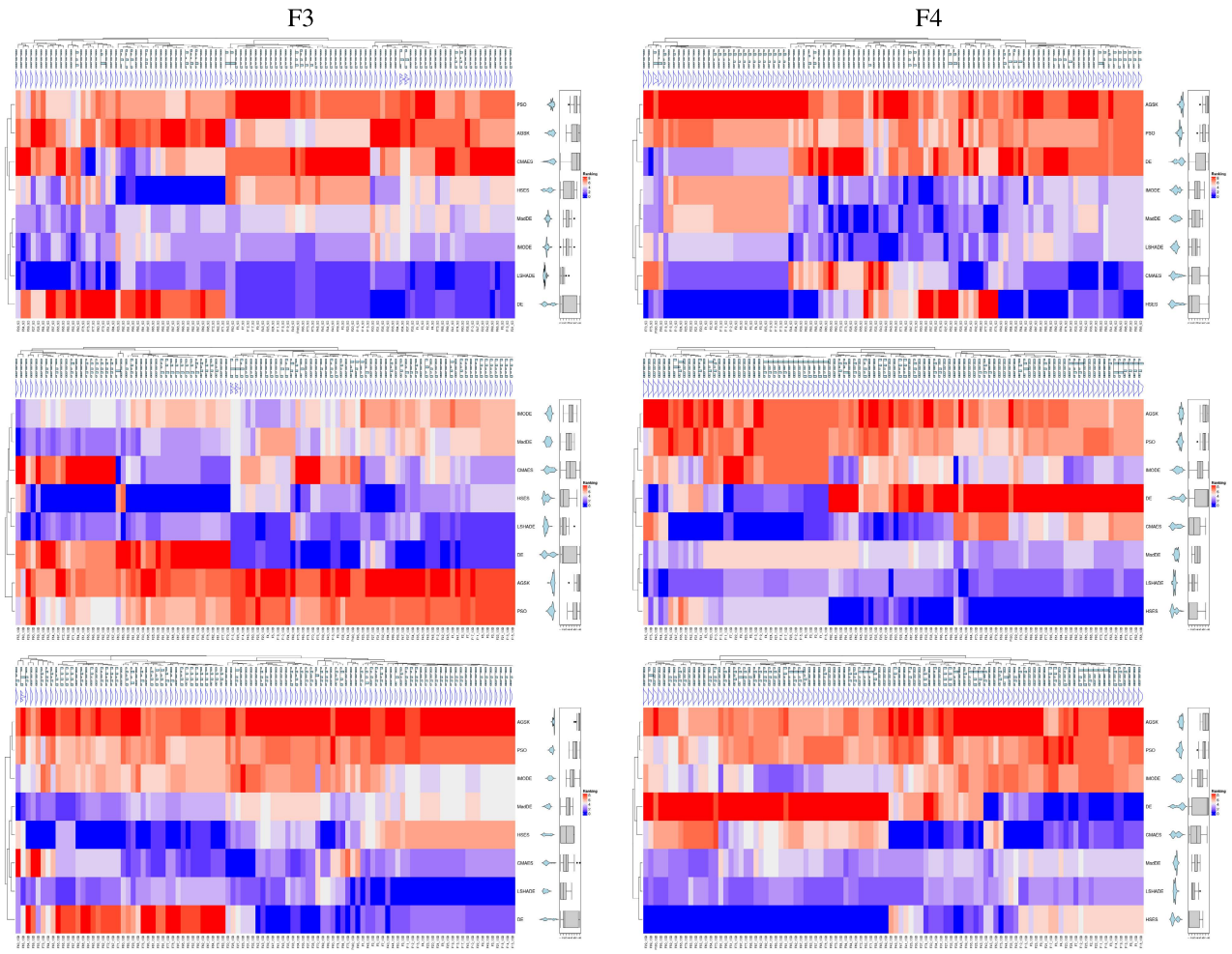

FIGURE 5. Comparison of the algorithms on different parameters settings for benchmark functions F3-F4, $D = \{5, 10, 15\}$.

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 patter 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

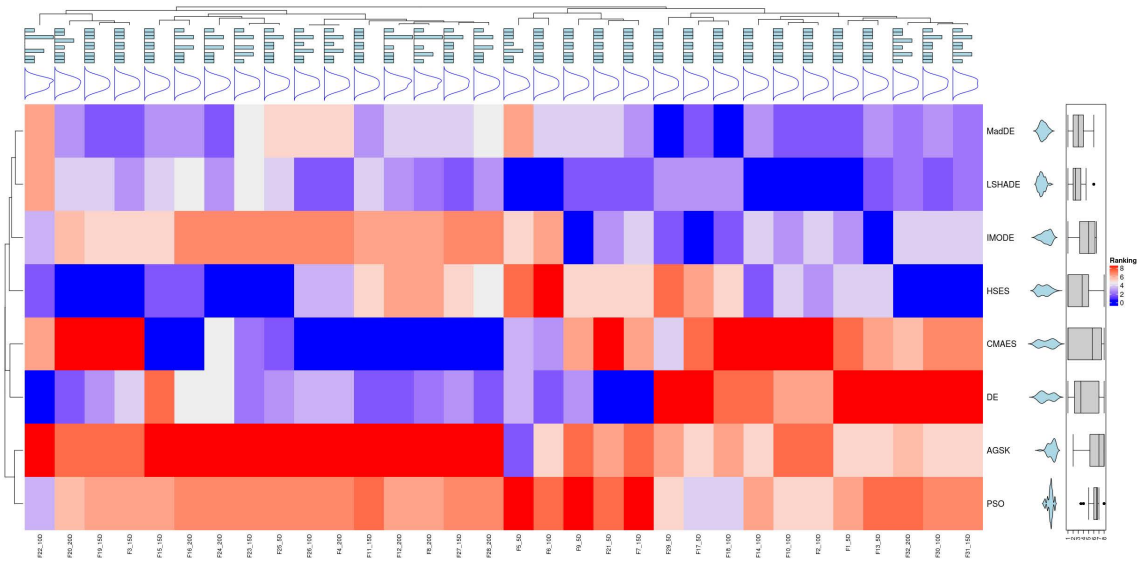

FIGURE 6. Comparison of the algorithms on the ambiguous benchmark set.

TABLE 2. Parametrizations for the ambiguous benchmark set.

ID	function	D	k	m	λ	ID	function	D	k	m	λ
1	F1	5	16	1	0.01	17	F3	5	16	0.9	0.01
2	F1	10	16	1	0.01	18	F3	10	16	0.9	0.01
3	F1	15	16	1	0.01	19	F3	15	16	0.9	0.01
4	F1	20	16	1	0.01	20	F3	20	16	0.9	0.01
5	F1	5	8	0.5	0.01	21	F3	5	8	0.9	0.9
6	F1	10	8	0.5	0.01	22	F3	10	8	0.9	0.9
7	F1	15	8	0.5	0.01	23	F3	15	8	0.9	0.9
8	F1	20	8	0.5	0.01	24	F3	20	8	0.9	0.9
9	F2	5	2	0.5	0.99	25	F4	5	16	0.1	0.1
10	F2	10	2	0.5	0.99	26	F4	10	16	0.1	0.1
11	F2	15	2	0.5	0.99	27	F4	15	16	0.1	0.1
12	F2	20	2	0.5	0.99	28	F4	20	16	0.1	0.1
13	F2	5	1	1	0.1	29	F4	5	4	0.9	0.01
14	F2	10	1	1	0.1	30	F4	10	4	0.9	0.01
15	F2	15	1	1	0.1	31	F4	15	4	0.9	0.01
16	F2	20	1	1	0.1	32	F4	20	4	0.9	0.01

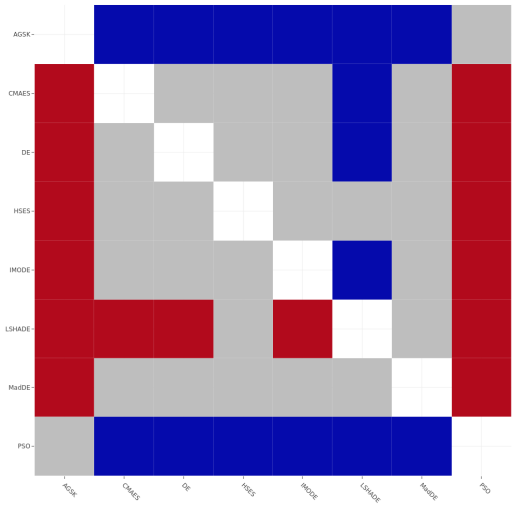

FIGURE 7. Results of the post-hoc test. For a given row: red cell – the algorithm in the row performs significantly better than the algorithm in the column; blue cell – performs significantly worse; grey cell – the ranking is ambiguous.

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 λ 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).

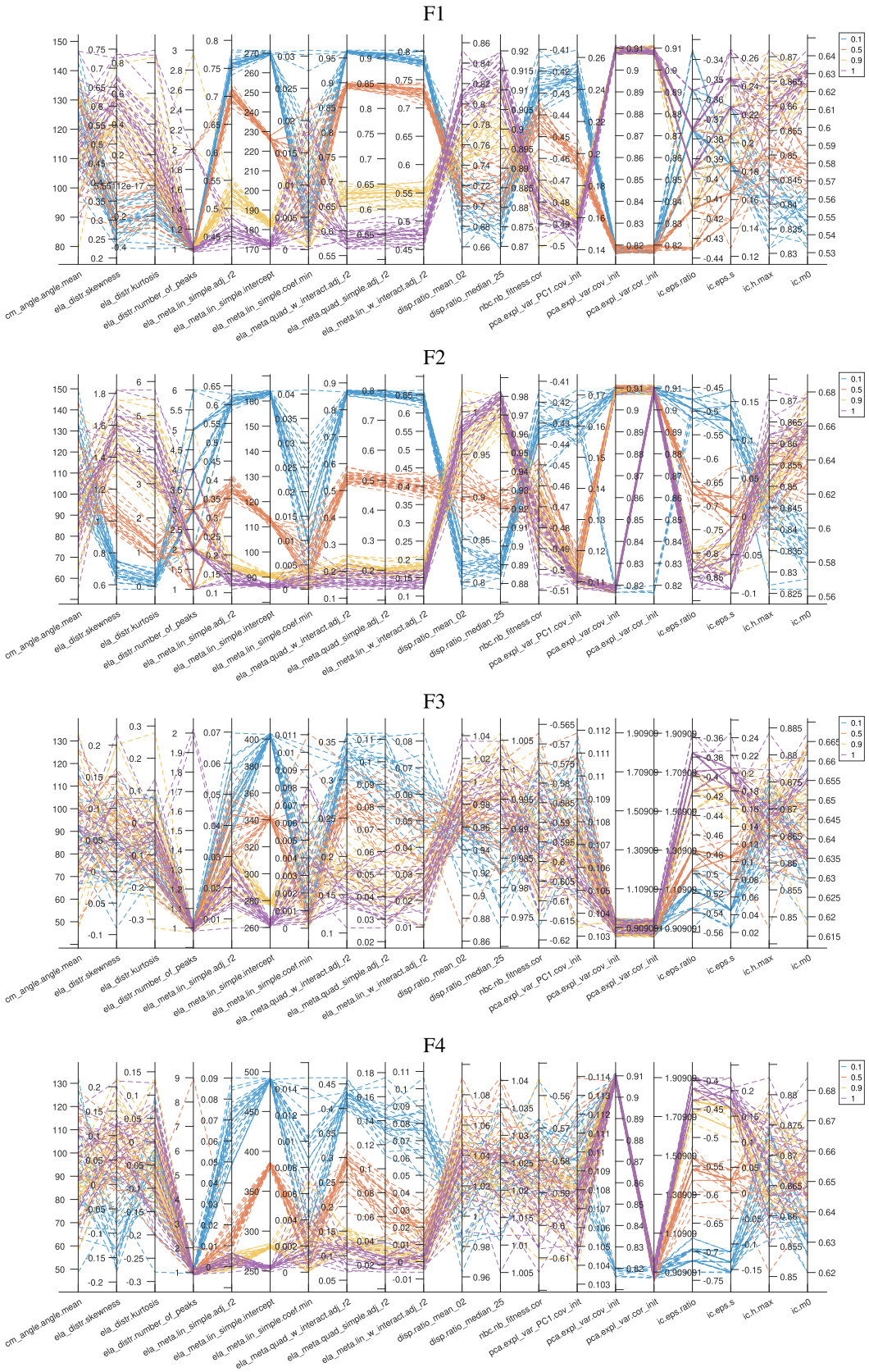

FIGURE 8. Parallel plots of the results of ELA for different parametrizations of F1-F4, $D = 10$, grouped by the value of the parameter m.

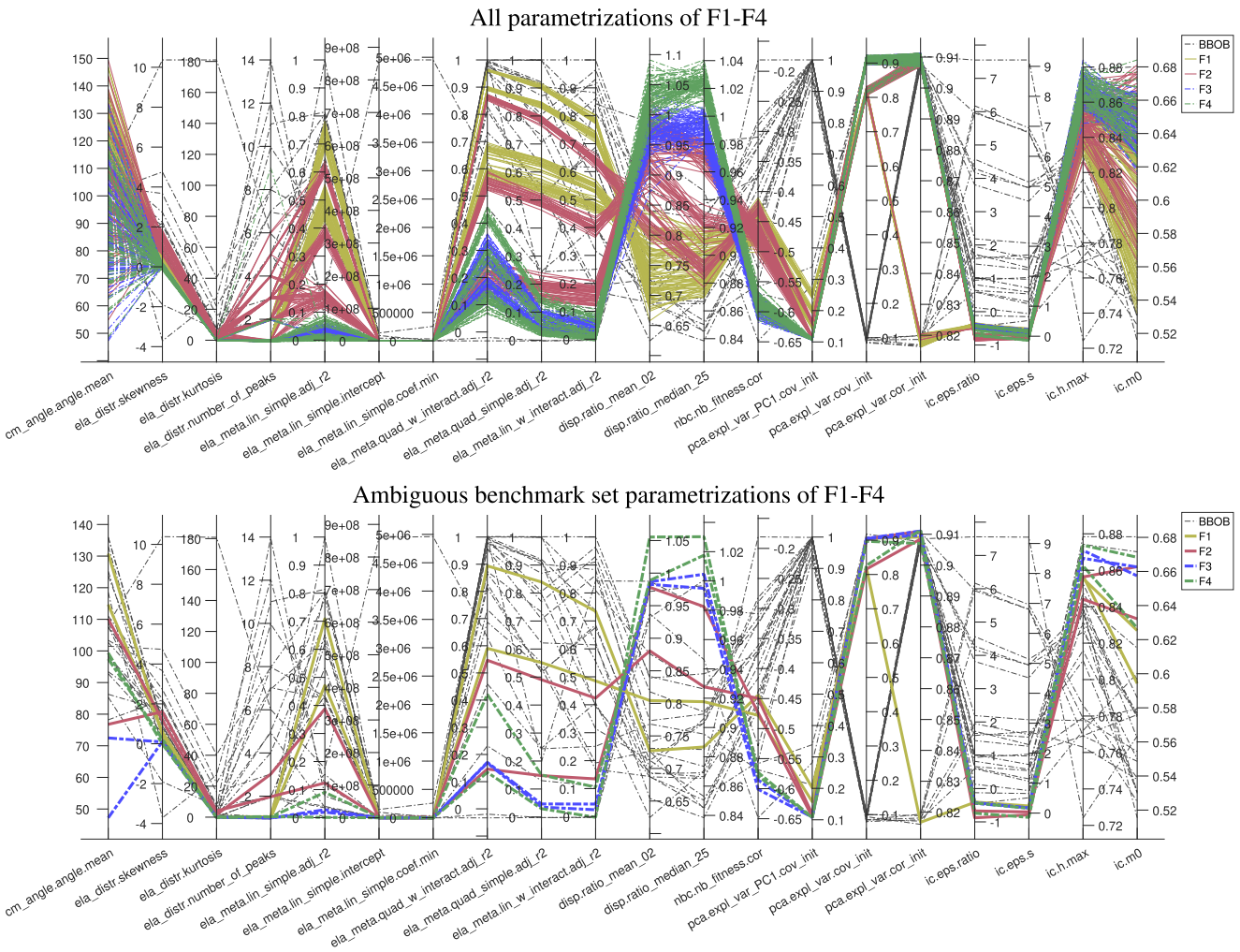

FIGURE 9. Comparison of ELA measures between different parametrizations of F1-F4 and BBOB, $D = 10$.

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

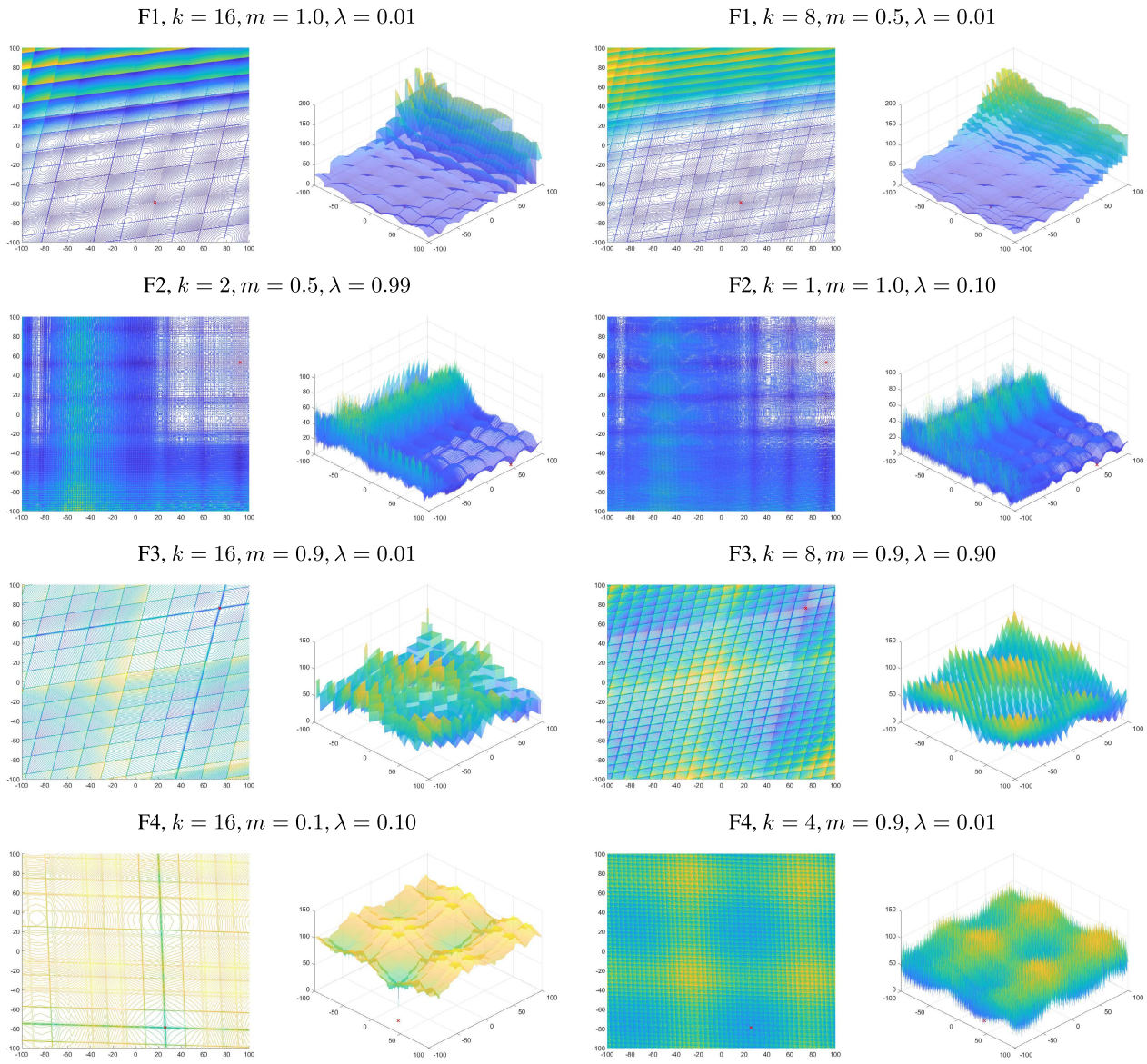

FIGURE 10. Contour and surface plots of the parametrization of F1-F4 used for the ambiguous benchmark set.

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

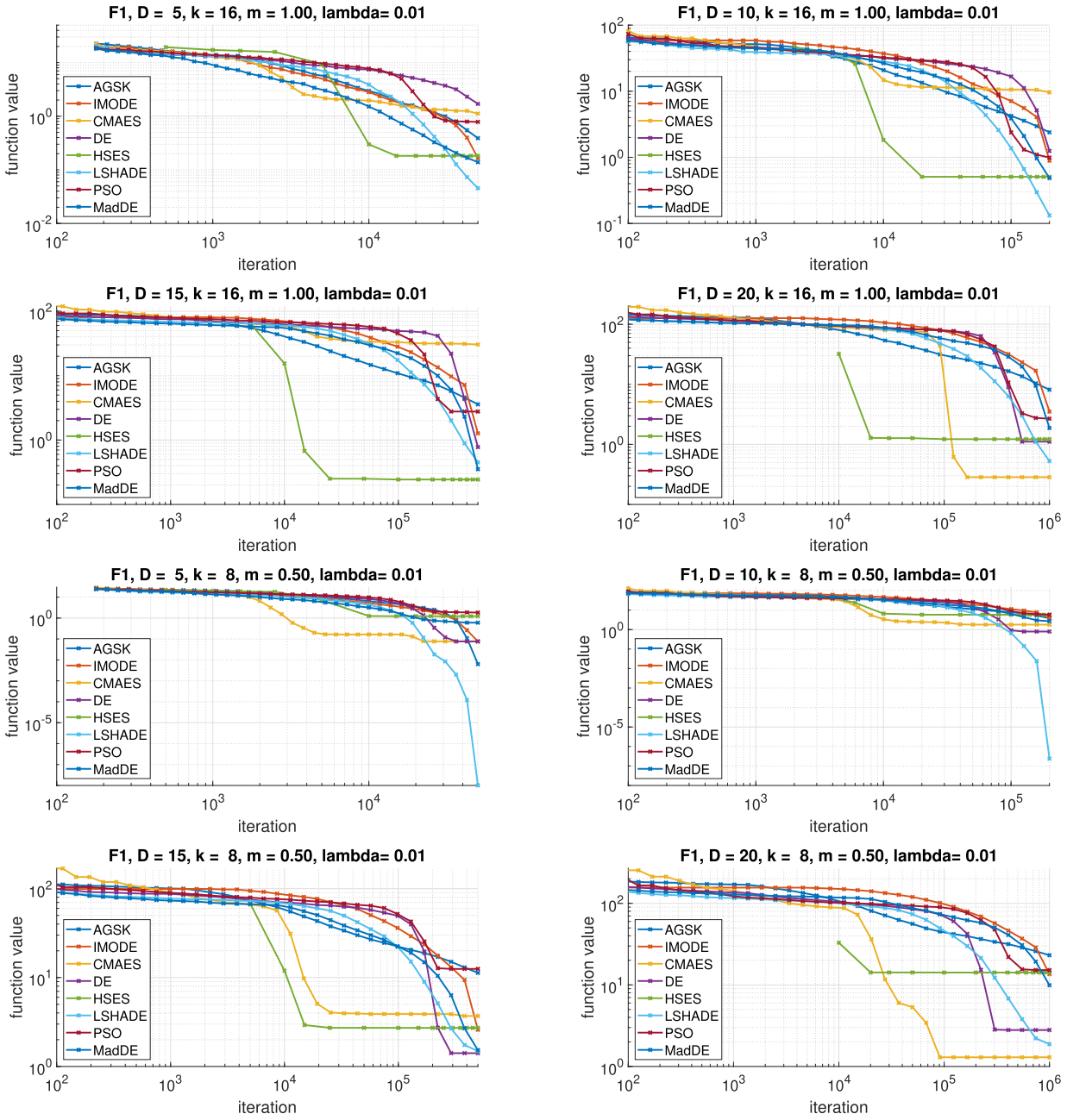

FIGURE 11. Convergence plots for F1 parametrizations (ID 1-8).

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_simple.intercept
- ela_meta.lin_simple.coef.min
- 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

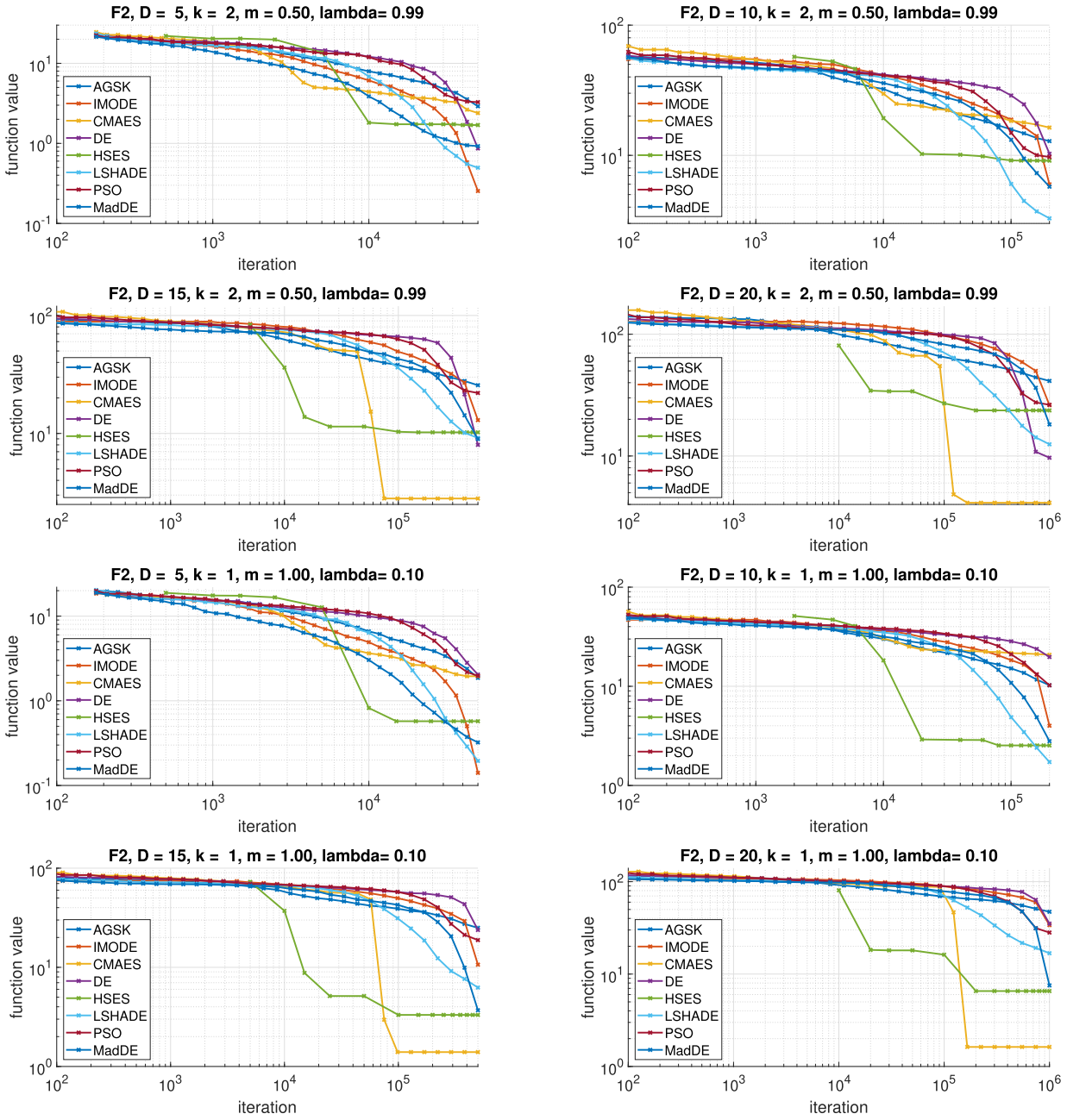

FIGURE 12. Convergence plots for F2 parametrizations (ID 9-16).

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

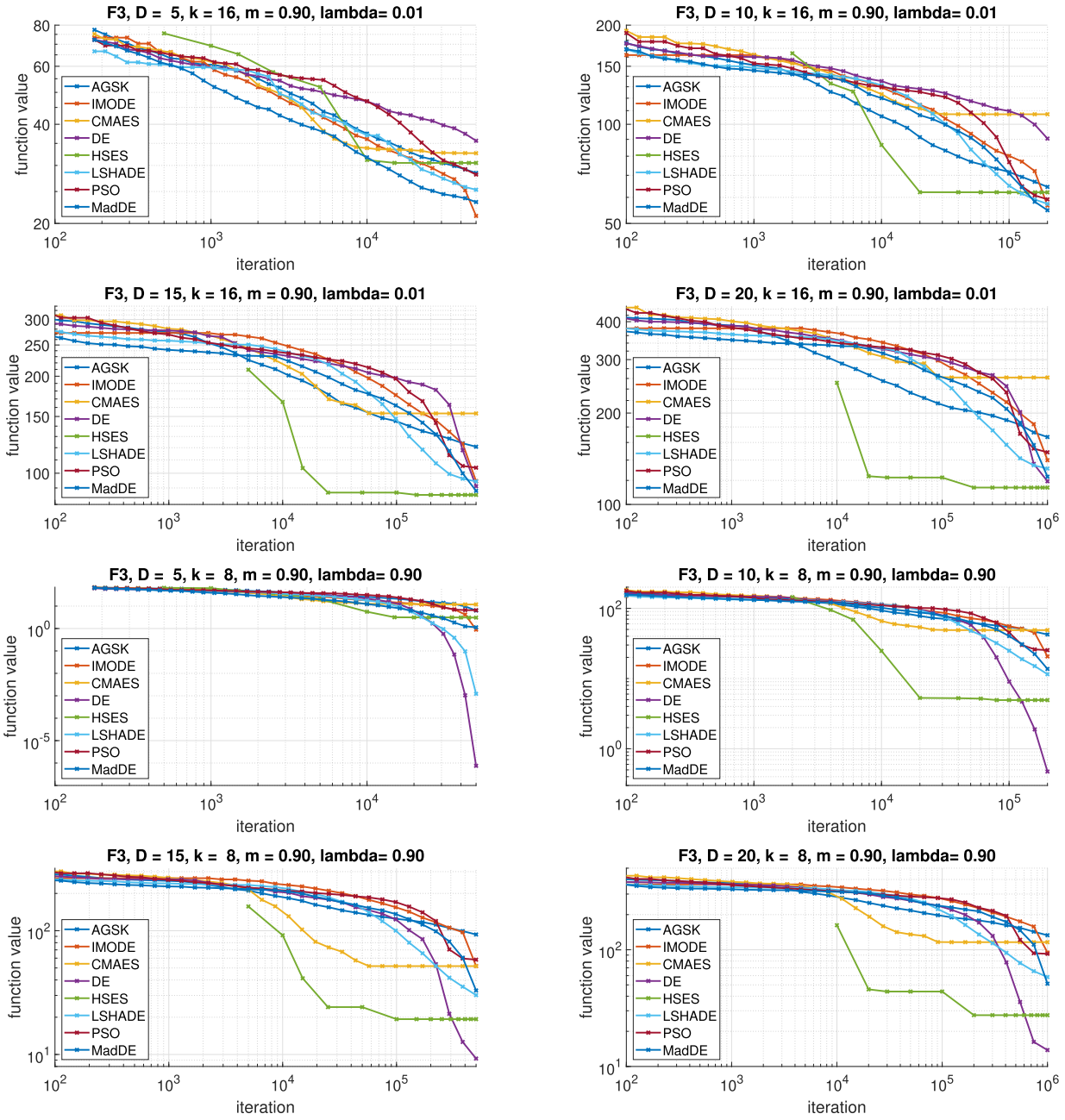

FIGURE 13. Convergence plots for F3 parametrizations (ID 17-24).

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,

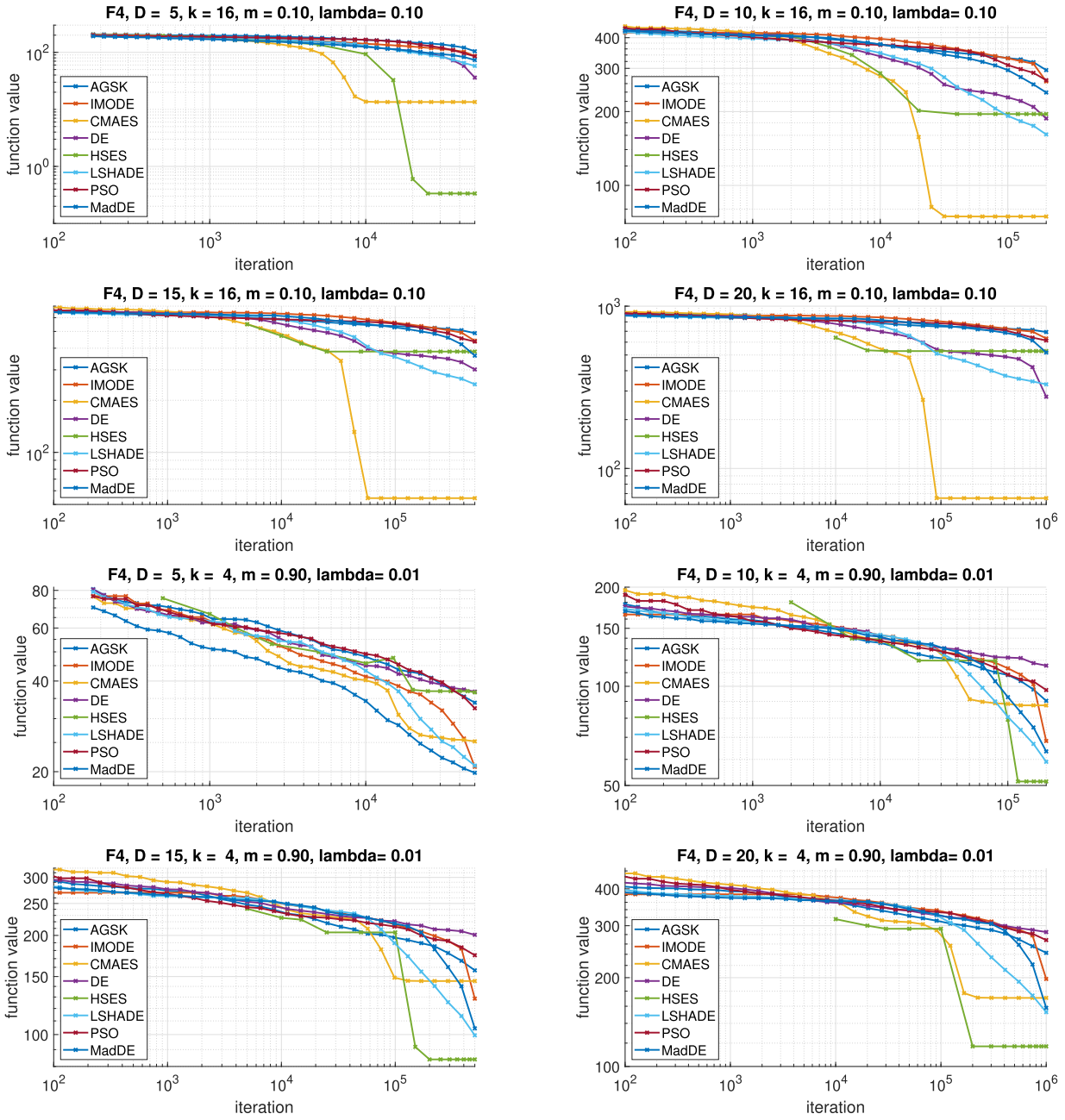

FIGURE 14. Convergence plots for F3 parametrizations (ID 25-32).

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	CMAES	DE	HSES	IMODE	LSHADE	MadDE	PSO
1	min	0.079	0	4.7E-05	0.148	0	0	0	0.148
	median	0.382	0.079	1.810	0.148	0.192	0.016	0.098	0.690
	mean	0.389	1.120	1.700	0.182	0.165	0.045	0.139	0.783
	max	0.629	6.674	2.978	0.533	0.375	0.227	0.377	1.822
	std	0.123	1.748	0.877	0.075	0.109	0.056	0.100	0.505
2	min	1.257	0	0	0.306	0.237	0.002	0.080	0.306
	median	2.407	6.692	0.341	0.464	0.805	0.139	0.475	0.751
	mean	2.389	9.639	1.250	0.507	0.889	0.132	0.485	0.993
	max	3.332	26.041	19.712	0.918	1.753	0.268	0.997	2.598
	std	0.486	9.996	3.616	0.119	0.407	0.087	0.163	0.606
3	min	2.534	0	0.158	0.079	0.621	0.254	0.158	1.066
	median	3.643	39.538	0.582	0.237	1.228	0.482	0.380	2.587
	mean	3.584	30.451	0.777	0.244	1.270	0.450	0.353	2.770
	max	4.641	49.380	2.816	0.464	2.680	0.649	0.533	6.899
	std	0.548	19.186	0.620	0.092	0.465	0.088	0.094	1.282
4	min	6.394	0	0.227	0.909	1.835	0.200	1.136	0.988
	median	7.750	0.237	1.105	1.210	3.430	0.526	1.882	2.292
	mean	8.132	0.284	1.110	1.221	3.512	0.527	1.871	2.679
	max	10.757	0.858	2.062	1.660	5.721	0.745	2.437	6.379
	std	1.144	0.187	0.482	0.156	1.049	0.122	0.277	1.340
5	min	0	0	0	0	0	0	0	0
	median	0	0	0	2.159	0	0	0	1.148
	mean	0.006	0.077	0.077	1.228	0.077	0	0.603	1.838
	max	0.189	1.148	1.148	3.308	1.148	0	2.296	7.452
	std	0.035	0.291	0.291	1.202	0.291	0	0.763	1.850
6	min	0	0	0	4.456	2.159	0	1.148	2.319
	median	4.202	0	0	5.604	3.950	0	2.296	5.380
	mean	3.818	1.779	0.790	5.796	3.870	2.4E-07	2.675	5.777
	max	6.568	13.668	4.456	7.901	5.798	6.1E-06	4.593	10.896
	std	1.645	3.678	1.252	0.803	1.212	1.1E-06	0.995	2.437
7	min	3.445	0	0	1.148	5.0E-06	0.003	0	3.444
	median	11.628	1.148	1.148	2.296	2.296	1.164	1.148	13.075
	mean	11.302	3.697	1.412	2.717	2.586	1.487	1.526	12.509
	max	14.614	35.888	5.741	5.741	4.456	3.455	3.445	20.485
	std	2.284	8.487	1.515	0.976	1.054	0.942	0.747	3.695
8	min	15.209	0	0	10.934	8.912	0.002	2.296	8.912
	median	22.856	1.148	2.296	14.379	13.231	2.306	10.497	15.633
	mean	23.114	1.297	2.797	14.180	13.583	1.877	9.913	15.192
	max	29.106	4.456	8.912	17.550	19.534	4.466	12.744	23.990
	std	3.309	1.258	2.218	1.731	2.636	1.186	2.273	4.007
9	min	0.351	0	0	0	0	0	0	0
	median	2.876	1.886	6.1E-05	2.173	0	0.003	0.807	3.365
	mean	2.908	2.396	0.865	1.695	0.254	0.497	0.925	3.274
	max	5.554	9.906	8.781	2.881	1.961	1.995	1.886	9.186
	std	1.355	2.719	2.037	0.882	0.659	0.832	0.942	2.023
10	min	7.886	0	0	5.664	1.886	0.002	1.605	1.605
	median	12.487	21.548	3.966	9.116	6.085	3.715	5.757	9.497
	mean	12.865	16.344	10.284	9.108	5.989	3.283	5.745	9.717
	max	17.286	37.279	28.568	14.860	9.641	6.727	9.033	17.369
	std	2.040	13.365	10.621	2.360	2.127	1.710	1.909	3.814
11	min	19.195	0	1.886	6.519	8.714	4.348	5.185	12.838
	median	25.994	2.173	5.953	10.248	12.758	9.845	8.918	20.023
	mean	25.629	2.807	8.006	10.219	12.961	9.169	8.968	22.022
	max	32.357	8.416	34.796	14.028	17.685	13.648	12.851	43.048
	std	3.492	2.678	6.217	1.690	2.222	2.560	1.848	7.163
12	min	30.890	0	3.210	15.752	19.499	6.852	10.187	15.232
	median	41.565	3.327	8.426	23.186	26.333	12.499	18.971	25.418
	mean	41.338	4.129	9.680	23.674	26.167	12.471	18.161	26.322
	max	50.257	10.308	21.103	30.395	36.428	18.686	24.438	43.032
	std	4.713	3.007	4.534	3.463	4.169	2.740	3.826	7.099
13	min	0	0	5.6E-08	0	0	0	0	0.199
	median	1.842	0.637	1.917	0.637	0	0.014	0.469	1.620
	mean	1.867	1.950	2.042	0.573	0.141	0.195	0.322	1.969
	max	3.528	8.261	6.146	1.274	0.470	0.695	0.640	6.072
	std	1.029	2.449	2.140	0.257	0.219	0.268	0.255	1.371
14	min	5.961	0	4.0E-04	1.474	1.274	0.485	0	0.199
	median	10.284	26.534	22.105	2.549	4.134	1.661	2.551	9.113
	mean	10.248	20.786	19.752	2.534	4.004	1.722	2.797	10.249
	max	14.187	30.922	30.011	3.385	7.986	3.130	9.179	26.090
	std	2.151	11.369	8.803	0.519	1.881	0.723	1.762	6.107
15	min	17.448	0	0.637	2.274	3.728	2.806	1.408	3.382
	median	25.269	1.274	12.385	3.186	9.151	5.617	3.156	17.325
	mean	25.000	1.396	23.789	3.311	10.639	6.248	3.687	18.817
	max	32.461	3.018	53.380	4.460	20.560	13.618	8.005	57.909
	std	3.840	0.807	22.321	0.605	4.464	3.025	1.503	12.130
16	min	37.808	0.199	5.2E-07	4.993	14.945	4.194	1.878	16.267
	median	47.335	1.374	20.635	6.416	35.142	16.225	5.674	24.982
	mean	47.233	1.626	35.135	6.549	33.864	16.808	7.541	28.150
	max	55.853	3.186	81.701	9.323	51.526	33.009	23.803	78.122
	std	4.450	0.889	33.104	1.095	9.050	7.673	4.969	12.278
17	min	24.497	20.920	29.237	27.239	13.817	21.721	18.211	20.209
	median	28.568	30.398	35.520	30.398	20.559	25.862	22.916	28.718
	mean	28.468	32.684	35.650	30.532	21.078	25.313	23.226	28.147
	max	31.597	51.998	41.569	33.557	27.393	27.175	27.811	34.698
	std	1.978	8.097	3.210	1.310	3.177	1.664	2.499	3.940
18	min	55.092	54.477	60.309	57.636	50.114	51.925	46.091	52.587
	median	64.854	126.734	94.517	62.089	56.761	57.102	54.738	57.903
	mean	64.553	107.265	90.495	62.136	56.653	57.323	54.832	59.166
	max	72.476	156.234	116.656	63.955	63.579	61.972	61.251	67.202
	std	3.653	38.127	15.928	1.761	3.729	2.392	2.863	4.328
19	min	104.287	81.724	77.044	81.716	84.433	87.715	76.012	85.912
	median	120.277	95.646	92.118	84.875	95.491	94.035	88.989	105.300
	mean	120.845	153.290	91.115	85.651	94.512	94.278	88.154	104.025
	max	137.601	264.754	111.408	90.047	102.123	102.945	93.989	127.015
	std	8.426	75.605	7.885	2.583	4.458	3.957	4.213	10.525
20	min	148.185	102.866	111.148	105.221	116.583	121.006	112.764	115.995
	median	168.455	320.296	119.584	113.839	140.981	130.682	123.086	148.106
	mean	166.773	261.881	119.236	113.669	139.884	131.311	123.438	148.521
	max	185.485	377.771	132.276	122.457	154.659	139.613	135.300	183.048
	std	10.307	105.028	4.649	4.272	10.055	5.071	5.052	15.519
21	min	0	0	0	0	0	0	0	0
	median	6.490	4.090	0	3.552	0	0	0	5.213
	mean	6.351	11.857	7.5E-07	3.078	0.893	0.001	1.130	6.553
	max	12.826	55.099	2.2E-05	3.552	5.138	0.037	3.762	15.020
	std	3.770	14.631	4.1E-06	1.228	1.668	0.007	1.669	4.667
22	min	33.289	0	0	3.552	8.699	3.387	3.552	7.649
	median	42.885	14.208	0	3.552	21.159	11.961	13.212	25.586
	mean	42.386	48.884	0.474	4.930	20.624	11.510	13.691	25.355
	max	50.719	128.454	3.552	8.690	33.462	17.541	23.428	45.074
	std	4.791	47.193	1.228	1.818	5.749	3.612	4.751	11.528
23	min	65.592	0	3.552	17.760	27.757	20.753	12.042	26.357
	median	94.560	14.378	8.067	18.100	52.652	29.963	33.199	59.561
	mean	93.180	51.745	9.269	19.290	51.787	30.153	32.884	58.324
	max	105.063	231.198	14.990	23.238	72.605	40.446	42.918	88.369
	std	8.714	75.848	4.243	1.974	10.185	4.553	7.000	16.076
24	min	90.555	0	0	22.898	70.394	43.377	31.612	57.931
	median	135.687	20.019	14.548	26.450	91.950	59.616	50.779	90.813
	mean	133.143	116.119	13.820	27.504	94.698	58.368	51.204	92.303
	max	152.926	361.609	22.216	37.065	122.852	68.916	65.893	137.916
	std	13.447	142.139	4.671	3.412	12.196	7.376	8.207	24.918
25	min	60.686	0	0.183	0.120	53.576	0	26.523	29.946
	median	104.889	0	37.116	0.279	86.615	62.168	76.352	84.601
	mean	104.439	13.471	36.067	0.335	86.386	57.876	73.217	85.128
	max	125.811	161.451	78.464	0.849	99.178	72.786	91.105	126.704
	std	14.153	41.309	20.464	0.208	10.621	17.322	13.625	18.336
26	min	163.671	0	135.256	154.755	233.503	130.362	208.422	207.208
	median	314.126	41.748	194.938	195.832	267.512	164.385	242.386	270.424
	mean	295.113	74.707	187.533	195.388	265.412	161.362	239.153	268.315
	max	331.847	329.923	222.756	237.361	300.395	187.579	261.766	330.771
	std	40.180	106.332	27.320	18.559	19.013	12.448	13.686	31.063
27	min	348.402	0	193.006	317.022	374.203	221.197	304.657	341.376
	median	509.195	41.849	311.162	386.469	446.116	249.563	368.799	436.817
	mean	488.725	54.362	301.460	382.444	441.602	247.384	362.159	436.950
	max	528.011	494.286	354.265	413.896	467.535	271.327	391.294	529.460
	std	50.179	89.360	43.729	18.832	24.525	13.193	25.341	38.330
28	min	548.602	0	0	506.788	576.439	273.658	473.872	530.989
	median	711.098	41.795	279.976	525.713	635.228	334.567	521.129	606.926
	mean	692.511	65.597	276.620	528.635	632.042	329.717	518.021	614.407
	max	727.464	580.101	527.866	554.467	672.459	364.781	563.720	715.581
	std	48.104	108.293	201.979	13.152	21.947	24.451	24.890	43.689
29	min	23.854	17.386	28.283	18.695	16.092	16.002	16.246	23.265
	median	34.351	21.897	37.156	39.615	20.530	20.433	19.454	32.242
	mean	33.905	25.213	36.893	36.677	20.750	20.953	19.828	32.490
	max	42.422	49.843	43.193	51.209	26.747	26.529	22.987	44.599
	std	4.011	9.400	3.599	9.305	2.768	2.137	1.618	5.791
30	min	78.816	39.269	104.742	37.788	54.182	52.321	55.594	78.915
	median	91.075	98.698	118.756	49.973	68.264	58.426	63.994	98.324
	mean	90.372	87.538	115.562	51.413	68.231	58.990	63.439	97.430
	max	102.035	123.785	122.909	77.245	83.318	65.581	73.942	114.089
	std	6.501	29.765	6.035	8.978	7.479	3.394	3.731	9.774
31	min	144.170	62.598	174.725	68.015	98.105	82.713	90.469	125.414
	median	155.725	174.524	200.374	85.331	127.193	99.219	103.677	173.904
	mean	156.636	145.368	200.810	84.046	128.558	99.454	104.418	174.087
	max	181.204	199.811	219.169	104.426	149.310	111.173	117.778	202.850
	std	9.036	51.971	10.301	10.350	12.461	5.827	7.043	17.711
32	min	204.304	80.782	256.491	94.565	171.158	135.534	131.067	214.665
	median	242.529	112.234	286.675	114.071	198.139	153.238	161.119	274.342
	mean	242.741	170.763	285.455	117.005	197.970	152.829	158.040	268.152
	max	275.858	271.524	306.145	138.448	223.666	164.808	177.022	309.456
	std	15.207	82.614	11.193	12.941	14.681	6.469	10.930	24.682

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. 927–944, Feb. 2019.

[2] O. Mersmann, M. Preuss, H. Trautmann, B. Bischl, 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-SO-BCO

[5] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box 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," *Algorithms*, 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] R. W. Garden and A. P. Engelbrecht, "Analysis and classification of optimisation benchmark functions and benchmark suites," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 1641–1649.
[12] Y.-W. Zhang and S. K. Halgamuge, "Similarity of continuous optimization problems from the algorithm performance perspective," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 2949–2957.
[13] L. A. Christie, A. E. I. Brownlee, and J. R. Woodward, "Investigating benchmark correlations when comparing algorithms with parameter tuning," in *Proc. Genetic Evol. Comput. Conf. Companion*, Jul. 2018, pp. 209–210.
[14] I. Fister, J. Brest, A. Iglesias, A. Galvez, S. Deb, and I. Fister, "On selection of a benchmark by determining the Algorithms' qualities," *IEEE Access*, vol. 9, pp. 51166–51178, 2021.
[15] T. Weise and Z. Wu, "Difficult features of combinatorial optimization problems and the tunable *w*-model benchmark problem for simulating them," in *Proc. Genetic Evol. Comput. Conf. Companion*. New York, NY, USA: ACM, 2018, pp. 1769–1776.
[16] J. Kudela, "Novel zigzag-based benchmark functions for bound constrained single objective optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2021, pp. 857–862.
[17] K. R. Opara, A. A. Hadi, and A. W. Mohamed, "Parametrized benchmarking: An outline of the idea and a feasibility study," in *Proc. Genetic Evol. Comput. Conf. Companion*. New York, NY, USA: ACM, 2020, pp. 197–198.
[18] P. Bujok and R. Polakova, "Eigenvector crossover in the efficient jSO algorithm," *MENDEL*, vol. 25, no. 1, pp. 65–72, Jun. 2019.
[19] A. W. Mohamed, A. A. Hadi, and A. K. Mohamed, "Gaining-sharing knowledge based algorithm for solving optimization problems: A novel nature-inspired algorithm," *Int. J. Mach. Learn. Cybern.*, vol. 11, pp. 1501–1529, Dec. 2020.
[20] A. W. Mohamed, A. A. Hadi, A. K. Mohamed, and N. H. Awad, "Evaluating the performance of adaptive GainingSharing knowledge based algorithm on CEC 2020 benchmark problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–8.
[21] A. Auger and N. Hansen, "A Restart CMA evolution strategy with increasing population size," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2, Sep. 2005, pp. 1769–1776.
[22] 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.
[23] 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.
[24] 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.
[25] 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.
[26] 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.
[27] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN)*, vol. 4, Nov./Dec. 1995, pp. 1942–1948.
[28] 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.
[29] 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*.
[30] 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.
[31] T. Eftimov, R. Hribar, U. 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.
[32] O. Mersmann, B. Bischl, 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.
[33] 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.
[34] 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 Evol. Comput.*, vol. 26, pp. 23–34, Feb. 2016.

JAKUB KUDELA received the M.S. degree in mathematical engineering and the Ph.D. degree in applied mathematics from the Brno University of Technology (BUT), in 2014 and 2019, respectively.

Since 2018, he has been a Research Assistant with the Institute of Automation and Computer Science, BUT. His research interests include the development of computational methods for various optimization problems and engineering applications.

RADOMIL MATOUSEK received the M.S. degree in applied computer science and the Ph.D. degree in technical cybernetics from the Brno University of Technology, in 1996 and 2004, respectively.

Since 2012, he has been an Associate Professor with the Institute of Automation and Computer Science, BUT. His research interests include the development of AI and computational methods for engineering applications.