

Received January 21, 2021, accepted January 31, 2021, date of publication February 9, 2021, date of current version April 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3058285

On Selection of a Benchmark by Determining the Algorithms' Qualities

IZTOK FISTER^{1,2}, (Member, IEEE), JANEZ BREST¹, (Senior Member, IEEE),
ANDRES IGLESIAS^{2,4}, AKEMI GALVEZ^{2,4}, SUASH DEB^{3,5}, (Senior Member, IEEE),
AND IZTOK FISTER, JR.¹, (Member, IEEE)

¹Faculty of Electrical Engineering and Computer Science, University of Maribor, 2000 Maribor, Slovenia

²Department of Applied Mathematics and Computational Sciences, University of Cantabria, 39005 Santander, Spain

³IT & Educational Consultant, Ranchi 834010, India

⁴Faculty of Sciences, Toho University, Funabashi 274-8510, Japan

⁵Distinguished Professorial Associate, Decision Sciences & Modeling Program, Victoria University, Melbourne, Victoria 8001, Australia

Corresponding author: Iztok Fister (iztok.fister@um.si)

This work was supported in part by the Slovenian Research Agency (Projects J2-1731 and L7-9421) under Grant P2-0041, in part by the Project PDE-GIR of the European Union's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska-Curie under Grant 778035, and in part by the Spanish Ministry of Science, Innovation and Universities (Computer Science National Program) of the Agencia Estatal de Investigación and European Funds EFRD (AEI/FEDER, UE) under Grant TIN2017-89275-R.

ABSTRACT The authors got the motivation for writing the article based on an issue, with which developers of the newly developed nature-inspired algorithms are usually confronted today: How to select the test benchmark such that it highlights the quality of the developed algorithm most fairly? In line with this, the CEC Competitions on Real-Parameter Single-Objective Optimization benchmarks that were issued several times in the last decade, serve as a testbed for evaluating the collection of nature-inspired algorithms selected in our study. Indeed, this article addresses two research questions: (1) How the selected benchmark affects the ranking of the particular algorithm, and (2) If it is possible to find the best algorithm capable of outperforming all the others on all the selected benchmarks. Ten outstanding algorithms (also winners of particular competitions) from different periods in the last decade were collected and applied to benchmarks issued during the same time period. A comparative analysis showed that there is a strong correlation between the rankings of the algorithms and the benchmarks used, although some deviations arose in ranking the best algorithms. The possible reasons for these deviations were exposed and commented on.

INDEX TERMS Evolutionary algorithms, benchmark functions, differential evolution.

I. INTRODUCTION

The purpose of the article is searching for an answer to the question how selection of the benchmark suite affects determining the quality of the newly developed nature-inspired algorithms. Primarily, the motivation for this study was caused by the comments of reviewers in journals in which researchers wish to present their work, that usually demand testing the quality of these algorithms on benchmarks of the newest version. The question is how false are the results obtained by violating the demand?

Nature-inspired algorithms have become standard for solving the hardest optimization problems, with which humans are confronted in the real-world. These algorithms operate with a population of solutions according to the principles

of operation of some animals, flowers, and even human society. Observing the principles found in nature, like tracing the ant trails, watching termites build their nests (mounds), inspecting wolves and their hunting habits in deep forests, investigating the flying traces of birds, and even admiring the small lightning bugs, called fireflies, in the young summer nights, have led to raising the development of nature-inspired algorithms. All these inspirations can be treated as optimization processes, while the mathematical formulation for their description presents a basis for building the optimization algorithms.

In general, there are two main families of nature-inspired algorithms: Evolutionary Algorithms (EA) [1], and Swarm Intelligence (SI) based algorithms [2]. Recently, we have witnessed a noticeable increase of newly developed nature-inspired algorithms, especially in the SI domain, where authors usually hide the true novelty of the proposed

The associate editor coordinating the review of this manuscript and approving it for publication was Bilal Alatas¹.

algorithm behind a metaphor of some natural or man-made process on which principle it works [3]. The only thing that really counts is their performance. Obviously, the newly developed nature-inspired algorithms can be tested on many problem testbeds, such as, for example [4].

Unfortunately, the majority of proprietary testbeds are devoted for solving continuous optimization problems and, in general, do not present any problem for modern stochastic nature-inspired algorithms. Indeed, there are two conferences that promote the development of new algorithms by issuing benchmark testbed problems and organizing the competitions, in which they can even compete against each other: The Genetic and Evolutionary Computation Conference (GECCO), and The Conference of Evolutionary Computation (CEC). The former organizes The Black Box Optimization Competition (BBComp) [5], which tries to hide the problem (thus the name black box) to be optimized from the experimenter, while in the latter case, the original problems are known in advance, but, for the purpose of the competition, the functions are modified using shifting, rotation, hybridization, and composition of more functions, in order to mask the original optimum and, thus, make searching for this much harder. Interestingly, both mentioned problem testbeds are changed from year to year to prevent experimenters from tuning their algorithms in a specific way, and, thus, outperforming the other algorithms. As speculated in [6], both types of problem testbeds introduced at both mentioned conferences are highly complex, and, therefore, the potential possibility of cheating is minimal.

Although several papers exist analyzing the behavior of different nature-inspired algorithms applied to different CEC benchmarks [7], [8], these observe the subject more from the algorithm's point of view, i.e., how good algorithms solve the benchmarks. To best of our knowledge, there are no papers that choose an opposite point of view, i.e., how the benchmarks influence the results of the algorithm in question. The pioneer work in this direction was made by Fister *et al.* [6], where the authors concluded that the contemporary benchmarks are hard enough for determining the true value of the algorithms. This means that the results obtained on the newest benchmark would not be very different than those observed on the older benchmarks.

The present article is an extension of the mentioned article. Actually, it goes a step forward by introducing a metric for comparing the rankings of algorithms achieved in different competitions, and by extending the comparison of the outstanding nature-inspired algorithms in solving the different function benchmark suites that have been issued in CEC Competitions on Single Objective Real-Parameter Numerical Optimization [?], [9], [10] during the last decade (i.e., from 2010 to 2020). The metric base on the Spearman rank correlation measuring the significance of the connection between two non-normally distributed variables. Ten algorithms were collected into this comparative study. These belong to both families of stochastic nature-inspired population-based algorithms. The members of the EAs family

base on the DE algorithm widened from its original version throughout the self-adaptive versions jDE [11] and SaDE [12] to the SHADE [13] and its improved variants LSHADE [14], iL-SHADE [15], jSO [16] and LSHADE_RSP [17]. The members of the SI-based family include the original Artificial Bee Colony (ABC) algorithm [18] and its self-adaptive variant SSEABC [19]. All algorithms were applied to the CEC'13 [9], CEC'14/16 [9], and CEC'17/18 [10] benchmark function suites.

Two research questions are set in this study: (1) Does using different CEC benchmark function suites influence the ranking of the specific algorithms, and (2) Does there exist a best algorithm, capable of outperforming the results of all the other algorithms on all the selected benchmarks. Thus, we were focused on the winners of the particular competitions. Unfortunately, among these less SI-based algorithms can be found.

The structure of the remainder the article is as follows: The descriptions of the algorithms in the comparative analysis are discussed in Section II. The characteristics of the observed benchmark function suites are described in detail in Section III. Section IV is devoted to illustrating the experiments and results. The article is concluded with Section V, where the performed work is examined and directions are outlined for the future work.

II. ALGORITHMS IN THE COMPARATIVE STUDY

This section is devoted to describing the stochastic population-based nature-inspired algorithms used in our comparative study. There are ten algorithms that belong to the EA and SI-based algorithm families. The following EAs were taken into consideration in the study:

- Original DE (Storn and Price [20]),
- Self-adaptive DE (jDE) (Brest *et al.* [11]),
- Self-adaptive DE (SaDE) (Qin *et al.* [12], [21]),
- Success-History based Adaptive DE (SHADE) (Tanabe and Fukunaga [13]).

Obviously, all the mentioned algorithms base on the original DE. Recently, the SHADE algorithm attracted the DE community extraordinarily due to its efficiency and implementation simplicity. Consequently, more variants of this algorithms were proposed later, from which the following were applied in our study:

- SHADE with Linear Population Size Reduction (LSHADE) (Tanabe and Fukunaga [14]),
- Improved LSHADE (iL-SHADE) (Brest *et al.* [15]),
- Single objective real-parameter optimization algorithm (jSO) (Brest *et al.* [16]),
- LSHADE algorithm with a Rank-based Selective Pressure strategy (LSHADE-RSP) (Stanovov *et al.* [17]).

On the other hand, SI-based algorithms were usually entered in the CEC Competitions on Real-Parameter Single-Objective Optimization. In our study, we selected the following members of this family:

- Artificial Bee Colony (ABC) (Karaboga and Basturk [18]),

TABLE 1. Collection of Algorithms in the Study.

Org.	Variant	Branch	Features	Year	Ref.
DE	Original		Fixed F , CR , NP	1995	[21]
	jDE		Self-adaptive F , CR	2006	[12]
	SaDE		Self-adaptive F , CR	2005	[13]
	SHADE	Original	2 mutation strategies	2013	[14]
		LSHADE	LSPR, 'DE/pBest/1/bin'	2014	[15]
		iL-SHADE ¹	init/update of $M_{CR}^{(t)}$, calc. of p_i	2016	[16]
		jSO ²	'DE/pBest_w/1/bin'	2017	[17]
		LSHADE_RSP ³	Rank selection	2018	[18]
ABC	Original		Crossover probability $1/D$	2007	[19]
	SSEABC		Adaptation of mutation strategy	2016	[20]

- Self-adaptive Search Equation-based Artificial Bee Colony (SSEABC) (Yavuz *et al.* [19]).

The main characteristic of the ABC is that it is one of the few SI-based algorithms using the crossover operator. Interestingly, this operator is used with probability of $1/D$, where D denotes the dimension of the problem. Such value enables slow convergence on the one hand, and avoids getting stuck into the local optima on the other.

Ten different stochastic nature-inspired population based algorithms were included in this study, where the collection of these is illustrated in Table 1. As can be seen from this table, all the observed algorithms have their origins in two nature-inspired algorithms, as follows: (1) DE and (2) ABC. The former belongs to the EAs, while the latter to the SI-based family. Among the variants of DE algorithms, where the original version with two self-adaptive variants jDE and SaDE are taken into consideration, the following improved branches of the SHADE variant are observed: LSHADE, iL-SHADE, jSO, and LSHADE_RSP. The ABC was one of the more efficient members of the SI-based algorithms, where its hybrid version SSEABC was recognized as one of the best SI-based algorithms in the 2016 CEC Competition on Real-Parameter Single-Objective Optimization.

The motivation behind collecting these algorithms was threefold: (1) To show the performance level that the original algorithm can achieve by solving the contemporary CEC benchmark suites, (2) To follow the evolution of the more successful branches appearing in CEC competitions that base on the SHADE algorithm, and (3) To compare the performance of two of the more powerful algorithm's families of nature-inspired algorithms.

The mentioned algorithms are discussed in detail in the remainder of the section.

A. DIFFERENTIAL EVOLUTION

DE was developed by Storn and Price in 1995 [20] and attracted the attention of the evolutionary community quickly.

¹Rank - #3 on CEC'16 Competition on real-parameter single-objective optimization

²Rank - #2 on CEC'17 Competition on real-parameter single-objective optimization

³Rank - #2 on CEC'18 Competition on real-parameter single-objective optimization

It has proven to be an appropriate tool for solving discrete as well as continuous optimization problems. Consequently, many DE variants have emerged for solving different, real-world problems. Indeed, DE belongs to a class of EA using real-valued representation of solutions. In each generation, these solutions undergo acting the operations of operators, like mutation, crossover, and selection.

In place of natural evolution, variation operators in DE (i.e., mutation and crossover) operate on vector differences that modify two randomly selected vectors according to the so-called mutation strategy. The basic mutation strategy, for instance, selects two solutions randomly, while their scaled difference is added to the third solution, in other words:

$$\mathbf{u}_i^{(t)} = \mathbf{x}_{r0}^{(t)} + F \cdot (\mathbf{x}_{r1}^{(t)} - \mathbf{x}_{r2}^{(t)}), \quad \text{for } i = 1, \dots, NP, \quad (1)$$

where $F \in [0.0, 1.0]$ is the scaling factor regulating the rate of modification, NP is the population size and $r0, r1, r2$ are randomly selected values in the interval $1, \dots, NP$. Let us notice that, although Price and Storn proposed the slightly different interval, i.e., $F \in [0.0, 2.0]$, the interval of values for parameter $F \in [0.1, 1.0]$ was enforced in the DE community during the time.

The crossover parameter regulates how many parameter values are copied to the trial vector either from the vector obtained after mutation (Eq. (1)), or from the parent vector. However, the crossover can be performed binomially (denoted as 'bin') or exponentially (denoted as 'exp'). Actually, the binomial crossover operation can be expressed mathematically as:

$$w_{i,j}^{(t)} = \begin{cases} u_{i,j}^{(t)}, & \text{rand}_j(0, 1) \leq CR \vee j = j_{rand}, \\ x_{i,j}^{(t)}, & \text{otherwise,} \end{cases} \quad (2)$$

where $CR \in [0.0, 1.0]$ controls the portion of parameters copied to the trial solution from the mutant vector. In addition, the condition $j = j_{rand}$ ensures that almost one element in the trial solution differs from the original solution $\mathbf{x}_i^{(t)}$.

There are many mutation strategies in DE. Therefore, a special notation has been introduced to describe their operation. In the case of base mutation strategy 'rand/1/bin', for example, the base vector is selected randomly, then, one vector difference is added to it, and, finally, the number of

modified parameters in the trial/offspring vector follows a binomial distribution.

The selection is expressed mathematically as follows:

$$\mathbf{x}_i^{(t+1)} = \begin{cases} \mathbf{w}_i^{(t)}, & \text{if } f(\mathbf{w}_i^{(t)}) \leq f(\mathbf{x}_i^{(t)}), \\ \mathbf{x}_i^{(t)}, & \text{otherwise.} \end{cases} \quad (3)$$

The DE selection is known under the name 'one-to-one', because the trial and the destination vector compete for surviving in the next generation, where the better between these according to the fitness function has more chance to survive into the next generation.

B. jDE ALGORITHM

In 2006, Brest *et al.* [11] proposed an effective DE variant, called jDE, where the DE control parameters F and CR are self-adapted during the run. This means that these parameters are added to the representation of individuals and undergo acting as the variation operators. Consequently, a representation of the individual in jDE is modified as follows:

$$\mathbf{x}_i^{(t)} = (x_{i,1}^{(t)}, x_{i,2}^{(t)}, \dots, x_{i,D}^{(t)}, F_i^{(t)}, CR_i^{(t)}).$$

Additionally, the jDE parameters F and CR are modified according to the following equations:

$$F_i^{(t+1)} = \begin{cases} F_l + \text{rand}_1 * (F_u - F_l), & \text{if } \text{rand}_2 < \tau_1, \\ F_i^{(t)}, & \text{otherwise,} \end{cases} \quad (4)$$

$$CR_i^{(t+1)} = \begin{cases} \text{rand}_3, & \text{if } \text{rand}_4 < \tau_2, \\ CR_i^{(t)}, & \text{otherwise,} \end{cases} \quad (5)$$

where: $\text{rand}_{i=1,\dots,4} \in [0, 1]$ are randomly generated values drawn from uniform distribution in the interval $[0, 1]$, τ_1 and τ_2 are learning rates, F_l and F_u are the lower and upper bounds of feasible values for parameter F , respectively.

C. SELF-ADAPTIVE DE

There are many mutation strategies in DE, where their application depends on the problem to be solved. Typically, it is not known in advance which DE mutation strategy performs well on a particular problem. In line with this, the proposed SaDE [12] selects between two different strategies according to the feedback from the search process. More precisely, the algorithm decides which of two strategies to select according to probabilities p_1 and $p_2 = 1 - p_1$ that were initialized to the equal value 0.5. Then, the proper DE mutation strategy is selected according to the equation:

$$\mathbf{u}_i^{(t)} = \begin{cases} \mathbf{x}_{r_0}^{(t)} + F \cdot (\mathbf{x}_{r_1}^{(t)} - \mathbf{x}_{r_2}^{(t)}), & \text{if } \mathcal{U}(0, 1) < p_1, \\ \mathbf{x}_i^{(t)} + F \cdot (\mathbf{x}_{\text{best}}^{(t)} - \mathbf{x}_i^{(t)}) \\ \quad + F \cdot (\mathbf{x}_{r_0}^{(t)} - \mathbf{x}_{r_1}^{(t)}), & \text{otherwise,} \end{cases} \quad (6)$$

where probabilities p_1 and p_2 are calculated using the expressions:

$$p_1 = \frac{ns_1 \cdot (ns_2 + nf_2)}{ns_2 \cdot (ns_1 + nf_1) + ns_1 \cdot (ns_2 + nf_2)}, \quad (7)$$

$$p_2 = 1 - p_1,$$

and ns_1 and nf_1 denote the number of successful and unsuccessful changes of trial solution using the first DE mutation

strategy, respectively, and ns_2 and nf_2 designate the number of successful and unsuccessful changes of the trial solution using the second DE mutation strategy, respectively.

Additionally, the behavior of the DE algorithm is controlled using three critical parameters: scale factor F , crossover rate CR and population size NP . While these parameters remain fixed by the original DE, two of the three parameters are self-adapted in SaDE. Thus, the F_i parameter is changed for each i -th individual in the interval $(0, 2]$ according to the following rule:

$$F_i = 2 \cdot \mathcal{N}(0.5, 0.3), \quad \text{for } i = 1, \dots, NP, \quad (8)$$

where $\mathcal{N}(0.5, 0.3)$ denotes a random number drawn from the Normal distribution with mean 0.5 and standard deviation 0.3. Modifying the crossover rate is more sophisticated in this algorithm, since the mean value of this parameter CR_m actually determines the characteristic of the Normal distribution, from which the new value of $CR_i = \mathcal{N}(CR_m, 0.1)$ is drawn. Let us mention that the value of variable CR_m is updated after each 25 generations, where the five last used values of CR_i are averaged, and each value of the same variable is effective for five generations.

D. SUCCESS-HISTORY BASED ADAPTIVE DE

For adapting the DE control parameters, the SHADE algorithm memorizes a historical memory M_{CR} and M_F with H entries for CR and F , respectively. At the beginning, the content of M_{CR_i} and M_{F_i} for $i = 1, \dots, H$ is initialized to 0.5. Then, the mentioned control parameters are adapted according to the following equation:

$$CR_i = \mathcal{N}(M_{CR,r_i}, 0.1), \quad F_i = \mathcal{C}(M_{F,r_i}, 0.1), \quad (9)$$

where $\mathcal{N}(\mu, \sigma)$ describes the randomly selected value drawn from Normal distribution with mean μ and standard deviation σ . $\mathcal{C}(\mu, \sigma)$ is the randomly selected value drawn from Cauchy distribution with mean μ and standard deviation σ , while r_i is the randomly selected value drawn from the uniform distribution in the interval $[1, H]$.

In order to update the historical memory, the number of successfully changed individuals is recorded in the S_{CR} and S_F success history, while the contents of the memories are modified as follows:

$$M_{CR,k}^{(t+1)} = \begin{cases} \text{mean}_{WA}(S_{CR}), & \text{if } S_{CR} \neq 0, \\ M_{CR,k}^{(t)}, & \text{otherwise,} \end{cases} \quad (10)$$

$$M_{F,k}^{(t+1)} = \begin{cases} \text{mean}_{WL}(S_F), & \text{if } S_F \neq 0, \\ M_{F,k}^{(t)}, & \text{otherwise,} \end{cases} \quad (11)$$

while k determines the position of the memory update. At the beginning, the position is set to $k = 1$. Then, k is incremented, until $k \leq H$, and the position $k = 1$ is set, when $k > H$. The function $\text{mean}_{WA}(S_{CR})$ in Eq. (10) denotes the weighted arithmetic mean that is calculated as follows:

$$\text{mean}_{WA}(S_{CR}) = \sum_{k=1}^{|S_{CR}|} w_k \cdot S_{CR,k}, \quad (12)$$

$$w_k = \frac{\Delta f_k}{\sum_{k=1}^{|S_{CR}|} \Delta f_k}, \quad (13)$$

where $\Delta f_k = |f(\mathbf{u}^{(t)}) - f(\mathbf{x}^{(t)})|$. The weighted Lehmer mean $mean_{WL}(S_F)$ in Eq. (11) is expressed as follows:

$$mean_{WL}(S_F) = \frac{\sum_{k=1}^{|S_F|} w_k \cdot S_{F,k}^2}{\sum_{k=1}^{|S_F|} w_k \cdot S_{F,k}}. \quad (14)$$

The SHADE algorithm applied the 'current-to-pbest/1/bin' DE mutation strategy that is expressed as follows:

$$v_i^{(t)} = x_i^{(t)} + F_i \cdot (x_{pbest}^{(t)} - x_i^{(t)}) + F_i \cdot (x_{r_0}^{(t)} - x_{r_1}^{(t)}), \quad (15)$$

where $F_i^{(t)}$ denotes the scaling factor corresponding to the i -th vector, and $x_{pbest}^{(t)}$ is a randomly selected value drawn from the top $NP \times p_i$ members in generation t . Thus, p_i is calculated as follows:

$$p_i = rand[p_{min}, 0.2], \quad (16)$$

where p_{min} is set such that the pbest individual can be selected between two vectors, i.e., $p_{min} = 2/NP$.

E. SHADE WITH LINEAR POPULATION SIZE REDUCTION

To enhance the performance of the SHADE algorithm further, the LSHADE introduced the Linear Population Size Reduction (LPSR) that affects the third DE control parameter, i.e., population size NP . This feature reduces the population size according to a decreasing linear function with maturing of the evolutionary search process. Although various adaptive population size methods were proposed in the past (e.g., GAVaPS by Arabas *et al.* [22]) that are capable of decreasing and increasing population size according to the feedback from the evolutionary search process, it seems that the simple LPSR is the most effective in improving EA performances by solving the continuous optimization problems [23].

The LPSR was implemented within the LSHADE algorithm according to the following equation:

$$NP^{(t+1)} = \left\lceil \left(\frac{NP_{min} - NP_{init}}{MAX_NFE} \cdot NFE + NP_{min} \right) \right\rceil, \quad (17)$$

where NP_{min} is set to the smallest feasible value that enables it to perform the variation operation (e.g., $NP_{min} = 4$), NFE is the current number of fitness function evaluations, NP_{init} is the initial population size and MAX_NFE is the maximum number of fitness function evaluations.

F. IMPROVED LSHADE

An extended version of LSHADE, called iL-SHADE, was proposed for the CEC-16 Competition on real-parameter single objective optimization. The latter incorporates four new features into the original LSHADE algorithms, as follows:

- Initialization of the history memory: The initial value of $M_{CR} = 0.8$ was employed for each of the H elements in place of $M_{CR} = 0.5$ as set in the original algorithm. Additionally, at least one element of the history memory was set to $M_{CR,k} = M_{F,k} = 0.8$.

- Update of the history memory $M_{CR}^{(t)}$: The Lehner mean function $mean_{WL}(S_{CR})$ was applied in place of the weighted arithmetic mean function $mean_{WA}(S_{CR})$.
- Adaptation of values $CR_i^{(t)}$ and $F_i^{(t)}$ according to an evolutionary search process maturity: The higher values of those parameters are not allowed at the beginning, while the lower ones not at the end of the optimization process.
- Calculation of the p_i value for the DE mutation strategy 'current-to-pbest/1/bin': The authors proposed the following equation for this calculation:

$$p_i = \frac{p_{max} - p_{min}}{MAX_NFE} \cdot NFE + p_{min}, \quad (18)$$

where p_{min} and p_{max} denote the predefined minimum and maximum constants, respectively.

Obviously, all the other features of the LSHADE, like LPSR, remained unchanged in this algorithm.

G. SINGLE OBJECTIVE REAL-PARAMETER OPTIMIZATION ALGORITHM

An extended version of the iL-SHADE, called jSO, was proposed for the CEC'17 competition on real-parameter single-objective optimization that introduced only one main improvement to the original algorithm, as follows:

- Application of the new mutation strategy 'current-to-pbest-w/1/bin' that is expressed as:

$$v_i^{(t)} = x_i^{(t)} + F_w \cdot (x_{pbest}^{(t)} - x_i^{(t)}) + F \cdot (x_{r_0}^{(t)} - x_{r_1}^{(t)}), \quad (19)$$

where

$$F_w = \begin{cases} 0.7 \cdot F, & \text{if } NFE < 0.2 \cdot MAX_NFE, \\ 0.8 \cdot F, & \text{if } NFE < 0.4 \cdot MAX_NFE, \\ 1.2 \cdot F, & \text{otherwise.} \end{cases} \quad (20)$$

The motivation behind Eq. (20) was to allow the evolutionary search process to make higher changes of an individual's position into the search space at the end of the optimization, when the diversity of the population starts to disappear gradually.

Although the change to the iL-SHADE is minor, the jSO had a crucial impact on improving the results of the CEC'17 competition.

H. LSHADE ALGORITHM WITH A RANK-BASED SELECTIVE PRESSURE STRATEGY

The extended version of LSHADE, called LSHADE-RSP, is another variant of the same base algorithm that was developed for the CEC'18 competition on real-parameter single-objective optimization. It proposes the so called rank-based mutation scheme 'current-to-pbest/r/bin' for avoiding the premature convergence and losing the population diversity. This strategy replaces the original 'current-to-pbest/1/bin' and is defined similarly as in the jSO algorithm, i.e.,:

$$v_i^{(t)} = x_i^{(t)} + F_w \cdot (x_{pbest}^{(t)} - x_i^{(t)}) + F \cdot (x_{pr_0}^{(t)} - x_{pr_1}^{(t)}), \quad (21)$$

except that, in place of randomly selected vectors in the second term of Eq. (21), these are selected using the rank selection typically used in Genetic Algorithms (GA) [24].

The rank-based selection in GAs operates as follows: At first, solutions are sorted according to their fitness values. This means that the individuals with the smaller fitness are ordered before those with the larger ones. Then, the rank values are assigned to each individuals according to the following expression:

$$rank_i = k \cdot (NP - i) + 1, \quad (22)$$

where variable k is a scaling factor set such that the largest rank means the better solution according to the fitness function. Based on the rank values, the probability of selection is calculated according to the following equation:

$$pr_i = \frac{rank(i)}{\sum_{i=1}^{NP} rank_i}. \quad (23)$$

Finally, the proportionate selection operator is applied with the ranked fitness values, and two parent solutions are selected to enter into the mutation strategy.

I. ARTIFICIAL BEE COLONY ALGORITHM

The ABC algorithm was introduced in 2005, and it is inspired by the natural behavior of bees. In the ABC algorithm, the artificial bee colony consists of three bee groups, as follows: employed bees, onlookers, and scouts. The employed bees discover food sources individually, in other words, only one employed bee exists for each food source. The employed bees share information about food sources with the onlooker bees in their hive. Then, the onlooker bees can choose which food sources to forage. The richer the food source with nectar, the more probability for visiting by the onlooker bees. Finally, employed bees whose food sources are exhausted by either employed or onlooker bees, become scouts.

Formally, the ABC algorithm supports three types of bees appearing sequentially within each search cycle:

- employed,
- onlooker,
- scout.

The purpose of the employed bees is to send them onto the food sources and to evaluate their nectar amounts. The onlooker bees are devoted to sharing the information about food sources with the employed bees, selecting the proper food source and evaluating their nectar amounts. Finally, the scouts are determined that are necessary for discovering the new food sources.

Before the ABC search process can take place, initialization is performed, while a termination condition is responsible for termination of the search cycle. Typically, the maximum number of function evaluations MAX_NFE is used as the termination condition.

To generate a candidate food position from the old one, the ABC uses the following expression:

$$\mathbf{v}_i^{(t)} = \mathbf{x}_i^{(t)} + \phi_{i,j}^{(t)} \cdot (\mathbf{x}_i^{(t)} - \mathbf{x}_k^{(t)}), \quad (24)$$

where $\phi_{i,j}^{(t)}$ is a scaling factor drawn randomly from the uniform distribution in the interval $[-1.0, 1.0]$, $\mathbf{x}_k^{(t)}$ is a randomly selected vector in the interval $\{1, \dots, NP\}$ and the relation $i \neq k$ holds. The onlooker bee chooses a food source according to its associated probability calculated as follows:

$$p_i = \frac{f(\mathbf{x}_i)}{\sum_{j=1}^{NP} f(\mathbf{x}_j)}, \quad (25)$$

where $f(\mathbf{x}_i)$ denotes the value of the fitness function.

Interestingly, Eq. (24) is used for each of the employed bees and those onlookers with probability higher than the probability assigned to the particular food source. Only one element of solution is changed in each modification operation. On the other hand, all individuals that do not change their fitness function values for predetermined limit cycles, become scouts, i.e., they are restarted.

J. SELF-ADAPTIVE SEARCH EQUATION-BASED ARTIFICIAL BEE COLONY

The SSEABC algorithm was developed for the CEC'16 competition on real-parameter single-objective optimization and introduced three strategies for exploring the problem search space: (1) Self-adaptive search equation determination, (2) A competitive local search selection, and (3) An incremental population size. Interestingly, the mutation strategy in this algorithm was constructed randomly according to the following equation:

$$x_{i,j}^{(t)} = term_1 + term_2 + term_3 + term_4, \text{ for } j = 1, \dots, m, \quad (26)$$

where m determines the number of changed elements in the vector $\mathbf{x}_i^{(t)}$ in the interval $[1, D]$, $term_1$ denotes the base vector, on which changes are performed (i.e., current, best or randomly selected), while $term_2$, $term_3$, and $term_4$ denote different combinations of the second term in Eq. (24) (e.g., $(\mathbf{x}_{best} - \mathbf{x}_{r1})$, $(\mathbf{x}_0 - \mathbf{x}_{r1})$, etc.), where each term $term_k$ is multiplied with different scale factors ϕ_k for $k = 1, \dots, 3$, respectively. Each component of the mentioned expression, i.e., (1) The number of successful updates are about to be maintained, and (2) Those components with the lowest success rates are to be eliminated from the set of available components for constructing the mutation strategy.

The SSEABC uses two local search strategies that compete between each other during the fixed budget of fitness function evaluations. In a competition phase, the best local search algorithm is determined that is then applied later in the deployment phase. If the best solution is not improved in this phase, the competition is turned back.

The incremental population size is based on Incremental Social Learning (ISL) [23]. This strategy adds a new solution to the population after a predefined number of generations. Thus, the new solution $\mathbf{x}_{new}^{(t)}$ is generated according to the following equation:

$$x_{new,j}^{(t)} = x_{ini,j}^{(t)} + \phi_{i,j} \cdot (x_{best,j}^{(t)} - x_{ini,j}^{(t)}), \quad (27)$$

TABLE 2. The CEC Competitions on Real-Parameter Single-Objective Optimization Function Benchmark Suites.

Type	CEC'13		CEC'14		CEC'17	
	Num.	Dimensions	Num.	Dimensions	Num.	Dimensions
Unimodal	5	{10,30,50,100}	3	{10,30,50,100}	2	{10,30,50,100}
Multi-modal	15	{10,30,50,100}	13	{10,30,50,100}	7	{10,30,50,100}
Composition	8	{10,30,50,100}	8	{10,30,50,100}	10	{10,30,50,100}
Hybrid	n/a	n/a	6	{10,30,50,100}	10	{10,30,50,100}
Summary	28	4	30	4	29	4

where $\mathbf{x}_{ini}^{(r)}$ denotes the randomly generated vector, and $\mathbf{x}_{best}^{(r)}$ the best solution found in the population so far.

III. THE CEC FUNCTION BENCHMARK SUITES

The aim of the CEC competitions is to compare the state-of-the-art stochastic search algorithms on an annual basis. These competitions include various types of benchmark problem suites, like: single-objective, large-scale, noisy, multi-objective, and constrained optimization. The CEC competitions provide specific test environments for detailed algorithm assessment and comparison [25]. Mainly, the test environment is especially popular for benchmarking the EAs.

In this study, we are focused on the CEC Competition on Real-Parameter Single-Objective Optimization benchmark function suites. Over time, several competitions were organized as part of the general CEC conference. More information about the particular competitions can be found on the web site of Prof. Suganthan [26], who is one of the main promoters of Evolutionary Computation (EC) and the main driving force for development of the new benchmark suites in the last two decades. Besides the definitions of benchmark suites, the results of the particular competition and the winning algorithm's codes can also be found, together with articles describing their operations in detail.

Indeed, the CEC Competitions on Real-Parameter Single-Objective Optimization were organized in the years 2005 [27], 2013 [28], 2014 and 2016 [9], and 2017 and 2018 [10]. Afterwards, this competition was renamed to the 100-Digit Challenge Special Session and the Competition on Single Objective Numerical Optimization in 2019 and 2020 [29]. Interestingly, the two competitions were organized biannually, i.e., the same problem benchmark suites were used in the years 2014 and 2016, and in the years 2017 and 2018. Usually, the benchmark functions are implemented in Matlab, C and Java programming languages. Due to the changed propositions and application of different termination conditions in the last two benchmarks, this study is focused on the problem benchmarks issued in the last decade, i.e., in the years: 2013, 2014 (2016), and 2017 (2018).

The characteristics of the mentioned function benchmark suites are illustrated in Table 2, from which it can be seen that the functions in the suites are divided into four types: (1) Unimodal, (2) Multi-modal, (3) Hybrid, and (4) Composition functions. Unimodal functions have a single global optimum without any local optima. Interestingly, the CEC'13

benchmark function suite did not support the hybrid functions. These have arisen in later issues of CEC benchmark suites. Unimodal functions are non-separable and rotated. Multi-modal functions are either separable or non-separable. Additionally, these are also rotated and/or shifted. The hybrid functions are developed such that the variables are divided randomly into some subcomponents, and then different basic functions are used for different subcomponents [9]. Composition functions consist of the sum of two or more basic functions. Hybrid functions are used in this suite as the basic functions for constructing composition functions. As a result, the characteristics of these hybrid and composition functions depend on the characteristics of the basic functions.

The separability of a function determines typically how difficult the function is for solving. That is, function $f(\mathbf{x}_i)$ is separable if its parameters $\mathbf{x}_i = [x_{i,1}, \dots, x_{i,D}]$ are independent. In general, the separable functions are considered to be the easiest. In contrast, the fully-nonseparable functions are usually the more difficult to solve [9]. The degree of separability could be controlled by dividing the object variables randomly into several groups, each of which contains a particular number of variables. Although some of the used functions are separable in their original forms, applied techniques such as Salomon's random coordinate rotation make them non-separable. Furthermore, the global optimum of the function could also be shifted and scaled. The search range of the problem variables is limited to $x_{i,j} \in [-100, 100]$ for all functions. Thus, the orthogonal (rotation) matrix is generated from standard normally distributed entries by Gram-Schmidt orthonormalization, while matrix data are saved in the corresponding files according to their dimensions.

The functions of dimensions $D = 10$, $D = 30$, and $D = 50$ were used in the CEC'13 competition benchmark suite, while the dimension of $D = 100$ was added for the newer CEC competition benchmarks (i.e., CEC'14 and CEC'17). Because the CEC'13 functions are defined for $D = 100$ as well, solving these was included into the study too. The maximum number of fitness function evaluations MAX_NFE was limited to $10,000D$ according to four observed dimensions $D = 10$, $D = 30$, $D = 50$, and $D = 100$, respectively.

IV. EXPERIMENTS AND RESULTS

The purpose of our experimental work was to show that the following two hypotheses hold: (1) Selecting the CEC Competition on a Real-Parameter Single-Objective Optimization

benchmark issued in the last decade does not influence the assessment of the quality of the newly developed algorithm significantly, and (2) It is possible to find the algorithm achieving the best results on all observed benchmarks. The first issue speculates that the performance of the newly developed algorithm can be tested on arbitrary CEC benchmarks issued in the last decade, while the second that there might exist some best algorithm capable of outperforming the results of all the other algorithms by solving all the problems in the test.

Three sources served to help us in implementing the algorithms in Table 1: (1) The DE, jDE, and SaDE algorithms based on the original implementation of DE in C/C++ taken from the official Berkeley University of California web sites [30], (2) ABC on the implementation of the original implementation in C/C++ found on Karaboga's web sites [31], while (3) The implementation of the other algorithms appearing in CEC competitions, i.e., SSEABC, SHADE, LSHADE, iL-SHADE, jSO, LSHADE_RSP, were downloaded from Prof. Suganthan's GitHub repository [32]. Although source codes were taken from different sources, all these codes were implemented in the same C++ programming language and compiled using the g++ compiler of the same version. In this case, we ensured the fairness of the results.

Two experiments were performed in order to justify the research questions set in Section I. The results of the experiments, in which ten different algorithms were involved by optimizing three CEC function benchmarks of four different dimensions of functions, are illustrated in the remainder of the section.

A. INFLUENCE OF CEC BENCHMARKS ON THE PERFORMANCE OF ALGORITHMS

The purpose of the test was to identify how the different CEC benchmarks affect the results of ten stochastic nature-inspired algorithms. In line with this, all the mentioned algorithms were applied to the CEC'13, CEC'14, and CEC'17 function benchmarks. The experimental setup complied with the regulations of the CEC Competition on Real-Parameter Single-Objective Optimization, except that the number of independent runs was set to 25 due to reducing the time complexity. Thus, each algorithm was applied to three different benchmarks using the same parameter setup as presented at the beginning of the section. The results were evaluated using two statistical tests, i.e., the Friedman non-parametric test and Spearman rank correlation. The former is appropriate for ranking the algorithms according to their performance, while the latter for establishing the dependencies between ranked algorithms and the observed benchmarks.

1) FRIEDMAN NON-PARAMETRIC TEST

An analysis of the results based on the Friedman non-parametric statistical tests, in which the results obtained by a particular algorithm optimizing all benchmark functions of a specific dimension, were evaluated according to five

standard statistical measures: *Best*, *Worst*, *Mean*, *Median*, and *StDev* values. Typically, these tests are conducted in order to estimate the quality of the results obtained by various nature-inspired algorithms for global optimization [33]. Indeed, the Friedman non-parametric test is a two-way analysis of variances by ranks. In the first step, the statistic test is calculated and converted to ranks. Then, the null hypothesis is stated, which assumes that medians between the ranks of all algorithms are equal. Here, a high value of rank means a better algorithm. The second step is performed only if a null hypothesis of a Friedman test is rejected. In this case, the post-hoc tests are conducted using the calculated ranks. Let us notice that the post-hoc analysis was not performed in our study, because we needed only the proper ranking of the algorithm's performance for the next step. Here, the test was conducted using a significance level of 0.05.

In summary, there were $25 \times 3 \times 4 \times 10 = 3,000$ independent runs, where 25 denotes the number of independent runs, 3 the number of benchmarks, 4 the number of different function dimensions, and 10 the number of algorithms in the tests. Thus, each algorithm produced results evaluated according to $n \times 5$ different statistical measures per single run per benchmark, where n determines the number of functions in a particular benchmark (see Table 2). These results represent ranks obtained after calculating the Friedman non-parametric tests. However, the classifiers composed from the average values of statistical measures obtained after 25 independent runs were applied in place of values gained after a single run.

The results of the conducted tests are illustrated in Fig. 1, which is divided into four parts according to the observed dimensions of the functions (i.e., $D = 10$, $D = 30$, $D = 50$, $D = 100$). Thus, each part consists of a table and a graph, where the former presents the results of the particular algorithm numerically, while the latter illustrates the same data graphically.

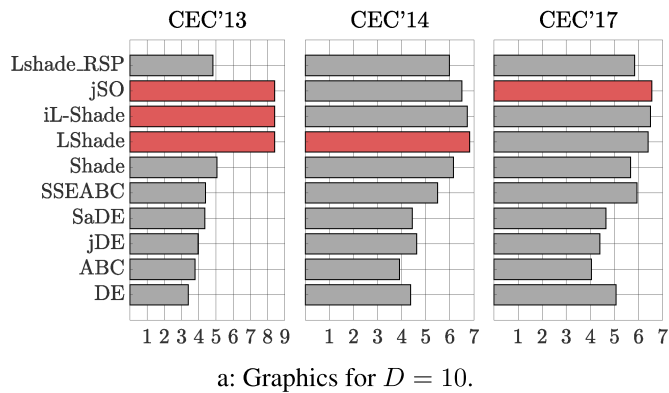
As can be seen from Fig. 1, the performance of the algorithms can be divided into three classes, i.e., the results of:

- the original algorithms (DE, ABC),
- the self-adaptive algorithms (jDE, SaDE, SSEABC),
- the SHADE branches (SHADE, LSHADE, iL-SHADE, jSO).

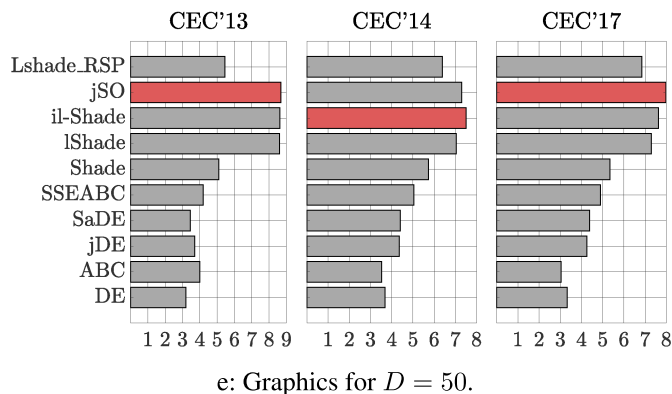
The results of the original algorithms showed that the ABC algorithm outperformed the results of the original DE on the CEC'13 benchmark suite on the one hand, while the DE algorithm achieved better results on the other suites. The self-adaptive SSEABC overcame the results of the other self-adaptive algorithms in tests, except by optimizing the functions of higher dimensions (i.e., $D = 100$), where the jDE algorithm was more powerful. More interesting is the comparison of the SHADE branch of algorithms, where the results achieved by the LSHADE, iL-SHADE, and jSO outperformed the results of the SHADE and LSHADE_RSP.

2) SPEARMAN RANK CORRELATION TEST

Spearman rank correlation is applied primarily when one wishes to assess whether the connection between two



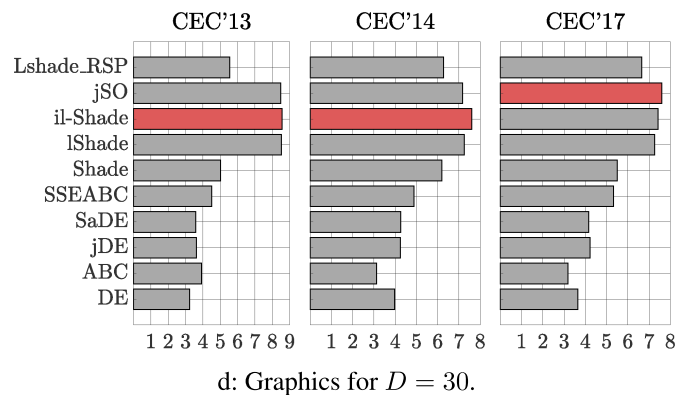
Algorithm	CEC benchmark		
	2013	2014	2017
DE	3.24	3.97	3.64
ABC	3.92	3.13	3.18
jDE	3.62	4.25	4.23
SaDE	3.58	4.26	4.16
SSEABC	4.50	4.88	5.33
SHADE	5.01	6.20	5.51
LSHADE	8.52	7.26	7.27
il-SHADE	8.56	7.60	7.43
jSO	8.49	7.17	7.60
LSHADE_RSP	5.55	6.27	6.66

c: Ranks by $D = 30$.

Algorithm	CEC benchmark		
	2013	2014	2017
DE	3.22	3.53	3.41
ABC	3.83	3.91	3.32
jDE	3.95	4.91	4.64
SaDE	3.95	4.43	4.27
SSEABC	4.23	4.54	4.42
SHADE	5.13	6.16	5.79
LSHADE	8.46	6.83	6.64
il-SHADE	8.52	7.23	7.71
jSO	8.60	7.22	7.95
LSHADE_RSP	5.10	6.26	6.84

g: Ranks by $D = 100$.

Algorithm	CEC benchmark		
	2013	2014	2017
DE	3.39	4.37	5.06
ABC	3.78	3.91	4.05
jDE	3.96	4.62	4.39
SaDE	4.36	4.44	4.65
SSEABC	4.39	5.49	5.93
SHADE	5.06	6.15	5.67
LSHADE	8.41	6.82	6.39
il-SHADE	8.41	6.73	6.49
jSO	8.41	6.50	6.55
LSHADE_RSP	4.83	5.98	5.83

b: Ranks by $D = 10$.

Algorithm	CEC benchmark		
	2013	2014	2017
DE	3.19	3.69	3.32
ABC	3.99	3.53	3.04
jDE	3.70	4.36	4.26
SaDE	3.45	4.40	4.38
SSEABC	4.20	5.05	4.89
SHADE	5.09	5.73	5.35
LSHADE	8.60	7.05	7.30
il-SHADE	8.62	7.51	7.63
jSO	8.69	7.30	7.99
LSHADE_RSP	5.45	6.39	6.84

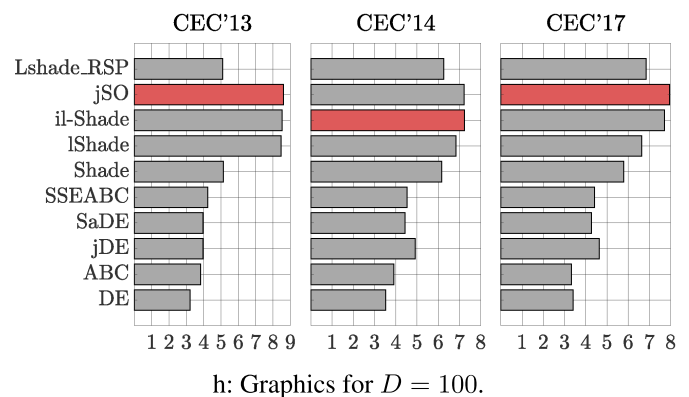
f: Ranks by $D = 50$.

FIGURE 1. The results of Friedman's non-parametric tests by various CEC benchmarks in competitions on real-parameter single-objective optimization.

non-normally distributed variables is significant or not. In our study, it was used to show how dependent rankings of

the particular algorithms are on the observed benchmarks. The Spearman correlation coefficient r_s is a non-parametric

measure of rank correlation that is expressed mathematically as [34]:

$$r_s = \frac{\text{cov}(r_X, r_Y)}{\sigma_{r_X} \cdot \sigma_{r_Y}}, \quad (28)$$

where X and Y are classifiers converted to ranks r_X and r_Y , respectively, σ_{r_X} and σ_{r_Y} the corresponding standard deviations of them, and $\text{cov}(r_X, r_Y)$ is a measure for joint variability of these variables, in other words:

$$\text{cov}(r_X, r_Y) = \frac{1}{N} \sum_{i=1}^N (r_{X,i} - \bar{r}_X) \cdot (r_{Y,i} - \bar{r}_Y). \quad (29)$$

In Eq. (29), \bar{r}_X and \bar{r}_Y denote means of ranks belonging to classifiers X and Y , while N is the number of algorithms in the test.

The null hypothesis in a Spearman test is set as $r_s = 0$, which asserts that variables do not correlate. This means that the test does not assume a linear relationship between two variables, but arbitrarily curved. To show how significant the relationship is between the pairs of observed variables [35], the Wilcoxon non-parametric test needs to be performed after calculation of the Spearman correlation coefficient r_s . The null hypothesis by the Wilcoxon test asserted that there was not a significant relationship between the two random variables. The result of this test is a p -value, denoting the probability of obtaining test results at least as extreme as the results actually observed, under the assumption that the null hypothesis is correct.

In our study, the Spearman rank correlation tests were performed between pairs of classifiers converted to ranks regarding the Friedman non-parametric tests, where each rank vector r_X represents the ranking of algorithms obtained by optimizing the particular CEC benchmark on different dimensions of the functions. Here, pairs represent combinations of two distinct rankings, obtained by optimizing particular CEC benchmarks. As a result, each rank vectors r_X and r_Y consisted of 4×10 elements (also samples), where 4 denotes the number of different dimensions of functions in the benchmark, and 10 is the number of algorithms. The same data were also entered in the Wilcoxon non-parametric test in order to calculate the corresponding p -values.

The results of the Spearman rank correlations are illustrated in Fig. 2 that are divided into two parts: a graph and a table. Graph 2a illustrates the distribution of data entered into the Spearman tests in the form of rank vectors. Table 2b presents the comparison of mutual comparisons between pairs of distinct benchmarks, represented as a matrix with rows designated benchmarks and columns values of Spearman correlation coefficient r_s and p -value as obtained by the Wilcoxon non-parametric test. For instance, the Spearman correlation coefficient by comparing rankings obtained by the CEC'13 and CEC'14 benchmarks is 0.95. This means, that the rankings obtained by optimizing the CEC'13 benchmark functions is not significantly different from the rankings obtained by optimizing the CEC'14 benchmark functions due to the p -value zero.

Interestingly, the p -values obtained for all the other comparison of benchmark pairs were also set to zero. This means that the rankings of a particular algorithm were not significantly different when the results of the same algorithms were compared on the different CEC benchmark suites. As a result, the first hypothesis can be accepted.

B. IN SEARCHING FOR THE BEST ALGORITHM

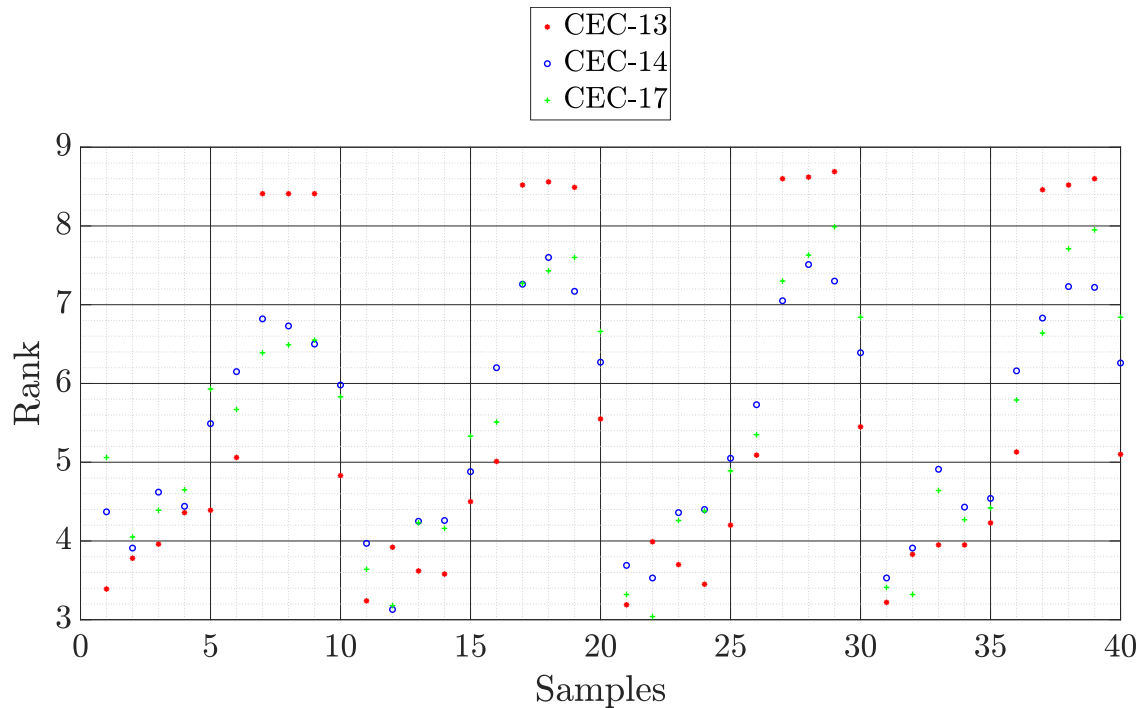
The purpose of this test was searching for the best algorithm that must be ranked in the first place by optimizing all the observed benchmarks. In line with this, the best algorithm according to the Friedman tests was identified and aggregated into Table 3, where rows denote the algorithms and columns their rankings in the first place by optimizing the observed benchmarks. Thus, the first place is denoted by a plus sign ('+'). We searched for the algorithm with the pluses in all columns (i.e., 12 pluses in one row).

From Table 3 it can be seen that three algorithms (i.e., LSHADE, iL-SHADE, jSO) achieved the same and highest rank by optimizing all the functions of dimension $D = 10$ collected in the CEC'13 benchmarks. This means that these problems became too easy for the state-of-the-art algorithms nowadays. Furthermore, the LSHADE achieved the best results 2-times, iL-SHADE 5-times, and jSO 7-times, by optimizing three different benchmark suites by different dimensions of the functions. When the number of pluses is analyzed from the benchmark's point of view, it can be concluded that the iL-SHADE was better on the CEC'14 benchmark functions, while jSO on CEC'13 and CEC'17. According to the presented results, we can conclude that the best algorithm does not exist. This fact obviously justifies that the second hypothesis must be rejected.

C. DISCUSSION

Our experimental work started with the assumption that selecting a CEC benchmark issued in the last decade does not affect the ranking of the newly developed algorithm significantly. On the other hand, some reviewers argued that the new CEC benchmarks are more relevant for determining the quality of the nature-inspired algorithms. Consequently, the winning algorithms solving these benchmarks might also be more relevant. As can be derived from these two assumptions, there must exist an algorithm capable of outperforming the results of all the other algorithms on all observed benchmarks. Unfortunately, the second assumption was rejected by the fact that iL-SHADE outperformed the results of jSO on the CEC'14 benchmark, and vice versa, that jSO outperformed the results of iL-SHADE on the CEC'17 benchmark. Are our assumptions or the derivation from them wrong?

Obviously, each coin has two sides. On the one hand, the purpose of organizing the CEC competitions is not to find a general problem solver capable of solving all problems, but encourage the development of new, more powerful nature-inspired algorithms, by solving increasingly difficult problems. The fact that the benchmarks' problems become harder and harder from year to year proves the fact that



a: Distribution of data.

Bench.	CEC' 13		CEC' 14		CEC' 17	
	r_s	p	r_s	p	r_s	p
CEC' 13	1.00	n/a	0.95	0.0	0.97	0.0
CEC' 14			1.00	n/a	0.92	0.0
CEC' 17					1.00	n/a

b: The results of the Spearman rank test.

FIGURE 2. The results of Spearman rank correlation test by various CEC benchmark function suites in competitions on real-parameter single-objective optimization.**TABLE 3.** The Best Algorithms According to Benchmark Function Suites.

Alg.	CEC' 13				CEC' 14				CEC' 17				Sum.
	10	30	50	100	10	30	50	100	10	30	50	100	
DE													0
jDE													0
SaDE													0
SHADE													0
LSHADE	+				+								2
iL-SHADE	+	+				+	+	+					5
jSO	+			+	+				+	+	+	+	7
LSHADE_RSP													0
ABC													0
SSEABC													0
Sum.	3	1	1	1	1	1	1	1	1	1	1	1	14

even three state-of-the-art algorithms are capable of finding the optima of all functions of dimension $D = 10$ collected in the CEC'13 benchmark. On the other hand, algorithms appearing in the particular competitions are adapted strongly to the demands of specific benchmarks. This means that even the winner of the CEC'17 competition, that is adapted to the demands of the current benchmark, is probably not capable

of coping with the performance of the winner of the previous competitions.

In general, could we pronounce the best algorithm on all CEC benchmark suites? In our opinion, the answer to this question gives us the famous "No Free Lunch" (NFL) theorem by Woolpert and Macready [36], asserting that any two optimization algorithms are equivalent when their

performance is averaged across all problems. How about selecting the CEC benchmarks? In our opinion, there is no big difference in using any of the CEC benchmarks from the last decade. All are hard enough for showing the advantages and weaknesses of the newly developed algorithms.

V. CONCLUSION

Developers of new stochastic nature-inspired algorithms are typically confronted with the issue of how good their product is when compared with the existing ones. Consequently, a lot of benchmark suites have emerged that enable assessment of the quality of algorithms objectively. In this sense, benchmark function suites prepared for competitions at the GECCO and CEC conferences present a suitable testbed for testing the quality of these algorithms. Because the CEC benchmarks are founded on more or less an annual basis, usually, the question arises how the particular benchmark suite influences the ranking of algorithms in these competitions?

This article is focused on the CEC Competitions on Real Parameter Single-Objective Optimization, with three benchmark suites taken into consideration, i.e., in the years 2013, 2014 (2016), and 2017 (2018). Ten stochastic nature-inspired algorithms were captured in this study: the original DE and ABC, the self-adaptive jDE, SaDE and SSEABC, and the SHADE branch of algorithms, incorporating the original SHADE, LSHADE, iL-SHADE, jSO and LSHADE_RSP.

The purpose of this study was twofold: (1) To show that using different CEC benchmark function suites does not influence their ranking crucially, and (2) To find the best algorithm capable of outperforming the results of the other algorithms on all benchmarks. In the sense of the first issue, the Spearman rank correlation was conducted, based on Friedman's ranks obtained by optimizing the particular CEC benchmark functions of four different dimensions (i.e., $D = 10$, $D = 30$, $D = 50$ and $D = 100$). The Spearman rank correlation tests with classifiers containing ranks obtained by classifying the results of ten different algorithms by solving the different CEC benchmark suites showed no significant differences among rankings. However, the rankings of the top state-of-the-art algorithms in tests showed, for instance, that the best ranked algorithm by solving the CEC'17 function suite is not the best by solving the CEC'14 suite and vice versa. The reason for this needs to be searched in the fact that both specific algorithms were issued at different times for different competitions, i.e., the former was tuned for solving the CEC'17 function suite optimally, while the latter for the CEC'14 suite. Obviously, this phenomenon confirms the NFL theorem.

Many directions exist for the future work, for example: to make a similar study for the other CEC competitions, to include the additional algorithms into the existing study, or to extend the same study to the GECCO function suite too.

REFERENCES

- [1] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd ed. Berlin, Germany: Springer, 2015.
- [2] C. Blum and D. Merkle, *Swarm Intelligence: Introduction and Applications*, 1st ed. Berlin, Germany: Springer, 2008.
- [3] K. Sörensen, "Metaheuristics—The metaphor exposed," *Int. Trans. Oper. Res.*, vol. 22, no. 1, pp. 3–18, 2015.
- [4] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimization problems," *Int. J. Math. Model. Numer. Optim.*, vol. 4, no. 2, pp. 150–194, 2013.
- [5] N. Hansen, T. Tusar, O. Mersmann, A. Auger, and D. Brockhoff, "COCO: The experimental procedure," 2016, *arXiv:1603.08776*. [Online]. Available: <http://arxiv.org/abs/1603.08776>
- [6] I. Fister, S. Deb, D. Fister, and I. Fister, "How does selecting a benchmark function suite influence the estimation of an Algorithm's quality?" in *Proc. 6th Int. Conf. Soft Comput. Mach. Intell. (ISCMI)*, Nov. 2019, pp. 95–100.
- [7] A. P. Piotrowski, "Review of differential evolution population size," *Swarm Evol. Comput.*, vol. 32, pp. 1–24, Feb. 2017.
- [8] M. S. Maucec and J. Brest, "A review of the recent use of differential evolution for large-scale global optimization: An analysis of selected algorithms on the CEC 2013 LSGO benchmark suite," *Swarm Evol. Comput.*, vol. 50, Nov. 2019, Art. no. 100428.
- [9] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China, Nanyang Technol. Univ., Singapore, Tech. Rep. 201212, 2013.
- [10] N. H. Awad, M. Z. Ali, J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained real-parameter numerical optimization," Nanyang Technol. Univ., Singapore, 2016.
- [11] 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.
- [12] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2, Dec. 2005, pp. 1785–1791.
- [13] R. Tanabe and A. Fukunaga, "Evaluating the performance of SHADE on CEC 2013 benchmark problems," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 1952–1959.
- [14] 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.
- [15] J. Brest, M. S. Maucec, and B. Boskovic, "iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 1188–1195.
- [16] J. Brest, M. S. Maucec, and B. Boskovic, "Single objective real-parameter optimization: Algorithm jSO," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2017, pp. 1311–1318.
- [17] V. Stanovov, S. Akhmedova, and E. Semenkin, "LSHADE algorithm with rank-based selective pressure strategy for solving CEC 2017 benchmark problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2018, pp. 1–8.
- [18] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, Oct. 2007.
- [19] G. Yavuz, D. Aydin, and T. Stutzle, "Self-adaptive search equation-based artificial bee colony algorithm on the CEC 2014 benchmark functions," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 1173–1180.
- [20] 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.
- [21] 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, Apr. 2009.
- [22] J. Arabas, Z. Michalewicz, and J. Mulawka, "GAVaPS—A genetic algorithm with varying population size," in *Proc. 1st IEEE Conf. Evol. Comput. IEEE World Congr. Comput. Intell.*, Jun. 1994, pp. 73–78.
- [23] M. A. M. D. Oca, T. Stutzle, K. Van den Enden, and M. Dorigo, "Incremental social learning in particle swarms," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 41, no. 2, pp. 368–384, Apr. 2011.
- [24] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, vol. 27, 1st ed. Reading, MA, USA: Addison-Wesley Longman Publishing, 1989.

- [25] 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.
- [26] P. N. Suganthan. *Suganthan's Home Page*. Accessed: Mar. 7, 2021. [Online]. Available: <https://www.ntu.edu.sg/home/epsnugan/>
- [27] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Kanpur Genet. Algorithms Lab., IIT Kanpur, Nanyang Technol. Univ., Singapore, Tech. Rep. 2005005, 2005.
- [28] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," Comput. Intell. Lab., Nanyang Technol. Univ., Singapore, Zhengzhou Univ., Zhengzhou, China, Tech. Rep. 201311, 2013.
- [29] K. V. Price, N. H. Awad, M. Z. Ali, and P. N. Suganthan, "Problem definitions and evaluation criteria for the 100-digit challenge special session and competition on single objective numerical optimization," Nanyang Technol. Univ., Singapore, 2018.
- [30] Berkeley University of California. *Differential Evolution (DE)*. Accessed: Jan. 18, 2021. [Online]. Available: <https://www1.icsi.berkeley.edu/~storn/code.html>
- [31] D. Karaboga. *Artificial Bee Colony (ABC) Algorithm Homepage*. Accessed: Mar. 15, 2021. [Online]. Available: <https://abc.erciyes.edu.tr/>
- [32] P. N. Suganthan. *Suganthan's GitHub Web Page*. Accessed: Feb. 22, 2021. [Online]. Available: <https://github.com/P-N-Suganthan>
- [33] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Ann. Math. Statist.*, vol. 11, no. 1, pp. 86–92, Mar. 1940.
- [34] J. H. Zar, "Significance testing of the spearman rank correlation coefficient," *J. Amer. Stat. Assoc.*, vol. 67, no. 339, pp. 578–580, Sep. 1972.
- [35] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull.*, vol. 1, no. 6, pp. 80–83, 1945.
- [36] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.



IZTOK FISTER (Member, IEEE) received the degree in computer science from the University of Ljubljana, in 1983, and the Ph.D. degree from the Faculty of Electrical Engineering and Computer Science, University of Maribor, in 2007. Since 2010, he has been a Teaching Assistant with the Computer Architecture and Languages Laboratory, Faculty of Electrical Engineering and Computer Science, University of Maribor. He is currently also with the University of Cantabria, Santander. His research interests include computer architectures, programming languages, operational researches, artificial intelligence, and evolutionary algorithms. In his research areas, he has published more original scientific articles, review articles, book chapters, edited books, and contributed to more than 100 international scientific conferences. He is also one of the presidents in the International Student Conference in Computer Science. In many journals with impact factor, he has been promoted as an outstanding reviewer.



JANEZ BREST (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in computer science from the University of Maribor, Maribor, Slovenia, in 1995, 1998, and 2000, respectively.

He is currently a Full Professor and also the Head of the Laboratory for Computer Architecture and Programming Languages, Institute of Computer Science, Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor. His research interests include evolutionary computation, bio-inspired algorithms, artificial intelligence, and optimization. His fields of expertise embrace programming languages, and parallel and distributed computing research.

Dr. Brest is an Associate Editor of the journal *Swarm and Evolutionary Computation*.



ANDRES IGLESIAS is currently with Toho University, Funabashi, Japan, and also with the University of Cantabria, Santander, Spain, where he is also the Director of the Computer Graphics and Artificial Intelligence (CGAI) Research Group. He has published more than 230 scientific articles (including journal articles in 21 categories of JCR) and 14 books, and holds two patents. His research interests include computer graphics, geometric modeling, artificial intelligence, and soft computing. He is the current chair of IFIP TC5 WG5.10 and an editorial board member of 15 international journals. He has been a chair/organizer of 55 international conferences and workshops and a program committee member of more than 300 international conferences. He has been a reviewer of about 200 international articles in journals (mostly in JCR journals) and more than 500 articles in international conferences, and expert evaluator of projects for NSF (USA), the European Commission Horizon 2020 and FP7, and other national and regional public research agencies in several countries.



AKEMI GALVEZ is currently with Toho University, Funabashi, Japan, also with the University of Cantabria, Santander, Spain, and also a member of the Computer Graphics and Artificial Intelligence (CGAI) Research Group. She has published more than 160 international articles in scientific journals and conferences, including articles in top journals of several categories of JCR, and holds two patents. Her research interests include geometric modeling and processing, shape reconstruction, artificial intelligence, soft computing, and their industrial applications. She is also an editorial board member of seven international journals. She has also been co-chair/co-organizer of prestigious international conferences such as ICMS'2006, ICCSA'2011, and Cyberworlds'2014, and reviewer/program committee member of several international conferences in the fields of computer graphics and artificial intelligence.



SUASH DEB (Senior Member, IEEE) received the B.E. degree in mechanical engineering from Jadavpur University, Kolkata, and the M.Tech. degree in computer science from the University of Calcutta. Besides being an IT & Educational Consultant, he is currently a Distinguished Professorial Associate with Victoria University, Melbourne, Australia. He published extensively in various reputed journals & also received a no. of awards/honors from peers of multiple organizations. His research interests include metaheuristics, swarm intelligence, and machine learning. He received the United Nations Fellowship in computer science from Stanford University, USA. He served as the Secretary for the IEEE Computational Intelligence Society (Kolkata Chapter), also the Founding President for the International Neural Network Society (INNS)—India Regional Chapter, and also the Founding Secretary General for the India International Congress on Computational Intelligence (IICCI).



IZTOK FISTER, JR. (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in computer science from the University of Maribor, Slovenia. He is currently an Assistant Professor with the University of Maribor. He has published more than 120 research articles in referred journals, conferences, and book chapters. His research interests include data mining, pervasive computing, optimization, and sport science. He has acted as a program committee member at more than 30 international conferences. Furthermore, he is a member of the editorial boards of five different international journals.

...