IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 6 February 2023, accepted 17 February 2023, date of publication 22 February 2023, date of current version 1 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3247954

RESEARCH ARTICLE

How Much Do Swarm Intelligence and Evolutionary Algorithms Improve Over a Classical Heuristic From 1960?

ADAM P. PIOTROWSKI[1], JAROSLAW J. NAPIORKOWSKI[1], AND AGNIESZKA E. PIOTROWSKA[2]
[1]Institute of Geophysics, Polish Academy of Sciences, 01-452 Warsaw, Poland
[2]Faculty of Polish Studies, University of Warsaw, 00-927 Warsaw, Poland

Corresponding author: Jaroslaw J. Napiorkowski (j.napiorkowski@igf.edu.pl)

This work was supported in part by the Ministry of Science and Higher Education of Poland within statutory activities under Grant 3841/E-41/S/2022; in part by the Institute of Geophysics, Polish Academy of Sciences through the Project HUMDROUGHT; and in part by the National Science Centre under Grant 2018/30/Q/ST10/00654.

ABSTRACT Hundreds of variants of Swarm Intelligence or Evolutionary Algorithms are proposed each year and numerous competitions and comparisons between algorithms may suggest rapid improvement in the field. However, such comparisons are often done between a limited number of methods and are based on averaged ranks of algorithms. This way they measure whether one method is on average ranked better than the others, without giving any information on how much improvement is in fact obtained. In this study we show a general comparison between 69 algorithms, starting from methods proposed in the 1960's up to variants developed in the early 2020's, on single-objective static numerical problems. Algorithms are compared on searching for a minimum of 30 different 50-dimensional mathematical functions, and on 22 real-world problems. We focus on the relative improvement achieved by various algorithms over a single-solution based method proposed in 1960 by Howard Rosenbrock. We find that the general improvement of Evolutionary Algorithms over Rosenbrock's algorithm is relatively limited. It is high for the artificial benchmarks, for which many Evolutionary Algorithms find solutions 10 times closer to the global optimum in terms of fitness than Rosenbrock's algorithm, but much lower for real-world problems. Improvement is also higher when performance averaged over many runs is compared, but lower when the best results from multiple runs are analyzed. In the last case, only the best Evolutionary Algorithms are able to find solutions of a "typical" real-world problem that are 2-3 times better in terms of fitness than those found by Rosenbrock's algorithm. The relative improvement of recently proposed algorithms is not much better than the improvement achieved by algorithms proposed over a decade ago.

INDEX TERMS Evolutionary algorithms, swarm intelligence, metaheuristics, performance, Rosenbrock's algorithm.

I. INTRODUCTION
Heuristic approaches devoted to solving optimization problems [1] are being developed at least since 1945 when George Polya published his famous book "How to solve it" [2]. The early 1960's saw the dawn of some classical algorithms that are still widely used, like Nelder-Mead simplex [3] or Rosenbrock's method [4]. In the mid-1970's the idea of biologically-inspired optimization algorithms appeared when Holland published his book on Genetic Algorithms [5] and Ingo Rechenberg worked on the first Evolutionary Strategies [6]. These algorithms were followed by other metaheuristics, including Simulated Annealing [7], Differential Evolution [8], and Particle Swarm Optimization [9], and since the late 1990's a swarm of inspiration-guided methods sprinkled in the optimization literature. The prefix "meta" before heuristics means that the method should be of general use. Metaheuristics are frequently called Evolutionary

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

Algorithms (EAs) – even if they have no direct link to the biological process of evolution – or Swarm Intelligence methods (we would not care about the difference between EAs and Swarm Intelligence in this paper), and hundreds of variants are proposed each year. EAs have found plentiful applications in almost any field of science, for example in developing self-learning robots [10], improving chemical reaction models [11], analysis of bio-diversity in prehistorical epochs [12], conservation of marine environment [13], facilitating health care services [14], development of artificial life forms [15], discovering new drugs [16] or searching for exoplanets [17]. They have also been widely used in various fields of engineering, including software faults prediction [18], smart grid development and control [19], system fault diagnosis [20], transmission congestion in power networks [21], development of charging stations for electrical vehicles [22], or planning of the renewable energy sources [23] [24]. Different kinds of EAs are applied to different kinds of problems – continuous, discrete or combinatorial, single-objective or multi-objective, static or dynamic, constrained or unconstrained [25]. In this paper we focus on the algorithms aiming at numerical, single-objective static problems.

There is a common belief that EAs are improving their performance – which means that the newer ones can either find better solutions in the search space or find solutions of similar quality quicker than the older ones. Such a view may be justified by the outcome of numerous competitions that are held each year [26] and the fact that some kinds of algorithms are steadily modified in a step-by-step manner [27]. However, it was also noted that algorithms that win subsequent competitions may be inferior to the winners of previous competitions, even if they were held on similar kinds of problems [28]. Apart from conference-related competitions, several papers appeared in which a comparison of several, but rarely more than 30 algorithms, was presented [29], [30], [31].

There is of course an endless problem how to measure the quality of optimizers, and different opinions on the performance of different methods may be the outcome of different rules set for comparison [32]. For example, in Black-Box Optimization Benchmarking suite (BBOB) competitions, the expected value to be reached is fixed, and algorithms compete by how quickly they can find a solution with required performance [33] when in the majority of IEEE benchmarks [34] the opposite view is set – the number of function calls is fixed, and algorithms are expected to find as good solution as possible within the pre-specified number of function calls. We also know that algorithms that perform relatively better on quick search would be outperformed by others when much more time is allowed, and vice versa [35]. This is in full accordance with No Free Lunch theorems for optimization [36], [37]. However, from a practical point of view, a different fundamental issue seems to be missed in the literature: a majority of studies focus on a rank-based comparison of algorithms, and ignore the question – How much improvement is obtained?

In the present paper we study the improvement obtained by 68 EAs, proposed between the 1960's and early 2020's, over a very basic optimization algorithm from 1960. Our comparison is restricted to single-objective static numerical problems and is based on the IEEE approach, hence we set the maximum number of function calls (MNFC), and algorithms search for the solution with the lowest objective function value within this computational time limit. To analyze the size of the improvement of EAs, we use a 50-dimensional IEEE benchmark test suite from CEC 2017 competition composed of 30 functions [34], and a set of 22 real-world problems [38].

In our research, we verify the improvement obtained by EAs over a simple, non-population-based algorithm proposed by Rosenbrock [4] that makes steps along the coordinate axes and occasionally modifies the step size and rotates the coordinate system (RA). This is a historical, but still sometimes used method that may be competitive to EAs, especially on unimodal or simple multimodal problems. If we know the global optimum (as in the case of CEC 2017 benchmarks), or at least the best solution known so far (as in the case of real-world problems), we may discuss the relative improvement of particular EA over RA. We focus on whether EAs in fact improve over RA, how much improvement is obtained, and whether the improvement obtained by newer EAs is indeed sufficiently larger than the improvement obtained by more historical algorithms to make a real difference for practitioners that wish to solve a particular problem. We aim at writing a simple and short paper focused only on the practical improvement of Swarm Intelligence and Evolutionary Algorithms in general, hence no detailed analysis of the possible reasons for successes or failures obtained by 69 algorithms is given. Such more algorithm-specific, detailed research is left for future papers.

The rest of the paper is organized as follows. In section II we briefly discuss various papers that aim at the comparison between Swarm Intelligence and Evolutionary Algorithms, or their different features. In section III we describe shortly the methods used in the study (test suites, algorithms, and performance criteria). In section IV we discuss the main results from both ranking-based comparisons between 69 algorithms, and the relative improvement of Swarm Intelligence and Evolutionary Algorithms over Rosenbrock's method. In section V we conclude the paper.

II. LITERATURE REVIEW

Various kinds of Swarm Intelligence and Evolutionary Algorithms have been compared multiple times in the literature, both theoretically and empirically, but we are not aware of papers in which a relative improvement of many novel algorithms over the classical ones would be addressed. In numerous guides a number of rules have been proposed that, at least in the opinion of their authors, should be followed when comparing various algorithms [39], [40], [41], [42]. In practice often it is considered that the safest choice is to follow the rules defined for some widely accepted benchmark sets, such as IEEE or BBOB ones [33], [34].

The main outcome of the theoretical approach to the comparison of Evolutionary Algorithms are No Free Lunch theorems for optimization [36]. According to No Free Lunch theorems, if the problem is implemented on a computer (hence discretized) and algorithms are non-revisiting, the performance of all possible heuristic methods averaged uniformly over all possible problems would be equal. However, this finding has limited practical applicability, as no one would average performance of optimization algorithms over all possible problems – at least because the fitness landscape of almost all of such problems would be a mix of random points, and hence would be of no interest to anyone.

In some other theoretical comparison studies similarities between various algorithms have been found, and the lack of novelty of many newer algorithms over the older ones has been revealed [43], [44], [45]. The fact that many new algorithms repeat operators of older methods just under novel nomenclature is one of the main problems that users may face when seeking for relevant Evolutionary Algorithms to solve particular problems [43]. A number of other papers address a theoretically different aspect of Evolutionary Algorithms, namely how they cope with different kinds of difficulties that may be faced when solving optimization problems [46], [47], [48]. Finally, from a theoretical point of view, the impact of various control parameter settings on the performance of different optimization algorithms has also been widely studied [49], [50], [51], [52], [53].

Despite the importance of theoretical studies, empirical comparisons between evolutionary algorithms seem to be more popular than theoretical ones, even though they are always limited by the scale of problems, setting of comparison rules, and algorithms chosen for the competition. A relatively wide-scale comparison between up to 30 various evolutionary algorithms has been presented in numerous papers [29], [30], [31], [54], [55]. In addition, each year multiple novel algorithms compete in different competitions on Evolutionary Computations (e.g. [28], [56]). However, the results of all such comparisons are rather ''local'', e.g. they show which algorithms perform better than the others, but it is hard to generalize them to gain opinions on the improvement in Swarm Intelligence or Evolutionary Algorithms in general.

In many important papers some specific features of evolutionary algorithms, or competition settings, have been compared. Such empirical tests may address diversified kinds of issues. For example, in [45], [57], [58], and [59] the impact of various initialization techniques on the performance of Evolutionary Algorithms is shown. A number of review papers [60], [61], [62], [63] analyze the impact of the population size of particular algorithms on the final performance. The impact of versatile other control parameters on the performance of specific kinds of Evolutionary Algorithms has also been addressed multiple times [64], [65], [66], [67]. As shown in numerous comparison papers, the performance of specific Evolutionary Algorithms would also depend on the number of allowed function calls [35], [55], [68], [69]. It is also known that the choice of the specific statistical test may affect the choice of the best algorithms [70], [71], [72], [73]. The comparison between algorithms may even to some degree be affected by the number of runs that are performed with each method [74]. Topology is another factor that may affect the effectiveness of many algorithms and is also studied in various comparison papers [75], [76], [77], [78]. In addition, the impact of the ensemble strategies [79], surrogate meta-models [80], multi-populations [81], or competition mechanisms [82], [83] on the performance of Evolutionary Algorithms has been analyzed. All such studies provide fruitful insight into the relative performance of Evolutionary Algorithms, or the impact of particular operators, control parameters, or other features of a specific method or competition settings. However, such papers still do not reveal how much improvement has been de facto obtained in recent years by Evolutionary Algorithms over classical, half-a-century-old optimizers.

III. METHODS

This section introduces briefly algorithms, benchmark sets, comparison criteria, and statistical tests used in the present study.

A. TEST PROBLEMS

To verify the range of improvement obtained by different EAs, two different types of numerical minimization problems are used: 30 mathematical, 50-dimensional functions (which we consider a trade-off between high- and low-dimensional cases) from IEEE CEC 2017 benchmarks [34] and 22 real-world problems of various difficulty and dimensionality [38] that come from different fields of science and industry. We set the maximum number of allowed function calls to the values proposed in the original papers in which the tests were introduced: to 10,000D for CEC 2017 benchmarks [34], where D is the problem dimensionality, and to 150,000 function calls for the real-world problems [38]. For each problem, every algorithm has been run 51 times [34]. For each run the solution with the lowest value of the objective function is remembered, hence we obtain a sample of 51 results per problem for each algorithm.

B. RELATIVE IMPROVEMENT MEASURE

In the classical comparison between algorithms, a rank-based ranking of algorithms is used. In such a case algorithms are ranked for each problem from the best one (rank 1) to the worst, usually according to the mean performance from all runs made on the specific problem. When setting the ranks of algorithms, often some very small threshold on the difference between methods is set; if the difference in the performance of some algorithms on a specific problem is lower than this threshold, these algorithms are given the same rank. As in many papers, we set this threshold to 10^{-8}. Then, ranks are averaged over all problems (e.g. [34]). This way the size of improvement, and the range of difference between algorithms, is lost.

In this study we focus on the scale of improvement obtained by EAs. To measure the relative improvement of a particular algorithm over RA, for each problem we need information on the function value in the global optimum (GM). In the case of CEC 2017 benchmarks GM is known and equals 0 for each problem [34]. However, in the case of real-world problems, the global minimum is often unknown. Hence, we assume that the value of GM used in this study to measure the relative improvement will be equal to the lowest value found for the particular real-world problem by any algorithm applied in this research, or, if it is lower, the value reported in the paper in which the winner of CEC 2011 competition that was held on the same 22 real-world problems was described [84]. We list the GM values that we have used for all real-world problems in Suppl. Table. 1.

In the majority of comparisons between algorithms only the mean performance (AV) averaged over all performed runs (51 in our case) is used as a comparison criterion. However, mean performance favor algorithms that avoid big failures in all runs, not necessarily those that at least in some runs can reach a noticeably better solution than competitors. Hence, in this paper we use two measures: the 51-runs averaged performance (AV), and the best performance obtained in 51 runs (BEST). The second measure may benefit algorithms that have a larger standard deviation of performance noted in different runs. We are interested in whether the results obtained would be consistent for both AV and BEST measures.

Having the GM value for the particular problem, we may define the relative improvement of algorithm A over RA on i^{th} problem as

$$IMPR_i(A, RA) = 100\left(1 - \frac{f(A) - GM}{f(RA) - GM}\right) \quad (1)$$

where $f()$ is either the average value of the performance from 51 runs (AV) or the best performance (BEST) found during 51 runs by the algorithm of interest. Here we assume that RA does not find exactly the global minimum for any problem. The idea is to have a measure to verify the relative performance between classical Rosenbrock's algorithm RA and any other algorithm A. We would like to see how much better (what we understand as: closer to the global optimum in terms of fitness) is algorithm A, with respect to RA. The proposed $IMPR_i$ measure is based on the measure of average convergence [148]; however, it is related not directly to GM, but to the difference between the performance obtained by RA and GM. $IMPR_i$ measure shows in percentage (%) how much closer to the global optimum (GM) in terms of performance is the solution found by the particular algorithm A, with respect to the solution found by the classical RA. For example: if the objective function value in the global optimum GM equals 0, the averaged performance from 51 runs equals 100 for RA, and 10 for algorithm A, $IMPR_i$ measure shows 90% improvement of A over RA. Note that the highest possible improvement of A is 100%, but there is no limit on the deterioration (or negative improvement) – if in the above example the averaged performance of algorithm A was 500, $IMPR_i$ would be negative and equal to -400%.

Note that AV and BEST measures refer to the specific problem. However, discussing the relative improvement of a particular algorithm on each specific problem would not be much effective. Rather, we need some aggregate information on the improvement. AV or BEST improvement averaged over all problems may be the simplest measure, but it is highly affected by the problems on which the specific algorithm A performs poorly, as there is no limit on the negative improvement (see discussion on two examples given above). Median AV or BEST improvement is insensitive to the scale of improvement on problems on which particular method performs especially well, or equally bad, hence shows a flattened picture for a "typical" problem. Having this in mind we will discuss the mean, median, lowest, and biggest improvement of AV and BEST measures for each algorithm.

C. EVOLUTIONARY ALGORITHMS COMPARED

In this study we compare the performance of 69 optimization algorithms (see Table 1). Algorithms are arranged historically, from the oldest (RA proposed in 1960, NMA proposed in 1965, etc) to the recent ones, proposed in 2021. Tested algorithms compose just a small fraction of EAs that were proposed so far and the choice of competitors is obviously subjective. We aim at comparison between a large number of methods, including both the best algorithms that we were aware of, and many less known ones. We have, though, tried to avoid metaphor-based methods in which novelty or performance could be doubtful – as discussed in landmark critical papers [43], [149]. However, we are aware that many widely appreciated, popular methods are anyway not included in the present test. Some tested algorithms we have programmed ourselves in MATLAB, but the majority of codes were obtained from their inventors, as indicated in Table 1.

Appropriate choice of control parameters for metaheuristics is often a deliberate issue. We are against fitting all control parameters to the same value, as depending on the specific operators of particular algorithms, different settings may be beneficial for each particular method. Hence, in this paper we use the simplest, but also one of the most fair approaches: the control parameters of tested algorithms are given as suggested in the source papers. A very few exceptions that needed to be made due to technical reasons are listed in Table 1. Also, the box-constraints handling methods employed were the same as in the source papers or computer codes, if they were specified there; otherwise, we used the rebounding method.

IV. RESULTS AND DISCUSSION

In this section, we first pay attention to the averaged-based ranking of algorithms across all problems. After that, we discuss the relative improvement obtained by particular methods over the RA algorithm from the 1960's.

TABLE 1. Algorithms compared. D– problem dimensionality. * - marks codes that have been obtained from other researchers, either on request or from the relevant web pages.

Chronological number	Short name	Long/descriptive name	Citation	Year	Comments
1	RA	Rosenbrock's algorithm	[4]	1960	Reference algorithm.
2	NMA	Nelder – Mead simplex	[3]	1965	A classical algorithm that is still used in various applications and packages.
3	DE	Differential Evolution	[8]	1995	Population size = 5D (but within [10,500]), F = 0.8, CR = 0.5, DE/rand/1 mutation.
4	PSO	Particle Swarm Optimization	[9]	1995	Population size = 40, $c_1 = c_2 = 1.49$, inertia weight decrease linearly with time within [0.9,0.4].
5	CLPSO	Comprehensive learning PSO	[85]*	2006	Population size = 40. State-of-the-art PSO variant. The code has been obtained from its authors on request.
6	AMALGAM	Self-adaptive multimethod search	[86]*	2009	Ensemble algorithm based to large extent on Covariance Matrix Adaptation Evolution Strategy. One of the first multimethod ensembles with re-starts and increasing population size. The code has been obtained from its authors on request.
7	JADE	Adaptive DE	[87]	2009	Population size = 100. State-of-the-art DE variant.
8	DEGL	DE with Global and Local neighborhood mutation operators	[88]	2009	Population size = 5D (but within [10,500]).
9	SADE	Self-Adaptive DE	[89]	2009	Population size = 50.
10	SFMDE	Super-Fit Memetic DE	[90]	2009	Population size = 100. A kind of ensemble memetic algorithm, composed of DE, PSO, NMA, and RA.
11	GA-MPC	Genetic Algorithm with multi-parent crossover	[84]*	2011	Population size = 90. The winner of the 2011 IEEE Competition on Evolutionary Computation (CEC). The code has been downloaded from the CEC2011 web page: www3.ntu.edu.sg/home/epnsugan/index_files/CEC11-RWP/CEC11-RWP.htm
12	AdapSS-JADE	JADE with adaptive strategy selection	[91]	2011	Population size = 100.
13	CDE	Clustering DE	[92]	2011	Population size = 100.
14	EPSDE	DE with ensemble of strategies and parameters	[93]	2011	Population size = 50.
15	jDElscop	Multi-strategies self-adaptive DE	[94]	2011	Population size is initialized at 10D (but within [50,500]) and is gradually reduced during the run.
16	SspDE	DE with self-adaptive strategy and parameters	[95]	2011	Population size = 100.
17	MDE_pBX	Memetic adaptive DE with new mutation and crossover	[96]*	2012	Population size = 100. The code has been obtained from its authors on request.
18	DE-SG	Differential Evolution with separated groups	[97]	2012	Population size = 2D (but within [20,500]).
19	SapsDE	DE with adaptive resizing mechanism	[98]	2013	Population size is variable, initialized with 1D (but within [10,500]).
20	PMS	Parallel Memetic Structures	[99]	2013	Non-population based EA inspired and partly based on RA.
21	AM-DEGL	Adaptive memetic DEGL	[100]	2013	Population size = 5D (but within [10,500]). The algorithm also uses NMA as sub-procedure.
22	ALC-PSO	PSO with ageing leader	[101]	2013	Population size = 20.
23	ATPS-DE	JADE with adaptive population tuning	[102]	2013	Population size is adaptive during the run, initialized with 5D (but within [50,200]).
24	L-SHADE	SHADE with linear population size reduction	[103]*	2014	A version of Successful History Adaptive DE with population size linearly reduced during a run from 18D at the beginning to 4 at the end. State-of-the-art algorithm. The code has been obtained from its authors on request.
25	CoBiDE	DE based on Covariance Matrix learning	[104]	2014	Population size = 60. DE algorithm that adaptively rotates coordinate system during the search moves, and uses a bimodal distribution setting of parameters. State-of-the-art algorithm.
26	LBBO	Linearized Biogeography-based Optimization	[105]*	2014	Population size = 50. The code has been obtained from its author's web page http://academic.csuohio.edu/simond/bbo/linearized/
27	Rcr-JADE	JADE with crossover reparation	[106]	2014	Population size = 100. The probability of crossover is updated according to real occurrence in successful moves.

TABLE 1. ***(Continued.)*** **Algorithms compared. D– problem dimensionality. * - marks codes that have been obtained from other researchers, either on request or from the relevant web pages.**

28	SPS-L-SHADE-EIG	L-SHADE with successful parent selection	[107]*	2015	Population size is linearly reduced from 19*D* to 4. The algorithm adaptively rotates the coordinate system. The code has been obtained from CEC 2015 web page www3.ntu.edu.sg/home/epnsugan/index _ files, currently it is available at https://github.com/P-N-Suganthan/CEC2015-Learning-Based.
29	JADE-EIG	JADE with eigenvector-based crossover	[108]	2015	Population size = 5*D* (but within [30,500]).
30	HCLPSO	Heterogeneous comprehensive learning PSO	[109]*	2015	Population size = 40. The code has been obtained from its authors on request.
31	IDE	DE with individual independent mechanism	[110]*	2015	Population size depends on the dimensionality. It is set to 50 for up to 10-dimensional problems, to 200 for over 50-dimensional problems, and is scaled linearly between 50 and 200 for problems with dimensionalities between 10 and 50. The code has been obtained from its authors on request.
32	JADE-AEPD	JADE with auto-enhanced population diversity	[111]	2015	Population size = 100.
33	JADEEP	JADE with evolution path	[112]	2015	Population size = 100.
34	MPADE	Adaptive DE with sub-populations	[113]	2016	Total population size = 200. The code has been obtained from its authors on request.
35	MPEDE	Multi-population DE ensemble	[114]	2016	Total population size = 250. The code has been obtained from its authors on request.
36	IILPSO	Inter-swarm interactive learning PSO	[115]	2016	Population size = 50 (divided into two swarms).
37	ANS	Across Neighborhood Search	[116]*	2016	Population size = 20. The code has been obtained from http://guohuawunudt.gotoip2.com/publications.html.
38	cNrGA	Non-revisiting Genetic Algorithm	[117]*	2016	Population size = 100. The code has been obtained from www.ee.cityu.edu.hk/~syyuen/Public/Code.html.
39	GLPSO	Genetic learning PSO	[118]	2016	Population size = 50. The method mixes properties of PSO and Genetic Algorithm.
40	HMJCDE	Hybrid memetic CoDE and JADE	[119]	2016	Population size = 100. The hybrid algorithm that merges two state-of-the-art DE variants.
41	L-SHADE-SPACMA	Hybrid algorithm that merges L-SHADE and CMA-ES	[120]*	2017	Population size is linearly decreased for 18*D* at the beginning of the search to 4 at the end. The code has been obtained from https://github.com/P-N-Suganthan/CEC2017-BoundContrained/blob/master/Codes-of-Top-Methods-and-results.zip.
42	L-SHADE-cnEpSin	Ensemble sinusoidal parameter adaptation L-SHADE	[121]*	2017	Population size is linearly decreased from 18*D* at the beginning of the search to 4 at the end. The algorithm is a hybrid of DE variants merged into L-SHADE framework. The code has been obtained from https://github.com/P-N-Suganthan/CEC2017-BoundContrained/blob/master/Codes-of-Top-Methods-and-results.zip.
43	EPSO	Ensemble of PSO variants	[122]*	2017	Population size = 40, divided into two uneven swarms. The code has been obtained from https://github.com/P-N-Suganthan/CODES/blob/master/2017-ASOC-EPSO.zip.
44	ETI-SHADE	SHADE with event-triggered scheme	[123]*	2017	Population size = 150. The code has been obtained from its authors on request.
45	HIVBBO	Hybrid Invasive Weed and Biogeography-based optimization	[124]*	2017	Population size = 100. The code has been obtained from http://embeddedlab.csuohio.edu/BBO/IWO.html.
46	L-JADE	JADE with linear population size reduction	[27]*	2018	Population size is linearly decreased from 18*D* at the beginning of the search to 4 at the end. This is a JADE algorithm with linear population size reduction added. The code has been obtained from the authors of L-MPEDE on request.
47	L-MPEDE	MPEDE with linear population size reduction	[27]*	2018	Population size is linearly decreased from 18*D* at the beginning of the search to 4 at the end. This is a MPEDE algorithm with linear population size reduction added. The code has been obtained from its authors on request.
48	DSO	Drone Squadron Optimization	[125]*	2018	Population size = 100 divided into 4 groups. The code has been obtained from https://github.com/melovv/DSO-MATLAB.
49	EFADE	Enhanced fitness-adaptive DE	[126]*	2018	Population size = 50. The code has been obtained from its authors on request.
50	L-SHADE-50	Simplification of L-SHADE variants	[127]	2018	Population size is linearly decreased from 18*D* at the beginning of the search to 4 at the end. In contradiction to the majority of new variants, this algorithm is simpler than the algorithms on which it is based. An algorithm proposed by the authors of the present papers.
51	L-SHADE-50-PWI	L-SHADE-50 with PSO-based inertia weight	[128]	2018	Population size is linearly decreased from 18D at the beginning of the search to 4 at the end. An algorithm proposed by the authors of the present papers.

TABLE 1. ***(Continued.)*** **Algorithms compared. D– problem dimensionality. * - marks codes that have been obtained from other researchers, either on request or from the relevant web pages.**

52	HSES	Hybrid Sampling Evolution Strategy	[129]*	2018	Population size = 200. The winner of the IEEE Competition on Evolutionary Computation in 2018. The code has been obtained from https://github.com/P-N-Suganthan?tab=repositories
53	EnsDE	Ensemble of DE algorithms	[130]*	2018	Population size = 100. State-of-the-art ensemble of DE variants. The code has been obtained from https://github.com/P-N-Suganthan?tab=repositories.
54	DEPSO	Dual environmental PSO	[131]*	2019	Population size = 50. The code has been obtained from its authors on request.
55	HARD-DE	Hierarchical archive-based DE	[132]*	2019	Population size is parabolically (quicker at the end of the search) decreased during a run from $25\ln(D)\sqrt{D}$ to 4. The code has been obtained from https://sites.google.com/view/zhenyumeng/
56	jDE	DE with diversity and adaptive population size	[133]*	2019	Population size is initialized with 50 but varies adaptively during search within the [8,5D] range. The algorithm is based on [135], but with a diversity-based mechanism. The code has been obtained from its authors on request.
57	SOMA T3A	Version of self-organizing migrating algorithm	[135]*	2019	Population size = 1500. The code has been obtained from https://github.com/P-N-Suganthan/CEC2019.
58	TAPSO	Triple-archive PSO	[136]*	2020	Population size = 60. The code has been obtained from its authors on request.
59	TbL-SHADE	Tuning-based mutation L-SHADE	[137]*	2020	Population size = 100. The code has been obtained from its authors on request.
60	IMODE	Improved multi-operator DE	[138]*	2020	Population size is variable and initialized to $6D^2$ (but no more than 10000). IMODE is the winner of the IEEE Competition on Evolutionary Computation 2020. The code has been obtained from https://github.com/P-N-Suganthan/2020-Bound-Constrained-Opt-Benchmark.
61	AGSK	Adaptive gaining-sharing knowledge algorithm	[139]*	2020	Population size was set by the authors of the algorithm to 40D if $D < 6$ and to 100 otherwise. AGSK is the second-best algorithm in the IEEE Competition on Evolutionary Computation 2020. The code has been obtained from https://github.com/P-N-Suganthan/2020-Bound-Constrained-Opt-Benchmark.
62	CIJADE	Hybrid DE	[140]*	2020	Population size = 100. The code has been obtained from its authors on request.
63	Di-DE	Depth information-based DE	[141]*	2020	Population size = 10D. The code has been obtained from https://sites.google.com/view/zhenyumeng/.
64	APGSK-IMODE	Hybrid of APGSK and IMODE	[142]*	2021	Population size is variable. It is initialized to 30D. The algorithm is a hybrid the of two best algorithms in the IEEE Competition on Evolutionary Computation 2020. The code has been obtained from https://github.com/P-N-Suganthan/2021-SO-BCO.
65	CS-DE	Cooperative strategy-based DE	[143]*	2021	Population size is variable, initialized with $25\ln(D)\sqrt{D}$ (but not fewer than 25). The code has been obtained from https://sites.google.com/view/zhenyumeng/.
66	ELSHADE-SPACMA	Enhanced L-SHADE-SPACMA	[144]*	2021	Population size is linearly reduced during a run from 18D at the beginning to 4 at the end. Hybrid algorithm. The code has been obtained from its authors on request.
67	FDBSFS	Fitness distance-based stochastic fractional search	[145]*	2021	Population size = 50. The code has been obtained from https://www.mathworks.com/matlabcentral/fileexchange/47565-stochastic-fractal-search-sfs.
68	HIP-DE	Historical population-based adaptive DE	[146]*	2021	Population size is variable, initialized to 15D. The code has been obtained from https://sites.google.com/view/zhenyumeng/.
69	MaDE	Bayesian hyperparameter-optimization DE	[147]*	2021	Population size is linearly decreasing during a run, from initial $2D^2$ to 4 at the end of the run. The code has been obtained from https://github.com/P-N-Suganthan/2021-SO-BCO.

A. RANK-BASED CLASSIFICATION OF ALGORITHM

Before we analyze the relative improvement over RA obtained by different EAs on benchmarks and real-world problems, some discussion on the classical rank-based ranking of algorithms is needed. In Table 2 the algorithms are listed from the best to the worst one according to the all-problems averaged ranks. In the left part of Table 2 the average ranking is based on 22 real-world problems from the 2011 set, and on the right – on the 50-dimensional version of 30 CEC 2017 benchmarks. To facilitate a comparison of rankings obtained by older and more recent EAs, the year of publication is associated with each algorithm in Table 2. In Table 3 similar information is given, but for algorithms arranged historically, with chronological numbers specified in Table 1; in Table 3 we add specific information on the average rank of each algorithm. To simplify the discussion on historical improvement, in Table 4 the average year of publication, and the most recent year of publication are presented for the 10 best and the 10 worst algorithms, and in Table 5 the average ranking of the 10 oldest, and the average ranking of the 10 most recent algorithms is given. The detailed results obtained by each algorithm on every problem are listed in Suppl. Tables 2-5.

TABLE 2. **Average-ranks-based ranking of algorithms arranged from the best to the worst for real-world 2011 problems and 50-dimensional CEC 2017 benchmarks.**

Real-world problems 2011			CEC benchmarks 2017		
Position in ranking	Publicatio year	Algorithm's name	Position in ranking	Publication year	Algorithm's name
1	2019	HARD_DE	1	2021	ELSHADE_SPACMA
2	2017	L_SHADE_cnEpSin	2	2017	L_SHADE_SPACMA
3	2016	MPADE	3	2017	L_SHADE_cnEpSin
4	2018	L_SHADE_50_PWI	4	2018	L_SHADE_50
5	2014	L_SHADE	5	2018	HSES
6	2020	CIJADE	6	2018	L_SHADE_50_PWI
7	2018	L_SHADE_50	7	2019	HARD_DE
8	2021	HIP_DE	8	2021	HIP_DE
9	2014	CoBiDE	9	2021	CS_DE
10	2021	CS_DE	10	2014	L_SHADE
11	2016	MPEDE	11	2009	AMALGAM
12	2017	L_SHADE_SPACMA	12	2015	SPS_L_SHADE_EIG
13	2015	SPS_L_SHADE_EIG	13	2017	ETI_SHADE
14	2014	Rcr_JADE	14	2018	L_MPEDE
15	2013	ATPS_DE	15	2018	L_JADE
16	2018	L_MPEDE	16	2020	Di_DE
17	2021	ELSHADE_SPACMA	17	2015	IDE
18	2016	HMJCDE	18	2015	JADE_EIG
19	2018	EFADE	19	2020	CIJADE
20	2015	JADE_AEPD	20	2016	MPADE
21	2009	JADE	21	2019	jDE
22	2015	JADEEP	22	2018	EnsDE
23	2018	L_JADE	23	2013	AM_DEGL
24	2020	TbL_SHADE	24	2014	Rcr_JADE
25	2015	JADE_EIG	25	2016	MPEDE
26	2013	AM_DEGL	26	2018	EFADE
27	2011	GA_MPC	27	2014	CoBiDE
28	2009	SADE	28	2013	ATPS_DE
29	2018	EnsDE	29	2015	JADE_AEPD
30	2013	SapsDE	30	2009	JADE
31	2011	AdapSS_JAD	31	2016	HMJCDE
32	2012	MDE_pBX	32	2015	JADEEP
33	2015	IDE	33	2011	AdapSS_JADE
34	2011	SspDE	34	2011	GA_MPC
35	2015	HCLPSO	35	2017	HIVBBO
36	2017	EPSO	36	2011	jDElscop
37	2017	ETI_SHADE	37	2013	SapsDE
38	2019	jDE	38	2019	DEPSO
39	2011	CDE	39	2009	SADE
40	2017	HIVBBO	40	2011	SspDE
41	2011	jDElscop	41	2015	HCLPSO
42	2009	AMALGAM	42	2017	EPSO
43	2011	EPSDE	43	2020	AGSK
44	2020	Di_DE	44	2009	DEGL
45	2016	GLPSO	45	2020	TbL_SHADE
46	2020	TAPSO	46	2021	APGSK_IMODE
47	2014	LBBO	47	2012	MDE_pBX
48	2021	FDBSFS	48	2006	CLPSO
49	2006	CLPSO	49	2011	EPSDE
50	2020	IMODE	50	2011	CDE
51	2018	HSES	51	2020	TAPSO
52	2021	APGSK_IMODE	52	2016	GLPSO
53	2020	AGSK	53	2012	DE_SG
54	2021	MaDE	54	1965	NMA
55	2016	IILPSO	55	2014	LBBO
56	2018	DSO	56	2020	IMODE
57	2019	DEPSO	57	2021	FDBSFS
58	1965	NMA	58	2016	IILPSO
59	2009	DEGL	59	2016	cNrGA
60	2012	DE_SG	60	2016	ANS
61	2016	cNrGA	61	2021	MaDE
62	1995	PSO	62	2018	DSO
63	2013	PMS	63	1995	PSO
64	2013	ALC_PSO	64	2009	SFMDE
65	2009	SFMDE	65	1960	RA
66	1960	RA	66	2013	ALC_PSO
67	2016	ANS	67	2013	PMS
68	1995	DE	68	1995	DE
69	2019	SOMA_T3A	69	2019	SOMA_T3A

TABLE 2. ***(Continued.)*** **Average-ranks-based ranking of algorithms arranged from the best to the worst for real-world 2011 problems and 50-dimensional CEC 2017 benchmarks.**

From Tables 2 and 3 we find that the HARD-DE algorithm from the year 2019, is the best method for real-world problems and the ELSHADE-SPACMA from 2021 is the winner on 50-dimensional CEC 2017 benchmarks. L-SHADE-cnEpSin, which was proposed in 2017, is the second-best method on real-world problems and the third-best on CEC 2017 benchmarks, which makes it the rank-based winner of the summarized competition. Such results suggest that at least some of the most recent algorithms are better than the older competitors. Indeed, from Table 4 we see that the average publication year of the 10 best algorithms, out of 69 tested, is close to 2018 for both real-world and benchmark problems, when the average year of publication of the 10 worst algorithms is between 2004-2006. Also, as given in Table 5, the mean ranking of the 10 newest algorithms is much better than the mean ranking of the 10 oldest methods. From Table 2 we see that among the best 20 algorithms on real-world problems, the oldest is from 2013 (ATPS-DE – which is ranked 15th). In the case of CEC 2017 benchmarks AMALGAM from 2009 is ranked 11th and is the only over-10-years-old method among the best 20 algorithms. This confirms that, as long as problem-averaged rank-based comparison of algorithms is considered, the older methods are not competitive with the newer ones.

From Table 2 we may also find that L-SHADE [103] based methods in general perform better than other versions of Swarm Intelligence and Evolutionary Algorithms that have been tested in the present study. We may recognize many variants of L-SHADE among the best ten optimizers, based on both the results obtained for real-world problems and CEC 2017 benchmark functions. As L-SHADE algorithms have also found so far many practical applications in different fields of science and engineering [150], [151], [152], [153], the present result may be considered as another confirmation of their superiority, or at least a high performance among other Evolutionary Algorithms.

On the other hand, some recently proposed algorithms may perform very poorly even according to the averaged ranks. From Table 4 we find that algorithms proposed in the

TABLE 3. Average ranks of algorithms on real-world problems 2011 and 50-dimensional CEC benchmarks 2017.

			Real-world problems 2011		CEC benchmarks 2017	
Chronological number	Algorithms name	Publication year	Position in ranking	Average rank	Position in ranking	Average rank
1	RA	1960	66	53.3	65	59.0
2	NMA	1965	58	49.3	54	50.5
3	DE	1995	68	57.8	68	67.4
4	PSO	1995	62	50.1	63	57.3
5	CLPSO	2006	49	41.1	48	45.1
6	AMAL-GAM	2009	42	36.8	11	13.8
7	JADE	2009	21	27.9	30	29.9
8	DEGL	2009	59	49.5	44	41.4
9	SADE	2009	28	30.8	39	37.2
10	SFMDE	2009	65	53.3	64	58.8
11	GA-MPC	2011	27	30.0	34	32.9
12	AdapSS-JADE	2011	31	31.4	33	31.1
13	CDE	2011	39	35.7	50	46.1
14	EPSDE	2011	43	37.0	49	45.3
15	jDElscop	2011	41	36.8	36	34.8
16	SspDE	2011	34	32.5	40	38.3
17	MDE_pBX	2012	32	31.4	47	43.6
18	DE-SG	2012	60	49.5	53	50.4
19	SapsDE	2013	30	31.3	37	35.3
20	PMS	2013	63	50.7	67	60.8
21	AM-DEGL	2013	26	29.6	23	26.4
22	ALC-PSO	2013	64	51.1	66	60.2
23	ATPS-DE	2013	15	25.0	28	29.1
24	L-SHADE	2014	5	18.8	10	12.5
25	CoBiDE	2014	9	19.9	27	28.2
26	LBBO	2014	47	40.4	55	51.0
27	Rcr-JADE	2014	14	24.7	24	26.9
28	SPS-L-SHADE-EIG	2015	13	24.6	12	14.6
29	JADE-EIG	2015	25	29.4	18	24.7
30	HCLPSO	2015	35	33.8	41	40.0
31	IDE	2015	33	32.0	17	24.6
32	JADE-AEPD	2015	20	27.3	29	29.4
33	JADEEP	2015	22	29.1	32	30.5
34	MPADE	2016	3	17.7	20	25.4
35	MPEDE	2016	11	22.4	14	27.4
36	IILPSO	2016	55	46.9	58	53.8
37	ANS	2016	67	54.3	60	55.9
38	cNrGA	2016	61	49.7	59	55.0
39	GLPSO	2016	45	39.8	52	47.8
40	HMJCDE	2016	18	26.2	31	30.0
41	L-SHADE-SPACMA	2017	12	22.7	2	7.8
42	L-SHADE-cnEpSin	2017	2	17.7	3	9.2
43	EPSO	2017	36	34.0	42	40.5
44	ETI-SHADE	2017	37	34.4	13	18.1
45	HIVBBO	2017	40	35.7	35	34.1
46	L-JADE	2018	23	29.3	15	20.1
47	L-MPEDE	2018	16	25.8	14	18.9
48	DSO	2018	56	48.1	62	56.4
49	EFADE	2018	19	27.0	26	28.1
50	L-SHADE-50	2018	7	19.8	4	9.3
51	L-SHADE-50-PWI	2018	4	17.8	6	10.3

TABLE 3. *(Continued.)* Average ranks of algorithms on real-world problems 2011 and 50-dimensional CEC benchmarks 2017.

52	HSES	2018	51	42.3	5	9.9
53	EnsDE	2018	29	31.2	22	25.9
54	DEPSO	2019	57	48.3	38	36.9
55	HARD-DE	2019	1	17.0	7	10.8
56	jDE	2019	38	35.2	21	25.7
57	SOMA T3A	2019	69	62.1	69	67.7
58	TAPSO	2020	46	40.1	51	47.6
59	TbL-SHADE	2020	24	29.3	45	41.7
60	IMODE	2020	50	42.0	56	51.9
61	AGSK	2020	53	43.7	43	40.6
62	CIJADE	2020	6	19.3	19	25.2
63	Di-DE	2020	44	39.0	16	23.0
64	APGSK-IMODE	2021	52	43.3	46	42.9
65	CS-DE	2021	10	21.0	9	12.1
66	ELSHADE-SPACMA	2021	17	26.0	1	7.3
67	FDBSFS	2021	48	40.5	57	53.1
68	HIP-DE	2021	8	19.8	8	11.5
69	MaDE	2021	54	43.7	61	56.2

TABLE 4. The average year of publication of the 10 best and 10 worst algorithms according to the average ranking, and the year of the most recent algorithm that is within the 10 best or 10 worst methods.

	Average year of algorithm's publication		Most recent year of algorithm's publication	
	Real-world problems 2011	CEC benchmarks 2017	Real-world problems 2011	CEC benchmarks 2017
Best 10 algorithms	2017.8	2018.4	2021	2021
Worst 10 algorithms	2004.8	2005.9	2019	2021

recent three years are both among the best 10, and among the worst 10 for each set of problems. SOMA_T3A from 2019 is the worst method out of 69 algorithms on both real-world problems and artificial benchmarks. MaDE from 2021 is ranked 54th on real-world problems and 61st on artificial benchmarks, and performs poorer than the classical Nelder-Mead algorithm (NMA) from 1965, which is ranked 58th and 54th, accordingly. Hence, according to the average ranking, some most recent methods are indeed better than all tested former competitors, but still many novel algorithms perform poorly, even when compared with much older methods.

B. RELATIVE IMPROVEMENT OVER ROSENBROCK'S METHOD

The method proposed by Rosenbrock in 1960 (RA) is extremely simple, does not use a population framework, and may be applied to solve any continuous optimization problem. This motivates us to analyze the relative improvement obtained by various EAs over such an old, simple, and flexible algorithm. As we found in the previous sections, RA is not

TABLE 5. The mean ranking obtained by the 10 oldest and 10 most recent algorithms, and the mean average rank of the 10 oldest and 10 most recent algorithms.

	Mean ranking of algorithms		Mean average-rank of algorithms	
	Real-world problems 2011	CEC benchmarks 2017	Real-world problems 2011	CEC benchmarks 2017
10 oldest algorithms	51.8	48.6	45.0	48.6
10 most recent algorithms	34.2	31.6	33.8	32.4

much competitive with EAs when the rank-based approach is used.

The meaning of the relative improvement of a particular algorithm over RA is defined in eq. 1. Note that 51 runs of each algorithm are made on every problem, hence the improvement may be counted for the averaged performance from 51 runs (AV), or for the best performance reached during 51 runs (BEST) made by the particular algorithm. As tests were made on 22 real-world problems and 30 mathematical functions, we illustrate 1. the improvement averaged over all problems, 2. the median improvement from all problems, 3. the lowest, and 4. the biggest improvement achieved across all problems. Such four statistics are separately shown for AV on 30 CEC 2017 benchmarks (Fig. 1), BEST on 30 CEC 2017 benchmarks (Fig. 2), AV on 22 real-world problems (Fig. 3) and BEST on 22 real-world problems (Fig. 4). Note that in Figs 1-4 the algorithm number 1 is RA, hence its relative improvement is always equal to 0. We remind that, according to eq. (1), for the specific problem there is no limit on the worst (lowest) improvement, but there is a maximum possible improvement (100%). Hence, to keep important details visible in Figs 1-4, we have limited the range of improvements shown to [-100%, 100%] for mean, median, and the largest improvement, and to [-1000%, 100%] for the lowest noted improvement; we assume that performances that are over 10 times poorer than performances obtained by RA may be classified as a big failure, does not matter how poor they are.

From the top pictures of Figs 1-4 one may note that, with some surprise, the mean relative improvement obtained by the majority of EAs over RA is negative, which means that if the improvement is averaged over all problems, the majority of EAs lead to the poorer solutions than RA. This is observed for both CEC 2017 benchmarks (Figs 1-2) and real-world problems (Figs 3-4), and both when the average (AV, Figs 1 and 3) or the best performance out of 51 runs on a particular problem (BEST, Figs 2 and 4) is considered. Why this is the case? Many EAs perform very poorly on a few specific problems, which is clearly seen in the third plot from the top of Figs 1-4. For example, on problem 3 from 50-dimensional CEC 2017 benchmarks, for which the value of the function in the global optimum equals 0, the 51-runs averaged performance of RA is slightly below 0.02 (see Suppl. Table 2). However, the 51-runs averaged performance of 36 EAs on this problem is higher than 100. Hence, over half of the tested EAs are 4 orders of magnitude weaker than the RA on this problem. Because the maximum possible relative improvement is 100% (see eq. 1), and there is no restriction on the negative improvement, such poor results on a single problem mean that the relative improvement averaged over all problems turns out to be negative. Similarly difficult for many EAs are real-world problems nr. 4, which aims at the optimal control of a stirred tank reactor, and nr. 9, which focuses on large-scale transmission pricing in power systems. Relatively good performance of RA on some tasks on which many EAs face problems means that, when the 51-runs averaged performance is considered (AV), only 25 EAs show positive relative improvement over RA on CEC 2017 benchmarks, and only 18 EAs – on real-world problems. Note that such very good performance of RA on a few problems would be lost from our view if average ranks were to be used for comparison among optimizers.

All this means that by searching for the positive mean relative improvement we may recognize EAs that avoid failures on any problem, rather than pointing at those that perform very well for many problems. Based on the 51-runs averaged performance (top pictures in Fig 1 and 3), we find that only AMALGAM from 2009, Rcr_JADE from 2014, MPEDE and HMJCDE from 2016, EPSO, ETI-SHADE and HIVBBO from 2017, and CS-DE from 2021 show the positive mean relative improvement over RA on both CEC 2017 benchmarks and 2011 real-world problems. When we consider both averaged, and the best performance from 51 runs, just a single EA, namely HMJCDE from 2016, shows consistent positive mean relative improvement over RA (top pictures of Figs 1-4). Interestingly, HMJCDE was not ranked especially high in rank-based analysis, being 18th and 31st on real-world problems and CEC 2017 benchmarks, respectively (see Tables 2 and 3). HMJCDE avoids a serious failure on any problem but also does not perform especially well on any tasks.

A different picture appears when we focus on median relative improvement over RA (second pictures from the top on Figs 1-4), where we compare improvements on a typical problem, skipping extremes. When we consider the averaged performance from 51 runs, median relative improvement over RA is generally positive for almost all EAs, with exception of the basic DE from 1995 and SOMA-T3A from 2019. When the best performance among 51 runs on real-world problems is considered (see Fig. 4), the median improvement is negative for 6 EAs. However, although the values of median improvement over RA are often positive, they highly differ for various algorithms and are much different for CEC 2017 benchmarks than for real-world problems. In the case of CEC 2017 benchmarks the median relative improvement over RA for the majority of algorithms exceeds 50%, and for some (AMALGAM from 2009, HSES from 2018, EL-SHADE-SPACMA from 2021) is close to 95%, both for the average (see Fig. 1) and the best performance from 51 runs (Fig. 2). It means that some EAs are able to find solutions 20 times

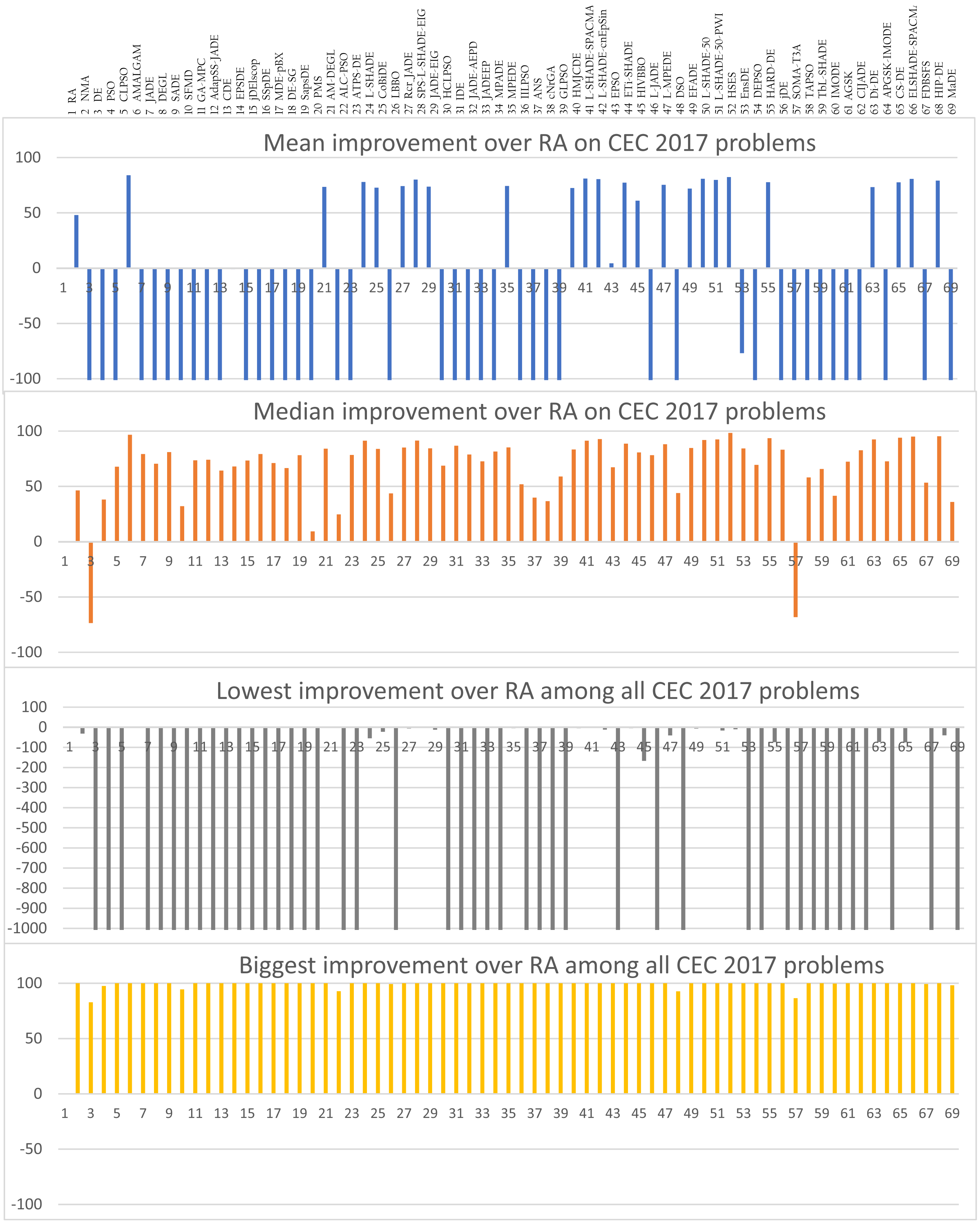


FIGURE 1. Relative improvement over RA of 51-runs average performance. Statistics computed over all CEC 2017 problems.

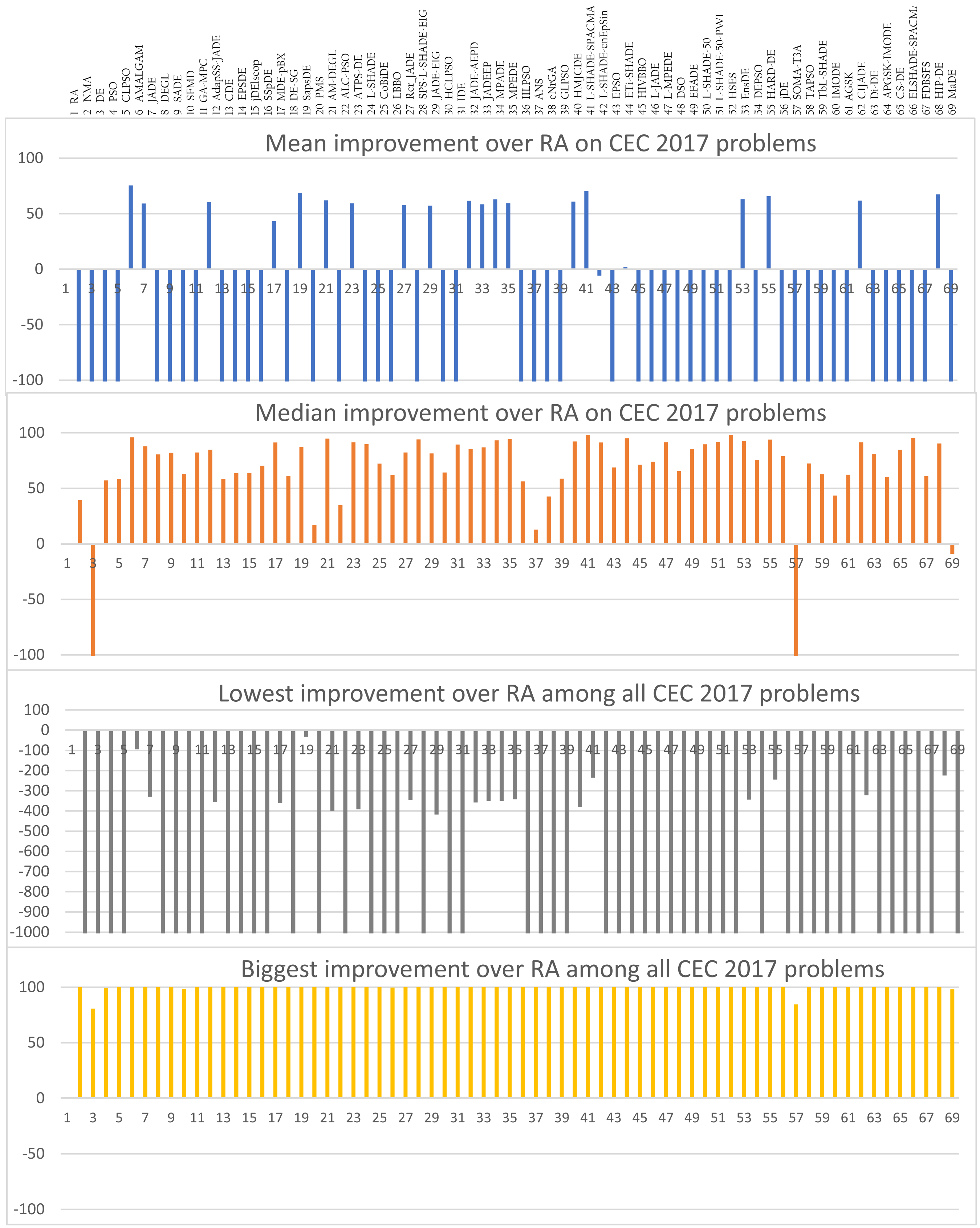


FIGURE 2. **Relative improvement over RA of 51-runs best performance. Statistics computed over all CEC 2017 problems.**

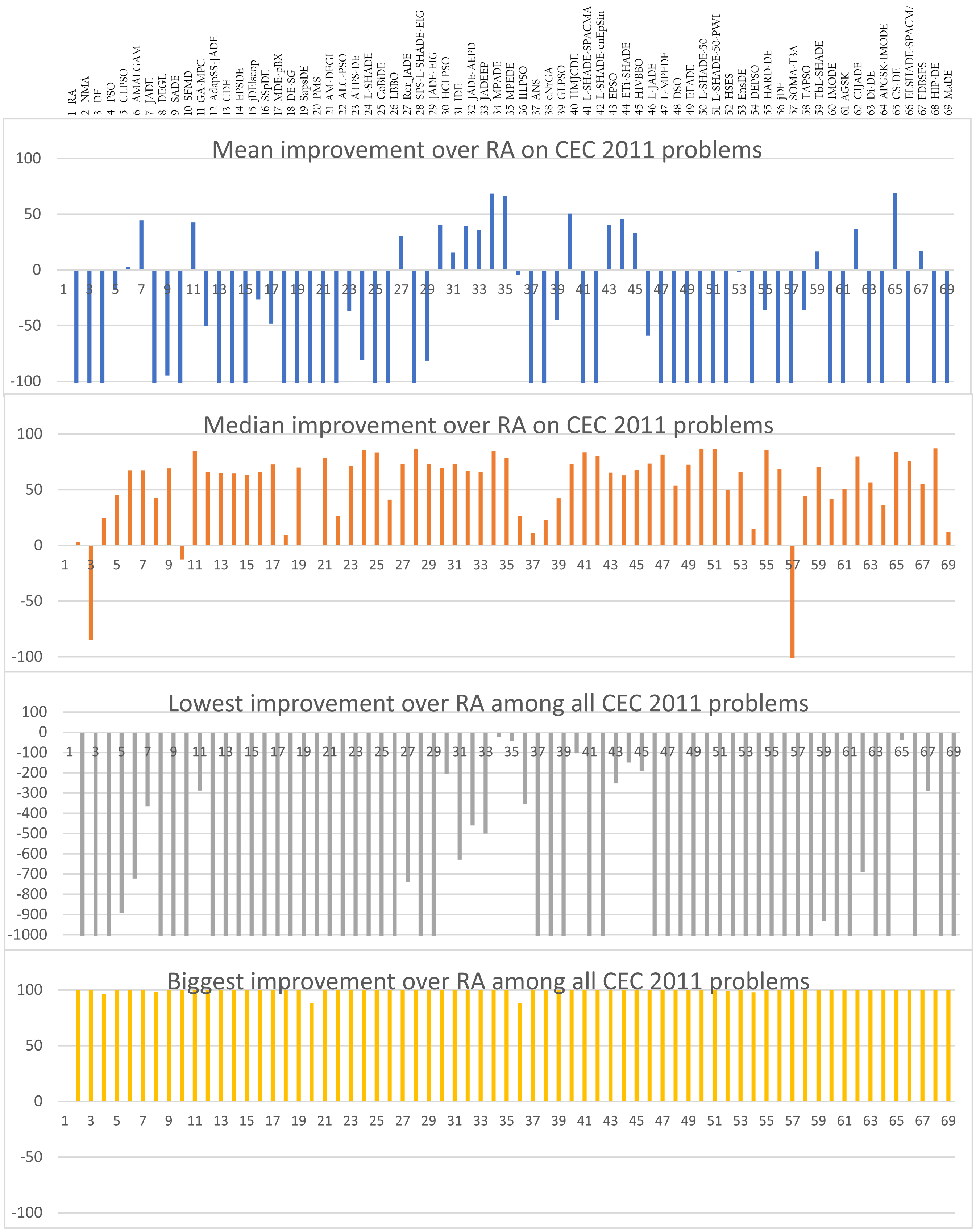


FIGURE 3. Relative improvement over RA of 51-runs average performance. Statistics computed over all real-world problems.

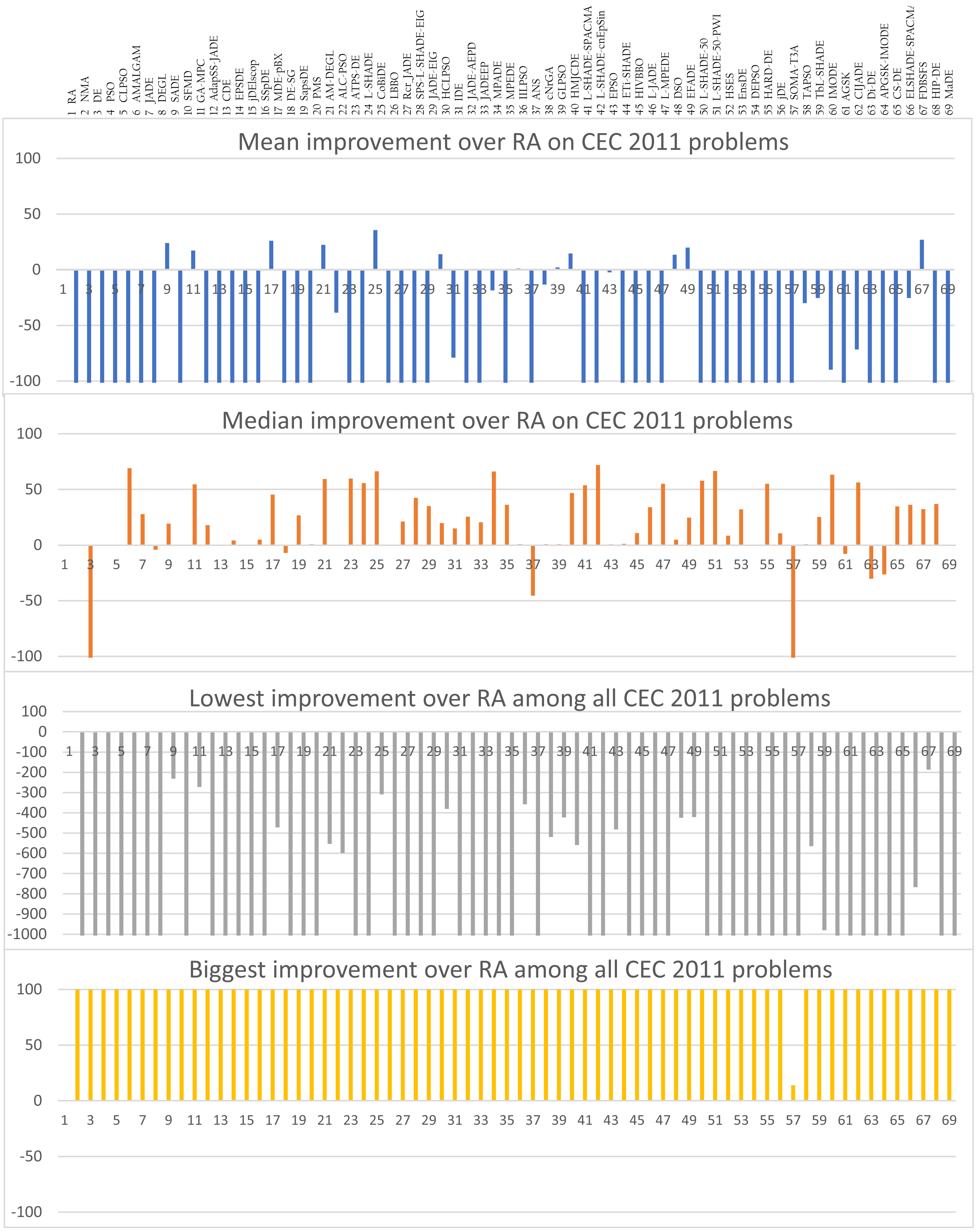


FIGURE 4. Relative improvement over RA of 51-runs best performance. Statistics computed over all real-world problems.

better than RA on a typical benchmark function. In the case of real-world problems, the median relative improvement over RA is lower for most EAs, and it does not reach 90% for any algorithm. If the results from the best runs are compared (see Fig. 4), the median relative improvement over RA on real-world problems is for the vast majority of EAs lower than 50%, and only a few best algorithms reach the value of 70% – what means that they on a typical problem find three times better solutions than RA.

The above discussion shows that the relative improvement obtained by EAs over RA on real-world problems is lower than the relative improvement obtained on CEC 2017 benchmarks. There may be a simple explanation for that: in the source papers, the majority of newly introduced EAs are tested and fitted to either mathematical benchmarks, or to a single specific real-world problem. Comparisons of newly proposed algorithms on a wider range of real-world problems are rare. Moreover, the majority of comparisons between EAs focus on the all-runs averaged performance. However, the comparisons based on the averaged performance may be in contradiction with the expectations of many practical users of Evolutionary Algorithms. Many practitioners would either run the algorithm just once on the problem they are interested in or would seek for the best solution of their problem in numerous runs. In both cases, the comparisons based on the averaged performance may be inadequate. Our results show that if we use a different measure, for example, comparing the best solutions found during all runs on real-world problems, the relative improvements of EAs over RA are observed, but are lower than one could expect.

V. CONCLUSION

When comparing Evolutionary Algorithms often the rank-based approach is used, in which only the ranking of methods is considered, not the actual performance. When ranking is averaged over many problems, a marginal improvement on numerous tasks becomes more important than a substantial improvement on some, or even many problems. When we apply the rank-based approach to analyze the performance of 69 heuristic optimizers on 22 real-world problems and 30 different 50-dimensional IEEE CEC 2017 benchmarks, three relatively recent Evolutionary Algorithms show the best performance: HARD-DE from 2019 (the winner on real-world problems), ELSHADE-SPACMA from 2021 (the winner on IEEE CEC 2017 benchmarks) and L-SHADE-cnEpSin from 2017 (that ranked 2nd and 3rd on these two sets of problems). Among the best 10 algorithms, out of 69, the majority have been proposed since the year 2017. Historical algorithms do not look much competitive: Rosenbrock's algorithm from 1960 is among the five poorest methods, and the famous Nelder and Mead Simplex from 1965 is ranked only slightly better. Although some of the most recent algorithms occupy the bottom positions of such ranking, in general, the newer algorithms look much better than the older ones.

Much different opinion on the performance of Evolutionary Algorithms may be drawn if, instead of the rank-based approach, the relative improvement on various problems over a historical Rosenbrock's algorithm from 1960 is considered for comparison. By relative improvement we mean how much closer, in terms of the objective function value, to the global optimum (if known), or to the best solution known so far, are the solutions found by a particular Evolutionary Algorithm than the solutions found by the reference approach – Rosenbrock's algorithm.

The range of improvement obtained by Evolutionary Algorithms over Rosenbrock's algorithm to a large degree depends on a specific measure that is used (average or best performance from many runs, average or median improvement from many problems). The problem with the improvement averaged over many problems is that the upper limit of the relative improvement is restricted by 100%, but there is no bottom limit on the deterioration of the results. The algorithms that on some problems perform by orders of magnitude poorer than Rosenbrock's approach cannot compensate for this poor result on the remaining problems. As a result, on average the majority of Evolutionary Algorithms note deterioration (negative relative improvement) over Rosenbrock's algorithm. Algorithms that on average gain substantial improvement over Rosenbrock's algorithm are not necessarily the newest ones. Such average improvement rarely exceeds 67% for 50-dimensional CEC benchmarks or 50% for real-world problems. Hence, on average the best Evolutionary Algorithms find solutions 2-3 times closer to the global optimum (in terms of fitness) than the historical Rosenbrock's algorithm.

The median measure is insensitive to the extremes, and it shows the improvement for a "typical" problem. The median value of the relative improvement over Rosenbrock's algorithm is for almost all Evolutionary Algorithms positive, but highly diversified. For many Evolutionary Algorithms, the median improvement exceeds 50%. In the case of the best Evolutionary Algorithms it even exceeds 90% on 50-dimensional CEC benchmarks, and 80% on real-world problems, which means that the solutions found by these methods on a typical problem may be 5-10 times better than the solutions found by the Rosenbrock's algorithm.

However, if one compares only the best results obtained by algorithms in many runs (instead of the performance that was averaged over all runs made), on a "typical" real-world problem only some Evolutionary Algorithms can find 2-3 times better solutions than the Rosenbrock's approach. As many practitioners would be mainly interested in the best solutions that may be found within many runs on real-world problems, not in the averaged performance on mathematical test functions, from a practical point of view these last values are especially important.

The recent algorithms do not show much larger relative improvement over Rosenbrock's method than the algorithms proposed a decade ago. This is because on a "typical" problem the recent algorithms frequently find only slightly better solutions than the older methods – which is sufficient to achieve a better position in ranking but has very limited practical meaning for the majority of users. As a result, despite many efforts, practically meaningful improvement of the performance on real-world problems is still a challenging task for Evolutionary Algorithms.

REFERENCES

[1] K. Sorensen, M. Sevaux, and F. Glover, "A history of metaheuristics," in *Handbook Heuristics*, R. Martí, P. Panos, M. Resende, Eds. Cham, Switzerland: Springer, 2018, pp. 1–18.

[2] G. Polya, *How to Solve it*. Princeton, NJ, USA: Princeton Univ. Press, 1945.

[3] J. A. Nelder and R. Mead, "A simplex-method for function minimization," *Comput. J.*, vol. 7, no. 4, pp. 308–313, 1965.

[4] H. H. Rosenbrock, "An automatic method for finding the greatest or least value of a function," *Comput. J.*, vol. 3, no. 3, pp. 175–184, Mar. 1960.

[5] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: Univ. Michigan Press, 1975.

[6] I. Rechenberg, "Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution," Department of Process Engineering, Dr.-Ing. thesis, Technical Univ. Berlin, Berlin, Germany, 1971.

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[8] R. Storn and K. V. Price, "Differential evolution—A simple and efficient adaptive scheme for global optimization over continuous spaces," Int. Comput. Sci. Inst., Berkeley, CA, USA, Tech. Rep., TR-95-012, 1995.

[9] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Piscataway, NJ, USA, Nov. 1995, pp. 1942–1948.

[10] A. E. Eiben and J. Smith, "From evolutionary computation to the evolution of things," *Nature*, vol. 521, no. 7553, pp. 476–482, May 2015.

[11] K. R. Opara and P. P. Oh, "Regularization and concave loss functions for estimation of chemical kinetic models," *Appl. Soft Comput.*, vol. 116, Feb. 2022, Art. no. 108286.

[12] J. X. Fan, S. Z. Shen, D. H. Erwin, P. M. Sadler, N. MacLeod, Q. M. Cheng, X. D. Hou, J. Yang, X. D. Wang, Y. Wang, H. Zhang, X. Chen, G. X. Li, Y. C. Zhang, Y. K. Shi, D. X. Yuan, Q. Chen, L. N. Zhang, C. Li, and Y. Y. Zhao, "A high-resolution summary of Cambrian to early Triassic marine invertebrate biodiversity," *Science*, vol. 367, pp. 272–277, Apr. 2020.

[13] S. S. Rabotyagov, T. D. Campbell, M. White, J. G. Arnold, J. Atwood, M. L. Norfleet, C. L. Kling, P. W. Gassman, A. Valcu, J. Richardson, R. E. Turner, and N. N. Rabalais, "Cost-effective targeting of conservation investments to reduce the Northern Gulf of Mexico hypoxic zone," *Proc. Nat. Acad. Sci. USA*, vol. 111, no. 52, pp. 18530–18535, Dec. 2014.

[14] K. M. Hassan, A. Abdo, and A. Yakoub, "Enhancement of health care services based on cloud computing in IoT environment using hybrid swarm intelligence," *IEEE Access*, vol. 10, pp. 105877–105886, 2022.

[15] S. Kriegman, D. Blackiston, M. Levin, and J. Bongard, "A scalable pipeline for designing reconfigurable organisms," *Proc. Nat. Acad. Sci. USA*, vol. 117, no. 4, pp. 1853–1859, 2020.

[16] P. Nowak-Sliwinska, A. Weiss, X. Ding, P. J. Dyson, H. van den Bergh, A. W. Griffioen, and C.-M. Ho, "Optimization of drug combinations using feedback system control," *Nature Protocols*, vol. 11, no. 2, pp. 302–315, Feb. 2016.

[17] S. L. Grimm, B.-O. Demory, M. Gillon, C. Dorn, E. Agol, A. Burdanov, and L. Delrez, "The nature of the TRAPPIST-1 exoplanets," *Astron. Astrophys.*, vol. 613, p. A68, May 2018.

[18] L. Yang, Z. Li, D. Wang, H. Miao, and Z. Wang, "Software defects prediction based on hybrid particle swarm optimization and sparrow search algorithm," *IEEE Access*, vol. 9, pp. 60865–60879, 2021.

[19] M. E. C. Bento, D. Dotta, R. Kuiava, and R. A. Ramos, "A procedure to design fault-tolerant wide-area damping controllers," *IEEE Access*, vol. 6, pp. 23383–23405, 2018.

[20] C.-Y. Lee and T.-A. Le, "An enhanced binary particle swarm optimization for optimal feature selection in bearing fault diagnosis of electrical machines," *IEEE Access*, vol. 9, pp. 102671–102686, 2021.

[21] M. Sarwar, A. S. Siddiqui, S. S. M. Ghoneim, K. Mahmoud, and M. M. F. Darwish, "Effective transmission congestion management via optimal DG capacity using hybrid swarm optimization for contemporary power system operations," *IEEE Access*, vol. 10, pp. 71091–71106, 2022.

[22] G. Rajendran, C. A. Vaithilingam, K. Naidu, A. A. Alsakati, K. S. P. Oruganti, and M. F. Fauzan, "Dynamic voltage stability enhancement in electric vehicle battery charger using particle swarm optimization," *IEEE Access*, vol. 10, pp. 97767–97779, 2022.

[23] N. Li, F. He, W. Ma, R. Wang, and X. Zhang, "Wind power prediction of kernel extreme learning machine based on differential evolution algorithm and cross validation algorithm," *IEEE Access*, vol. 8, pp. 68874–68882, 2020.

[24] S. Gao, K. Wang, S. Tao, T. Jin, H. Dai, and J. Cheng, "A state-of-the-art differential evolution algorithm for parameter estimation of solar photovoltaic models," *Energy Convers. Manage.*, vol. 230, Feb. 2021, Art. no. 113784.

[25] X. Yu and M. Gen, *Introduction to Evolutionary Algorithms*. London, U.K.: Springer-Verlag, 2010.

[26] D. Molina, A. LaTorre, and F. Herrera, "An insight into bio-inspired and evolutionary algorithms for global optimization: Review, analysis, and lessons learnt over a decade of competitions," *Cognit. Comput.*, vol. 10, no. 4, pp. 517–544, Aug. 2018.

[27] A. P. Piotrowski and J. J. Napiorkowski, "Step-by-step improvement of JADE and SHADE-based algorithms: Success or failure?" *Swarm Evol. Comput.*, vol. 43, pp. 88–108, Dec. 2018.

[28] U. Skvorc, T. Eftimov, and P. Korsec, "CEC real-parameter optimization competitions: Progress from 2013 to 2018," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Wellington, New Zealand, Jun. 2019, pp. 3126–3133, doi: 10.1109/CEC.2019.8790158.

[29] T. Liao, D. Molina, and T. Stützle, "Performance evaluation of automatically tuned continuous optimizers on different benchmark sets," *Appl. Soft Comput.*, vol. 27, pp. 490–503, Feb. 2015.

[30] P. Bujok, J. Tvrdík, and R. Poláková, "Comparison of nature-inspired population-based algorithms on continuous optimisation problems," *Swarm Evol. Comput.*, vol. 50, Nov. 2019, Art. no. 100490.

[31] R. Solgi and H. A. Loáiciga, "Bee-inspired metaheuristics for global optimization: A performance comparison," *Artif. Intell. Rev.*, vol. 54, no. 7, pp. 4967–4996, Oct. 2021.

[32] J. Swan, S. Adriaensen, A. E. I. Brownlee, K. Hammond, C. G. Johnson, A. Kheiri, F. Krawiec, J. J. Merelo, L. L. Minku, E. Ozcan, G. L. Pappa, P. Garcia-Sanchez, K. Sorensen, S. Voss, M. Wagner, and D. R. White, "Metaheuristics 'in the large,'" *European J. Oper. Res.*, vol. 29, pp. 393–406, Mar. 2022.

[33] K. Varelas, O. A. E. Hara, D. Brockhoff, N. Hansen, D. M. Nguyen, T. Tušar, and A. Auger, "Benchmarking large-scale continuous optimizers: The bbob-largescale testbed, a COCO software guide and beyond," *Appl. Soft Comput.*, vol. 97, Dec. 2020, Art. no. 106737.

[34] N. H. Awad, M. Z. Ali, P. N. Suganthan, J. J. Liang, and B. Y. Qu, "Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization," Nanyang Technol. Univ., Singapore, Tech. Rep., 2016.

[35] A. P. Piotrowski, M. J. Napiorkowski, J. J. Napiorkowski, and P. M. Rowinski, "Swarm intelligence and evolutionary algorithms: Performance versus speed," *Inf. Sci.*, vol. 384, pp. 34–85, Apr. 2017.

[36] D. H. Wolper and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[37] M. Koppen, D. H. Wolpert, and W. G. Macready, "Remarks on a recent paper on the 'no free lunch' theorems," *IEEE Trans. Evol. Comput.*, vol. 5, no. 3, pp. 295–296, Jun. 2001.

[38] S. Das and P. N. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems," Jadavpur Univ., Nanyang Technol. Univ., Kolkata, India, Tech. Rep., 2010.

[39] J. D. Ser, E. Osaba, D. Molina, X.-S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P. N. Suganthan, C. A. C. Coello, and F. Herrera, "Bio-inspired computation: Where we stand and what's next," *Swarm Evol. Comput.*, vol. 48, pp. 220–250, Aug. 2019.

[40] 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.

[41] A. LaTorre, D. Molina, E. Osaba, J. Poyatos, J. D. Ser, and F. Herrera, "A prescription of methodological guidelines for comparing bio-inspired optimization algorithms," *Swarm Evol. Comput.*, vol. 67, Dec. 2021, Art. no. 100973.

[42] 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.

[43] K. Sörensen, "Metaheuristics—The metaphor exposed," *Int. Trans. Oper. Res.*, vol. 22, no. 1, pp. 3–18, Jan. 2015.

[44] M. A. Lones, "Mitigating metaphors: A comprehensible guide to recent nature-inspired algorithms," *Social Netw. Comput. Sci.*, vol. 1, no. 1, p. 49, Jan. 2020.

[45] A. Tharwat and W. Schenck, "Population initialization techniques for evolutionary algorithms for single-objective constrained optimization problems: Deterministic vs. stochastic techniques," *Swarm Evol. Comput.*, vol. 67, Dec. 2021, Art. no. 100952.

[46] T. Weise, R. Chiong, and K. Tang, "Evolutionary optimization: Pitfalls and booby traps, evolutionary optimization: Pitfalls and booby traps," *J. Comput. Sci. Technol.*, vol. 27, no. 5, pp. 907–936, 2012.

[47] K. M. Malan and A. P. Engelbrecht, "A survey of techniques for characterising fitness landscapes and some possible ways forward," *Inf. Sci.*, vol. 241, pp. 148–163, Aug. 2013.

[48] Z.-H. Zhan, L. Shi, K. C. Tan, and J. Zhang, "A survey on evolutionary computation for complex continuous optimization," *Artif. Intell. Rev.*, vol. 55, no. 1, pp. 59–110, Jan. 2022.

[49] T. Chen, K. Tang, G. Chen, and X. Yao, "A large population size can be unhelpful in evolutionary algorithms," *Theor. Comput. Sci.*, vol. 436, no. 2, pp. 54–70, Jun. 2012.

[50] M. Clerc and J. Kennedy, "The particle swarm—Explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[51] F. van den Bergh and A. P. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Inf. Sci.*, vol. 176, no. 8, pp. 937–971, 2006.

[52] C. W. Cleghorn and A. P. Engelbrecht, "Particle swarm stability: A theoretical extension using the non-stagnate distribution assumption," *Swarm Intell.*, vol. 12, no. 1, pp. 1–22, Mar. 2018.

[53] K. R. Opara and J. Arabas, "Differential evolution: A survey of theoretical analyses," *Swarm Evol. Comput.*, vol. 44, pp. 546–558, Feb. 2019.

[54] A. E. Ezugwu, O. J. Adeleke, A. A. Akinyelu, and S. Viriri, "A conceptual comparison of several Metaheuristic algorithms on continuous optimisation problems," *Neural Comput. Appl.*, vol. 32, no. 10, pp. 6207–6251, May 2020.

[55] A. Kazikova, M. Pluhacek, and R. Senkerik, "How does the number of objective function evaluations impact our understanding of metaheuristics behavior?" *IEEE Access*, vol. 9, pp. 44032–44048, 2021.

[56] K. V. Price, N. H. Awad, M. Z. Ali, and P. N. Suganthan, "The 2019 100-digit challenge on real-parameter, single-objective optimization: Analysis of results," Nanyang Technol. Univ., Singapore, Tech. Rep., 2019. [Online]. Available: http://www.ntu.edu.sg/home/epnsugan

[57] A. P. Engelbrecht, "Particle swarm optimization: Velocity initialization," in *Proc. World Congr. Comput. Intell., Congr. Evol. Comput.*, Jun. 2012, pp. 1–8.

[58] I. Poikolainen, F. Neri, and F. Caraffini, "Cluster-based population initialization for differential evolution frameworks," *Inf. Sci.*, vol. 297, pp. 216–235, Mar. 2015.

[59] Q. Li, S.-Y. Liu, and X.-S. Yang, "Influence of initialization on the performance of metaheuristic optimizers," *Appl. Soft Comput.*, vol. 91, Jun. 2020, Art. no. 106193.

[60] R. Mallipeddi and P. N. Suganthan, "Empirical study on the effect of population size on differential evolution algorithm," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2008, pp. 3663–3670.

[61] A. P. Piotrowski, "Review of differential evolution population size," *Swarm Evol. Comput.*, vol. 32, pp. 1–24, Feb. 2017.

[62] A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, "Population size in particle swarm optimization," *Swarm Evol. Comput.*, vol. 58, Nov. 2020, Art. no. 100718.

[63] C. M. Fernandes, N. Fachada, J. L. J. Laredo, J. J. Merelo, and A. C. Rosa, "Population sizing of cellular evolutionary algorithms," *Swarm Evol. Comput.*, vol. 58, Nov. 2020, Art. no. 100721.

[64] D. Zaharie, "Influence of crossover on the behavior of differential evolution algorithms," *Appl. Soft Comput.*, vol. 9, no. 3, pp. 1126–1138, 2009.

[65] F. Neri and V. Tirronen, "Scale factor local search in differential evolution," *Memetic Comput.*, vol. 1, no. 2, pp. 153–171, Jun. 2009.

[66] J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, and A. Abraham, "Inertia weight strategies in particle swarm optimization," in *Proc. 3rd World Congr. Nature Biologically Inspired Comput.*, New York, NY, USA, 2011, pp. 640–647.

[67] C. Huang, Y. Li, and X. Yao, "A survey of automatic parameter tuning methods for metaheuristics," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 201–216, Apr. 2020.

[68] P. Pošık, W. Huyer, and L. Pál, "A comparison of global search algorithms for continuous black box optimization," *Evol. Comput.*, vol. 20, no. 4, pp. 509–541, Dec. 2012.

[69] Y. D. Sergeyev, D. E. Kvasov, and M. S. Mukhametzhanov, "On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget," *Sci. Rep.*, vol. 8, no. 1, p. 453, Jan. 2018.

[70] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[71] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.

[72] N. Veček, M. Mernik, and M. Črepinšek, "A chess rating system for evolutionary algorithms: A new method for the comparison and ranking of evolutionary algorithms," *Inf. Sci.*, vol. 277, pp. 656–679, Sep. 2014.

[73] J. Carrasco, S. García, M. M. Rueda, S. Das, and F. Herrera, "Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review," *Swarm Evol. Comput.*, vol. 54, May 2020, Art. no. 100665.

[74] N. Veček, M. Črepinšek, and M. Mernik, "On the influence of the number of algorithms, problems, and independent runs in the comparison of evolutionary algorithms," *Appl. Soft Comput.*, vol. 54, pp. 23–45, May 2017.

[75] T. Blackwell and J. Kennedy, "Impact of communication topology in particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 689–702, Aug. 2019.

[76] N. Lynn, M. Z. Ali, and P. N. Suganthan, "Population topologies for particle swarm optimization and differential evolution," *Swarm Evol. Comput.*, vol. 39, no. 4, pp. 24–35, 2018.

[77] Q. Liu, W. Wei, H. Yuan, Z.-H. Zhan, and Y. Li, "Topology selection for particle swarm optimization," *Inf. Sci.*, vol. 363, pp. 154–173, Oct. 2016.

[78] X. Zhou, Y. Wu, M. Zhong, and M. Wang, "Artificial bee colony algorithm based on multiple neighborhood topologies," *Appl. Soft Comput.*, vol. 111, Nov. 2021, Art. no. 107697.

[79] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Ensemble strategies for population-based optimization algorithms—A survey," *Swarm Evol. Comput.*, vol. 44, pp. 695–711, Feb. 2019.

[80] A. Díaz-Manríquez, G. Toscano, and C. A. C. Coello, "Comparison of metamodeling techniques in evolutionary algorithms," *Soft Comput.*, vol. 21, no. 19, pp. 5647–5663, Oct. 2017.

[81] H. Ma, S. Shen, M. Yu, Z. Yang, M. Fei, and H. Zhou, "Multi-population techniques in nature inspired optimization algorithms: A comprehensive survey," *Swarm Evol. Comput.*, vol. 44, pp. 365–387, Feb. 2019.

[82] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, Feb. 2015.

[83] M.-C. Yuen, S.-C. Ng, and M.-F. Leung, "An improved competitive mechanism based particle swarm optimization algorithm for multi-objective optimization," in *Proc. 10th Int. Conf. Inf. Sci. Technol. (ICIST)*, London, U.K., Sep. 2020, pp. 209–218.

[84] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "GA with a new multi-parent crossover for solving IEEE-CEC2011 competition problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, New Orleans, LA, USA, Jun. 2011, pp. 1034–1040.

[85] J. J. Liang, A. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jul. 2006.

[86] J. A. Vrugt, B. A. Robinson, and J. M. Hyman, "Self-adaptive multimethod search for global optimization in real-parameter spaces," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 243–259, Apr. 2009.

[87] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

[88] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 526–553, Jun. 2009.

[89] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Sep. 2009.

[90] A. Caponio, F. Neri, and V. Tirronen, "Super-fit control adaptation in memetic differential evolution frameworks," *Soft Comput.*, vol. 13, nos. 8–9, pp. 811–831, Jul. 2009.

[91] W. Gong, Á. Fialho, Z. Cai, and H. Li, "Adaptive strategy selection in differential evolution for numerical optimization: An empirical study," *Inf. Sci.*, vol. 181, no. 24, pp. 5364–5386, 2011.

[92] Z. Cai, W. Gong, C. X. Ling, and H. Zhang, "A clustering-based differential evolution for global optimization," *Appl. Soft Comput.*, vol. 11, no. 1, pp. 1363–1379, Jan. 2011.

[93] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1679–1696, 2011.

[94] J. Brest and M. S. Maučec, "Self-adaptive differential evolution algorithm using population size reduction and three strategies," *Soft Comput.*, vol. 15, no. 11, pp. 2157–2174, Nov. 2011.

[95] Q.-K. Pan, P. N. Suganthan, L. Wang, L. Gao, and R. Mallipeddi, "A differential evolution algorithm with self-adapting strategy and control parameters," *Comput. Oper. Res.*, vol. 38, no. 1, pp. 394–408, Jan. 2011.

[96] S. Islam, S. Das, S. Ghosh, S. Roy, and P. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Trans. Syst., Man, Cybern., B, Cybern*, vol. 42, no. 2, pp. 482–500, Apr. 2012.

[97] A. P. Piotrowski, J. J. Napiorkowski, and A. Kiczko, "Differential evolution algorithm with separated groups for multi-dimensional optimization problems," *Eur. J. Oper. Res.*, vol. 216, no. 1, pp. 33–46, Jan. 2012.

[98] X. Wang and S. Zhao, "Differential evolution algorithm with self-adaptive population resizing mechanism," *Math. Problems Eng.*, vol. 2013, pp. 1–14, Jan. 2013.

[99] F. Caraffini, F. Neri, G. Iacca, and A. Mol, "Parallel memetic structures," *Inf. Sci.*, vol. 227, no. 4, pp. 60–82, 2013.

[100] A. P. Piotrowski, "Adaptive memetic differential evolution with global and local neighborhood-based mutation operators," *Inf. Sci.*, vol. 241, no. 12, pp. 164–194, Aug. 2013.

[101] W.-N. Chen, J. Zhang, Y. Lin, N. Chen, Z.-H. Zhan, H. S.-H. Chung, Y. Li, and Y.-H. Shi, "Particle swarm optimization with an aging leader and challengers," *IEEE Trans. Evol. Comput.*, vol. 17, no. 2, pp. 241–258, Apr. 2013.

[102] W. Zhu, Y. Tang, J.-A. Fang, and W. Zhang, "Adaptive population tuning scheme for differential evolution," *Inf. Sci.*, vol. 223, pp. 164–191, Feb. 2013.

[103] R. Tanabe and A. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proc. IEEE Congr. Evol. Comput.*, Bejing, China, Jul. 2014, pp. 1658–1665.

[104] Y. Wang, H.-X. Li, T. Huang, and L. Li, "Differential evolution based on covariance matrix learning and bimodal distribution parameter setting," *Appl. Soft. Comput.*, vol. 18, pp. 232–247, May 2014.

[105] D. Simon, M. G. H. Omran, and M. Clerc, "Linearized biogeography-based optimization with re-initialization and local search," *Inf. Sci.*, vol. 267, pp. 140–157, May 2014.

[106] W. Gong, Z. Cai, and Y. Wang, "Repairing the crossover rate in adaptive differential evolution," *Appl. Soft Comput.*, vol. 15, pp. 149–168, Feb. 2014.

[107] S. M. Guo, J. S. H. Tsai, C. C. Yang, and P. H. Hsu, "A self-optimization approach for L-SHADE incorporated with eigenvector-based crossover and successful-parent-selecting framework on CEC 2015 benchmark set," in *Proc. IEEE Congr. Evol. Comput.*, Sendai, Japan, May 2015, pp. 1003–1010.

[108] S.-M. Guo and C.-C. Yang, "Enhancing differential evolution utilizing eigenvector-based crossover operator," *IEEE Trans. Evol. Comput.*, vol. 19, no. 1, pp. 31–49, Feb. 2015.

[109] N. Lynn and P. N. Suganthan, "Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation," *Swarm Evol. Comput.*, vol. 24, pp. 11–24, Oct. 2015.

[110] L. Tang, Y. Dong, and J. Liu, "Differential evolution with an individual-dependent mechanism," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 560–574, Aug. 2015.

[111] M. Yang, C. Li, Z. Cai, and J. Guan, "Differential evolution with auto-enhanced population diversity," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 302–315, Feb. 2015.

[112] Y. L. Li, Z. H. Zhan, Y. J. Gong, W. N. Chen, J. Zhang, and Y. Li, "Differential evolution with an evolution path: A DEEP evolutionary algorithm," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1798–1810, Sep. 2015.

[113] L. Cui, G. Li, Q. Lin, J. Chen, and N. Lu, "Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations," *Comput. Oper. Res.*, vol. 67, pp. 155–173, Mar. 2016.

[114] G. Wu, R. Mallipeddi, P. N. Suganthanc, R. Wang, and H. Chen, "Differential evolution with multi-population based ensemble of mutation strategies," *Inf. Sci.*, vol. 329, pp. 329–345, Jan. 2016.

[115] Q. Qin, S. Cheng, Q. Zhang, L. Li, and Y. Shi, "Particle swarm optimization with interswarm interactive learning strategy," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2238–2251, Oct. 2016.

[116] G. Wu, "Across neighborhood search for numerical optimization," *Inf. Sci.*, vol. 329, pp. 597–618, Feb. 2016.

[117] Y. F. Lou and S. Y. Yuen, "Non-revisiting genetic algorithm with adaptive mutation using constant memory," *Memetic Comput.*, vol. 8, no. 3, pp. 189–210, 2016.

[118] Y. J. Gong, J. J. Li, Y. Zhou, Y. Li, H. S. H. Chung, Y. H. Shi, and J. Zhang, "Genetic learning particle swarm optimization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.

[119] G. Li, Q. Lin, L. Cui, Z. Du, Z. Liang, J. Chen, N. Lu, and Z. Ming, "A novel hybrid differential evolution algorithm with modified CoDE and JADE," *Appl. Soft Comput.*, vol. 47, pp. 577–599, Oct. 2016.

[120] A. W. Mohamed, A. A. Hadi, A. M. Fattouh, and K. M. Jambi, "LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, San Sebastian, Spain, Jun. 2017, pp. 145–152, doi: 10.1109/CEC.2017.7969307.

[121] N. H. Awad, M. Z. Ali, and P. N. Suganthan, "Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 372–379, doi: 10.1109/CEC.2017.7969336.

[122] N. Lynn and P. N. Suganthan, "Ensemble particle swarm optimizer," *Appl. Soft Comput.*, vol. 55, pp. 533–548, Jun. 2017.

[123] W. Du, S. Y. S. Leung, Y. Tang, and A. V. Vasilakos, "Differential evolution with event-triggered impulsive control," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 244–257, Jan. 2017.

[124] G. Khademi, H. Mohammadi, and D. Simon, "Hybrid invasive weed/biogeography-based optimization," *Eng. Appl. Artif. Intell.*, vol. 64, pp. 213–231, Sep. 2017.

[125] V. V. de Melo and W. Banzhaf, "Drone squadron optimization: A novel self-adaptive algorithm for global numerical optimization," *Neural Comput. Appl.*, vol. 30, no. 10, pp. 3117–3144, Nov. 2018.

[126] A. W. Mohamed and P. N. Suganthan, "Real-parameter unconstrained optimization based on enhanced fitness-adaptive differential evolution algorithm with novel mutation," *Soft Comput.*, vol. 22, no. 10, pp. 3215–3235, 2018.

[127] A. P. Piotrowski and J. J. Napiorkowski, "Some metaheuristics should be simplified," *Inf. Sci.*, vol. 427, pp. 32–62, Feb. 2018.

[128] A. P. Piotrowski, "L-SHADE optimization algorithms with population-wide inertia," *Inf. Sci.*, vol. 468, pp. 117–141, Nov. 2018.

[129] G. Zhang and Y. Shi, "Hybrid sampling evolution strategy for solving single objective bound constrained problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Rio de Janeiro, Brazil, Jul. 2018, pp. 1–7, doi: 10.1109/CEC.2018.8477908.

[130] G. Wu, X. Shen, H. Li, H. Chen, A. Lin, and P. N. Suganthan, "Ensemble of differential evolution variants," *Inf. Sci.*, vol. 423, pp. 172–186, Jan. 2018.

[131] J. Zhang, X. Zhu, Y. Wang, and M. Zhou, "Dual-environmental particle swarm optimizer in noisy and noise-free environments," *IEEE Trans. Cybern.*, vol. 49, no. 9, pp. 2011–2021, Jun. 2019.

[132] Z. Meng and J.-S. Pan, "HARD-DE: Hierarchical archive based mutation strategy with depth information of evolution for the enhancement of differential evolution on numerical optimization," *IEEE Access*, vol. 7, pp. 12832–12854, 2019.

[133] R. Poláková, J. Tvrdík, and P. Bujok, "Differential evolution with adaptive mechanism of population size according to current population diversity," *Swarm Evol. Comput.*, vol. 50, Nov. 2019, Art. no. 100519.

[134] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.

[135] Q. B. Diep, I. Zelinka, S. Das, and R. Senkerik, "SOMA T3A for solving the 100-digit challenge," in *Proc. Swarm, Evol. Memetic Comput. Conf.*, Maribor, Slovenia, 2019, pp. 155–165.

[136] X. Xia, L. Gui, F. Yu, H. Wu, B. Wei, Y.-L. Zhang, and Z.-H. Zhan, "Triple archives particle swarm optimization," *IEEE Trans. Cybern.*, vol. 50, no. 12, pp. 4862–4875, Dec. 2020.

[137] X. Sun, L. Jiang, Y. Shen, H. Kang, and Q. Chen, "Success history-based adaptive differential evolution using turning-based mutation," *Mathematics*, vol. 8, no. 9, p. 1565, Sep. 2020.

[138] 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)*, Glasgow, U.K., Jul. 2020, pp. 1–8, doi: 10.1109/CEC48606.2020.9185577.

[139] 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)*, Glasgow, U.K., Jul. 2020, pp. 1–8, doi: 10.1109/CEC48606.2020.9185901.

[140] J.-S. Pan, N. Liu, and S.-C. Chu, "A hybrid differential evolution algorithm and its application in unmanned combat aerial vehicle path planning," *IEEE Access*, vol. 8, pp. 17691–17712, 2020.

[141] Z. Meng, C. Yang, X. Li, and Y. Chen, "Di-DE: Depth information-based differential evolution with adaptive parameter control for numerical optimization," *IEEE Access*, vol. 8, pp. 40809–40827, 2020.

[142] A. W. Mohamed, A. A. Hadi, P. Agrawal, K. M. Sallam, and A. K. Mohamed, "Gaining-sharing knowledge based algorithm with adaptive parameters hybrid with IMODE algorithm for solving CEC 2021 benchmark problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Kraków, Poland, Jun. 2021, pp. 841–848, doi: 10.1109/CEC45853.2021.9504814.

[143] Z. Meng, Y. Zhong, and C. Yang, "CS-DE: Cooperative strategy based differential evolution with population diversity enhancement," *Inf. Sci.*, vol. 577, pp. 663–696, Oct. 2021.

[144] A. A. Hadi, A. W. Mohamed, and K. M. Jambi, "Single-objective real-parameter optimization: Enhanced LSHADE-SPACMA algorithm," in *Heuristics for Optimization and Learning* (Studies in Computational Intelligence), vol. 906. Cham, Switzerland: Springer, 2021, pp. 103–121.

[145] S. Aras, E. Gedikli, and H. T. Kahraman, "A novel stochastic fractal search algorithm with fitness-distance balance for global numerical optimization," *Swarm Evol. Comput.*, vol. 61, Mar. 2021, Art. no. 100821.

[146] Z. Meng and C. Yang, "Hip-DE: Historical population based mutation strategy in differential evolution with parameter adaptive mechanism," *Inf. Sci.*, vol. 562, pp. 44–77, Jul. 2021.

[147] 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)*, Kraków, Poland, Jun. 2021, pp. 832–840, doi: 10.1109/CEC45853.2021.9504792.

[148] A. H. Halim, I. Ismail, and S. Das, "Performance assessment of the metaheuristic optimization algorithms: An exhaustive review," *Artif. Intell. Rev.*, vol. 54, pp. 2323–2409, Oct. 2021.

[149] Z. Ma, G. Wu, P. N. Suganthan, A. Song, and Q. Luo, "Performance assessment and exhaustive listing of 500+ nature-inspired metaheuristic algorithms," *Swarm Evol. Comput.*, vol. 77, Mar. 2023, Art. no. 101248, doi: 10.1016/j.swevo.2023.101248.

[150] S. K. Goudos, G. V. Tsoulos, G. Athanasiadou, M. C. Batistatos, D. Zarbouti, and K. E. Psannis, "Artificial neural network optimal modeling and optimization of UAV measurements for mobile communications using the L-SHADE algorithm," *IEEE Trans. Antennas Propag.*, vol. 67, no. 6, pp. 4022–4031, Jun. 2019.

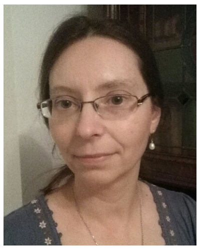
[151] P. P. Biswas, P. N. Suganthan, and G. A. J. Amaratunga, "Minimizing harmonic distortion in power system with optimal design of hybrid active power filter using differential evolution," *Appl. Soft Comput.*, vol. 61, pp. 486–496, Dec. 2017.

[152] P. P. Biswas, P. N. Suganthan, G. Wu, and G. A. J. Amaratunga, "Parameter estimation of solar cells using datasheet information with the application of an adaptive differential evolution algorithm," *Renew. Energy*, vol. 132, pp. 425–438, Mar. 2019.

[153] Y. Nasir, W. Yu, and K. Sepehrnoori, "Hybrid derivative-free technique and effective machine learning surrogate for nonlinear constrained well placement and production optimization," *J. Petroleum Sci. Eng.*, vol. 186, Mar. 2020, Art. no. 106726.

ADAM P. PIOTROWSKI received the Ph.D. and Habilitation degrees from the Institute of Geophysics, Polish Academy of Sciences, in 2005 and 2014, respectively. He is currently working on evolutionary computation, neural networks, thermal aspects of natural water bodies, and rainfall-runoff modeling. He is the first author of more than 40 journal articles. He is an Associate Editor of *Swarm and Evolutionary Computation* journal.

JAROSLAW J. NAPIORKOWSKI received the M.Sc. degree in electronics from the Department of Electronics, Warsaw University of Technology, Warsaw, Poland, and the Ph.D. degree in earth sciences from the Institute of Geophysics, Polish Academy of Sciences, Warsaw. He was a Postdoctoral Scholar at University College Dublin and a Humboldt Fellow at Ruhr University Bochum. He was a recipient of the Tison Award.



AGNIESZKA E. PIOTROWSKA received the Ph.D. degree from the University of Warsaw, Warsaw, Poland, in 2008. She is currently working with the University of Warsaw. Her main research interests include historical linguistics, bio-inspired optimization methods and links between biological senses, brain functions, human cognition, and language.