Swarm and Evolutionary Computation 43 (2018) 88–108

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Step-by-step improvement of JADE and SHADE-based algorithms: Success or failure?

Adam P. Piotrowski *, Jaroslaw J. Napiorkowski

Institute of Geophysics, Polish Academy of Sciences, Ks. Janusza 64, 01-452 Warsaw, Poland

ARTICLE INFO

Keywords:
Differential Evolution
Evolutionary Algorithms
Benchmark problems
Comparison of metaheuristics
Population size
Computational speed

ABSTRACT

For about two decades Differential Evolution (DE) algorithms belong to the most successful optimization metaheuristics. Among plentiful DE versions proposed so far those that are based on mutation strategies and control parameter adaptation methods introduced within JADE and SHADE variants show especially encouraging performance. However, many modifications of JADE or SHADE are developed simultaneously by various researchers, and are rarely compared or discussed against each other. As most of JADE or SHADE-based variants are tested on recently introduced sets of artificial benchmark functions with a single, pre-specified maximum number of function calls, it remains unclear how they would perform on real-world problems, or when the number of allowed function calls differs from the standard values. In this study in-deep insight into the performance of twenty two JADE/SHADE-based variants on a large sets of artificial benchmarks and real-world problems is presented. The impact of the pre-assumed maximum number of function calls and the algorithm population size on the results is verified and discussed. Finally, overall comparison of algorithms that originate from JADE or SHADE against other kinds of metaheuristics is presented. The main aim of the study is to point out these among recently introduced JADE or SHADE-based operators that turn out to be especially successful, and to determine conditions under which they either achieve desired results or fail.

1. Introduction

The wide popularity of Differential Evolution (DE) algorithms [64] leads to sizable interest in developing their novel variants. Although DE algorithms are modified or hybridized with other approaches in various ways [19,20,52], most frequently the attention is focused on mutation strategies and adaptation of algorithm control parameters. Since 2009 we witness a massive influx of DE variants based on both the mutation strategy and the control parameters adaptation method proposed by Zhang and Sanderson within JADE algorithm [81]. JADE has been called one of a few "important variants of DE" in a major DE review published in 2011 [19]. Among plentiful JADE-based methods, a modification that introduces a memory of successful history to control parameter adaptation scheme called SHADE [65] gains much attention, partly because SHADE-based variants often turned out among the winners of recently held IEEE Competitions on Evolutionary Computation (CEC). SHADE has been ranked the fourth best (losing only to some CMA-ES-based [37] methods) out of 21 algorithms in CEC′2013 competition, L-SHADE [66] has been the winner of CEC′2014 competition, SPS-L-SHADE-EIG [36] has been the winner of CEC′2015, L-SHADE-EpSin [3] has been one among two joint-winners of CEC′2016 and jSO (a L-SHADE-based algorithm) [12], L-SHADE-cnEpSin [7] and L-SHADE-SPACMA [50] were ranked among the best four algorithms at CEC′2017 competition. Such results contribute to high reputation of JADE/SHADE-based algorithms, but need some caution, as three out of four recent IEEE CEC competitions which have been won by SHADE-based algorithms were held on the same set of 30 artificial benchmark problems proposed for CEC′2014 [45], and the fourth (CEC′2017 [5]) was based on artificial problems developed in a similar way. This may bias the results, as it is well known that the selection of problems affects the ranking of compared algorithms, not only with respect to the No Free Lunch theorems [75]. For example in Ref. [26] it was shown that JADE, proposed in 2009, in fact does not outperform the best methods tested within CEC′2005 competition, and in Ref. [54] it was found that various algorithms, including JADE, perform much differently when one search for either the minimum or the maximum of exactly the same functions. Lai and Zhou [42] showed that, depending on the problem which is solved, hybrid algorithms may, or may not outperform the basic methods which were hybridized. Very recently, Molina et al. [51] showed that even the winners of three IEEE CEC competitions based on relatively similar kinds of artificial

* Corresponding author.
E-mail addresses: adampp@igf.edu.pl (A.P. Piotrowski), jnn@igf.edu.pl (J.J. Napiorkowski).

https://doi.org/10.1016/j.swevo.2018.03.007
Received 3 August 2017; Received in revised form 8 January 2018; Accepted 11 March 2018
Available online 17 March 2018

benchmark functions (CEC′2013, CEC′2014 and CEC′2015) may be ranked differently, depending on which among the three classification criteria (i.e. those based on CEC′2013, CEC′2014 or CEC′2015) is applied.

Although many JADE/SHADE-based variants have been proposed within a few recent years, most of them have not been mutually compared (with exception of off-line comparisons among algorithms submitted for the same IEEE CEC competition), even though they often have been tested against the first JADE or the first SHADE approach. There is, however, one important exception – in Ref. [51] three L-SHADE-based variants were compared (L-SHADE [66], L-SHADE-ND [60] and SPS-L-SHADE-EIG [36]), overall showing that none of them is superior to the other two. Also, in almost every publication with new JADE/SHADE-based variant the proposed approach is tested only with the single pre-defined number of allowed function calls. Although this is a standard procedure used for many years when testing metaheuristics, as shown in Refs. [56,58] the efficiency of various algorithms may largely depend on the number of allowed function calls, and the values on which the most frequently algorithms are tested do not necessarily have to meet the need of practitioners that may wish to use them. Finally, it is well known that the performance of many metaheuristics may be affected by the choice of their control parameters [23,79], especially the population size [55]. How different values of control parameters could be recommended for different problems may be learned from Ref. [36] for SPS-L-SHADE-EIG variant of SHADE algorithm that uses the linear population size reduction method.

This study carries out an inter-comparison among a relatively large number of JADE/SHADE-based variants on artificial-benchmark and real-world problems with various numbers of allowed function calls and different population sizes. Our main goal is to find whether the development of JADE and SHADE-based algorithms is an example of "excellent evolution", as praised in Ref. [26], or is just "adding stuff" [39]. We also wish to determine which operators and procedures recently introduced to JADE or SHADE variants are most beneficial, and which are of marginal, if any importance for both types of problems. What may be of special interest to practitioners, we aim at finding whether the results would be consistent when very small, classical and very large number of function calls is available. Verifying the practical importance of numerous control parameters that appear in various JADE/SHADE-based variants is impossible in any short study, but we aim at studying at least the impact of the choice of population size, which is the main control parameter of every DE variant.

We point out the most efficient improvements of JADE/SHADE-based variants from practitioners' point of view, and suggest the possible ways for further development of such algorithms. However, because the number of citations to the first JADE paper alone [81] in Google Scholar well exceeds 1400 at the time of writing, not to mention papers that refer to other JADE/SHADE-based variants, the study have to be inevitably incomplete. We are afraid that during the time needed to find, understand and discuss all modifications of JADE or SHADE algorithms at least similarly large number of new JADE/SHADE-based approaches could be published. Hence, in this paper we mention only a subjectively selected fraction of papers that have been published, and test only a subjectively selected fraction of algorithms that we have mentioned. This incompleteness is a drawback of our study, but it also justifies it – simply because of the wide interest in the topic.

The rest of the paper is structured as follows. In section 2 a brief history of development of JADE and SHADE-based algorithms is outlined. Section 3 introduces comparison criteria and JADE and SHADE-based variants that are inter-compared in this study. The results are discussed from various points of view in section 4, and section 5 concludes the paper.

2. Variants of JADE/SHADE algorithms – a brief history

The basic DE algorithm was proposed by Storn and Price [64] and since then has been the subject of many modifications. Here we do not aim neither at discussing the behaviour of DE variants, nor at introducing the history of their development; the interested reader is referred to the major review papers [19,20,52]. The goal of this section is only to outline a brief history of DE sub-variants based on JADE [81] or SHADE [65].

With respect to previous DE variants, there were two main novelties proposed within JADE [81] algorithm in 2009. The first one was the current-to-*p*best mutation strategy that, to effectively guide the search, uses information from one of *p*% of best members of the population. Earlier majority of DE mutation strategies used either only randomly selected individuals, what led to too random moves, or incorporated the best solution to guide the search, what led to premature convergence. The second novelty was the adaptation method for the two main DE control parameters, namely the scaling factor F by which mutation vector is multiplied and the crossover parameter CR. Soon after publication of JADE, Peng et al. [53] introduced its important modification by using weighted mean to F and CR control parameter adaptation, where the weights depend on the relative improvement achieved recently by individuals that used the particular pair of control parameter values. This adaptation with weighted mean has been later applied within SHADE algorithm [65], in which a memory to store successful control parameter values has been added, and has also been extended to multiobjective DE variant [46].

Setting Peng et al. [53] modification apart, the original JADE algorithm [81] has been modified in a number of other papers. Two JADE-based variants with a number of mutation strategies used with adaptively modified probabilities were introduced in Refs. [28,29]. Another variant of JADE with ranking-based mutation operators has been proposed in Ref. [30]. A method to "repair" the value of crossover parameter has been introduced in Ref. [31] and further modified in order to be used within micro-JADE (i.e. JADE with very small population size) in Ref. [13]. Yi et al. [78] also used the crossover repairing approach in their variant of JADE in which the probability of choosing particular solution among *p*best ones depends on the relative performance of individuals, and a specific mechanism to retain some useful individuals that lost selection procedure is added. In Ref. [17] a distributed variant of DE was proposed that uses JADE-based control parameters adaptation scheme, and mutation strategies that are based on the current-to-*p*best mutation strategy introduced in JADE. Both in algorithm introduced in Ref. [17] and in the earlier one proposed in Ref. [25] the use of individuals within mutation strategy based on JADE to some extent depends on the Euclidean distances between the members of the current population. Control parameter adaptation methods introduced within JADE were also implemented into heuristics that hybridize Differential Evolution with either Genetic Algorithm [2] or Artificial Bee Colony Optimization [83]. In a few papers [73,84] different population size adaptation schemes were introduced into modified versions of JADE. Segundo et al. [61] proposed the JADE variant with control parameters that are generated from Tsallis distribution, instead of Normal and Cauchy distributions that were used in classical JADE [81]. In Ref. [77] a method to avoid stagnation by enhancing population diversity was incorporated into JADE. A modification of JADE that traces its evolution path in a way inspired by CMA-ES [37] was proposed in Ref. [43]. Authors of [44] proposed a hybridized version of modified JADE and CoDE algorithms [72]. Guo et al. [34] introduced a variant of JADE with eigenvector-based crossover operator in order to facilitate the search for non-separable problems. MDE-pBX algorithm [40], although differs from JADE in various ways, uses JADE-based control parameter adaptation scheme and a mutation strategy that is largely based on the one proposed in JADE. Also multi-population ensemble algorithm MPEDE introduced in Ref. [76] uses both the mutation strategy and the control parameter adaptation methods from JADE. The JADE-based current-to-*p*best mutation strategy has been used within a multi-objective portfolio optimization algorithm described in Ref. [48]. Authors of various other DE variants refer to JADE as an inspiration (to mention just [74]), even though their algorithms share only slight similarity with JADE.

The SHADE algorithm introduced in 2013 [65] has also quickly

become the kick-off point for new variants, frequently together with JADE. Guo et al. [35] proposed successful-parent-selection (SPS) framework and introduced it to both SHADE and JADE. In Ref. [21] an event-trigged impulsive (ETI) control scheme-based operator has been added to both JADE and SHADE in order to regulate the positions of individuals. Gong et al. [32] included surrogate models into both JADE and SHADE; the idea was to benefit from high performance of JADE or SHADE and from computational speed of surrogate models. Various variants of crossover operators within SHADE algorithm has been implemented in Refs. [14,67]. Authors of [82] introduced ARDE method that, inspired by an idea of using an archive for selected solutions from JADE, keeps successful difference vectors archived for future mutation operations. ARDE also applies the control parameters adaptation scheme proposed in SHADE.

A year after introducing SHADE, Tanabe and Fukunaga proposed its variant with linear population size reduction (L-SHADE) [66], in which population size is initialized to be 18 times larger than the problem dimensionality, and is linearly decreased to just 4 individuals at the end of the search. L-SHADE was applied to solve various constrained problems in Ref. [80], and to find the most efficient location of wind turbines in a windfarm [9]. As L-SHADE become the winner of CEC′2014 competition, a number of other researchers worked on its further modifications, especially those participating in IEEE CEC conferences held in 2015, 2016 and 2017. Many best performing algorithms at these three recent IEEE CEC competitions on unconstrained single-objective optimization problems were based on L-SHADE. SPS-L-SHADE-EIG [36] that merges L-SHADE with eigenvector-based crossover operator [34] and successful-parent-selection framework [35] was the winner of CEC′2015 competition. L-SHADE-EpSin [3] that introduces two sinusoidal-based adaptation strategies for scaling factor parameter (to be used just within the first half of the search, leaving classical L-SHADE-based adaptation during the second half) and a local search procedure within L-SHADE, was called the co-winner of CEC′2016 competition. However, as shown in Ref. [57], the local search procedure used within L-SHADE-EpSin is structurally biased and over-frequently samples solutions located close to the origin of the coordinate system, what affects the results achieved. The L-SHADE-EpSin has been again modified in Ref. [7], where an adaptive method of choosing which among two sinusoidal adaptation strategies of scaling factor to use, and specific, neighborhood-related CMA-based adaptation of crossover control parameter were added. This method, called L-SHADE-cnEpSin, was also highly ranked in CEC′2017 competition. In another study [8] similar CMA-based rotation of coordinate system during crossover was tested within classical L-SHADE. L-SHADE, in which scaling parameter adaptation is performed differently in the first and the second half of the search (as in L-SHADE-EpSin [3]), has also been hybridized with CMA-ES in Ref. [50]. The resulting algorithm, called L-SHADE-SPACMA, was ranked fourth at CEC′2017 competition. In iL-SHADE algorithm [11] the modified mechanism of managing the historical memory values was introduced. Although this algorithm did not achieve spectacular successes, its modified version called jSO [12], in which classical DE/current-to-pbest/1 strategy introduced by Zhang and Sanderson [81] for JADE was slightly modified, turned out the second best method of CEC′2017 competition. Apart from IEEE CEC conferences, an algorithm based on L-SHADE that modifies mutation strategies and control parameter adaptation methods has been recently introduced in Ref. [1]. L-SHADE has also been hybridized with stochastic fractal search procedures [4] and cultural algorithms [6]. Finally, a different, network-based kind of the linear population size reduction procedure was proposed for SHADE in Ref. [70].

3. Methodology

As discussed in section 2, there are many algorithms based on JADE or SHADE, but the question remains how much, and under which conditions, the proposed modifications could lead to the improvements in performance. To find that 22 different JADE or SHADE-based variants are tested (see Table 1) on thirty 50-dimensional CEC′2014 artificial benchmarks and 22 various-dimensional real-world problems from CEC′2011 set. The choice of both sets of numerical minimization problems is motivated by two factors that stress their diversity. Firstly, CEC′2014 is composed of the artificial benchmark functions, when CEC′2011 of the real-world problems. Secondly, CEC′2014 set have been frequently used in papers in which modified JADE or SHADE variants were introduced, contrary to CEC′2011 set which was rarely applied. One may be interested if the supposed improvement observed on one set would be confirmed on the other.

As suggested in the source papers of both test sets [18,45], in the case of CEC′2014 problems the maximum number of function calls (NFC) is set to $10\,000 \cdot D$ (where D is the problem dimensionality) and in the case of CEC′2011 problems to 150 000. However, to verify the performance of various JADE/SHADE-based variants under different conditions, we also perform tests with the NFC values five times larger (called NFC · 5), five times lower (NFC / 5) than the suggested ones, and, because some practical applications are time consuming, with only 5000 function calls. Independent tests on the 50-dimensional CEC′2014 problems are hence performed with the numbers of function calls set to 5000, 100 000, 500 000 and 2 500 000; independent tests on the CEC′2011 problems are performed with the numbers of function calls set to 5000, 30 000, 150 000 and 750 000. However, because on computationally the most time-consuming problem F3 from the CEC′2011 set all algorithms achieve almost equal performance when the number of function calls is set to 150 000, we skip this single problem when 750 000 function calls are used. This allows us to save about 40% of computational time without affecting the results.

In addition, each JADE/SHADE-based variant is tested with three different population sizes. When the number of function calls is larger than 5000, those three population sizes are: the default value (PS), the value twice smaller (PS / 2) and twice larger (PS · 2). When the number of function calls is set to 5000, it would be a waste of time to test algorithms with larger population sizes, as rather very low population size may be successful. Hence, when 5000 function calls are allowed, algorithms are tested with: default population size (PS), the value twice smaller (PS / 2) and the population size set to 20. As default value we consider either the value suggested in the source paper if this choice was motivated there, or $PS = 100$ that is the most frequent choice in DE [55] (see Table 1 for detailed choices for each specific algorithm). In case of variants with the linear population size reduction (L), the smallest (final) population size (set to 4) is kept the same in each variant, but three initial (the highest) population size values are tested: the default one (PS), twice smaller (PS / 2) and either twice larger (PS · 2, when the number of function calls is higher than 5000) or set to 20 (when the number of function calls is set to 5000). The final population size is not modified as its size is theoretically justified as the lowest number of individuals required for mutation strategies. In case of JADE-based variants with other adaptive population sizes (ATPS-DE [84] and SapsDE [73]) the upper and the lower limits of population size are tested with the default, twice smaller and, when the number of function calls is higher than 5000, twice larger values. When the number of function calls is set to 5000, the third variant of ATPS-DE [84] is used with the upper and the lower limit of population size set to 20, and both the lower bound and the initial value of population size in SapsDE [73] are set to 20 (in SapsDE there is no upper bound). Because CEC′2011 set includes both very low- and high-dimensional problems (dimensionality vary from 1 to 216), in case of some algorithms which population size is suggested to depend on the problem dimensionality, the additional bounds are set on the population size to avoid too small populations when D is very small, or too large populations when D is high (see Table 1 for details). Summarizing, every JADE/SHADE-based variant is tested 24 times: on two test sets, with four different values of maximum number of function calls and with three different population sizes. For clarity, all 24 considered test settings are summarized in Table 2. Each algorithm is run 51 times on every problem

Table 1
Tested JADE-based variants. Abbreviations: DE – Differential Evolution; D – problem dimensionality. [*] – marks algorithms which MATLAB codes were obtained from authors of the source papers or relevant web pages (see Comments). Default value of population size used for particular variant is bolded. When the number of function calls is set to 5000, algorithms are not tested with the largest population size. Instead, they are tested with population size (or initial population size in case of linear population size reduction, and bounds of population size in case of ATPS-DE and SapsDE) set to 20.

short name	long name	reference	year	three tested variants of population size when number of function calls is higher than 5000	comments
AdapSS-JADE	JADE with adaptive strategy selection	[28]	2011	50; **100**; 200;	Modified variant of JADE [81] with an archive, based on specific concept of self-adaptation. AdapSS-JADE uses four mutation strategies, each with different and adaptively modified probability. The variant with normalized average reward method is used, as the best among four ones proposed in Ref. [28].
ATPS-DE	JADE with adaptive population tuning scheme	[84]	2013	variable within [25,100], initialized with $2.5 \cdot D$ within bounds; **variable within [50,200], initialized with $5 \cdot D$ within bounds**; variable within [100,400], initialized with $10 \cdot D$ within bounds;	Variant of JADE [81] with population size variable within pre-specified range. The algorithm may increase or decrease population size during run depending on how successful the recent generations are. We decided to initialize population size with $5 \cdot D$ individuals as default value when this number is within the bounds, otherwise the default initial population size is set to the value on the bound. In case of twice lower and twice higher populations both the bound limits and initialization population size are respectively increased or decreased.
ETI-SHADE	SHADE with event-triggered impulsive control scheme	[21]	2017	75; **150**; 300;	Event-trigged impulsive control (ETI) scheme has been tested within various DE variants in Ref. [21]. The best results were obtained by authors of [21] when ETI is coupled with SHADE (ETI-SHADE), hence such variant is tested in this paper. Authors of [21] performed tests with various population sizes and stressed that, when ETI is implemented within SHADE, population size should be higher than in case of most DE algorithms.
HMJCDE	Hybrid framework combining modified JADE and modified CoDE	[44]	2016	50; **100**; 200;	In Ref. [44] first, a modified versions of JADE and CoDE [72] are proposed, then these two modified versions are coupled together into a hybrid HMJCDE algorithm. We use the proposed hybrid here.
JADE	JADE	[81]	2009	50; **100**; 200;	The first variant of JADE, which inspired all other versions discussed in this study. The version of JADE with an archive is used. In Ref. [81] various population size settings were set by authors for problems of different dimensionality, without a specified rule. *NP* was set to 100 for 30-dimensional problems. In recent years setting population size to 100 become the most frequent choice [55], hence we kept $NP = 100$ as default value here.
JADE-AEPD	DE with auto-enhanced population diversity	[77]	2015	50; **100**; 200;	Auto-enhanced population diversity (AEPD) has been implemented into a few DE variants. In Ref. [77] the population size of 20 is suggested for various variants with AEPD. However, from figure that show relation between performance of JADE-AEPD and population size (Fig. 3 in Ref. [77]) one finds that for JADE-AEPD variant the performance is better when *NP* is larger than 20. Hence we set the same default value of population size as for the majority of other tested JADE-based variants ($NP = 100$).
JADE-EIG	JADE with eigenvector-based crossover operation	[34]	2015	ceil($2.5 \cdot D$) within [250,15]; **$5 \cdot D$ within [500,30]**; $10 \cdot D$ within [1000,60];	In Ref. [34] an eigenvector-based crossover operator has been added into various DE algorithms, including JADE with the external archive. The population sizes has been suggested to depend on the problem dimensionality by $5 \cdot D$. We decided to keep this value as default one in our tests, as long as it is not outside the assumed bounds.
JADEEP	JADE with an Evolution Path	[43]	2015	50; **100**; 200;	In JADEEP algorithm operators based on evolution path are added, and some JADE elements are modified. The idea of tracking evolution path is inspired by CMA-ES [37] approach. The population size of JADEEP is not specified for most problems tested in Ref. [43], but population size of classical DE variants with evolution path added, which were also applied in Ref. [43], is set to 100 for most problems.
L-JADE	JADE with linear population size reduction	–	–	linearly decreasing from: $36 \cdot D$ to 4; **$18 \cdot D$ to 4**; $9 \cdot D$ to 4;	Unpublished JADE algorithm with linear population size reduction method implemented as in L-SHADE [66]. Linear population size reduction is the only difference between JADE [81] and L-JADE.
L-MPEDE [*]	MPEDE with linear population size reduction	–	–	linearly decreasing from: $36 \cdot D$ to 4; **$18 \cdot D$ to 4**; $9 \cdot D$ to 4;	Unpublished MPEDE algorithm with linear population size reduction method implemented as in L-SHADE [66]. The code has been obtained from dr Wu's web page (http://guohuawunudt.gotoip2.com/publications.html).
L-SHADE	SHADE with linear population size reduction	[66]	2014	linearly decreasing from: $36 \cdot D$ to 4;	The first paper that introduces linear population size mechanism to DE algorithm.

(*continued on next page*)

Table 1 (*continued*)

short name	long name	reference	year	three tested variants of population size when number of function calls is higher than 5000	comments
				18·D to 4; 9·D to 4;	
L-SHADE-cnEpSin [*]	Modified ensemble sinusoidal parameter adaptation incorporated into SHADE with linear population size reduction and eigenvector-based crossover	[7]	2017	linearly decreasing from: 36·D to 4; **18·D to 4**; 9·D to 4;	L-SHADE-*cn*EpSin, one of the best algorithms of IEEE CEC′2017 competition on single objective-optimization, is a modification of L-SHADE-EpSin [3] that has won IEEE CEC′2016 competition. L-SHADE-*cn*EpSin differes from L-SHADE-EpSin by making some control parameters of sinusoidal function adaptive, skipping local search procedure and adding eigenvector-based crossover operator. The code has been obtained from IEEE CEC′2017 organizer's web page (http://www.ntu.edu.sg/home/epnsugan/).
L-SHADE-EpSin-NLS [*]	Ensemble sinusoidal parameter adaptation incorporated into SHADE with linear population size reduction without local search	[3,57]	2016	linearly decreasing from: 36·D to 4; **18·D to 4**; 9·D to 4;	L-SHADE-EpSin is one of two joint winners of CEC′2016 competition. The code has been obtained from CEC′2016 organizer's web page (www3.ntu.edu.sg/home/epnsugan/index_files). L-SHADE-EpSin uses two sinusoidal functions for scale factor (F) parameter adaptation during the first half of the search, and classical SHADE-based control parameter adaptation mechanism during the second half of the search. L-SHADE-EpSin introduced in Ref. [2] has also incorporated a local search procedure that is implemented only once, just before the termination of the search. However, as shown in Ref. [57], the local search procedure of L-SHADE-EpSin is structurally biased (see Ref. [41] for the discussion on structural bias in population-based metaheuristics) and sample solutions located close to the origin of the coordinate system more frequently than those located in other parts of the search space, what affects the results obtained on some artificial benchmarks. Hence, to achieve a fair comparison against other variants, local search procedure is discarded from the version used in this study, called L-SHADE-EpSin-NLS (NLS means "no local search"), as discussed in details is [57]. Apart from the lack of local search procedure, L-SHADE-EpSin-NLS works exactly the same as L-SHADE-EpSin [3].
L-SHADE-SPACMA [*]	SHADE with linear population size reduction and semi-parameter adaptation hybridized with Covariance Matrix Adaptation – Evolutionary Strategy	[50]	2017	linearly decreasing from: 36·D to 4; **18·D to 4**; 9·D to 4;	The algorithm hybridizes L-SHADE [66] enhanced with modified control parameter adaptation mechanism with CMA-ES [37]. The adaptation mechanism of L-SHADE is divided into two different procedures, one of them is executed during the first, the other during the second part of the search. The code has been obtained from IEEE CEC′2017 organizer's web page (http://www.ntu.edu.sg/home/epnsugan/).
MDE-pBX	Modified DE with p-best crossover	[40]	2012	50; **100**; 200;	MDE_pBX is loosely based on JADE and introduces novel DE crossover and mutation operators.
MPADE	Adaptive DE algorithm using multiple sub-populations	[17]	2016	100; **200**; 400;	A complicated algorithm based on JADE-originated mutation strategy and control parameters adaptation method. MPADE uses three different mutation strategies. Each strategy is applied by a group of best, moderate or poorest individuals, and each benefit from information obtained from different sub-groups of individuals. Additional mechanism of selection is introduced, where some among the poorest individuals in the population that survive to the end of particular generation may be replaced with the best individuals that lost in the selection procedure during this generation.
MPEDE [*]	Multi-population ensemble DE	[76]	2016	125; **250**; 500;	MPEDE is a new self-adaptive DE variant that brings together the benefits from both adaptive and distributed DE methods. In the original paper [76] population size was set fixed to 250, what is justified by the need to manage sub-populations of uneven size. The code has been obtained from dr Wu's web page (http://guohuawunudt.gotoip2.com/publications.html).
Rcr-JADE	Crossover-rate repaired JADE	[31]	2014	50; **100**; 200;	The variant of JADE with archive [81] that modifies ("repairs") the adaptation rule used for the crossover parameter.
SapsDE	Differential Evolution algorithm with self-adaptive population resizing mechanism	[73]	**2013**	variable with only lower limit set to 25, initialized with max[0.5·D, 25]; **variable with only lower limit set to 50, initialized with max [1·D,50]**;	Self-adaptive DE variant, based on the concepts from JADE with archive. Contrary to the original JADE, population size is adaptively modified during run; in default case the lower limit of population size is set to 50 according to [73], but there is no upper limit. As there is no maximum population size in SapsDE, we decided to initialize algorithm with max [1·D,50] as default value. In cases with twice lower and

(*continued on next page*)

Table 1 (*continued*)

short name	long name	reference	year	three tested variants of population size when number of function calls is higher than 5000	comments
				variable with only lower limit set to 100, initialized with max[2·D, 100];	twice higher population sizes both initialization range and lower limit of the population size are respectively increased or decreased.
SHADE	Success-history based parameter adaptation for DE	[65]	2013	50; **100**; 200;	SHADE is based on JADE, with adding a memory to store successful control parameter values, and weighted mean to F and CR control parameter adaptation. The weights of weighted mean used during F and CR adaptation depend on the relative improvement achieved recently by individuals that used the particular pair of control parameter values to generate F and CR in that generation.
SPS-L-SHADE-EIG [*]	Success-history-based parameter adaptation DE (SHADE) with successful-parent-selecting framework, eigenvector-based crossover and linear population size reduction	[36]	2015	linearly decreasing from: 38·D to 4; **19·D to 4**; ceil(9.5·D) to 4;	A version of DE variant called SHADE [65] with linear population size reduction, eigenvector-based crossover and successful-parent-selecting framework. The winner of one of competitions held during CEC′2015 meeting. The code has been obtained from CEC′2015 organizer's web page (www3.ntu.edu.sg/home/epnsugan/index_files). The values of parameters defined in Ref. [36] as "default" are used. Note that the default initial population size is slightly larger than in case of other variants with linear population size reduction, as suggested in Ref. [36].
SPS-SHADE [*]	Successful-parent-selecting framework for SHADE	[35]	2015	50; **100**; 200;	Successful-parent-selecting framework has been introduced in Ref. [35] to a number of DE variants, including SHADE. Although in Ref. [35] the population size of all algorithms was related to problem dimensionality by $NP = 5 \cdot D$, we set default value of SPS-SHADE to 100, the value proposed for SHADE in Ref. [65], to facilitate comparison between SPS-SHADE and the initial variant of SHADE.

with each specified setting, as required in Ref. [45].

The selection of specific JADE/SHADE-based variants that are tested is obviously subjective and disputable. The chosen 22 algorithms are given in Table 1. We test three main variants: the first JADE [81], the first SHADE [65] and SHADE with the linear population size reduction L-SHADE [66]. We include two among the best performing L-SHADE-based algorithms from IEEE CEC′2017 competition, L-SHADE-cnEpSin [7] and L-SHADE-SPACMA [50]. Then we apply two other IEEE CEC winners: SPS-L-SHADE-EIG [36] and L-SHADE-EpSin [3]. However, to make the comparison fair, we use L-SHADE-EpSin variant with the structurally-biased local search procedure switched off, calling it L-SHADE-EpSin-NLS (for more details the reader is referred to [57]). We also test separately variants from which SPS-L-SHADE-EIG collected its operators, namely JADE-EIG [34] and SPS-SHADE [35]. Among recent SHADE modifications we select ETI-SHADE that use event-trigged mechanism [21]. From plentiful variants of JADE we test Rcr-JADE [31], because, as discussed in section 2, it become an inspiration for a number of other algorithms, two variants of JADE with different population size adaptation mechanisms (SapsDE and ATPS-JADE [73,84]) and variants that introduce versatile other modifications: AdapSS-JADE [28],

Table 2
Variants of IEEE CEC benchmark problems, numbers of allowed function calls and population sizes used. NFC – means default number of function calls (suggested in source paper of CEC problems); PS – means default number of population size (often suggested in the source paper of the algorithm, see Table 1).

benchmark set	number of problems	dimensionality	numbers of function calls	population size
CEC′2014	30	50 for each problem	Default NFC = 10 000·D = 500 000	PS - default, as suggested in the source paper PS / 2 - twice smaller than in the source paper PS · 2 - twice higher than in the source paper
			NFC / 5 = 2000·D = 100 000	PS - default, as suggested in the source paper PS / 2 – twice smaller than in the source paper PS · 2 - twice higher than in the source paper
			NFC · 5 = 50 000·D = 2 500 000	PS - default, as suggested in the source paper PS / 2 – twice smaller than in the source paper PS · 2 - twice higher than in the source paper
			5000	PS - default, as suggested in the source paper PS / 2 - twice smaller than in the source paper PS = 20 - equal to 20
CEC′2011	22	problem-dependent, from 1 to 216	Default NFC = 150 000	PS - default, as suggested in the source paper PS / 2 - twice smaller than in the source paper PS · 2 - twice higher than in the source paper
			NFC / 5 = 30 000	PS - default, as suggested in the source paper PS / 2 – twice smaller than in the source paper PS · 2 - twice higher than in the source paper
			NFC · 5 = 750 000	PS - default, as suggested in the source paper PS / 2 – twice smaller than in the source paper PS · 2 - twice higher than in the source paper
			5000	PS - default, as suggested in the source paper PS / 2 - twice smaller than in the source paper PS = 20 - equal to 20

HMJCDE [44], JADE-AEPD [77], JADEEP [43] and MPADE [17]. We also test two variants that are loosely related to JADE, namely MDE-pBX [40] and MPEDE [76]; they use some concepts from JADE, but within an independently developed algorithmic structure. Finally, as the linear population size reduction from L-SHADE become very popular in recent approaches, we also test versions of JADE and MPEDE with added linear population size reduction (L-JADE and L-MPEDE); in such case the default initial and final *PS* are set to $18 \cdot D$ and 4, respectively, as in L-SHADE [66]. Although, as far as we know, such variants were not proposed so far in any paper, they may help us answering the question to what extent the linear population size reduction alone affects the performance by comparing SHADE with L-SHADE, JADE with L-JADE and MPEDE with L-MPEDE.

The control parameter values suggested in the source papers are generally used for each variant, with a very few exceptions discussed in Table 1. Note that in any case the number of p% individuals cannot be lower than 1 (if larger minimum values of p% has been given in their source papers, those suggestions are retained in this study). Codes collected from the authors of respective procedure, or from official web page, are marked in Table 1 with [*]. In codes that we wrote oneself, reflection [38] has been used as bounds handling method. All algorithms were run in MATLAB.

Algorithms are compared separately for each among 24 test cases (two problem sets, four values of maximum number of function calls and three population sizes) according to their averaged rankings, obtained as follows. First, for each test case all 22 variants are ranked independently for every problem. The best function values found during each run by the specific algorithm with a particular maximum number of function calls and a particular population size are averaged over 51 runs (for raw results, see Suppl. Tables 1–48). Then, algorithms are ranked from the one with the lowest 51-run averaged objective function value (which is the best and receives rank 1), to the one with the highest 51-run averaged objective function value (rank 22 – the worst). As occasionally the results obtained by various variants may differ marginally, it is assumed that when the difference between the 51-run averaged objective function values achieved by some algorithms on specific problem is smaller than 10^{-10}, such algorithms receive equal rank (for example two algorithms may share rank 5.5, what means that none algorithm was ranked 5th or 6th). Finally, ranks achieved by each algorithm for the particular test set, number of function calls and population size are averaged over 30 (in case of CEC′2014) or 22 (in case of CEC′2011) problems. Such averaged rankings and, for simplicity of reading the results, the final place achieved by each algorithm in each ranking, are given in Tables 3–6.

The statistical significance of pair-wise comparisons among 22 JADE/SHADE-based variants is tested by means of the Friedman's test with the post-hoc Shaffer's static procedure [62] at $\alpha = 0.05$, as suggested in Refs. [27,69] for experiments where a fair comparison among a number of already existing methods is needed. The results are given in Suppl. Tables 49-50 for each among 24 considered comparisons. The respective codes of statistical tests were obtained from www.cmpe.boun.edu.tr/~ulas/m2test/details.php [69]. As many JADE/SHADE-based variants share large similarities, it is not surprising that frequently the differences between their results are not statistically significant, even if one method is constantly slightly better than the some others.

As for various problems and various numbers of function calls specific algorithms may perform better with different population sizes, we also show the inter-comparison among all 22 JADE/SHADE variants with three various population sizes together in Tables 7–9. In such a case we compare together 66 "variants" – 22 algorithms, each with three different population sizes. Tests with the number of function calls set to 5000 are given separately in Table 9, as they use different population size settings than tests made with three other numbers of function calls.

Finally, one may be interested in verifying the performance of JADE/SHADE-based algorithms against other types of metaheuristics. However, as all JADE/SHADE-based algorithms are relatively similar when compared with other kinds of methods, for such tests we selected three examples from 22 tested JADE/SHADE variants: one of the best, one of the poorest and one among those that perform moderately. Such three JADE/SHADE variants are compared against 12 metaheuristics discussed in Table 10, including three DE variants that have little in common with JADE or SHADE, three Particle Swarm Optimization (PSO) algorithms, two Genetic Algorithms (GA) and four algorithms from different families

Table 3
Ranking of 22 JADE-based algorithms averaged over all thirty 50-dimensional CEC′2014 problems for each considered variant of maximum number of function calls and population size setting. NFC – means classical value of maximum number of function calls (500 000); NFC / 5 – means five times lower value of maximum number of function calls (hence 100 000); NFC · 5 – means five times higher value of maximum number of function calls (hence 2 500 000); 5000 – means 5000 function calls; PS – means population size suggested by the authors of particular method; PS / 2 – means twice smaller population size than suggested by the authors of particular method; PS · 2 – means twice higher population size than suggested by the authors of particular method; 20 – means population size (in L-SHADE initial population size) set to 20.

	NFC PS	NFC PS / 2	NFC PS · 2	NFC / 5 PS	NFC / 5 PS / 2	NFC / 5 PS · 2	NFC · 5 PS	NFC · 5 PS / 2	NFC · 5 PS · 2	5000 PS	5000 PS / 2	5000 20
AdapSS-JADE	13.32	14.70	13.23	12.30	13.92	8.28	13.48	14.92	14.08	10.43	7.70	10.73
ATPS-DE	11.20	11.30	10.97	11.23	10.15	13.20	11.18	10.97	11.35	16.77	14.60	8.53
ETI-SHADE	7.62	8.83	14.47	15.37	10.35	16.90	8.55	8.98	7.73	14.63	12.77	10.87
HMJCDE	12.90	12.63	11.58	10.02	13.02	11.23	12.18	12.28	12.52	16.83	12.33	14.10
JADE	14.73	15.18	13.95	11.77	13.15	7.78	14.68	15.25	14.90	10.53	7.80	10.23
JADE-AEPD	13.15	10.77	13.78	11.50	13.15	8.27	11.72	9.03	13.35	11.33	8.07	10.90
JADE-EIG	12.30	10.98	12.32	13.23	12.55	15.67	11.02	10.32	11.67	18.27	16.40	13.30
JADEEP	13.48	16.07	13.52	11.73	14.02	8.17	15.38	17.18	14.67	7.57	7.23	12.83
L-JADE	11.15	9.28	11.70	13.30	11.68	13.90	10.37	9.42	10.73	12.00	15.80	7.43
L-MPEDE	10.58	8.50	10.85	14.73	11.03	17.27	9.13	8.13	9.70	13.43	18.37	12.40
L-SHADE	8.65	7.85	9.37	8.88	8.58	9.23	8.63	7.53	9.48	8.27	12.13	6.00
L-SHADE-cnEpSin	6.62	5.50	5.77	5.63	5.28	7.43	6.27	6.13	6.50	6.00	10.07	10.67
L-SHADE-EpSin-NLS	5.15	4.77	4.75	4.57	4.18	6.73	5.47	5.27	5.88	4.60	9.00	12.23
L-SHADE-SPACMA	5.22	5.55	4.98	6.80	4.52	8.87	5.90	6.40	5.37	9.90	13.50	12.40
MDE-pBX	17.92	19.12	15.88	14.83	17.98	9.62	17.15	19.02	16.02	4.70	5.07	15.23
MPADE	13.48	14.38	12.27	12.13	14.65	14.43	13.53	14.22	13.05	20.60	18.30	17.10
MPEDE	13.20	13.58	12.88	14.47	12.58	15.83	13.92	13.53	12.83	19.73	17.60	16.03
Rcr-JADE	13.85	15.10	13.08	11.80	13.40	7.75	14.90	15.40	13.85	9.43	5.83	9.67
SapsDE	17.07	16.93	15.15	13.27	16.35	9.82	16.13	15.68	16.90	3.20	4.20	13.67
SHADE	11.27	11.47	10.98	9.30	9.80	11.65	12.35	12.68	11.77	8.63	6.57	6.00
SPS-L-SHADE-EIG	6.15	5.17	8.13	12.23	9.13	15.43	5.53	4.80	6.10	11.87	16.93	10.40
SPS-SHADE	14.00	15.33	13.38	13.90	13.52	15.53	15.52	15.85	14.55	14.27	12.73	12.27

Table 4
Final place in each competition on 50-dimensional CEC'2014 problems based on averaged ranks from Table 3. NFC – means classical value of maximum number of function calls (500 000); NFC / 5 – means five times lower value of maximum number of function calls (hence 100 000); NFC · 5 – means five times higher value of maximum number of function calls (hence 2 500 000); 5000 – means 5000 function calls; PS – means population size suggested by the authors of particular method; PS / 2 – means twice smaller population size than suggested by the authors of particular method; PS · 2 – means twice higher population size than suggested by the authors of particular method; 20 – means population size (in L-SHADE initial population size) set to 20.

	NFC PS	NFC PS / 2	NFC PS · 2	NFC / 5 PS	NFC / 5 PS / 2	NFC / 5 PS · 2	NFC · 5 PS	NFC · 5 PS / 2	NFC · 5 PS · 2	5000 PS	5000 PS / 2	5000 20
AdapSS-JADE	15	16	15	14	18	7	14	16	17	10	6	9
ATPS-DE	9	11	7	7	7	14	10	11	9	18	16	4
ETI-SHADE	5	7	20	22	8	21	5	7	5	17	14	10
HMJCDE	12	13	9	6	13	12	12	12	12	19	12	19
JADE	20	18	19	10	14.5	4	17	17	20	11	7	6
JADE-AEPD	13	9	18	8	14.5	6	11	8	15	12	8	11
JADE-EIG	11	10	12	15	11	19	9	10	10	20	18	17
JADEEP	16.5	20	17	9	19	5	19	21	19	5	5	16
L-JADE	8	8	10	17	10	15	8	9	8	14	17	3
L-MPEDE	7	6	6	20	9	22	7	6	7	15	22	14.5
L-SHADE	6	5	5	4	4	9	6	5	6	6	11	1.5
L-SHADE-cnEpSin	4	3	3	2	3	2	4	3	4	4	10	8
L-SHADE-EpSin-NLS	1	1	1	1	1	1	1	2	2	2	9	12
L-SHADE-SPACMA	2	4	2	3	2	8	3	4	1	9	15	14.5
MDE-pBX	22	22	22	21	22	10	22	22	21	3	2	20
MPADE	16.5	15	11	12	20	16	15	15	14	22	21	22
MPEDE	14	14	13	19	12	20	16	14	13	21	20	21
Rcr-JADE	18	17	14	11	16	3	18	18	16	8	3	5
SapsDE	21	21	21	16	21	11	21	19	22	1	1	18
SHADE	10	12	8	5	6	13	13	13	11	7	4	1.5
SPS-L-SHADE-EIG	3	2	4	13	5	17	2	1	3	13	19	7
SPS-SHADE	19	19	16	18	17	18	20	20	18	16	13	13

Table 5
Ranking of 22 JADE-based algorithms averaged over all 22 real-world CEC'2011 problems (21 in case of NFC · 5 variants, as computationally demanding problem F3 for which each algorithm achieves almost equal performance is skipped) for each considered variant of maximum number of function calls and population size setting. NFC – means classical value of maximum number of function calls (150 000); NFC / 5 – means five times lower value of maximum number of function calls (hence 30 000); NFC · 5 – means five times higher value of maximum number of function calls (hence 750 000); 5000 – means 5000 function calls; PS – means population size suggested by the authors of particular method; PS / 2 – means twice smaller population size than suggested by the authors of particular method; PS · 2 – means twice higher population size than suggested by the authors of particular method; 20 – means population size (in L-SHADE initial population size) set to 20.

	NFC PS	NFC PS / 2	NFC PS · 2	NFC / 5 PS	NFC / 5 PS / 2	NFC / 5 PS · 2	NFC · 5 PS	NFC · 5 PS / 2	NFC · 5 PS · 2	5000 PS	5000 PS / 2	5000 20
AdapSS-JADE	14.68	15.27	12.55	11.32	10.84	12.23	16.50	15.18	15.75	13.07	11.27	13.18
ATPS-DE	10.91	12.59	11.77	13.41	13.66	12.45	13.00	12.68	10.68	12.41	12.64	11.50
ETI-SHADE	15.14	9.82	18.45	16.91	15.25	15.32	9.86	10.50	13.36	15.25	14.48	13.41
HMJCDE	12.18	9.82	13.77	15.50	13.07	14.59	10.36	10.59	10.50	18.11	16.20	15.86
JADE	12.91	14.68	11.82	9.86	10.34	10.64	14.36	14.73	13.84	11.52	9.95	12.32
JADE-AEPD	12.64	12.64	10.64	9.77	11.02	10.55	12.00	11.64	12.89	11.59	9.55	11.68
JADE-EIG	12.86	11.18	14.68	14.41	15.57	14.41	11.55	11.82	10.41	14.20	15.18	9.70
JADEEP	13.77	14.82	11.41	9.45	10.39	9.45	14.86	15.00	14.32	8.07	8.61	12.50
L-JADE	13.05	10.82	13.50	12.41	13.93	11.36	9.73	9.59	11.18	10.18	13.14	7.64
L-MPEDE	11.27	10.00	11.70	13.91	13.48	14.36	9.57	9.68	9.00	14.18	15.55	10.95
L-SHADE	8.59	9.09	9.68	9.86	10.16	9.59	7.18	7.05	7.57	6.68	8.77	5.82
L-SHADE-cnEpSin	7.05	8.00	6.82	8.36	8.52	8.09	7.55	7.55	8.09	6.52	8.52	11.52
L-SHADE-EpSin-NLS	7.64	7.95	7.18	8.18	8.52	8.45	8.59	8.18	8.18	6.32	7.77	11.30
L-SHADE-SPACMA	9.09	9.50	9.77	10.27	9.43	10.18	9.05	9.18	9.09	8.57	10.05	9.98
MDE-pBX	13.91	16.18	11.14	7.27	11.11	5.73	16.91	16.91	16.27	6.05	7.82	13.98
MPADE	7.18	8.14	8.41	14.95	12.80	17.50	8.86	8.86	8.57	19.84	18.43	17.52
MPEDE	10.09	9.05	13.05	15.00	11.52	16.68	9.09	9.14	9.52	18.84	17.66	12.23
Rcr-JADE	11.59	13.36	8.36	7.00	8.98	8.73	13.16	14.05	12.61	11.39	8.64	10.45
SapsDE	14.32	15.68	12.23	7.41	11.43	6.00	16.77	15.77	16.23	7.50	6.14	12.73
SHADE	9.09	11.45	11.39	11.73	7.25	12.73	11.18	12.59	11.66	10.34	9.95	6.77
SPS-L-SHADE-EIG	11.23	8.95	10.75	12.73	12.66	11.50	6.91	6.27	7.59	10.84	12.09	8.82
SPS-SHADE	13.82	14.00	13.93	13.27	13.07	12.45	15.95	16.05	15.68	11.52	10.59	13.14

of methods. The comparison between these 15 metaheuristics is performed in a similar way as the comparison among 22 JADE/SHADE-based variants discussed earlier, but tests are performed only with default values of function calls and population sizes. For inter-comparison among various kinds of metaheuristics when very different numbers of function calls are allowed the reader is referred to [56]. Results are given in Tables 11–14, averaged rankings and final positions of algorithms are summarized in Table 15, and statistical significance is shown in Tables 16 and 17.

4. Results

From Tables 3–6 (that show averaged ranking of 22 JADE/SHADE-based variants on each among 24 competitions), Tables 7–9 (in which variants with all three population size settings are compared together) and Suppl. Tables 1–48 (with detailed results obtained by each variant on every problem) one may find various conclusions regarding the practical usefulness of different JADE/SHADE-based methods. We organized our discussion in 10 sub-sections (4.1–4.10), each addressing a different

Table 6
Final place in each competition on CEC′2011 problems based on averaged ranks from Table 5. NFC – means classical value of maximum number of function calls (150 000); NFC / 5 – means five times lower value of maximum number of function calls (hence 30 000); NFC · 5 – means five times higher value of maximum number of function calls (hence 750 000); 5000 – means 5000 function calls; PS – means population size suggested by the authors of particular method; PS / 2 – means twice smaller population size than suggested by the authors of particular method; PS · 2 – means twice higher population size than suggested by the authors of particular method; 20 – means population size (in L-SHADE initial population size) set to 20.

	NFC PS	NFC PS / 2	NFC PS · 2	NFC / 5 PS	NFC / 5 PS / 2	NFC / 5 PS · 2	NFC · 5 PS	NFC · 5 PS / 2	NFC · 5 PS · 2	5000 PS	5000 PS / 2	5000 20
AdapSS-JADE	21	20	16	11	9	13	20	19	20	16	13	18
ATPS-DE	8	14	13	16	19	14.5	15	15	11	15	15	10
ETI-SHADE	22	8.5	22	22	21	20	10	10	16	19	17	19
HMJCDE	12	8.5	19	21	16.5	19	11	11	10	20	20	21
JADE	15	18	14	8.5	7	10	17	17	17	12.5	9.5	14
JADE-AEPD	13	15	7	7	10	9	14	12	15	14	8	12
JADE-EIG	14	12	21	18	22	18	13	13	9	18	18	5
JADEEP	17	19	11	6	8	6	18	18	18	6	5	15
L-JADE	16	11	18	13	20	11	9	8	12	8	16	3
L-MPEDE	10	10	12	17	18	17	8	9	6	17	19	8
L-SHADE	4	6	5	8.5	6	7	2	2	1	4	7	1
L-SHADE-cnEpSin	1	2	1	5	2.5	3	3	3	3	3	4	11
L-SHADE-EpSin-NLS	3	1	2	4	2.5	4	4	4	4	2	2	9
L-SHADE-SPACMA	5.5	7	6	10	5	8	6	7	7	7	11	6
MDE-pBX	19	22	9	2	11	1	22	22	22	1	3	20
MPADE	2	3	4	19	15	22	5	5	5	22	22	22
MPEDE	7	5	17	20	13	21	7	6	8	21	21	13
Rcr-JADE	11	16	3	1	4	5	16	16	14	11	6	7
SapsDE	20	21	15	3	12	2	21	20	21	5	1	16
SHADE	5.5	13	10	12	1	16	12	14	13	9	9.5	2
SPS-L-SHADE-EIG	9	4	8	14	14	12	1	1	2	10	14	4
SPS-SHADE	18	17	20	15	16.5	14.5	19	21	19	12.5	12	17

topic. In most sub-sections we first refer to the rankings shown in Tables 3–6 for each competition separately, and later refer to results based on comparison among variants with all three population size settings considered together (Tables 7–9). Finally, the sub-section 4.11 is devoted to the comparison between JADE/SHADE-based variants and other types of metaheuristics; the results of such comparison are given in Tables 11–17.

4.1. Diversification of rankings of JADE/SHADE-based algorithms averaged over artificial benchmarks and real-world problems

From Tables 3 and 5 we find that higher differences in averaged rankings are observed for the 50-dimensional CEC′2014 problems than for the real-world CEC′2011 ones. This suggest that CEC′2014 problems are more selective than CEC′2011 ones, what is supported by the statistical comparison (see Suppl. Tables 49-50). Although one-by-one differences between most pairs of algorithms are not statistically significant at $\alpha = 0.05$, more frequently such differences are statistically significant in the case of CEC′2014 than in the case of CEC′2011 problems. Statistically significant differences between the performances of pairs of algorithms are observed most frequently for tests on CEC′2014 problems with: 1) the classical number of function calls but the reduced population size (NFC, *PS*/2); 2) the reduced number of function calls and increased population size (NFC/5, *PS*·2); 3) the increased number of function calls and the reduced population size (NFC·5, *PS*/2); 4. 5000 function calls and the population size set higher than 20. Such cases represent settings under which various algorithms have rarely been tested so far, hence they may be considered independent from conditions for which authors tuned their methods. Overall, few among JADE/SHADE-based algorithms perform statistically significantly better (at $\alpha = 0.05$) than competitors on the real-world problems. Hence, fitting modified variant to the artificial benchmarks is simpler than to the various real-world problems.

Tables 7–9, in which variants with all three population sizes are compared together, confirm the observations discussed above, with one exception – when the number of function calls is set to 5000, averaged ranking is highly diversified among algorithms with different population sizes for both CEC′2014 and CEC′2011 problems (see Table 9).

4.2. Searching for the best JADE/SHADE variants when computational budget is not very low

In this sub-section we exclude tests performed with the number of function calls set to 5000.

From Table 4 we find that L-SHADE-EpSin-NLS (the winner of CEC′2016 with the local search procedure skipped, see Refs. [3] and [57]) is, according to average ranking, in each among 9 tests performed on 50-dimensional CEC′2014 problems either the winner or the second best method. In the competitions performed on CEC′2011 problems L-SHADE-EpSin-NLS is always among the best four methods. None other JADE/SHADE-based variant tested is ranked so high under all competition settings.

Other algorithms that are well ranked for all settings are L-SHADE-cnEpSin (worst ranking – 5th) and L-SHADE (worst ranking – 9th). L-SHADE-SPACMA is better ranked on CEC′2014 problems (worst ranking – 8th; best ranking – 1st), than on CEC′2011 ones (worst ranking – 10th; best ranking – 5th). SPS-L-SHADE-EIG performs similarly well for CEC′2014 and CEC′2011 problems, winning three out of 18 competitions, and is ranked the second best for three others. However, for some settings of the population size and function calls SPS-L-SHADE-EIG perform poorly. Interestingly, all these algorithms (L-SHADE-EpSin-NLS, L-SHADE-cnEpSin, L-SHADE, L-SHADE-SPACMA and SPS-L-SHADE-EIG) were introduced and performed highly in recent IEEE CEC competitions, all use the linear population size reduction mechanisms, none of them has been introduced in a journal paper (considering that L-SHADE-NLS [57] is just a simplification of L-SHADE-EpSin [3]) and all were created by adding some modifications step-by-step to the previous algorithms (an approach that is sometimes criticized as adding stuff [39]). However, we must also note that results achieved on CEC′2014 and CEC′2011 may seriously differ in pointing out at specific best approach. For example, L-SHADE-SPACMA performs much better on CEC′2014 artificial benchmarks; on the other hand MPADE, although never better than 11th on CEC′2014 problems, is actually the second best choice for CEC′2011 problems with standard NFC.

These findings are fully confirmed by the results given in Tables 7 and 8

Table 7
Averaged ranking of 22 JADE-based algorithms tested together with three population sizes. The ranking is averaged over all 50-dimensional CEC'2014 problems for variants with number of function calls set to NFC, NFC / 5 and NFC · 5. Variant with number of function calls set to 5000 is included in Table 9. NFC – means classical value of maximum number of function calls (150 000); NFC / 5 – means five times lower value of maximum number of function calls (hence 30 000); NFC · 5 – means five times higher value of maximum number of function calls (hence 750 000); PS – means population size suggested by the authors of particular method; PS / 2 – means twice smaller population size than suggested by the authors of particular method; PS · 2 – means twice higher population size than suggested by the authors of particular method.

algorithm	population size	averaged ranking			final position		
		NFC	NFC / 5	NFC · 5	NFC	NFC / 5	NFC · 5
AdapSS-JADE	PS	36.13	34.10	38.20	37	33	43
ATPS-DE	PS	31.35	30.73	31.25	22	21	27
ETI-SHADE	PS	20.87	41.50	23.83	12	55	13
HMJCDE	PS	36.18	28.22	34.63	38	16	35
JADE	PS	39.72	31.40	41.03	48	24	49
JADE-AEPD	PS	35.55	31.00	32.87	35	22.5	33
JADE-EIG	PS	34.98	35.37	31.47	32.5	40	28
JADEEP	PS	38.40	32.53	43.17	44	28	52
L-JADE	PS	29.40	34.23	29.03	19	35	22
L-MPEDE	PS	27.28	37.87	24.68	18	46	18
L-SHADE	PS	24.47	25.85	24.57	14	13	17
L-SHADE-cnEpSin	PS	16.60	15.47	17.12	8	5	8
L-SHADE-EpSin-NLS	PS	13.85	13.70	15.65	2	4	5
L-SHADE-SPACMA	PS	13.30	19.10	16.50	1	8	7
MDE-pBX	PS	50.25	39.50	49.28	65	50.5	60
MPADE	PS	37.58	32.90	37.92	42	32	41
MPEDE	PS	34.98	36.30	38.33	32.5	45	44
Rcr-JADE	PS	37.20	31.00	40.90	40	22.5	48
SapsDE	PS	46.78	38.30	45.18	63	47.5	57
SHADE	PS	31.62	26.43	35.67	23	14	38
SPS-L-SHADE-EIG	PS	15.72	32.70	15.07	5	29	4
SPS-SHADE	PS	38.82	38.50	44.58	46	49	54
AdapSS-JADE	PS / 2	43.90	34.57	49.78	55	38	62.5
ATPS-DE	PS / 2	36.07	24.77	38.55	36	11	45
ETI-SHADE	PS / 2	30.17	24.90	34.03	20	12	34
HMJCDE	PS / 2	41.77	32.73	42.62	54	30.5	51
JADE	PS / 2	44.88	32.73	49.10	58	30.5	59
JADE-AEPD	PS / 2	33.63	31.97	32.42	27	26	31
JADE-EIG	PS / 2	34.18	30.47	36.70	31	20	40
JADEEP	PS / 2	46.43	35.57	53.77	62	42	65
L-JADE	PS / 2	26.73	23.23	30.65	17	9	25
L-MPEDE	PS / 2	26.38	23.77	29.98	16	10	23
L-SHADE	PS / 2	22.37	16.67	25.68	13	6	19
L-SHADE-cnEpSin	PS / 2	17.43	8.47	21.53	9	3	12
L-SHADE-EpSin-NLS	PS / 2	16.17	7.27	19.70	6	1	11
L-SHADE-SPACMA	PS / 2	17.55	7.33	23.93	10	2	14
MDE-pBX	PS / 2	58.32	44.27	58.58	66	57	66
MPADE	PS / 2	46.03	39.50	47.78	61	50.5	58
MPEDE	PS / 2	41.62	29.67	44.80	53	17	56
Rcr-JADE	PS / 2	44.25	32.52	49.30	57	27	61
SapsDE	PS / 2	50.10	40.60	49.78	64	53	62.5
SHADE	PS / 2	37.48	26.85	43.97	41	15	53
SPS-L-SHADE-EIG	PS / 2	14.18	18.20	17.90	3	7	9
SPS-SHADE	PS / 2	45.63	34.27	50.65	60	36	64
AdapSS-JADE	PS · 2	38.60	35.93	36.43	45	44	39
ATPS-DE	PS · 2	34.02	47.07	28.90	30	58	21
ETI-SHADE	PS · 2	41.60	53.83	17.93	52	64	10
HMJCDE	PS · 2	32.30	42.47	32.12	24	56	30
JADE	PS · 2	40.63	34.20	39.22	50	34	47
JADE-AEPD	PS · 2	40.45	35.73	34.67	49	43	36
JADE-EIG	PS · 2	33.72	49.67	30.07	28	61	24
JADEEP	PS · 2	41.10	34.63	38.90	51	39	46
L-JADE	PS · 2	36.67	48.67	28.80	39	60	20
L-MPEDE	PS · 2	32.52	55.03	24.28	25	66	15
L-SHADE	PS · 2	30.38	38.30	24.42	21	47.5	16
L-SHADE-cnEpSin	PS · 2	18.43	29.90	15.90	11	18	6
L-SHADE-EpSin-NLS	PS · 2	16.35	30.13	14.82	7	19	3
L-SHADE-SPACMA	PS · 2	15.67	34.47	13.27	4	37	1
MDE-pBX	PS · 2	44.20	35.53	41.18	56	41	50
MPADE	PS · 2	32.65	47.77	32.73	26	59	32
MPEDE	PS · 2	35.13	51.83	31.60	34	63	29
Rcr-JADE	PS · 2	37.88	31.53	35.50	43	25	37
SapsDE	PS · 2	44.92	40.33	44.75	59	52	55
SHADE	PS · 2	33.75	41.43	31.08	29	54	26
SPS-L-SHADE-EIG	PS · 2	24.70	51.20	14.17	15	62	2
SPS-SHADE	PS · 2	39.02	54.33	38.12	47	65	42

Table 8
Averaged ranking of 22 JADE-based algorithms tested together with three population sizes. The ranking is averaged over all CEC'2011 problems for variants with number of function calls set to NFC, NFC / 5 and NFC · 5 (in case of NFC · 5, problem F3 is skipped). Variant with number of function calls set to 5000 is included in Table 9. NFC – means classical value of maximum number of function calls (150 000); NFC / 5 – means five times lower value of maximum number of function calls (hence 30 000); NFC · 5 – means five times higher value of maximum number of function calls (hence 750 000); PS – means population size suggested by the authors of particular method; PS / 2 – means twice smaller population size than suggested by the authors of particular method; PS · 2 – means twice higher population size than suggested by the authors of particular method.

algorithm	population size	averaged ranking			final position		
		NFC	NFC / 5	NFC · 5	NFC	NFC / 5	NFC · 5
AdapSS-JADE	PS	38.41	30.98	45.50	47	32	56
ATPS-DE	PS	29.77	36.00	36.91	24	38	41
ETI-SHADE	PS	41.55	44.89	27.82	58	53	24
HMJCDE	PS	32.55	40.93	29.36	28	48	26
JADE	PS	34.18	28.11	41.05	34	26	49
JADE-AEPD	PS	33.32	28.52	35.18	32	27	39
JADE-EIG	PS	34.82	38.66	31.23	36	43	31
JADEEP	PS	36.86	26.93	41.32	43	22	50
L-JADE	PS	36.77	35.80	27.64	42	37	23
L-MPEDE	PS	30.64	39.30	26.52	26	45	20
L-SHADE	PS	24.32	29.07	19.45	8	30	3
L-SHADE-cnEpSin	PS	18.95	24.80	20.86	1	18	7
L-SHADE-EpSin-NLS	PS	21.00	24.52	23.36	3	17	10
L-SHADE-SPACMA	PS	25.32	29.20	25.36	12	31	15
MDE-pBX	PS	37.95	19.98	47.91	46	7.5	62
MPADE	PS	19.73	39.25	26.18	2	44	17.5
MPEDE	PS	26.05	41.30	26.45	15	49	19
Rcr-JADE	PS	29.09	19.02	36.93	21	5	42
SapsDE	PS	39.32	22.80	46.36	51	15	58
SHADE	PS	24.68	31.25	31.91	9	33	34
SPS-L-SHADE-EIG	PS	30.27	36.48	19.27	25	41	1
SPS-SHADE	PS	38.91	36.34	46.50	49	40	59
AdapSS-JADE	PS / 2	42.32	21.16	48.55	60	12	63
ATPS-DE	PS / 2	36.18	26.66	43.27	39	21	54
ETI-SHADE	PS / 2	28.55	31.75	37.91	19	35	45
HMJCDE	PS / 2	29.45	26.07	37.36	23	19	44
JADE	PS / 2	41.00	19.84	47.77	56	6	61
JADE-AEPD	PS / 2	35.32	20.57	38.18	38	11	46
JADE-EIG	PS / 2	29.14	31.66	36.32	22	34	40
JADEEP	PS / 2	41.36	19.98	47.64	57	7.5	60
L-JADE	PS / 2	31.23	28.89	30.25	27	28.5	29
L-MPEDE	PS / 2	28.82	27.57	31.50	20	25	32
L-SHADE	PS / 2	23.64	20.25	23.45	7	10	11
L-SHADE-cnEpSin	PS / 2	21.41	17.89	24.86	4	3	14
L-SHADE-EpSin-NLS	PS / 2	21.55	18.34	26.91	5	4	21
L-SHADE-SPACMA	PS / 2	25.36	20.11	29.55	13	9	27
MDE-pBX	PS / 2	45.64	21.98	52.91	64	13	66
MPADE	PS / 2	23.36	27.48	31.73	6	24	33
MPEDE	PS / 2	27.09	22.48	32.68	17	14	36
Rcr-JADE	PS / 2	37.14	16.75	45.91	45	2	57
SapsDE	PS / 2	42.09	23.89	49.77	59	16	64
SHADE	PS / 2	33.05	13.89	41.77	30	1	51
SPS-L-SHADE-EIG	PS / 2	24.77	27.39	20.41	10	23	6
SPS-SHADE	PS / 2	39.95	26.48	51.14	52	20	65
AdapSS-JADE	PS · 2	40.05	48.34	40.98	53	57	48
ATPS-DE	PS · 2	38.59	49.80	28.50	48	60	25
ETI-SHADE	PS · 2	55.05	53.02	34.32	66	64	37
HMJCDE	PS · 2	43.32	52.98	26.18	61	63	17.5
JADE	PS · 2	36.64	45.84	37.34	40	55	43
JADE-AEPD	PS · 2	34.23	46.02	35.16	35	56	38
JADE-EIG	PS · 2	46.09	48.89	27.36	65	59	22
JADEEP	PS · 2	36.73	43.75	38.82	41	52	47
L-JADE	PS · 2	45.23	45.16	30.86	62	54	30
L-MPEDE	PS · 2	39.25	49.98	23.55	50	61	12
L-SHADE	PS · 2	32.95	41.48	20.34	29	50	5
L-SHADE-cnEpSin	PS · 2	25.23	36.16	20.27	11	39	4
L-SHADE-EpSin-NLS	PS · 2	25.73	37.75	21.16	14	42	8
L-SHADE-SPACMA	PS · 2	33.14	40.80	24.64	31	47	13
MDE-pBX	PS · 2	33.41	28.89	42.41	33	28.5	53
MPADE	PS · 2	27.77	57.25	22.59	18	66	9
MPEDE	PS · 2	40.14	55.07	25.55	54	65	16
Rcr-JADE	PS · 2	26.50	40.20	31.98	16	46	35
SapsDE	PS · 2	40.68	32.68	44.64	55	36	55
SHADE	PS · 2	36.93	48.75	30.16	44	58	28
SPS-L-SHADE-EIG	PS · 2	34.93	42.93	19.41	37	51	2
SPS-SHADE	PS · 2	45.57	50.11	41.86	63	62	52

Table 9
Averaged ranking of 22 JADE-based algorithms tested together with three population sizes when number of function calls is set to 5000. The ranking is averaged separately over all CEC′2014 and over all CEC′2011 problems. PS – means population size suggested by the authors of particular method; PS / 2 – means twice smaller population size than suggested by the authors of particular method; 20 – means that population size (initial population size in L-SHADE) is set to 20.

algorithm	population size	CEC′2014		CEC′2011	
		averaged ranking	final position	averaged ranking	final position
		5000	5000	5000	5000
AdapSS-JADE	PS	46.70	49	50.48	58
ATPS-DE	PS	58.73	63	48.95	56
ETI-SHADE	PS	55.20	61	54.48	63
HMJCDE	PS	57.30	62	58.75	64
JADE	PS	47.23	50	48.02	53
JADE-AEPD	PS	48.23	52	48.48	55
JADE-EIG	PS	61.10	64	51.07	59.5
JADEEP	PS	41.87	42	43.30	48
L-JADE	PS	49.53	54	44.91	49
L-MPEDE	PS	53.50	60	51.48	61
L-SHADE	PS	42.23	43	38.36	42
L-SHADE-cnEpSin	PS	36.83	36	37.25	39
L-SHADE-EpSin-NLS	PS	34.40	35	36.00	37
L-SHADE-SPACMA	PS	45.93	47	39.57	43
MDE-pBX	PS	31.47	33	33.86	34
MPADE	PS	62.90	66	60.66	66
MPEDE	PS	62.50	65	59.52	65
Rcr-JADE	PS	45.17	46	48.07	54
SapsDE	PS	30.07	31	40.64	44
SHADE	PS	43.67	44	46.75	52
SPS-L-SHADE-EIG	PS	49.97	55	45.84	51
SPS-SHADE	PS	52.27	59	49.11	57
AdapSS-JADE	PS / 2	27.40	28	34.45	36
ATPS-DE	PS / 2	44.20	45	36.50	38
ETI-SHADE	PS / 2	38.30	38	41.11	46
HMJCDE	PS / 2	40.03	40	45.57	50
JADE	PS / 2	27.43	29	32.09	32
JADE-AEPD	PS / 2	28.47	30	31.00	30.5
JADE-EIG	PS / 2	49.27	53	41.02	45
JADEEP	PS / 2	26.50	26	30.73	29
L-JADE	PS / 2	46.23	48	37.41	41
L-MPEDE	PS / 2	51.97	58	43.23	47
L-SHADE	PS / 2	38.03	37	30.00	27
L-SHADE-cnEpSin	PS / 2	33.23	34	28.25	25
L-SHADE-EpSin-NLS	PS / 2	31.20	32	27.09	24
L-SHADE-SPACMA	PS / 2	41.83	41	31.00	30.5
MDE-pBX	PS / 2	18.67	19.5	25.23	23
MPADE	PS / 2	51.57	57	51.57	62
MPEDE	PS / 2	50.50	56	51.07	59.5
Rcr-JADE	PS / 2	23.23	22	28.32	26
SapsDE	PS / 2	19.73	21	25.18	22
SHADE	PS / 2	24.80	25	32.25	33
SPS-L-SHADE-EIG	PS / 2	48.20	51	37.32	40
SPS-SHADE	PS / 2	39.90	39	34.36	35
AdapSS-JADE	20	13.70	7	20.39	18
ATPS-DE	20	10.80	4	18.00	10
ETI-SHADE	20	13.83	8	20.41	19
HMJCDE	20	18.00	15	22.91	20
JADE	20	13.23	6	18.50	12
JADE-AEPD	20	14.03	9	17.09	8
JADE-EIG	20	18.10	16	14.43	5
JADEEP	20	17.50	14	18.77	13
L-JADE	20	10.47	3	13.32	3
L-MPEDE	20	17.40	13	19.57	17
L-SHADE	20	8.27	2	10.32	1
L-SHADE-cnEpSin	20	16.27	11	18.39	11
L-SHADE-EpSin-NLS	20	18.67	19.5	17.98	9
L-SHADE-SPACMA	20	18.13	17	16.91	7
MDE-pBX	20	23.47	23	25.07	21
MPADE	20	27.13	27	30.66	28
MPEDE	20	24.03	24	19.11	14
Rcr-JADE	20	12.93	5	15.64	6
SapsDE	20	18.23	18	19.34	15
SHADE	20	8.07	1	11.05	2
SPS-L-SHADE-EIG	20	14.47	10	13.45	4
SPS-SHADE	20	16.77	12	19.41	16

4.3. Searching for the best JADE/SHADE variants when only 5000 function calls are available

Tests performed with just 5000 function calls lead to much different ranking of algorithms. However, such rankings highly depend on the problem set and, more importantly, population size.

If we run algorithms with single specific population size settings (*PS*, *PS*/2 or 20, see Tables 3–6) SapsDE, MDE-pBX, SHADE, L-SHADE, L-SHADE-EpSin-NLS and L-SHADE-cnEpSin seems to perform well. L-SHADE-EpSin-NLS, which was praised when computational budgets were higher, is the 2nd best method for both CEC′2014 and CEC′2011 problems when the classical population size (*PS*) is used, but deteriorates (even to 12th place) when smaller population sizes are set. SapsDE is the winner for three out of six competitions, but performs poorly when the population size is set to 20. MDE-pBX is among the three best methods when the population size is higher than 20, but otherwise performs poorly.

However, from Table 9 we find that when 5000 function calls are allowed, almost always the best performance is achieved by specific algorithm when the population size is set to 20, hence the very good rankings obtained in the comparisons with higher population sizes are of minor importance. According to Table 9, SHADE and L-SHADE with the population size set to 20 are clearly the best choices, much better than all their recently modified variants. It seems that the modifications proposed in recent CEC competitions are generally oriented at exploration and slow down exploitation, hence are not suitable when the time budget is very low.

4.4. Searching for the worst JADE/SHADE variants

This is a difficult task. None method may be classified among the poorest in every competition. MDE-pBX and SapsDE are in almost every competition held on CEC′2014 problems among the poorest, and frequently perform poorly also on CEC′2011 problems. However, in some specific competitions they are ranked the 1st or the 2nd best methods. On the other hand, considering all 24 competitions, SPS-SHADE is at best ranked 12th, even though it is never the worst variant. It is also interesting that, although 24 competitions were performed, half of tested variants are never among the best three algorithms. Such list of 11 variants that never perform good enough include AdapSS-JADE, ATPS-DE, ETI-SHADE, HMJCDE, JADE, JADE-AEPD, JADE-EIG, JADEEP, L-MPEDE, MPEDE and SPS-SHADE. As one may note, MDE-pBX and SapsDE that were discussed above are not on this list of “never good” methods.

Tables 7–9 confirm that MDE-pBX, SapsDE and SPS-SHADE, irrespectively of the population size setting, perform poorly on CEC′2014 problems. However, MDE-pBX may be useful for real-world problems when the number of function calls is relatively low (set to NFC / 5). SapsDE and SPS-SHADE may hardly be recommended for any competition settings considered in this study.

4.5. Importance of the linear population size reduction

Overall, best JADE/SHADE-based algorithms use the linear population size reduction. From Table 4 we also find that L-JADE and L-MPEDE are better than JADE and MPEDE in 8 and 9 out of 11 competitions held on CEC′2014 problems, respectively. JADE and MPEDE may be better than their variants with the linear population size reduction only when low computational budgets are considered (NFC / 5 or 5000). On CEC′2014 problems L-SHADE and SPS-L-SHADE-EIG are always better than SHADE and SPS-SHADE when tests with 5000 function calls are excluded, although in the case of SPS-L-SHADE-EIG this may be due to eigenvector-based crossover. Such results seems to confirm the usefulness of the linear population size reduction. However, it has been first proposed for SHADE-based algorithm during CEC′2014 competition [66], and was since then almost always successfully used within

Table 10
Twelve algorithms tested against selected JADE-based methods (L-SHADE-EpSin, SapsDE and JADE). Abbreviations: DE – Differential Evolution; PSO – Particle Swarm Optimization; GA – Genetic Algorithm; D – problem dimensionality. [*] – marks algorithms which MATLAB codes were obtained from authors of the source papers or relevant web pages (see Comments).

short name	description	reference	year	population size	comments
AMALGAM [*]	A multialgorithm genetically adaptive method for single objective optimization	[71]	2009	variable during run, for CEC′2011 initialized with 10 when $D < 20$; 15 when $D < 40$; 20 when $D \geq 40$; for CEC′2014 initialized with 20	AMALGAM variant that merges CMA-ES [37], GA and PSO [22] is used, as suggested and described in Ref. [71]. AMALGAM makes a number of sub-runs within time budget; for the conditions of commencing sub-run, see Ref. [71]. In each consecutive sub-run the population is twice larger, until it become 32 times larger than in the first sub-run. The MATLAB code of AMALGAM has been obtained from prof. Vrugt, first author of [71].
cNrGA [*]	Non-revisiting GA with adaptive mutation using constant memory	[47]	2016	100	A variant of GA that remembers most of its history to avoid re-evaluations of already sampled solutions. The code of cNrGA has been obtained from author's web page www.ee.cityu.edu.hk/~syyuen/Public/Code.html.
CoBiDE	Differential Evolution based on covariance matrix learning and bimodal distribution parameter setting	[74]	2014	60	One of two first DE variants (the second has been introduced in Ref. [34]) that pay attention to the problem of coordinate system rotation.
GA-MPC [*]	Genetic algorithm with multi-parent crossover	[24]	2011	90	GA that was the winner of CEC′2011 competition. The MATLAB code of GA-MPC has been obtained from CEC′2011 web page (www3.ntu.edu.sg/home/epnsugan/index_files/CEC′11-RWP/CEC′11-RWP.htm). One change to the original code has been made – if objective function is "not a number" (what may happen in a few CEC′2011 problems even if solutions within a bounds are sampled) in the original code 0 had been set; in the modified code a very large number (10^{100}) is set in such a case.
GLPSO	Genetic Learning PSO	[33]	2016	50	A recent PSO [22] variant that mixes various elements from PSO and GA. Velocities of particles are initialized within 20% of bounds span, and are restricted within this range during run.
HCLPSO [*]	Heterogeneous comprehensive learning PSO	[49]	2015	40	Relatively recent PSO variant. The maximum velocities are set to 20% of the bounds span. The initial velocities are generated randomly form uniform distribution within the allowed margin. The code has been obtained from one of co-authors of [49] paper.
IDE [*]	DE with an Individual-independent mechanism	[68]	2015	for CEC′2011 depends on D: 50 if $D \leq 10$, 200 if $D > 50$, linear approximation within [50,200] interval if $10 < D \leq 50$; for CEC′2014 set to 200	A novel variant of DE that introduces concepts of individual-dependent setting of control parameters and mutation strategies that are based on function values found by particular individuals. The code has been obtained from one of co-authors of [68] paper.
IILPSO	PSO with Interswarm Interactive Learning Strategy	[59]	2016	50	A multi-swarm PSO variant with specific scheme of information sharing between swarms. The maximum velocities are set to 12.5% of the bounds span. The initial velocities are generated randomly form uniform distribution within the allowed margin.
jDElscop	Self-adaptive DE algorithm using population size reduction and three strategies	[10]	2011	variable during run, for CEC′2011 initialized with min [$10 \cdot D$,500]; for CEC′2014 initialized with 500	Self-adaptive DE variant with the population diminishing strategy used during the search. During run the population size of jDElscop is diminished by half three times, after 25%, 50% and 75% allowed function calls is used. However, in this study the population size is not diminished if after that it would be lower than 5.
LBBO [*]	Linearized Biogeography-based Optimization	[63]	2014	50	This Biogeography-based optimization variant uses local search technique and is able to re-initialize during run. The code of LBBO has been obtained from Prof. Simon's web page (http://academic.csuohio.edu/simond/bbo/linearized/). A change to Systematic Search procedure has been made, to prevent it from using too many function calls during run (in the original code in this procedure it was not checked whether the number of allowed function calls has been exceeded or not).
PMS	Parallel Memetic Structures	[15]	2013	1	PMS is non-population based heuristic, partly based on RA and some DE operators. Similarly to the approach proposed in Ref. [39], its construction is based on Ockham's razor principle, that is so frequently violated in modern metaheuristics.
SP-UCI [*]	Shuffled complex evolution with principal components analysis – University of California at Irvine	[16]	2011	$4 \cdot (2 \cdot D+1)$	The variant of SP-UCI with 4 simplexes is used. The MATLAB code of SP-UCI has been obtained from Dr. Chu, first author of [16].

Table 11
51-run mean performances of 15 various algorithms on 50-dimensional CEC′2014 problems with maximum number of function calls set to default value (500 000).

	L-SHADE-EpSin-NLC	SapsDE	JADE	AMALGAM	cNrGA	CoBiDE	GA-MPC	GLPSO
F1	8.12E-06	7.59E+04	1.36E+04	1.45E-13	8.74E+06	2.80E+05	4.00E+05	5.94E+05
F2	3.29E-14	2.84E-14	1.16E-13	8.69E-14	3.02E+05	1.30E-01	4.65E+03	7.41E+03
F3	5.68E-14	1.39E-01	3.84E+03	8.92E-14	3.16E+04	4.62E-03	1.55E+01	2.17E+03
F4	6.24E+01	4.93E+01	1.57E+01	5.40E+00	1.07E+02	3.37E+01	5.68E+01	7.22E+01
F5	2.02E+01	2.04E+01	2.04E+01	2.01E+01	2.00E+01	2.03E+01	2.03E+01	2.09E+01
F6	1.04E-02	9.18E+00	1.68E+01	7.30E-02	2.43E+01	8.16E+00	4.89E+00	2.15E+01
F7	9.81E-14	5.51E-03	3.00E-03	1.40E-13	1.91E-01	4.46E-15	3.00E-03	1.14E-02
F8	4.43E-10	1.95E-02	1.14E-13	4.89E-02	7.91E-01	2.01E-14	5.05E+01	1.40E-13
F9	2.94E+01	7.19E+01	5.29E+01	4.46E+00	8.03E+01	8.86E+01	5.47E+01	1.07E+02
F10	2.95E-02	7.07E+00	8.08E-03	1.66E+01	5.94E-01	1.31E+01	1.13E+03	1.59E+00
F11	3.10E+03	3.95E+03	3.84E+03	4.16E+02	4.39E+03	3.77E+03	3.58E+03	4.97E+03
F12	2.02E-01	3.65E-01	2.65E-01	6.59E-02	1.38E-01	3.07E-01	7.91E-02	1.29E-01
F13	2.03E-01	3.97E-01	3.23E-01	1.57E-01	3.52E-01	3.63E-01	4.25E-01	4.49E-01
F14	1.96E-01	3.33E-01	3.01E-01	3.01E-01	5.31E-01	2.78E-01	3.73E-01	3.76E-01
F15	5.37E+00	1.22E+01	7.18E+00	4.80E+00	2.27E+01	6.17E+00	5.52E+00	1.36E+01
F16	1.66E+01	1.79E+01	1.78E+01	1.79E+01	1.82E+01	1.78E+01	1.72E+01	1.77E+01
F17	3.63E+02	2.39E+03	2.46E+03	2.27E+03	3.59E+06	1.09E+04	2.91E+04	9.64E+04
F18	1.75E+01	2.87E+02	1.87E+02	1.51E+02	1.55E+03	6.85E+01	3.23E+02	1.39E+03
F19	9.56E+00	1.20E+01	1.37E+01	1.13E+01	2.27E+01	6.85E+00	5.87E+00	1.36E+01
F20	6.18E+00	1.67E+02	6.05E+03	1.83E+02	7.39E+04	3.12E+01	2.17E+02	1.57E+03
F21	3.11E+02	1.32E+03	1.35E+03	1.57E+03	3.64E+06	2.91E+03	1.75E+04	6.61E+04
F22	1.06E+02	5.46E+02	5.14E+02	2.23E+02	9.97E+02	5.16E+02	9.05E+02	8.97E+02
F23	3.44E+02	3.44E+02	3.44E+02	3.44E+02	3.44E+02	3.44E+02	3.37E+02	3.44E+02
F24	2.68E+02	2.79E+02	2.75E+02	2.57E+02	2.67E+02	2.68E+02	2.69E+02	2.70E+02
F25	2.05E+02	2.24E+02	2.19E+02	2.04E+02	2.18E+02	2.07E+02	2.00E+02	2.24E+02
F26	1.00E+02	1.00E+02	1.02E+02	1.00E+02	1.41E+02	1.04E+02	1.00E+02	1.59E+02
F27	3.12E+02	7.23E+02	4.60E+02	3.41E+02	9.79E+02	4.72E+02	1.22E+03	9.67E+02
F28	1.14E+03	1.33E+03	1.13E+03	1.14E+03	1.50E+03	1.20E+03	3.68E+02	1.76E+03
F29	8.04E+02	8.47E+02	8.86E+02	7.15E+02	4.17E+03	9.69E+02	2.15E+02	1.44E+06
F30	8.43E+03	1.08E+04	9.92E+03	9.48E+03	1.46E+04	8.71E+03	1.23E+03	1.24E+04

	HCLPSO	IDE	IILPSO	jDElscop	LBBO	PMS	SP-UCI
F1	6.27E+05	1.16E+06	6.95E+05	4.10E+05	1.21E+05	4.61E+05	5.30E+04
F2	1.41E+02	1.10E+00	3.47E+03	2.20E+01	2.22E+03	7.14E+03	1.35E+03
F3	1.50E+03	4.20E+00	1.82E+03	4.28E-03	6.10E-06	3.01E+04	1.25E-06
F4	8.81E+01	5.51E+01	1.10E+02	9.51E+01	2.47E+01	7.47E+01	5.08E+01
F5	2.02E+01	2.03E+01	2.00E+01	2.05E+01	2.00E+01	2.00E+01	2.11E+01
F6	1.47E+01	2.79E-02	3.77E+01	3.00E+01	3.74E+01	3.82E+01	8.48E+00
F7	6.99E-03	3.09E-03	2.09E-02	1.29E-11	1.93E-03	1.17E-02	3.87E-04
F8	4.61E-13	5.07E-07	9.48E-01	7.80E-14	0.00E+00	1.67E-07	5.59E+01
F9	1.02E+02	5.85E+01	2.17E+02	1.38E+02	2.95E+02	3.96E+02	4.85E+01
F10	2.87E+00	3.96E+01	6.20E+00	3.69E-02	6.84E+01	7.51E-02	1.26E+04
F11	4.06E+03	4.34E+03	4.97E+03	5.65E+03	5.92E+03	6.35E+03	1.29E+04
F12	1.51E-01	3.63E-01	1.95E-01	5.29E-01	3.75E-01	1.76E-01	3.33E+00
F13	3.45E-01	3.04E-01	4.59E-01	4.50E-01	4.46E-01	5.16E-01	3.16E-01
F14	2.78E-01	2.74E-01	3.41E-01	2.74E-01	4.07E-01	4.57E-01	4.42E-01
F15	1.11E+01	6.33E+00	4.17E+01	1.73E+01	2.69E+01	6.18E+01	5.87E+00
F16	1.79E+01	1.89E+01	1.85E+01	1.84E+01	1.93E+01	1.88E+01	2.15E+01
F17	1.38E+05	4.93E+03	2.51E+05	8.91E+03	4.20E+04	1.76E+05	2.48E+03
F18	1.75E+02	5.08E+01	6.73E+02	6.73E+01	1.60E+02	2.62E+03	1.83E+02
F19	1.50E+01	9.64E+00	2.55E+01	1.33E+01	6.09E+01	2.53E+01	1.26E+01
F20	1.57E+03	3.63E+01	2.38E+03	5.67E+01	1.99E+02	3.23E+04	1.86E+02
F21	1.21E+05	1.20E+03	1.03E+05	1.84E+03	6.04E+04	2.42E+05	1.61E+03
F22	6.56E+02	3.37E+02	1.18E+03	5.83E+02	1.50E+03	1.51E+03	7.62E+02
F23	3.44E+02	3.44E+02	3.44E+02	3.44E+02	3.44E+02	3.44E+02	3.44E+02
F24	2.60E+02	2.57E+02	2.64E+02	2.58E+02	2.73E+02	2.87E+02	2.78E+02
F25	2.12E+02	2.07E+02	2.36E+02	2.06E+02	2.18E+02	2.24E+02	2.00E+02
F26	1.02E+02	1.08E+02	1.53E+02	1.00E+02	1.14E+02	1.04E+02	1.56E+02
F27	5.60E+02	3.10E+02	1.43E+03	1.00E+03	1.32E+03	1.32E+03	5.19E+02
F28	1.65E+03	1.25E+03	4.39E+03	1.21E+03	3.30E+03	5.68E+03	1.24E+03
F29	1.22E+03	9.57E+02	7.00E+05	1.03E+03	2.17E+03	6.22E+06	8.98E+02
F30	1.05E+04	9.67E+03	1.43E+04	8.03E+03	1.03E+04	1.30E+04	1.08E+04

SHADE-based algorithms tested on artificial benchmark problems (CEC′2014 and CEC′2017). Hence, the question remains whether it will be also useful when solving real-world problems, e.g. those from CEC′2011 set.

In the case of CEC′2011 problems algorithms that use the linear population size reduction are also ranked among the best methods, but are not necessarily the winners. In tests with NFC / 5, depending on the population size settings Rcr-JADE, SHADE or MDE-pBX perform best. When only 5000 function calls is available, SHADE and L-SHADE perform almost equally well. For classical NFC, MPADE is ranked the 2nd best algorithm. Also comparison among the specific variants tested with and without linear population size reduction is inconclusive on CEC′2011 problems. L-JADE perform better than JADE in 6 out of 12 competitions, but is poorer when NFC and *PS* are set to the classical values. L-MPEDE is better than MPEDE in 7 out of 12 competitions, but again excluding the case with default NFC and *PS* settings. On the other hand, L-SHADE outperforms SHADE, and SPS-L-SHADE-EIG outperforms SPS-SHADE in 11 out of 12 competitions. Hence, the usefulness of the linear population size reduction is not clear for real-world problems. The linear population size reduction improves the results obtained by SHADE-based variants,

Table 12
Standard deviations of the results given in Table 11.

	L-SHADE-EpSin-NLC	SapsDE	JADE	AMALGAM	cNrGA	CoBiDE	GA-MPC	GLPSO
F1	4.52E-05	5.73E+04	1.06E+04	4.32E-14	3.29E+06	1.24E+05	1.55E+05	2.96E+05
F2	1.04E-14	0.00E+00	2.71E-13	1.83E-14	1.90E+06	3.70E-01	5.29E+03	8.34E+03
F3	0.00E+00	2.06E-01	2.41E+03	5.94E-14	1.38E+04	6.87E-03	2.69E+01	2.06E+03
F4	4.69E+01	4.01E+01	3.59E+01	2.10E+01	2.99E+01	3.54E+01	3.39E+01	3.20E+01
F5	4.80E-02	4.48E-02	3.36E-02	1.54E-01	1.17E-02	2.80E-01	2.25E-01	3.82E-01
F6	7.27E-02	4.66E+00	5.32E+00	2.16E-01	3.57E+00	3.11E+00	1.77E+00	3.26E+00
F7	3.95E-14	6.10E-03	5.49E-03	4.87E-14	1.38E-01	2.23E-14	4.91E-03	1.30E-02
F8	2.96E-09	1.39E-01	0.00E+00	2.46E-01	9.19E-01	4.38E-14	1.20E+01	1.06E-13
F9	5.62E+00	1.27E+01	7.36E+00	3.64E+00	1.59E+01	1.54E+01	1.25E+01	2.17E+01
F10	1.71E-02	2.81E+01	1.02E-02	2.96E+01	9.83E-01	3.28E+00	5.23E+02	1.96E+00
F11	3.17E+02	3.92E+02	3.34E+02	5.02E+02	8.98E+02	6.41E+02	9.74E+02	7.25E+02
F12	2.68E-02	6.27E-02	3.62E-02	7.39E-02	3.60E-02	2.92E-01	3.00E-02	5.28E-02
F13	1.71E-02	6.87E-02	5.51E-02	4.57E-02	7.21E-02	6.40E-02	7.63E-02	7.78E-02
F14	2.25E-02	1.25E-01	6.31E-02	4.44E-02	2.55E-01	2.65E-02	1.63E-01	1.45E-01
F15	4.67E-01	2.40E+00	7.75E-01	7.76E-01	7.08E+00	1.10E+00	1.28E+00	3.27E+00
F16	4.52E-01	7.49E-01	3.87E-01	1.33E+00	8.17E-01	8.75E-01	1.09E+00	9.34E-01
F17	1.52E+02	1.56E+03	7.60E+02	5.06E+02	1.70E+06	7.84E+03	1.64E+04	4.63E+04
F18	5.94E+00	3.29E+02	8.71E+01	5.01E+01	1.12E+03	4.31E+01	4.86E+02	1.14E+03
F19	1.22E+00	9.79E+00	6.11E+00	3.69E+00	1.36E+01	1.28E+00	1.35E+00	2.59E+00
F20	2.40E+00	1.24E+02	6.54E+03	1.18E+02	3.45E+04	9.54E+00	1.99E+02	1.02E+03
F21	1.24E+02	7.12E+02	3.87E+02	4.11E+02	2.97E+06	3.68E+03	1.54E+04	4.10E+04
F22	6.37E+01	1.85E+02	1.43E+02	1.25E+02	3.56E+02	1.72E+02	2.73E+02	2.47E+02
F23	2.93E-13	4.22E-13	4.99E-13	6.29E-04	2.28E-02	3.68E-13	6.29E-13	1.81E-07
F24	1.28E+00	3.91E+00	1.90E+00	1.31E+00	6.55E+00	2.85E+00	6.24E+00	6.96E+00
F25	1.32E-01	9.07E+00	5.43E+00	2.02E+00	3.71E+00	1.18E+00	1.01E-01	6.69E+00
F26	2.07E-02	1.64E-01	1.40E+01	8.04E-02	7.81E+01	1.95E+01	5.67E-02	4.95E+01
F27	2.00E+01	1.05E+02	8.32E+01	2.64E+01	9.57E+01	7.19E+01	1.83E+02	9.97E+01
F28	3.53E+01	2.41E+02	4.94E+01	7.19E+01	1.16E+02	6.51E+01	3.90E+00	4.87E+02
F29	2.96E+01	4.89E+01	6.90E+01	1.41E+02	5.09E+03	2.15E+02	3.06E+00	7.17E+06
F30	4.07E+02	1.10E+03	6.90E+02	6.62E+02	2.81E+03	4.64E+02	4.13E+02	2.30E+03

	HCLPSO	IDE	IILPSO	jDElscop	LBBO	PMS	SP-UCI
F1	2.16E+05	3.32E+05	3.75E+05	1.34E+05	1.52E+05	2.60E+05	2.95E+04
F2	1.96E+02	1.63E+00	4.99E+03	4.70E+01	7.07E+03	9.07E+03	1.48E+03
F3	8.12E+02	2.78E+00	1.11E+03	1.14E-02	1.14E-05	1.39E+04	7.53E-06
F4	1.48E+01	3.57E+01	3.65E+01	5.37E+00	4.11E+01	5.25E+01	4.12E+01
F5	5.75E-02	6.35E-02	8.92E-03	3.50E-02	2.91E-06	7.76E-03	3.85E-02
F6	3.42E+00	5.63E-02	3.52E+00	1.46E+00	4.20E+00	3.75E+00	2.50E+00
F7	6.89E-03	4.62E-03	4.11E-02	1.85E-11	4.25E-03	1.46E-02	1.99E-03
F8	9.49E-14	1.68E-06	9.01E-01	5.33E-14	0.00E+00	2.67E-08	1.07E+01
F9	1.73E+01	8.64E+00	3.54E+01	1.30E+01	4.69E+01	9.66E+01	8.38E+00
F10	1.65E+01	5.83E+01	2.70E+00	3.84E-02	9.57E+01	2.74E-02	3.37E+02
F11	4.33E+02	5.94E+02	6.77E+02	3.63E+02	8.13E+02	9.40E+02	4.40E+02
F12	5.17E-02	6.54E-02	4.58E-02	6.43E-02	1.64E-01	5.32E-02	2.87E-01
F13	5.79E-02	2.63E-02	8.56E-02	4.07E-02	9.50E-02	9.46E-02	3.94E-02
F14	2.91E-02	2.67E-02	1.23E-01	2.60E-02	7.23E-02	1.99E-01	2.51E-02
F15	3.08E+00	1.21E+00	1.86E+01	1.79E+00	9.59E+00	2.20E+01	1.65E+00
F16	7.54E-01	5.51E-01	7.18E-01	4.26E-01	7.11E-01	7.52E-01	2.56E-01
F17	1.23E+05	3.17E+03	3.32E+05	7.24E+03	7.06E+04	1.01E+05	4.58E+02
F18	8.45E+01	2.15E+01	7.10E+02	2.45E+01	6.30E+01	1.25E+03	1.99E+01
F19	2.85E+00	9.76E-01	1.17E+01	1.84E+00	2.27E+01	1.97E+01	2.85E+00
F20	1.10E+03	9.22E+00	1.85E+03	8.52E+00	3.64E+02	1.43E+04	4.35E+01
F21	6.47E+04	2.46E+02	5.26E+04	8.17E+02	1.33E+05	2.07E+05	3.73E+02
F22	2.54E+02	1.05E+02	3.33E+02	1.20E+02	3.19E+02	3.77E+02	3.56E+02
F23	9.77E-13	4.35E-13	7.02E-03	3.79E-13	2.98E-04	3.28E-05	2.22E-11
F24	3.71E+00	2.33E+00	7.06E+00	3.31E+00	4.69E+00	4.79E+01	1.71E+00
F25	2.46E+00	6.24E-01	8.06E+00	4.43E-01	6.81E+00	1.49E+01	2.30E-12
F26	1.40E+01	2.71E+01	5.02E+01	3.73E-02	3.47E+01	1.95E+01	4.34E+01
F27	1.58E+02	2.31E+01	2.86E+02	7.10E+01	9.75E+01	2.58E+02	6.60E+01
F28	1.62E+02	8.56E+01	1.06E+03	2.03E+01	8.32E+02	1.37E+03	7.57E+01
F29	1.91E+02	1.26E+02	4.98E+06	2.03E+02	9.47E+02	1.36E+07	1.10E+02
F30	1.01E+03	4.92E+02	3.68E+03	1.57E+02	1.53E+03	2.10E+03	1.24E+03

but not necessarily those based on JADE. It is also more helpful when the number of function calls is large or very small (5000). Such conclusions are confirmed by results given in Tables 7–9

4.6. Comparison between the basic SHADE and JADE

From Tables 4 and 6 we find that SHADE variant is always better than JADE when the number of allowed function calls is set to the classical value, or higher. When computational budget is low, the relation between JADE and SHADE depends on the population size and problem set, but JADE still cannot be considered better. From Tables 7–9 we also confirm that for each considered computational budget SHADE with the best variant of the population size setting is better than JADE with any variant of the population size setting. Hence, we may easily recommend using SHADE instead of JADE, irrespective of the kinds of problems, computational budget or population size.

4.7. Usefulness of modifications of JADE

Apart from SHADE and the linear population size reduction, a number

Table 13
51-run mean performances of 15 various algorithms on real-world CEC′2011 problems with maximum number of function calls set to default value (150 000).

	L-SHADE-EpSin-NLC	SapsDE	JADE	AMALGAM	cNrGA	CoBiDE	GA-MPC	GLPSO
F1	1.81E+00	1.23E+00	3.73E-01	4.79E+00	1.39E+01	0.00E+00	1.58E+00	1.01E+01
F2	−2.62E+01	−2.48E+01	−2.31E+01	−1.83E+01	−2.36E+01	−2.11E+01	−2.74E+01	−2.69E+01
F3	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05
F4	1.81E+01	1.84E+01	1.43E+01	1.47E+01	1.86E+01	1.93E+01	1.42E+01	1.56E+01
F5	−3.64E+01	−3.61E+01	−3.56E+01	−2.99E+01	−3.32E+01	−3.49E+01	−3.40E+01	−3.45E+01
F6	−2.92E+01	−2.91E+01	−2.90E+01	−2.34E+01	−2.56E+01	−2.88E+01	−2.78E+01	−2.66E+01
F7	1.13E+00	1.11E+00	1.19E+00	6.05E-01	1.34E+00	6.46E-01	1.03E+00	9.53E-01
F8	2.20E+02	2.20E+02	2.20E+02	2.20E+02	2.20E+02	2.20E+02	2.20E+02	2.20E+02
F9	1.82E+03	2.14E+03	2.11E+03	3.01E+03	1.72E+03	1.76E+03	3.89E+03	2.43E+03
F10	−2.16E+01	−2.13E+01	−2.14E+01	−1.94E+01	−1.45E+01	−2.17E+01	−2.17E+01	−2.07E+01
F11.1	5.20E+04	5.24E+04	5.24E+04	5.21E+04	5.25E+04	5.23E+04	5.26E+04	5.25E+04
F11.2	1.77E+07	1.74E+07	1.73E+07	1.74E+07	1.76E+07	1.77E+07	1.08E+06	1.79E+07
F11.3	1.54E+04	1.55E+04	1.54E+04	1.55E+04	1.55E+04	1.54E+04	1.54E+04	1.55E+04
F11.4	1.81E+04	1.82E+04	1.82E+04	1.81E+04	1.92E+04	1.81E+04	1.83E+04	1.92E+04
F11.5	3.27E+04	3.29E+04	3.30E+04	3.28E+04	3.30E+04	3.27E+04	3.28E+04	3.30E+04
F11.6	1.24E+05	1.34E+05	1.31E+05	1.36E+05	1.38E+05	1.28E+05	1.37E+05	1.39E+05
F11.7	1.84E+06	1.93E+06	1.90E+06	2.15E+06	1.97E+07	1.91E+06	2.15E+06	2.05E+06
F11.8	9.30E+05	9.42E+05	9.37E+05	9.57E+05	1.01E+06	9.38E+05	9.92E+05	9.54E+05
F11.9	9.38E+05	1.00E+06	9.71E+05	9.59E+05	1.73E+06	9.51E+05	1.21E+06	1.26E+06
F11.10	9.30E+05	9.43E+05	9.38E+05	9.53E+05	1.01E+06	9.38E+05	1.05E+06	9.48E+05
F12	1.55E+01	1.60E+01	1.69E+01	1.62E+01	1.81E+01	1.40E+01	1.25E+01	1.55E+01
F13	1.57E+01	1.74E+01	1.73E+01	1.89E+01	2.14E+01	1.25E+01	1.05E+01	2.12E+01

	HCLPSO	IDE	IILPSO	jDElscop	LBBO	PMS	SP-UCI
F1	9.58E-01	1.11E+00	1.20E+01	0.00E+00	2.96E+00	1.34E+01	1.15E+01
F2	−2.67E+01	−2.65E+01	−2.64E+01	−2.13E+01	−2.69E+01	−2.04E+01	−1.12E+01
F3	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05	1.15E-05
F4	1.41E+01	1.46E+01	1.42E+01	1.92E+01	1.39E+01	1.40E+01	2.01E+01
F5	−3.55E+01	−3.53E+01	−3.44E+01	−3.46E+01	−3.60E+01	−3.42E+01	−1.92E+01
F6	−2.82E+01	−2.87E+01	−2.50E+01	−2.88E+01	−2.73E+01	−2.83E+01	−1.34E+01
F7	1.05E+00	8.96E-01	1.26E+00	1.21E+00	8.00E-01	1.23E+00	1.76E+00
F8	2.20E+02	2.20E+02	2.23E+02	2.20E+02	2.20E+02	1.57E+03	2.20E+02
F9	1.62E+03	6.22E+03	6.28E+03	9.51E+03	6.00E+04	1.32E+03	9.65E+03
F10	−2.12E+01	−2.16E+01	−1.80E+01	−2.16E+01	−1.47E+01	−1.25E+01	−2.14E+01
F11.1	5.27E+04	5.85E+04	5.22E+04	2.38E+05	5.14E+04	5.23E+04	5.23E+04
F11.2	1.75E+07	1.91E+07	1.76E+07	1.78E+07	2.19E+07	1.76E+07	1.93E+07
F11.3	1.54E+04	1.54E+04	1.55E+04	1.54E+04	1.55E+04	1.55E+04	1.55E+04
F11.4	1.91E+04	1.84E+04	1.91E+04	1.82E+04	1.83E+04	1.90E+04	1.81E+04
F11.5	3.30E+04	3.28E+04	3.31E+04	3.28E+04	3.29E+04	3.32E+04	3.27E+04
F11.6	1.37E+05	1.35E+05	1.43E+05	1.36E+05	1.53E+05	1.47E+05	1.24E+05
F11.7	2.00E+06	1.96E+06	2.08E+06	2.17E+06	4.43E+07	2.74E+06	1.81E+06
F11.8	9.57E+05	9.60E+05	1.23E+06	1.05E+06	3.46E+06	8.35E+06	9.36E+05
F11.9	1.39E+06	1.38E+06	1.75E+06	1.54E+06	3.11E+06	6.64E+06	9.46E+05
F11.10	9.51E+05	9.62E+05	1.21E+06	1.06E+06	2.72E+06	7.23E+06	9.36E+05
F12	1.50E+01	1.42E+01	1.52E+01	1.75E+01	1.57E+01	2.50E+01	1.50E+01
F13	1.86E+01	1.61E+01	2.34E+01	1.25E+01	2.00E+01	2.89E+01	2.05E+01

of other modifications of JADE has been tested in this study. Here we consider approaches that share similarities with original JADE, but differ in details. All such modifications, including adaptive strategy selection (AdapSS-JADE), auto-enhanced population diversity (JADE-AEPD), eigenvector-based crossover (JADE-EIG), the use of information on evolution path (JADEEP) or repairing the value of crossover parameter (Rcr-JADE) leads to some improvement on CEC′2014 problems when the number of function calls is set to the classical value. However, such improvements with respect to JADE are smaller than the improvements achieved by L-SHADE.

For real-world problems (CEC′2011) AdapSS-JADE, JADE-AEPD, JADE-EIG and JADEEP overall perform similarly to JADE. Only Rcr-JADE is better than JADE in each competition, and in some tests the difference is large. Rcr-JADE is also ranked better than any among other six methods most similar to JADE (JADE itself, AdapSS-JADE, JADE-AEPD, JADE-EIG, JADEEP and L-JADE) when the number of function calls and the population size are set to the classical values. Rcr-JADE performs especially well on real-world problems when the number of function calls is relatively low. However, when the number of function calls is high (NFC · 5), JADE-EIG is a better choice than Rcr-JADE (both in case of CEC′2011 and CEC′2014 problems). This agrees with the results achieved by SPS-L-SHADE-EIG (that also uses eigenvector-based crossover), which on real-world problems performs best among all 22 variants when the number of function calls is large. Hence, among various modifications of JADE not based on SHADE, repairing the crossover rate parameter (Rcr-JADE) and using eigenvector-based crossover operator (JADE-EIG) are two the most promising ones.

4.8. Usefulness of modifications of SHADE

As SHADE is a very recent DE variant that has been quickly modified by adding the linear population size reduction, only a few modifications of the basic SHADE algorithm with the fixed population size may be found in the literature. Two of them are tested in this study, one introduces event trigged mechanism (ETI-SHADE) in order to avoid premature convergence or stagnation, the second introduces a method to take advantage of formerly successful individuals that at some stage of the evolution were removed from the population (SPS-SHADE). SPS-SHADE performs poorer than SHADE in our tests, hence cannot be recommended. The success of SPS-L-SHADE-EIG algorithm should rather be attributed to the use of the linear population size reduction and eigenvector-based crossover. Adding archive to keep some individuals that loose during selection within SPS procedure may unnecessarily complicate the approach, as in JADE and SHADE similar archive already exists, see Ref. [81]. ETI-SHADE generally improves the performance of SHADE on CEC′2014 problems when number of function calls is high

Table 14
Standard deviations of the results given in Table 13.

	L-SHADE-EpSin-NLC	SapsDE	JADE	AMALGAM	cNrGA	CoBiDE	GA-MPC	GLPSO
F1	3.96E+00	3.42E+00	1.31E+00	5.92E+00	6.93E+00	0.00E+00	4.01E+00	5.75E+00
F2	5.59E-01	8.34E-01	8.21E-01	5.92E+00	4.16E+00	2.79E+00	7.01E-01	1.58E+00
F3	1.68E-19	2.47E-19	1.77E-19	1.28E-18	4.77E-13	2.05E-19	1.71E-21	1.47E-19
F4	3.32E+00	3.31E+00	9.84E-01	4.58E-01	3.17E+00	2.74E+00	1.39E+00	2.65E+00
F5	5.30E-01	6.60E-01	6.43E-01	2.86E+00	1.78E+00	1.09E+00	1.49E+00	1.86E+00
F6	4.24E-03	2.42E-01	7.56E-02	3.15E+00	3.25E+00	6.92E-01	1.75E+00	3.13E+00
F7	9.03E-02	9.83E-02	8.70E-02	9.09E-02	1.85E-01	1.09E-01	2.87E-01	1.52E-01
F8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F9	3.61E+02	7.03E+02	4.48E+02	1.01E+03	8.62E+02	4.57E+02	6.74E+03	8.69E+02
F10	9.91E-02	1.03E+00	2.20E-01	2.42E+00	2.08E+00	1.06E-01	1.21E-01	1.69E+00
F11.1	5.06E+02	4.83E+02	5.04E+02	5.97E+02	5.54E+02	6.17E+02	1.09E+03	6.83E+02
F11.2	4.42E+04	5.67E+04	3.44E+04	2.61E+04	9.20E+04	6.65E+04	2.29E+04	1.08E+05
F11.3	1.09E+01	1.19E+01	4.45E+00	9.33E+00	2.00E+01	6.16E-01	1.85E+00	2.21E+01
F11.4	3.37E+01	1.06E+02	2.50E+02	3.52E+01	1.61E+02	3.60E+01	1.11E+02	1.91E+02
F11.5	1.16E+01	1.20E+02	9.08E+01	5.62E+01	8.36E+01	2.10E+01	2.40E+01	6.14E+01
F11.6	4.60E+02	4.84E+03	4.82E+03	5.51E+03	2.54E+03	1.26E+03	2.37E+03	4.01E+03
F11.7	1.21E+04	1.77E+04	2.88E+04	3.02E+05	4.47E+07	1.38E+04	2.62E+05	1.76E+05
F11.8	9.82E+02	3.58E+03	2.11E+03	3.63E+04	1.32E+05	2.22E+03	4.52E+04	2.28E+04
F11.9	1.15E+03	7.48E+04	1.13E+05	2.34E+04	3.28E+05	1.92E+04	1.42E+05	1.40E+05
F11.10	1.16E+03	1.01E+04	3.67E+03	1.06E+04	1.36E+05	2.02E+03	1.14E+05	6.94E+03
F12	7.75E-01	2.15E+00	1.54E+00	2.11E+00	2.61E+00	2.13E+00	3.61E+00	2.90E+00
F13	2.42E+00	2.87E+00	2.43E+00	2.21E+00	3.24E+00	2.83E+00	2.57E+00	2.91E+00
	HCLPSO	**IDE**	**IILPSO**	**jDElscop**	**LBBO**	**PMS**	**SP-UCI**	
F1	2.98E+00	3.41E+00	5.47E+00	0.00E+00	5.14E+00	6.17E+00	7.96E+00	
F2	1.31E+00	9.99E-01	2.49E+00	9.53E-01	1.14E+00	5.09E+00	1.08E+00	
F3	1.51E-19	1.58E-19	1.62E-19	1.75E-19	9.36E-15	1.30E-18	1.40E-19	
F4	2.80E-01	1.49E+00	2.39E-01	2.90E+00	4.13E-02	2.52E-01	2.37E+00	
F5	9.83E-01	1.06E+00	1.54E+00	6.64E-01	7.37E-01	1.34E+00	1.25E+00	
F6	1.14E+00	8.05E-01	2.98E+00	1.90E-01	1.97E+00	8.75E-01	1.07E+00	
F7	1.55E-01	1.26E-01	1.95E-01	1.03E-01	1.20E-01	2.26E-01	1.19E-01	
F8	0.00E+00	0.00E+00	1.83E+01	0.00E+00	0.00E+00	1.53E+03	1.96E+00	
F9	8.24E+02	1.08E+03	7.52E+03	2.33E+03	2.77E+04	6.19E+02	5.13E+03	
F10	2.28E-01	1.49E-01	2.82E+00	1.46E-01	2.96E+00	2.08E+00	6.24E-01	
F11.1	6.72E+02	1.45E+03	3.84E+02	1.13E+05	4.81E+02	5.53E+02	4.29E+02	
F11.2	5.70E+04	8.21E+04	7.73E+04	5.41E+04	2.84E+05	1.02E+05	3.28E+05	
F11.3	1.85E+00	9.39E-01	1.37E+01	3.07E-05	3.71E+00	3.74E+01	1.45E+01	
F11.4	1.36E+02	1.07E+02	1.45E+02	7.46E+01	5.92E+01	3.10E+02	4.11E+01	
F11.5	3.68E+01	3.27E+01	9.31E+01	4.42E+01	4.13E+01	1.11E+02	1.67E+01	
F11.6	2.22E+03	1.28E+03	4.91E+03	2.15E+03	9.33E+03	6.81E+03	6.25E+02	
F11.7	1.23E+05	2.08E+04	2.46E+05	1.95E+05	2.76E+07	7.18E+05	1.08E+04	
F11.8	3.15E+04	1.13E+04	4.03E+05	4.51E+04	3.83E+06	5.53E+06	1.83E+03	
F11.9	2.23E+05	6.70E+04	4.84E+05	1.13E+05	2.59E+06	3.61E+06	2.46E+03	
F11.10	2.15E+04	1.22E+04	4.55E+05	4.37E+04	2.31E+06	5.15E+06	1.77E+03	
F12	1.67E+00	1.76E+00	2.81E+00	1.86E+00	2.34E+00	1.00E+01	3.18E+00	
F13	2.70E+00	3.40E+00	2.63E+00	2.56E+00	2.83E+00	6.43E+00	1.07E+00	

Table 15
Mean rankings of 15 tested algorithms on 50-dimensional CEC′2014 and real-world CEC′2011 problems, when maximum number of function calls is set to default values (500 000 for CEC′2014 and 150 000 for CEC′2011).

	averaged ranking		final position	
	CEC′2014	CEC′2011	CEC′2014	CEC′2011
L-SHADE-EpSin-NLS	3.03	4.66	1	1
SapsDE	7.57	6.52	8	4
JADE	6.35	5.98	4	3
AMALGAM	3.75	8.07	2	8
cNrGA	11.40	11.39	13	14
CoBiDE	5.92	4.73	3	2
GA-MPC	6.67	6.70	6	5
GLPSO	10.45	9.07	12	11
HCLPSO	8.17	7.16	9	7
IDE	6.38	7.11	5	6
IILPSO	12.00	10.64	14	13
jDElscop	7.45	8.55	7	9
LBBO	10.12	9.25	11	12
PMS	12.33	11.50	15	15
SP-UCI	8.42	8.68	10	10

enough (NFC or NFC · 5), but fails on CEC′2011 ones. Hence, none of the two modifications may be especially recommended here.

4.9. Impact of the population size

Discussion given in this sub-section is based on Tables 7–9 only, in which all algorithms with each among three considered population size variants are compared together. In the case of both CEC′2011 and CEC′2014 problems one finds results that should be expected. For most algorithms the population size suggested by the authors of particular method is a good choice when the classical number of function calls is used (NFC); it should be increased when the number of available function calls is much higher than often assumed (NFC · 5), or decreased when this number is much lower (NFC / 5 or 5000). Also, the best algorithm (ranked 1) for NFC case is the one that uses *PS*, and for NFC / 5 – the one that uses *PS* / 2. When only 5000 function calls are available, population size of 20 individuals is by far the best choice. For NFC · 5 competitions, the best results on CEC′2014 problems are achieved by algorithm that uses *PS* · 2, but in case of CEC′2011 ones – algorithms with *PS* perform slightly better. However, the differences in averaged rankings between the best and the third best approaches that use *PS* and the second best algorithm that uses *PS* · 5 are marginal, and most algorithms perform better for CEC′2011 problems with NFC · 5 when their population sizes are twice larger than classically set. Hence, we obtain a clear conclusion – use default population size when the allowed number of function calls is classical, and increase or decrease it accordingly when larger or lower

Table 16
The statistical significance of pair-wise comparisons among 15 various algorithms on 50-dimensional CEC′2014 problems by means of Friedman's test with post-hoc Shaffer's procedure at 5% significance level. "+" – difference between two algorithms is statistically significant; empty cell – difference between two algorithms is not statistically significant.

CEC′2014	L-SHADE-EpSin-NLS	SapsDE	JADE	AMALGAM	cNrGA	CoBiDE	GA-MPC	GLPSO	HCLPSO	IDE	IILPSO	jDElscop	LBBO	PMS	SP-UCI
L-SHADE EpSin-NLS		+			+			+	+		+	+	+	+	+
SapsDE	+										+			+	
JADE					+			+			+			+	
AMALGAM					+			+	+		+		+	+	+
cNrGA	+		+	+		+	+			+		+			
CoBiDE					+			+			+		+	+	
GA-MPC					+						+			+	
GLPSO	+		+	+		+				+					
HCLPSO	+			+										+	
IDE					+			+			+			+	
IILPSO	+	+	+	+		+	+			+		+			
jDElscop	+				+						+			+	
LBBO	+			+		+									
PMS	+	+	+	+		+	+		+	+		+			+
SP-UCI	+			+										+	

Table 17
The statistical significance of pair-wise comparisons among 15 various algorithms on real-world CEC′2011 problems by means of Friedman's test with post-hoc Shaffer's procedure at 5% significance level. "+" – difference between two algorithms is statistically significant; empty cell – difference between two algorithms is not statistically significant.

CEC′2011	L-SHADE-EpSin-NLS	SapsDE	JADE	AMALGAM	cNrGA	CoBiDE	GA-MPC	GLPSO	HCLPSO	IDE	IILPSO	jDElscop	LBBO	PMS	SP-UCI
L-SHADE EpSin-NLS					+						+			+	
SapsDE					+									+	
JADE					+									+	
AMALGAM															
cNrGA	+	+	+			+	+								
CoBiDE					+						+			+	
GA-MPC					+									+	
GLPSO															
HCLPSO															
IDE															
IILPSO	+					+									
jDElscop															
LBBO															
PMS	+	+	+			+	+								
SP-UCI															

number of function calls is allowed.

4.10. Relative performance of JADE/SHADE-based algorithms with respect to the number of allowed function calls

On CEC′2014 problems, L-SHADE-EpSin-NLS, L-SHADE-cnEpSin and L-SHADE-SPACMA are best methods when the number of function calls is not very low (higher than 5000, see Tables 4 and 7). L-SHADE-SPACMA is better ranked for larger, L-SHADE-EpSin-NLS for lower computational budgets. When computational budget is very low (5000), SHADE or L-SHADE are the best two algorithms.

On CEC′2011 problems L-SHADE-cnEpSin is the best approach when the classical number of function calls is used. When the computational budget is lower (NFC / 5), SHADE or Rcr-JADE may be recommended, when higher (NFC · 5) SPS-L-SHADE-EIG is the best choice. When computational budget is very low (5000), again SHADE or L-SHADE outperform all other variants.

Overall, SHADE-based algorithms win comparisons for each computational budget. Advanced modifications of L-SHADE are useful when the number of function calls is not very low, otherwise simple SHADE or L-SHADE algorithms perform better.

Most algorithms that were created by adding some modifications to JADE, if the linear population size reduction procedure is unused, are ranked relatively better when the number of allowed function calls is low.

4.11. Performance of JADE/SHADE-based algorithms against other kinds of metaheuristics

In comparison against other metaheuristics three JADE/SHADE-based variants are selected: one of the overall best methods (L-SHADE-EpSin-NLS), the classical JADE that often shows moderate performance (JADE) and one of the poorest methods loosely based on JADE (SapsDE). Such three algorithms are tested with the default population sizes (*PS* case from Table 1) against 12 very different kinds of metaheuristics discussed in Table 10 on 50-dimensional CEC′2014 benchmarks and CEC′2011 real-world problems with the classical numbers of function calls (i.e. 500 000 for CEC′2014 and 150 000 for CEC′2011). 51-run averaged performances of such 15 algorithms, accompanied with their standard deviations, are given in Tables 11–14 In Table 15 averaged ranks and final positions of algorithms on CEC′2014 and CEC′2011 sets are given; the statistical comparison is presented in Tables 16 and 17

From Table 15 we find that L-SHADE-EpSin-NLS is the best among 15 compared algorithms both on CEC′2014 and CEC′2011 problems. Moreover, all three JADE/SHADE-based variants are among best four approaches on real-world problems, even though the differences are

often not statistically significant (see Table 17). The relative performances of JADE/SHADE-based methods are slightly poorer on CEC′2014 problems, as SapsDE is ranked only 8th best out of 15 algorithms. According to our results, among non-JADE/SHADE-based methods only CoBiDE (DE variant without clear links with JADE), GA-MPC (the winner of CEC′2011 competition) and to some extent AMALGAM (a multi-algorithm that uses CMA-ES as sub-procedure) may overall be competitive to the methods based on JADE or SHADE.

In recent years one may find two contradictory opinions on developing algorithms by adding various modifications step-by-step to already existing methods. In Refs. [15,39] such an approach is criticized as "adding stuff" and is considered inefficient, when in Ref. [26] this way of stepwise development of JADE/SHADE algorithms was called a "good example of evolution". Our results may be interpreted differently by skeptics and enthusiasts of JADE/SHADE methods.

In our opinion there is no doubt that adding elements one-by-one to existing algorithms may lead to various problems. We may give two examples here: 1) the best settings of 15 control parameters of SPS-L-SHADE-EIG algorithm for various problems may differ by orders of magnitude [36]; 2) some elements of L-SHADE-EpSin algorithm [3] are structurally-biased [57]. Such problems are probably en effect of "adding stuff" approach, criticized in Refs. [15,39]. Moreover, skeptical reader may argue that many recent JADE/SHADE-based variants were developed for solving CEC′2014 problems, hence their good performance on these benchmarks reported in our study is not surprising, and that on the real-world problems the differences between performances of various algorithms are small and rarely statistically significant.

However, by enthusiasts our findings may be read as the confirmation of the importance of JADE/SHADE-based algorithms for the future of metaheuristics. According to the discussed results, even when simplified and used with only one default value of control parameters, the two algorithms mentioned in the previous paragraph (SPS-L-SHADE-EIG and L-SHADE-EpSin-NLS), as well as many other JADE/SHADE-based methods, outperform the vast majority of other metaheuristics tested, both very simple [15] and highly complicated [63,71], on the artificial benchmarks and real-world problems. Even though on some problems differences are larger, on others smaller, the final results are in favor of JADE/SHADE-based methods.

5. Conclusions

In this study we briefly outline a history of such Differential Evolution algorithms that originated from JADE [81] variant. Methods that are based on JADE attract large attention in recent years and some of them, especially those referred to as L-SHADE, are probably among the most effective Evolutionary Algorithms. We perform an inter-comparison among 22 different JADE/SHADE-based variants that were proposed between 2009 and 2017. Tests are performed on both the 50-dimensional artificial benchmarks from CEC′2014 [45] and the real-world problems from CEC′2011 collection [18]. To discuss how successes or failures of various JADE/SHADE-based modifications depend on the computational budget we perform independent tests for four different numbers of function calls: 1) the values suggested for CEC′2014 and CEC′2011 problems, 2) the values five times larger than suggested, 3) the values five times smaller than suggested, and 4) very small computational budget set to 5000 function calls. We also consider the impact of population size, the main Differential Evolution control parameter, by performing tests with three different population size settings: 1) the population size suggested in source paper of particular method or in a review [55], 2) the population size twice lower and 3) the population size twice larger than suggested. For very small computational budget we skip tests with twice larger population sizes, instead testing algorithms with very small population size set to 20 individuals.

Although some specific JADE or SHADE-based variants may not perform better than the original JADE, this study shows that overall the evolution of this family of methods is successful, confirming intuitive claims from Ref. [26]. The main and simplified chain of evolution may be expressed as follows: DE [64] → JADE [81] → JADE with weighted adaptation of control parameters [53] → SHADE [65] → L-SHADE [66], and from L-SHADE within the last three years a number of successful variants were created, including SPS-L-SHADE-EIG [36], L-SHADE-EpSin [3], its simplified version (L-SHADE-EpSin-NLS [57]), L-SHADE-cnEpSin [7] and L-SHADE-SPACMA [50]. Each element of this chain of evolution performs generally better than the former, irrespectively of the problem set, if only the number of function calls is not too low. However, when the number of function calls is very low, the superiority of the recent modifications of L-SHADE over SHADE are doubtful.

Among 22 JADE-based variants tested in this study, two algorithms perform the best overall: L-SHADE-cnEpSin [7], one of the best algorithms in CEC′2017 competition, and L-SHADE-EpSin-NLS [3], the co-winner of CEC′2016 competition used without structurally-biased [57] local search procedure (hence NLS added to the name of the L-SHADE-EpSin procedure proposed originally in Ref. [3], see Ref. [57] for details). Based on our tests these variants must be recommended for both the artificial benchmark and the real-world problems when the number of allowed function calls is set to the classical values.

If one is interested in artificial benchmarks only, L-SHADE-SPACMA [50] is the best choice when the number of function calls is either classical or higher. On the real-world problems, when the computational budget is large, SPS-L-SHADE-EIG [36] approach achieves the best performance. For the real-world problems with other computational budgets SHADE [65], MPADE [17] or Rcr-JADE [31] can also be recommended as an alternatives. When the computational budget is very low (5000 function calls), basic SHADE [65] or L-SHADE [66] must be recommended for both the real-world and benchmark problems.

Generally speaking, SHADE-based variants outperform those based on initial JADE. Although the majority of best performing variants use the linear population size reduction procedure (hence L is added to their names) [66], the efficiency of this procedure is confirmed by our tests for SHADE-based algorithms only, especially when the number of allowed function calls is large. In the case of non-SHADE-based variants, or when the number of allowed function calls is low, the linear population size reduction procedure is important only for solving the artificial benchmarks, but not real-world problems.

Some JADE-based variants proposed in recent years does not lead to much improvement over JADE, according to the tests performed in this study. Excluding SHADE variants with the linear population size reduction, only three among JADE-based variants perform much better than the original JADE, namely MPADE [17], eigenvector-based crossover (JADE-EIG, [34]) and repairing the value of crossover parameter (Rcr-JADE, [31]). The efficiency of eigenvector-based crossover is also confirmed by the success of SHADE-based SPS-L-SHADE-EIG approach [36]) that include such procedure.

Overall, tests performed in this study confirm that JADE/SHADE-based methods are a "good example of evolution" [26] and may be very useful for solving various kinds of problems. We agree that not all JADE/SHADE-based variants may be recommended, but even the biological evolution, with four billion years of experience, often have to make a number of failures until yet another step becomes both innovative and successful.

Acknowledgments

This work was supported within statutory activities No 3841/E-41/S/2017 of the Ministry of Science and Higher Education of Poland.

We would like to thank Prof. Ponnuthurai N. Suganthan for providing the MATLAB codes of MDE-pBX and HCLPSO algorithms, Prof. Jasper A. Vrugt for providing the MATLAB code of AMALGAM, Dr Tang for providing the MATLAB code of IDE and Dr. Chu for providing the MATLAB code of SP-UCI. We are also grateful to the authors of [3,7,24,34–36,47,50,63,69,76] for making the codes of cNrGA, GA-MPC, LBBO, L-SHADE-cnEpSin, L-SHADE-EpSin, L-SHADE-SPACMA, MPEDE,

SPS-SHADE and SPS-L-SHADE-EIG algorithms, as well as the eigenvector-based crossover procedure and code of Shaffer's post-hoc statistical procedure widely available on the web.

Appendix A. Supplementary data

Supplementary data related to this article can be found at https://doi.org/10.1016/j.swevo.2018.03.007.

References

[1] M.Z. Ali, N.H. Awad, P.N. Suganthan, R.G. Reynolds, An adaptive multipopulation Differential Evolution with dynamic population reduction, IEEE Trans. Cybern. 47 (9) (2017) 2768–2779.

[2] M.Z. Ali, N.H. Awad, P.N. Suganthan, A.M. Shatnawi, R.G. Reynolds, An improved class of real-coded Genetic Algorithms for numerical optimization, Neurocompuing 275 (2018) 155–166.

[3] N.H. Awad, M.Z. Ali, P.N. Suganthan, R.G. Reynolds, An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems, in: Proc of the IEEE Congress on Evolutionary Computation, Vancouver, Canada, 2016, https://doi.org/10.1109/CEC.2016.7744163.

[4] N.H. Awad, M.Z. Ali, P.N. Suganthan, E. Jaser, A decremental stochastic fractal differential evolution for global numerical optimization, Inf. Sci. 372 (2016) 470–491.

[5] N.H. Awad, M.Z. Ali, J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem Defnitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-parameter Numerical Optimization, Technical Report, Nanyang Technological University, Singapore, 2016.

[6] N.H. Awad, M.Z. Ali, P.N. Suganthan, R.G. Reynolds, CADE: a hybridization of cultural algorithm and differential evolution for numerical optimization, Inf. Sci. 378 (2017) 215–241.

[7] N.H. Awad, M.Z. Ali, P.N. Suganthan, Ensemble sinusoidal Differential Covariance Matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems, in: 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 2017, https://doi.org/10.1109/CEC.2017.7969336.

[8] N.H. Awad, M.Z. Ali, P.N. Suganthan, R.G. Reynolds, A.M. Shantawi, A novel Differential crossover strategy based on Covariance Matrix Learning with Euclidean neighborhood for solving real-world problems, in: 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 2017, https://doi.org/10.1109/CEC.2017.7969337.

[9] P.P. Biswas, P.N. Suganthan, G.A.J. Amaratunga, Optimal placement of wind turbines in a windfarm using L-SHADE algorithm, in: 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 2017, https://doi.org/10.1109/CEC.2017.7969299.

[10] J. Brest, M.S. Maucec, Self-adaptive differential evolution algorithm using population size reduction and three strategies, Soft Comput. 15 (11) (2011) 2157–2174.

[11] J. Brest, M.S. Maucec, B. Boskovic, iL-SHADE: improved L-SHADE algorithm for single objective real-parameter optimization, in: 2016 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, 2016, https://doi.org/10.1109/CEC.2016.7743922.

[12] J. Brest, M.S. Maucec, B. Boskovic, Single objective real-parameter optimization: algorithm jSO, in: 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 2017, https://doi.org/10.1109/CEC.2017.7969456.

[13] C. Brown, Y.C. Jin, M. Leach, M. Hodgson, μJADE: adaptive differential evolution with a small population, Soft Comput. 20 (10) (2016) 4111–4120.

[14] P. Bujok, J. Tvrdík, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L.A. Zadeh, J.M. Zurada, Adaptive differential evolution: SHADE with competing crossover strategies, in: Artificial Intelligence and Soft Computing Series, Lecture Notes in Computer Science, vol. 9119, Springer, 2015, pp. 329–339.

[15] F. Caraffini, F. Neri, G. Iacca, A. Mol, Parallel memetic structures, Inf. Sci. 227 (2013) 60–82.

[16] W. Chu, X. Gao, S. Sorooshian, A new evolutionary search strategy for global optimization of high-dimensional problems, Inf. Sci. 181 (22) (2011) 4909–4927.

[17] L.Z. Cui, G.H. Li, Q.Z. Lin, J.Y. Chen, N. Lu, Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations, Comput. Oper. Res. 67 (2016) 155–173.

[18] S. Das, 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, Dec. 2010.

[19] S. Das, P.N. Suganthan, Differential Evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 27–54.

[20] S. Das, S.S. Mullick, P.N. Suganthan, Recent advances in Differential Evolution – an updated survey, Swarm Evol. Comput. 27 (2016) 1–30.

[21] W. Du, S.Y.S. Leung, Y. Tang, A.V. Vasilakos, Differential Evolution with event-triggered impulsive control, IEEE Trans. Cybern. 47 (1) (2017) 244–257.

[22] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proc. 6th Int. Symp, Micromachine Human Sci., Nagoya, Japan, 1995, pp. 39–43.

[23] A.E. Eiben, S.K. Smit, Parameter tuning for configuring and analyzing evolutionary algorithms, Swarm Evol. Comput. 1 (1) (2011) 19–31.

[24] S.M. Elsayed, R.A. Sarker, D.L. Essam, GA with a new multi-parent crossover for constrained optimization, IEEE Congr. Evol. Comput. (2011) 857–864.

[25] M.G. Epitropakis, D.K. Tasoulis, N.G. Pavlidis, V.P. Plagianakos, M.N. Vrahatis, Enhancing Differential Evolution utilizing proximity-based mutation operations, IEEE Trans. Evol. Comput. 15 (1) (2011) 99–119.

[26] C. Garcia-Martinez, P.D. Gutierrez, D. Molina, M. Lozano, F. Herrera, Since CEC 2005 competition on Real-parameter optimisation: a decade of research, Progress and comparative analysis's weakness, Soft Comput. 21 (19) (2017) 5573–5583.

[27] S. Garcia, F. Herrera, An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, J. Mach. Learn. Res. 9 (2008) 2677–2694.

[28] W.Y. Gong, A. Fialho, Z.H. Cai, H. Li, Adaptive strategy selection in Differential Evolution for numerical optimization: an empirical study, Inf. Sci. 181 (24) (2011) 5364–5386.

[29] W.Y. Gong, Z.H. Cai, C.X. Ling, H. Li, Enhanced Differential Evolution with adaptive strategies for numerical optimization, IEEE Trans. Syst. Man Cybern. B Cybern. 41 (2) (2011) 397–413.

[30] W.Y. Gong, Z.H. Cai, Differential Evolution with ranking-based mutation operators, IEEE Trans. Cybern. 43 (6) (2013) 2066–2081.

[31] W.Y. Gong, Z.H. Cai, Y. Wang, Repairing the crossover rate in adaptive differential evolution, Appl. Soft Comput. 15 (2014) 149–168.

[32] W.Y. Gong, A.M. Zhou, C.H. Zhai, A multioperator search strategy based on cheap surrogate models for evolutionary optimization, IEEE Trans. Evol. Comput. 19 (5) (2015) 746–758.

[33] Y.J. Gong, J.J. Li, Y. Zhou, Y. Li, H.S.H. Chung, Y.H. Shi, J. Zhang, Genetic learning particle swarm optimization, IEEE Trans. Cybern. 46 (10) (2016) 2277–2290.

[34] S.M. Guo, C.C. Yang, Enhancing Differential Evolution utilizing eigenvector-based crossover operator, IEEE Trans. Evol. Comput. 19 (1) (2015) 31–49.

[35] S.M. Guo, C.C. Yang, P.H. Hsu, J.S.H. Tsai, Improving differential evolution with a successful-parent-selecting framework, IEEE Trans. Evol. Comput. 19 (5) (2015) 717–730.

[36] S.M. Guo, J.S.H. Tsai, C.C. Yang, 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 Congress on Evolutionary Computation, Sendai, Japan, 2015, pp. 1003–1010.

[37] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation, in: Proc. IEEE Int. Conf. Evol. Comput., Nagoya, Japan, 1996, pp. 312–317.

[38] S. Helwig, J. Branke, S. Mostaghim, Experimental analysis of bound handling techniques in particle swarm optimization, IEEE Trans. Evol. Comput. 17 (2) (2013) 259–271.

[39] G. Iacca, F. Neri, E. Mininno, Y.S. Ong, M.H. Lim, Ockham's Razor in memetic computing: three stage optimal memetic exploration, Inf. Sci. 188 (2012) 17–43.

[40] S.M. Islam, S. Das, S. Ghosh, S. Roy, P.N. Suganthan, An adaptive Differential Evolution algorithm with novel mutation and crossover strategies for global numerical optimization, IEEE Trans. Syst. Man Cybern. B Cybern. 42 (2) (2012) 482–500.

[41] A.V. Kononova, D.W. Corne, P. De Wilde, V. Shneer, F. Caraffini, Structural bias in population-based algorithms, Inf. Sci. 298 (2015) 468–490.

[42] X.S. Lai, Y. Zhou, Success rates analysis of three hybrid algorithms on SAT instances, Swarm Evol. Comput. 34 (2017) 119–129.

[43] Y.L. Li, Z.H. Zhan, Y.J. Gong, W.N. Chen, J. Zhang, Y. Li, Differential evolution with an evolution path: a DEEP evolutionary algorithm, IEEE Trans. Cybern. 45 (9) (2015) 1798–1810.

[44] G.H. Li, Q.Z. Lin, L.Z. Cui, Z.H. Du, Z.P. Liang, J.Y. Chen, N. Lu, Z. Ming, A novel hybrid differential evolution algorithm with modified CoDE and JADE, Appl. Soft Comput. 47 (2016) 577–599.

[45] J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-parameter Numerical Optimization, Technical Report 201311, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, 2013. China and Nanyang Technological University, Singapore.

[46] Q.Z. Lin, C.Y. Tang, Y.P. Ma, Z.H. Du, J.Q. Li, J.Y. Chen, Z. Ming, A novel adaptive control strategy for decomposition-based multiobjective algorithm, Comput. Oper. Res. 78 (2017) 94–107.

[47] Y. Lou, S.Y. Yuen, Non-revisiting genetic algorithm with adaptive mutation using constant memory, Memet. Comput. 8 (2016) 189–210.

[48] K.T. Lwin, R. Qu, B.L. MacCarthy, Mean-VaR portfolio optimization: a nonparametric approach, Eur. J. Oper. Res. 260 (2017) 751–766.

[49] N. Lynn, P.N. Suganthan, Heterogeneous comprehensive learning Particle Swarm Optimization with enhanced exploration and exploitation, Swarm Evol. Comput. 24 (2015) 11–24.

[50] A.W. Mohamed, A.A. Hadi, A.M. Fattouh, K.M. Jambi, LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems, in: 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 2017, https://doi.org/10.1109/CEC.2017.7969307.

[51] D. Molina, F. Moreno-Garcia, F. Herdera, Analysis among winners of different IEEE CEC competitions on Real-parameters optimization: is there always improvement?, in: 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 2017, https://doi.org/10.1109/CEC.2017.7969392.

[52] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artif. Intell. Rev. 33 (1–2) (2010) 61–106.

[53] F. Peng, K. Tang, G. Chen, Z. Yao, Multi-start JADE with knowledge transfer for numerical optimization, in: Proc of the IEEE Congress on Evolutionary Computation, Trondheim, Norway, 2009, pp. 1889–1895.

[54] A.P. Piotrowski, Regarding the rankings of optimization heuristics based on artificially-constructed benchmark functions, Inf. Sci. 297 (2015) 191–201.

[55] A.P. Piotrowski, Review of differential evolution population size, Swarm Evol. Comput. 32 (2017) 1–24.
[56] A.P. Piotrowski, M.J. Napiorkowski, J.J. Napiorkowski, P.M. Rowinski, Swarm intelligence and evolutionary algorithms: performance versus speed, Inf. Sci. 384 (2017) 34–85.
[57] A.P. Piotrowski, J.J. Napiorkowski, Some metaheuristics should be simplified, Inf. Sci. 427 (2018) 32–62.
[58] P. Posik, W. Huyer, L. Pal, A comparison of global search algorithms for continuous black box optimization, Evol. Comput. 20 (4) (2012) 509–541.
[59] Q. Qin, S. Cheng, Q. Zhang, L. Li, Y. Shi, PSO with interswarm interactive learning strategy, IEEE Trans. Cybern. 46 (10) (2016) 2238–2251.
[60] K.M. Sallam, R.A. Sarker, D.L. Essam, S.M. Elsayed, Neurodynamic Differential Evolution algorithm and solving CEC2015 competition problems, in: 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, Japan, 2015, https://doi.org/10.1109/CEC.2015.7257003.
[61] E.H.D. Segundo, A.L. Amoroso, V.C. Mariani, L.D. Coelho, Thermodynamic optimization design for plate-fin heat exchangers by Tsallis JADE, Int. J. Therm. Sci. 113 (2017) 136–144.
[62] J.P. Shaffer, Modified sequentially rejective multiple test procedures, J. Am. Stat. Assoc. 81 (395) (1986) 826–831.
[63] D. Simon, M.G.H. Omran, M. Clerc, Linearized biogeography-based optimization with re-initialization and local search, Inf. Sci. 267 (2014) 140–157.
[64] R. Storn, K.V. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[65] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: Proc. IEEE Congress on Evolutionary Computation, Cancun, Mexico, 2013, pp. 71–78.
[66] R. Tanabe, A. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in: Proc. IEEE Congress on Evolutionary Computation, Bejing, China, 2014, pp. 1658–1665.
[67] R. Tanabe, A. Fukunaga, Reevaluating exponential crossover in differential evolution, in: Proceedings of Parallel Problem Solving from Nature, Ljubljana, September, 2014, pp. pp.201–210.
[68] L. Tang, Y. Dong, J. Liu, Differential evolution with an individual-dependent mechanism, IEEE Trans. Evol. Comput. 19 (4) (2015) 560–574.
[69] A. Ulas, O.T. Yildiz, E. Alpaydin, Cost-conscious comparison of supervised learning algorithms over multiple data sets, Pattern Recogn. 45 (2012) 1772–1781.
[70] A. Viktorin, M. Pluhacek, R. Senkerik, Network based linear population size reduction in SHADE, in: 2016 International Conference on Intelligent Networking and Collaborative Systems (INCoS), IEEE, 2016, pp. 86–93.
[71] J.A. Vrugt, B.A. Robinson, J.M. Hyman, Self-adaptive multimethod search for global optimization in real-parameter spaces, IEEE Trans. Evol. Comput. 13 (2) (2009) 243–259.
[72] Y. Wang, Z.X. Cai, Q.F. Zhang, Differential Evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (1) (2011) 55–66.
[73] X. Wang, S.G. Zhao, Differential Evolution algorithm with self-adaptive population resizing mechanism, Math. Probl Eng. (2013), 419372, https://doi.org/10.1155/2013/419372.
[74] Y. Wang, H.X. Li, T.W. Huang, L. Li, Differential Evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18 (2014) 232–247.
[75] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[76] G.H. Wu, R. Mallipeddi, P.N. Suganthan, R. Wang, H.K. Chen, Differential Evolution with multi-population based ensemble of mutation strategies, Inf. Sci. 329 (2016) 329–345.
[77] M. Yang, C.H. Li, Z.H. Cai, J. Guan, Differential Evolution with auto-enhanced population diversity, IEEE Trans. Cybern. 45 (2) (2015) 302–315.
[78] W.C. Yi, Y.Z. Zhou, L. Gao, X.Y. Li, J.H. Mou, An improved adaptive differential evolution algorithm for continuous optimization, Expert Syst. Appl. 44 (2016) 1–12.
[79] A. Zamuda, J. Brest, Self-adaptive control parameters' randomization frequency and propagations in differential evolution, Swarm Evol. Comput. 25 (2015) 72–99.
[80] A. Zamuda, Adaptive constraint handling and success history Differential Evolution for CEC 2017 constrained real-parameter optimization, in: 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, Spain, 2017, https://doi.org/10.1109/CEC.2017.7969601.
[81] J. Zhang, A.C. Sanderson, JADE: adaptive Differential Evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (5) (2009) 945–958.
[82] X. Zhang, X. Zhang, Improving differential evolution by differential vector archive and hybrid repair method for global optimization, Soft Comput. 21 (23) (2017) 7107–7116.
[83] J.J. Zhou, X.F. Yao, Multi-population parallel self-adaptive differential artificial bee colony algorithm with application in large-scale service composition for cloud manufacturing, Appl. Soft Comput. 56 (2017) 379–397.
[84] W. Zhu, Y. Tang, J.A. Fang, W.B. Zhang, Adaptive population tuning scheme for differential evolution, Inf. Sci. 223 (2013) 164–191.