Information Sciences 468 (2018) 117–141

Contents lists available at ScienceDirect

Information Sciences

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

L-SHADE optimization algorithms with population-wide inertia

Adam P. Piotrowski

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

ARTICLE INFO

Article history:
Received 16 March 2018
Revised 7 August 2018
Accepted 12 August 2018
Available online 13 August 2018

Keywords:
Differential evolution
Evolutionary algorithms
Particle swarm optimization
Benchmark problems
Comparison of metaheuristics
Linear population size reduction

ABSTRACT

Numerous Differential Evolution algorithms (DE) have been proposed during last twenty years for numerical optimization problems. Recently a number of successful history-based adaptive DE variants with linear population size reduction (L-SHADE) have been considered among the most efficient Evolutionary Algorithms. Various L-SHADE algorithms become the winners of numerous IEEE CEC competitions. In this study we show that the performance of L-SHADE variants may be improved by adding a population-wide inertia term (PWI) to the mutation strategies. The PWI term represent an averaged direction and size of moves that were successful in the previous generation. Hence, by introducing PWI into mutation strategy we boost the moves of L-SHADE individuals in the direction that on average led to the improvement in the previous generation. The PWI term is implemented into four L-SHADE variants proposed during 2014–2018 period. Empirical tests are performed on 60 artificial benchmark problems from IEEE CEC'2014 and IEEE CEC'2017 test sets, and on 22 real-world problems from IEEE CEC'2011. For each considered test set every L-SHADE variant performs better with PWI term than without it. Finally, four PWI-based L-SHADE variants are compared against 16 different metaheuristics. For each among three sets of problems PWI-based L-SHADE algorithms win the comparison, and only two among 16 other metaheuristics may be considered competitive on some sets of problems.

1. Introduction

Differential Evolution (DE) algorithms [39] are frequently considered among the most effective and popular population-based Evolutionary Algorithms for single-objective continuous optimization problems [9,29]. Of course, as the idea of general superiority of metaheuristics must be forgotten since the advent of No Free Lunch theorems [46], this effectiveness is inevitably limited to the problems that may be of interest to researchers [21]. Since the development of the basic DE version in mid-1990's [39], very large number of DE variants has been proposed. The rich history of DE research will not be given here, as it may be found in the numerous reviews [9,29], some of which offer an insight into hierarchical evolution of DE algorithms [1,35], or address a frequently neglected problem of theoretical analysis of DE behaviour [30].

Apart from hybridizing DE with other algorithms, the main ways of research in DE include the development of mutation strategies that govern the moves of individuals, and the methods of adaptive modification of DE control parameters during run. Although many approaches to both problems have beet discussed in the literature (see [1,9]), during last decade the ideas developed within a specific sub-class of DE called JADE [31,35,49] turned out especially successful. Among various JADE variants proposed in recent years, those that follow successful history-based adaptive DE (SHADE [40]) and its variant

E-mail address: adampp@igf.edu.pl

https://doi.org/10.1016/j.ins.2018.08.030
0020-0255/

with linear population size reduction (L-SHADE, [41]) attract main attention, and were a subject of a recent review [35]. Many L-SHADE-based variants have become the winners of recent IEEE Competitions in Evolutionary Computation (IEEE CEC). The first SHADE algorithm was ranked the fourth best method in CEC'2013 competition, L-SHADE was the winner of CEC'2014 competition, SPS-L-SHADE-EIG [18] was the winner of CEC'2015 competition, L-SHADE-EpSin [2] was the joint-winner of CEC'2016 competition, and three other L-SHADE-based algorithms (jSO [7], L-SHADE-cnEpSin [4] and L-SHADE-SPACMA [27]) were among four best methods in CEC'2017 competition. SPS-L-SHADE-EIG [18] was also found to be the best approach among 33 various metaheuristics tested on 22 real-world problems from different fields of science when computational budget (or the number of allowed function calls) is large [33]. L-SHADE-based algorithms have also found various practical applications, like minimizing distortion in power system [5] or locating the wind turbines in a windfarm [6], and were introduced into constrained optimization in [48].

Considering the performance and large popularity of L-SHADE algorithms, in this study their simple modification is proposed by adding a population-wide inertia (PWI) term into the so-called DE/current-to-*p*best/1 mutation strategy [49] that is commonly used by almost all L-SHADE variants. PWI vector represents the average direction and size of successful moves performed by individuals in the previous generation. Adding PWI term into the mutation strategy means that the moves performed by individuals in the subsequent generation will be pushed in the direction that was successful in the previous generation. The PWI approach is partly based on the idea of inertia weight introduced into PSO by Shi and Eberhart [37], a DEEP algorithm that remembers its evolution path [23], or conjugate gradient methods [20], that also keep information on previous directions for the subsequent moves. As the PWI vector depends on the successful moves performed in the previous generation only, values of PWI coordinates change dynamically and quickly during search, what allows the algorithm to benefit from the recent experience, but prevents trapping population in the local optima. The concept of PWI require only one additional control parameter, namely defining the minimum number of successful individuals in the particular generation that are needed to compute their average move, which impact is discussed in details in Section 5.2. The proposed PWI-based mutation is implemented into four L-SHADE variants, including the basic L-SHADE [41], L-SHADE-cnEpSin [4], L-SHADE-SPACMA [27] and L-SHADE-50 [34] (which is a simplified version of L-SHADE-EpSin [2], a co-winner of CEC'2016 competition) and is tested on large number of benchmarks and real-world problems.

2. Differential evolution

In this section we briefly introduce the basic DE algorithm. Without the loss of generality we consider continuous minimization problems and search for the global minimum solution $\mathbf{x}^*$:

$$f(\mathbf{x}*) = \min_{\mathbf{x}\in\mathbf{\Omega}} f(\mathbf{x}) \tag{1}$$

where $f(\mathbf{x})$ is the real-valued function, $\mathbf{x}$ is a D-dimensional vector, $\mathbf{x} = \{x^1, ..., x^D\}$, with the domain $\Omega \subseteq \mathbf{R}^D$.

The basic population-based DE algorithm [39] works as follows. First, in generation $g = 0$ a population of NP individuals (solution vectors) $\mathbf{x}_{i,g} = \{x^1_{i,g}, ..., x^D_{i,g}\}$, $i = 1,...,NP$, is randomly generated from the uniform distribution

$$x^j_{i,0} = L^j + rand^j_i(0,1) \cdot (U^j - L^j); \quad j = 1, ..., D; \quad i = 1, ..., NP \tag{2}$$

where $rand_i{}^j(0,1)$ is a random value within [0,1] interval generated separately for each jth element of ith individual and L^j and U^j are problem-specific bounds (box-constraints) that define the subset $\prod_{j=1}^{D} [L^j, U^j]$ of the search space $\mathbf{R}^D$. NP, called population size, is the first control parameter of DE that has to be determined by the user. In each subsequent generation g all individuals $\mathbf{x}_{i,g}$ ($i = 1,..., NP$) perform mutation, crossover and selection in a loop. Many mutation variants have been proposed for DE algorithms, but in the initial DE so called DE/rand/1 mutation is used, in which the parent individual $\mathbf{x}_{i,g}$ is mutated as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F \cdot (\mathbf{x}_{r2,g} - \mathbf{x}_{r3,g}) \tag{3}$$

where $r1$, $r2$ and $r3$ are randomly selected and mutually different integers from the range [1, NP] (which are also different from parent individual, hence $r1 \neq r2 \neq r3 \neq i$) and F (so called scale factor) is the second control parameter of DE. After that, crossover between the trial vector $\mathbf{v}_{i,g}$ and $\mathbf{x}_{i,g}$ occur

$$u^j_{i,g} = \begin{cases} v^j_{i,g} & if \quad rand^j_i(0,1) \le CR \quad or \quad j = j_{rand,i} \\ x^j_{i,g} & otherwise \end{cases} \tag{4}$$

and some boundary handling is applied (for an example see [49]), to assure that all solutions are within bounds. In Eq. (4) CR, the probability of crossover, is the third control parameter of the basic DE, and $j_{rand,i}$ is a randomly selected integer from [1, D] range. The objective function is called for $\mathbf{u}_{i,g}$ and $f(\mathbf{u}_{i,g})$ is compared with $f(\mathbf{x}_{i,g})$ in operation called selection

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & if \quad f(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & otherwise \end{cases} \tag{5}$$

According to the above selection rule, for the next generation the better among the ($\mathbf{u}_{i,g}$, $\mathbf{x}_{i,g}$) pair is proceeded, and the poorer solution is discarded. The above procedures are repeated for each among *NP* individuals, creating the population of *NP* solutions for the next generation $(g+1)$. The algorithm proceeds until the stopping conditions are reached, which are often defined by setting the maximum number of function calls (*MNFC*).

All three control parameters of the basic DE [39], namely *NP*, *F*, and *CR*, are fixed by the user and do not change during run. The most frequently *NP* is either related to the problem dimensionality *D* by $NP = 5 \cdot D$ or $NP = 10 \cdot D$, or set fixed, irrespectively of problem dimensionality, to some value between 50 and 200 [9,39]. The values of *F* and *CR* are often set between 0 and 1, but in the case of *F* one may find various exceptions [9]. However, the basic version of DE with fixed choices of control parameters is now rather outdated, and such parameters are adaptively modified during run in the majority of recent DE versions [9], including L-SHADE [41].

3. L-SHADE

In this section we briefly introduce L-SHADE [41] algorithm and its three modified versions (L-SHADE-cnEpSin [4], L-SHADE-SPACMA [27] and L-SHADE-50 [34]), for which the PWI approach proposed in this study will be implemented in Section 4. Detailed discussion on history of development of various L-SHADE-based variants may be found in a recent review [35].

3.1. L-SHADE

In L-SHADE [41] algorithm the large initial population size is gradually reduced during run, what is marked by index *g* (NP_g). L-SHADE population size is initialized according to Eq. (2) with $NP_0 = 18 \cdot D$ (where *D* is the problem dimensionality), and decreases linearly in subsequent generations to just 4 individuals at the end of the search.

For effective adaptation of control parameters, L-SHADE uses five memory sets. In order to maintain algorithm diversity, an external archive **A** (set **A** is initially empty ($\mathbf{A} = \emptyset$), but its maximum size varies during run by $|\mathbf{A}| = NA_g$, where NA_g is linearly related to NP_g) is used to preserve parent vectors which were worse than the trial vectors during selection. Sets **MF** and **MCR** of predefined size $|\mathbf{MF}| = |\mathbf{MCR}| = H$ (where *H* is a control parameter of L-SHADE algorithm) store in memory the information on crossover (*MCR*) and scale factor (*MF*) values that performed well in the past generations. They will be used to generate individual-specific CR_i and F_i values for each particular generation. All elements in **MF, MCR** are initially set to 0.5. Finally, **SF** and **SCR** sets store all CR_i and F_i values that generated a trial vector which turned out better than the parent in particular generation. **SF** and **SCR** sets are initially empty, and are emptied at the beginning of each generation; their size varies during the run, as it is equal to the number of successful offspring in each generation.

L-SHADE works as follows. At the beginning of each generation for every individual $\boldsymbol{x}_{i,g}$ from population $\mathbf{P_g}$ of current size $|\mathbf{P_g}| = NP_g$ control parameters' values F_i and CR_i are generated according to:

$$ri = randi(1, H) \tag{6}$$

$$F_i = randc_i(MF_{ri}, 0.1) \tag{7}$$

$$CR_i = \begin{cases} 0 & if \quad MCR_{ri} = -1 \\ randn_i(MCR_{ri}, 0.1) & otherwise \end{cases} \tag{8}$$

where *randi* is a random integer generated within a specified interval [1, *H*], $randc_i$ and $randn_i$ are random numbers generated respectively from Cauchy and Normal distributions; all these values are generated independently for each individual $\boldsymbol{x}_{i,g}$. When either F_i or CR_i is outside [0,1] interval, it is set to the closest bound (0 or 1), with the exception of the case $F_i \leq 0$, when F_i is regenerated. The specific MCR_k values may equal -1; such value is attributed to MCR_k if in generation when such *k*th element of **MCR** was set improvements were obtained only with CR_i control parameters that were equal to 0. Then, in each generation g, the following DE/current-to-*p*best/1 mutation strategy taken from JADE [49] is used to create a trial vector

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_i \cdot (\mathbf{x}_{pbest,g} - \mathbf{x}_{i,g}) + F_i \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \tag{9}$$

where $\mathbf{x}_{pbest,\,g}$ is, for each *i* and *g*, randomly selected from the *p*% of best individuals in the population $\mathbf{P_g}$ (*p*% is another control parameter of L-SHADE), and *r2* is randomly selected from the union of the current population $\mathbf{P_g}$ and the external archive **A**. External archive is initially empty, and during the run is gradually filled with parent individuals $\boldsymbol{x}_{i,g}$ that loose the selection procedure (Eq. (5)) against the offspring, until the maximum archive size equal to NA_g is reached. Then each time the new individual $\boldsymbol{x}_{i,g}$ is to be moved into archive after losing selection, one randomly selected individual from the archive is discarded (as NA_g decreases with *g*, a respective number of randomly selected individuals is also discarded whenever NA_g decrease). After mutation the crossover is performed, but as CR_i values vary for each individual (and generation *g*), Eq. (4) needs to be re-written as

$$u_{i,g}^{j} = \begin{cases} v_{i,g}^{j} & if \quad rand_i^j(0,1) \leq CR_i \quad or \quad j = j_{rand,i} \\ x_{i,g}^{j} & otherwise \end{cases} \tag{10}$$

Following crossover, bounds handling procedure proposed in [49] is applied as follows:

$$u_{i,g}^{j} = \begin{cases} \left(L^j + x_{i,g}^{j}\right)/2 & if \quad u_{i,g}^{j} < L^j \\ \left(G^j + x_{i,g}^{j}\right)/2 & if \quad u_{i,g}^{j} > G^j \end{cases} \tag{11}$$

Selection in L-SHADE is performed as in the basic DE following Eq. (5), with one exception – if the parent individual loses, it is moved to the archive **A**. In L-SHADE the control parameters' values F_i and CR_i that in generation g were used by those newly created individuals that have won the selection procedure (Eq. (5)) are stored in memory sets **SCR** and **SF**. At the beginning of each generation g, both memories are cleared ($\mathbf{SCR} = \emptyset$, $\mathbf{SF} = \emptyset$). Hence after each generation g they include only control parameters' values that turned out successful in this generation g. When generation g ends, if at least one improvement was noted (hence **SCR** and **SF** are not empty) the kth element of **MF** and **MCR** (k is initialized with 1 and after each generation is increased by 1 until it reaches H; when $k = H$ in particular generation g, in the generation $g+1$ index k is set back to 1) is replaced as follows

$$MF_k = \frac{\sum_{l=1}^{|SF|} w_l \cdot SF_l^2}{\sum_{l=1}^{|SF|} w_l \cdot SF_l} \tag{12}$$

$$MCR_k = \frac{\sum_{l=1}^{|SCR|} w_l \cdot SCR_l^2}{\sum_{l=1}^{|SCR|} w_l \cdot SCR_l} \tag{13}$$

$$w_l = \frac{\Delta f_l}{\sum_{a=1}^{|SCR|} \Delta f_a} \tag{14}$$

$$\Delta f_l = \left| f(u_{l,g}) - f(x_{l,g}) \right| \tag{15}$$

Note that always $|SF| \equiv |SCR|$. If improvements were obtained only with CR_i control parameters that were equal to 0, then MCR_k is set to -1 (instead of using Eq. (13)). If there were no improvements in generation g, MF_k and MCR_k remain unchanged.

At the end of each generation the linear population size reduction is performed by discarding the $NP_{g-1} - NP_g$ worst individuals from the population $\mathbf{P_g}$, with

$$NP_{g+1} = round\left(\frac{NP_{\min} - NP_0}{MNFC} \cdot NFC + NP_0\right) \tag{16}$$

where *round* procedure returns the nearest integer number; NP_0 is the population size at the beginning of the search; NP_{min} is the lowest possible population size; NFC is the number of already used function calls; $MNFC$ is the maximum number of allowed function calls.

The algorithm proceeds until the maximum number of function calls is used. The control parameter values of L-SHADE are set to: $NP_0 = 18 \bullet D$, $NP_{min} = 4$, $H = 6$, $NA_g = 2.6 \bullet NP_g$ and $p\% = 11\%$ (or 0.11), following [41].

3.2. L-SHADE-50

The L-SHADE-50 algorithm [34] differs from L-SHADE by keeping the values of control parameters F_i fixed to 0.5 for each i during the first half of the search, hence as long as $NFC < 0.5 \cdot MNFC$. During the second half of the search the control parameters F_i follows adaptive scheme as in L-SHADE. L-SHADE-50 was developed as a significant simplification of L-SHADE-EpSin [2], the co-winner of IEEE CEC'2016, hence keeps the control parameters used in L-SHADE-EpSin, namely: $NP_0 = 18 \bullet D$, $NP_{min} = 4$, $H = 5$, $NA_g = 1.4 \bullet NP_g$ and $p\% = 11\%$. For detailed discussion on reasons of simplifying L-SHADE-EpSin into L-SHADE-50, and on the importance of keeping F_i fixed during large part of the search, see [34].

3.3. L-SHADE-cnEpSin

L-SHADE-cnEpSin [4] algorithm is also a modified version of L-SHADE-EpSin [2], the co-winner of IEEE CEC2016. Our description of L-SHADE-cnEpSin is based on [4] and MATLAB code available from IEEE CEC'2017 web page (http://www.ntu.edu.sg/home/EPNSugan). L-SHADE-cnEpSin works as L-SHADE with the two differences discussed in sub-Sections 3.3.1 and 3.3.2.

3.3.1. Adaptation of F_i *by means of two sinusoidal functions*

During the first 50% of function calls L-SHADE-cnEpSin uses different method of F_i adaptation than L-SHADE. During the second half of the run (when $NFC \geq 0.5 \cdot MNFC$) the F_i values are updated as in L-SHADE. As long as $NFC < 0.5 \cdot MNFC$, the values of F_i at generation g are updated using this one among the following two sinusoidal functions

$$F_i = 0.5 \cdot \left(\sin\left(2\pi \cdot freq \cdot g + \pi\right) \cdot \frac{g_{\max} - g}{g_{\max}} + 1 \right) \tag{17}$$

$$F_i = 0.5 \cdot \left(\sin\left(2\pi \cdot fq_i \cdot g\right) \cdot \frac{g}{g_{\max}} + 1 \right) \tag{18}$$

that produced relatively larger number of successful offspring during the recent 20 generations. In Eqs. (17) and (18) g and g_{max} are the current generation number and the predefined maximum number of generations, *freq* is the control parameter with fixed, predefined value, and fq_i is yet another control parameter, whose values are adaptively modified in a similar way as the values of CR_i. As long as $NFC \leq 0.5 \cdot MNFC$, at the beginning of each generation g for each individual $\boldsymbol{x}_{i,g}$ in the population $\mathbf{P_g}$, the specific value of fq_i is generated as

$$fq_i = randc_i(MFQ_{ri}, 0.1) \tag{19}$$

where $randc_i$ is a random number generated from Cauchy distribution and ri is the index generated by Eq. (6) (the same as used for CR_i and F_i). The value of fq_i is regenerated if $fq_i \leq 0$ or truncated to 1 if $fq_i > 1$. MFQ_{ri} is selected randomly from an additional external memory **MFQ** ($|\mathbf{MFQ}| = H$, each element is set initially to 0.5) that stores the mean values of successful fq_i from the former generations. Values of fq_i that are successful in particular generation are remembered in the set **SFQ** whose size depends on the number of successful offspring in a particular generation. At the beginning of each generation **SFQ** is cleared ($\mathbf{SFQ} = \emptyset$), and during current generation g each value of fq_i associated with each offspring $\boldsymbol{u}_{i,g}$ that wins selection procedure according to Eq. (5) is remembered within memory set **SFQ**. Note that in L-SHADE-cnEpSin fq_i is remembered within **SFQ** irrespectively which equation, (17) or (18), was used to generate $\boldsymbol{u}_{i,g}$, even though fq_i had no impact on $\boldsymbol{u}_{i,g}$ if Eq. (17) was used. After generation g, if at least one improvement was noted, the kth element of **MFQ** (k is the same index as used in Eqs. (12) and (13)) is replaced by

$$MFQ_k = \frac{\sum_{l=1}^{|SFQ|} w_l \cdot SFQ_l{}^2}{\sum_{l=1}^{|SFQ|} w_l \cdot SFQ_l} \tag{20}$$

where w_l is defined as in Eqs. (14) and (15).

To specify which sinusoidal function Eqs. (17) or ((18)) will be used in generation g, the following approach is applied. For each among the first 20 generations (so called learning period, *LP*) which among two sinusoidal functions is used is determined randomly, with equal probability set to 0.5. In each subsequent generation the values $\mathrm{pf}_{1,g}$ and $\mathrm{pf}_{2,g}$ are determined as follows

$$pf_{1,g} = \frac{S_{1,g}}{S_{1,g} + S_{2,g}} \tag{21}$$

$$pf_{2,g} = \frac{S_{2,g}}{S_{1,g} + S_{2,g}} \tag{22}$$

where

$$S_{1,g} = \frac{\sum_{i=g-LP}^{g-1} ns_{1,i}}{\sum_{i=g-LP}^{g-1} ns_{1,i} + \sum_{i=g-LP}^{g-1} nf_{1,i}} + \varepsilon \tag{23}$$

$$S_{2,g} = \frac{\sum_{i=g-LP}^{g-1} ns_{2,i}}{\sum_{i=g-LP}^{g-1} ns_{2,i} + \sum_{i=g-LP}^{g-1} nf_{2,i}} + \varepsilon \tag{24}$$

and if $\mathrm{pf}_{1,g} > pf_{2,g}$, the sinusoidal function defined by Eq. (17) is used, otherwise the one defined by Eq. (18) is applied. In Eqs. (23) and (24) $\mathrm{ns}_{1,g}$ and $\mathrm{ns}_{2,g}$ means the number of successes achieved in each among *LP* (set to 20) recent generations when sinusoidal functions defined by Eqs. (17) ($ns_{1,i}$) and (18) ($ns_{2,i}$) were used; $\mathrm{nf}_{1,g}$ and $\mathrm{nf}_{2,g}$ are respective numbers of failures. However, when the particular sinusoidal function has not been used at generation g (as values of $\mathrm{pf}_{1,g}$ and $\mathrm{pf}_{2,g}$ depend on generation, not individual, the same sinusoidal function is used for the whole generation), the respective values of $\mathrm{ns}_{1,g}$ and $\mathrm{nf}_{1,g}$ (or $\mathrm{ns}_{2,g}$ and $\mathrm{nf}_{2,g}$) are set to 1, not to 0. ε in Eqs. (23) and (24) is set to 0.01.

Note that even though during the first 50% function calls F_i are not generated as in L-SHADE approach, still the successful values of F_i obtained during particular generation g are stored in **SF** memory, and values for **MF** set are computed according to Eq. (12). Since $NFC \geq 0.5 \cdot MNFC$, the values of F_i at subsequent generation are updated as in L-SHADE by Eq. (7), using the current elements from **MF** set.

Following the debate given in [34] we may express some doubts whether the complicated procedure discussed in this sub-section is useful in L-SHADE-cnEpSin. Possibly the procedure could be much simplified, as was the case of L-SHADE-EpSin [2]. However, in this study we aim at introducing PWI-based mutation, not at critical discussion of each considered variant of L-SHADE. Hence, we introduce PWI term directly into the version of L-SHADE-cnEpSin algorithm proposed in [4].

3.3.2. Using covariance matrix learning with euclidean neighborhood

In L-SHADE-cnEpSin the crossover may be performed in one of two ways. With the probability pc (which is a control parameter of the algorithm) the crossover for particular individual at particular generation is performed by means of eigenvector-based operation, otherwise classical crossover is applied. The eigenvector-based crossover is defined as follows.

At the beginning of each generation g the Euclidean distance between each individual and the best individual in the population ($\boldsymbol{x}_{best,g}$) is found and $\boldsymbol{x}_{best,g}$, together with its $NP \cdot pb - 1$ (pb being another control parameter of L-SHADE-cnEpSin) closest individuals, form a set of $k = 1,\ldots,NP \cdot pb$ solutions that are used to compute covariance matrix $\mathbf{C}_g$ ($D \times D$). Each element of $\mathbf{C}_g$ is defined as

$$c_g(a,b) = \frac{\sum_{k=1}^{NP \cdot pb} \left(x_{k,g}^a - \overline{x_g^a}\right)\left(x_{k,g}^b - \overline{x_g^b}\right)}{NP \cdot pb - 1}$$
$$\overline{x_g^a} = \frac{\sum_{k=1}^{NP \cdot pb} x_{k,g}^a}{NP \cdot pb} \tag{25}$$

where a and b are two dimensions of D, and $\overline{x_g^b}$ is computed similarly to $\overline{x_g^a}$, but for dimension b. Then matrix of eigenvectors ($\mathbf{E}_g$) of $\mathbf{C}_g$ are obtained by its factorization

$$\mathbf{C}_g = \mathbf{E}_g \mathbf{H}_g (\mathbf{E}_g)^T \tag{26}$$

where $\mathbf{H}_g$ is a diagonal matrix which diagonal elements are the corresponding eigenvalues of $\mathbf{E}_g$.

At the beginning of each generation g a random number $rand_g$ is generated from [0,1] interval. If $rand_g < pc$ then eigenvector-based crossover is performed by all individuals in the generation g in Eigen coordinate system, hence

$$\begin{aligned} \mathbf{x}'_{i,g} &= (\mathbf{E}_g)^T \mathbf{x}_{i,g} \\ \mathbf{v}'_{i,g} &= (\mathbf{E}_g)^T \mathbf{v}_{i,g} \end{aligned} \tag{27}$$

$$u'^j_{i,g} = \begin{cases} v'^j_{i,g} & if \quad rand_i^j(0,1) \leq CR_i \quad or \quad j = j_{rand,i} \\ x'^j_{i,g} & otherwise \end{cases} \tag{28}$$

and finally

$$\mathbf{u}_{i,g} = \mathbf{E}_g \mathbf{u}'_{i,g} \tag{29}$$

Otherwise (if $rand_g \geq pc$) the classical crossover defined in Eq. (10) is used for the whole population.

In L-SHADE-cnEpSin the value of $H = 5$, $freq = 0.5$, $ps = 0.5$ and $pc = 0.4$, other control parameters are set the same as for L-SHADE algorithm.

3.4. L-SHADE-SPACMA

L-SHADE-SPACMA [27] is a hybridized version of CMA-ES [19] and L-SHADE with so-called semi-parameter adaptation of F_i. The role of CMA-ES is basically restricted to serving as a variant of mutation strategy within L-SHADE framework, as discussed below. We describe L-SHADE-SPACMA algorithm according to [27] and MATLAB code available on IEEE CEC'2017 web page (http://www.ntu.edu.sg/home/EPNSugan). L-SHADE-SPACMA differs from the classical L-SHADE algorithm (discussed in Section 3.1) in three features that are discussed in the subsequent sub-sections: 1. the elements taken from CMA-ES, 2. the selection of either L-SHADE or CMA-ES-based mutation, 3. the way semi-parameter adaptation of F_i is performed.

3.4.1. CMA-ES elements in L-SHADE-SPACMA

CMA-ES algorithm is a well known Evolutionary Strategy [19]. Within L-SHADE-SPACMA the CMA-ES version from [19] is hybridized with L-SHADE.

In each generation g of L-SHADE-SPACMA each individual $\boldsymbol{x}_{i,g}$ creates offspring $\boldsymbol{v}_{i,g}$ either by means of DE/current-to-*p*best/1 mutation used in L-SHADE (Eq. (9)), or by means of CMA-ES-based equation (how this choice is performed will be discussed in Section 3.4.2) that is based on multivariate Gaussian distribution

$$\mathbf{v}_{i,g} = N\left(\mathbf{m}_g, {\sigma_g}^2 \mathbf{C}_g\right) \tag{30}$$

In Eq. (30) $\boldsymbol{m}_{g=0}$ is an initial D-dimensional vector, which elements are randomly generated within [0,1], and in subsequent generations g location of $\boldsymbol{m}_g$ is computed as a weighted average of μ best individuals from the population $\mathbf{P}_g$

$$\mathbf{m}_{g+1} = \sum_{i=1}^{\mu} w_{i,g}\mathbf{xs}_{i:\lambda,g} \tag{31}$$

$$w_{i,g} = wt_{i,g} / \sum_{i=1}^{\mu} wt_{i,g} \tag{32}$$

$$wt_{i,g} = \ln(\mu + 0.5) - \ln(i) \tag{33}$$

In Eq. (31) $\mathbf{xs}_{i:\lambda,g}$ means i^{th} best individual from the population $\mathbf{P_g}$ (e.g. $\mathbf{xs}_{1,g}$ is the best, $\mathbf{xs}_{\lambda,g}$ is the λ^{th} best individual in the population $\mathbf{P}_g$ in generation g, λ being the CMA-ES parameter) and $\mu = \lfloor NP_g/2 \rfloor$ (however, $\mu = NP_g/2$ is used in the code given at http://www.ntu.edu.sg/home/EPNSugan).

The step size σ_g and the covariance $\mathbf{C}_g$ in Eq. (30) are initialized as $\sigma_0 = 0.5$ and $\mathbf{C}_0 = \mathbf{I}$. We skip here the lengthy discussion how σ_g and $\mathbf{C}_g$ are modified in each generation and refer the reader to classical publication by Hansen [19]. We must, however, note that in L-SHADE-SPACMA the factor c_σ/d_σ discussed in [19] is computed only for the initial population, despite the fact that NP_g changes during run in L-SHADE-SPACMA.

It is important to remember that all trial vectors $\boldsymbol{v}_{i,g}$ (both those generated by Eq. (9) and by Eq. (30)) follow crossover (Eq. (10)) and bounds handling and selection (Eq. (11)), and that after each generation of L-SHADE-SPACMA the modification of CMA-ES parameters is based on the whole population $\mathbf{P}_g$, not only on the solutions that were generated during mutation based on CMA-ES (Eq. (30)). Hence, CMA-ES-based part of L-SHADE-SPACMA is not non-elitist approach, contrary to the classical CMA-ES [19], as Eq. (30) is used just as a kind of specific mutation strategy within L-SHADE framework.

3.4.2. Choosing L-SHADE or CMA-ES-based mutation

Whether L-SHADE or CMA-ES-based mutation is used by particular individual $\boldsymbol{x}_{i,g}$ in the particular generation g is governed by adaptively modified control parameter $FCP_{i,g}$. If

$$FCP_{i,g} \geq rand(0,1) \tag{34}$$

L-SHADE-based mutation (Eq. (9)) is used in generation g by $\boldsymbol{x}_{i,g}$, otherwise CMA-ES-based one (Eq. (30)) is applied for this individual. Values of $rand(0,1)$ are generated for each i and g.

The value of $FCP_{i,g}$ follows similar adaptation as F_i or CR_i. Initially, **MFCP** memory is created with H values of $FCP_{i,g}$, each equal to 0.5. At every generation g for each individual $\boldsymbol{x}_{i,g}$ one among H values of FCP stored in **MFCP** is chosen according to Eq. (6) and this value is used in Eq. (34).

After the generation g, if at least one improvement was noted, the kth element of **MFCP** (like in case of **MF** and **MCR**, k is initialized with 1 and after each generation is increased by 1 until it reaches H; when $k = H$ in particular generation g, in the generation $g+1$ index k is set back to 1) is modified as follows:

$$MFCP_k = cp \cdot MFCP_k + (1 - cp) \cdot \frac{dw1}{dw1 + dw2} \tag{35}$$

where

$$\begin{aligned} dw1 &= \sum_{\forall i \in L-SHADE \& f(x_{i,g}) > f(u_{i,g})} f(x_{i,g}) - f(u_{i,g}) \\ dw2 &= \sum_{\forall i \in CMA-ES \& f(x_{i,g}) > f(u_{i,g})} f(x_{i,g}) - f(u_{i,g}) \end{aligned} \tag{36}$$

If $MFCP_k$, the output from Eq. (35), is outside [0.2,0.8] bounds, it is truncated to 0.2 or 0.8, respectively. By $i \in L-SHADE$ we understand that $\boldsymbol{x}_{i,g}$ individual used L-SHADE-based mutation (Eq. (9)) in generation g; $i \in CMA-ES$ means that $\boldsymbol{x}_{i,g}$ individual used CMA-ES-based mutation (Eq. (30)). Hence, $dw1$ is the sum of improvements achieved by L-SHADE-based mutation in generation g; $dw2$ is the respective sum of improvements obtained by CMA-ES-based mutation. In Eq. (35) cp is a control parameter of the algorithm.

3.4.3. Semi-parameter adaptation of F_i

In semi-parameter adaptation approach introduced in L-SHADE-SPACMA F_i is, like in L-SHADE-50 and L-SHADE-cnEpSin, treated differently during the first and the second half of the search. As long as $NFC < 0.5 \cdot MNFC$, the values of F_i are generated randomly for each individual at every generation according to

$$F_i = 0.45 + 0.1 \cdot rand(0,1) \tag{37}$$

hence they do not follow any adaptation method, but are also not fully fixed. Since $NFC \geq 0.5 \cdot MNFC$ the values of F_i follows L-SHADE-based adaptation approach.

In L-SHADE-SPACMA the values of $H = 5$ and $cp = 0.8$, other control parameters are set the same as for L-SHADE and CMA-ES (see [27] and code in http://www.ntu.edu.sg/home/EPNSugan).

4. Population-wide inertia in L-SHADE

In the majority of DE algorithms, including L-SHADE variants, in each generation every individual makes a move in a specific direction. Such moves may be of very different length, and may lead to improvement (hence be successful) or not. Only successful moves modify the location of individuals in the next generation. If in the particular generation the majority of successful moves is performed in similar directions, it may be due to the fitness landscape that improves in roughly that way. The information that better solutions are found in specific direction may be used to focus moves of individuals in the next generation. The simplest way to do so is adding a kind of population-wide inertia (PWI) term, which represent the average direction and size of successful moves performed by individuals in the last generation, into the mutation strategy.

In this section we propose a modification of L-SHADE variants by introducing PWI term into DE/current-to-*p*best/1 [49] mutation strategy (Eq.(9)) that is used by L-SHADE methods. In *D*-dimensional search space the population-wide inertia term is a *D*-dimensional vector that represent population-averaged successful moves from the previous generation (*g*-1), hence it will be denoted as $\mathbf{PWI}_{g\text{-}1}$. It is defined as follows. If the number of improvements obtained during generation *g*-1 (called $\mathrm{NI}_{g\text{-}1}$) is larger than mi_g, then $\mathbf{PWI}_{g\text{-}1}$ is computed as

$$\mathbf{PWI}_{g-1} = \frac{\sum_{i=1}^{NP_{g-1}} \mathbf{mv}_{i,g-1}}{NI_{g-1}} \tag{38}$$

$$\mathbf{mv}_{i,g-1} = \begin{cases} \mathbf{u}_{i,g-1} - \mathbf{x}_{i,g-1} & if \quad f(\mathbf{u}_{i,g-1}) \le f(\mathbf{x}_{i,g-1}) \\ \mathbf{0} & if \quad f(\mathbf{u}_{i,g-1}) > f(\mathbf{x}_{i,g-1}) \end{cases} \tag{39}$$

otherwise (if $\mathrm{NI}_{g\text{-}1} \le mi_g$)

$$\mathbf{PWI}_{g-1} \equiv \mathbf{0} \tag{40}$$

For all individuals that were successful in the last generation *g*-1, $\mathbf{mv}_{i,g-1}$ is simply their move in *D*-dimensional search space performed in generation *g*-1; for unsuccessful individuals $\mathbf{mv}_{i,g-1} = \mathbf{0}$ (hence all coordinates of *D*-dimensional vector are set to 0). Unsuccessful moves are not considered in determining the averaged direction $\mathbf{PWI}_{g\text{-}1}$, hence the sum in Eq. (38) is divided by $\mathrm{NI}_{g\text{-}1}$, the number of successful individuals (not all moves performed). Note that due to the crossover operation, a number of elements in particular $\mathbf{mv}_{i,g-1}$ vector may equal 0 even when the move was successful, because after crossover almost always $\mathbf{u}_{i,g-1}$ contains some elements from $\mathbf{x}_{i,g-1}$.

The role of PWI term is to attract individuals in the direction that turned out successful in the previous generation. However, PWI should not dominate the search by forcing all individuals to move in the specific way; otherwise the exploration capabilities of the algorithm could be lost. Hence, the PWI term is added like another difference vector into the DE/current-to-*p*best/1 mutation strategy, and is multiplied by the same scaling factor F_i as the two classical difference vectors, what may be written as

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_i \cdot (\mathbf{x}_{pbest,g} - \mathbf{x}_{i,g}) + F_i \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) + F_i \cdot \mathbf{PWI}_{g-1} \tag{41}$$

Note that $\mathbf{PWI}_{g\text{-}1}$ vector, being the direction of population-averaged successful moves from the previous generation, is the same for each individual in the particular generation, but varies from generation to generation. Of course, in the initial generation $\mathbf{PWI}_0 \equiv \mathbf{0}$.

The only control parameter added by the proposed modification is mi_g that controls the minimum number of improvements required to compute $\mathbf{PWI}_g$. It prevents situations when the direction of population-averaged successful moves would be based on successes achieved by too few individuals, what could increase the possibility of trapping the algorithm in the local minimum. Note that because we introduce PWI-based mutation into L-SHADE algorithms, which population size vary during run, the value of mi_g may be either related to current population size (NP_g) and accordingly change during run (hence the index *g*), or be fixed to a value between 0 and 3. We assume the upper bound at 3 when mi_g is not related to population size, as mi_g should be lower than the lowest population size in L-SHADE, set classically to 4. We propose to relate the value of mi_g with the current population size by:

$$mi_g = 0.01 \cdot NP_g \tag{42}$$

The impact of this choice on the performance of L-SHADE variants will be discussed in Section 5.2.

The idea of PWI-based mutation is implemented into four L-SHADE variants, namely L-SHADE [41], L-SHADE-50 [34], L-SHADE-cnEpSin [4] and L-SHADE-SPACMA [27]. The pseudocodes of all four methods with PWI-based mutation (hence we add PWI to each name, e.g. L-SHADE-PWI) are given in Figs 1–4. As one may note, PWI-based mutation is very simple and may be easily added to any L-SHADE, or even other DE variants. In the case of L-SHADE-SPACMA, the concept of PWI is used within DE-based mutation, but not within CMA-ES-based equation (Eq. (30)) – see the pseudocode in Fig. 3.

5. Results

The population-wide inertia (PWI) based mutation strategy is tested within four L-SHADE algorithms (L-SHADE [41], L-SHADE-cnEpSin [4], L-SHADE-SPACMA [27] and L-SHADE-50 [34]) on three sets of problems: 50-dimensional artificial benchmarks from CEC'2017 [3] and CEC'2014 [24] (each set is composed from 30 problems), as well as 22 real-world problems

L-SHADE-PWI pseudocode:

---

1. set generation counter $g = 0$ and problem-related information: D (problem dimensionality), $MNFC$ (maximum number of function calls);
2. set control parameters: $NP_{g=0} = 18 \cdot D$, $NP_{min} = 4$, $H = 6$, $p\% = 11\%$, $NA_{g=0} = 2.6 \cdot NP_{g=0}$, $mi_{g=0} = 0.01 \cdot NP_{g=0}$;
3. create memory sets: **MF**, **MCR** ($|\mathbf{MF}| = |\mathbf{MCR}| = H$, $MF_i = MCR_i = 0.5$, $i = 1{:}H$), **A** ($|\mathbf{A}| = 0$), **SF**, **SCR**, $\mathbf{PWI}_{g=0} = \mathbf{0}$;
4. set L^d and U^d as lower and upper bounds of d-th variable ($d = 1,\ldots,D$);
5. initialize population $\mathbf{P_g}$ of individuals $\mathbf{x}_{i,g}$ according to eq. (2);
6. find objective function values $f(\mathbf{x}_{i,g})$ for all individuals in $\mathbf{P_g}$;
7. set the number of function calls used $NFC = NP_0$;
8. find the best solution found so far $\mathbf{x}_{best,0}$;
9. in the first generation the initial population is used, hence $NP_1 = NP_0$;
10. **while** $NFC < MNFC$
11. $g = g + 1$;
12. $|\mathbf{SF}| = |\mathbf{SCR}| = 0$, $|\mathbf{A_{temp}}| = 0$;
13. **for** $i = 1{:}\ NP_g$
14. generate ri according to eq. (6);
15. generate F_i according to eq. (7);
16. generate CR_i according to eq. (8);
17. choose randomly $\mathbf{x}_{pbest,g}$ for i among round($NP_g \cdot p\%$) best individuals in $\mathbf{P_g}$;
18. perform mutation following eq. (41);
19. perform crossover following eq. (10);
20. perform bounds handling following eq. (11) and selection following eq. (5);
21. **if** $f(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{i,g})$
22. $\mathbf{x}_{i,g} \rightarrow \mathbf{A_{temp}}$, $CR_i \rightarrow \mathbf{SCR}$, $F_i \rightarrow \mathbf{SF}$;
23. **end if;**
24. $NFC = NFC + 1$;
25. **end for;**
26. compute mi_g according to eq. (42);
27. compute $\mathbf{PWI_g}$ according to eqs (38)-(40);
28. reduce population size to NP_{g+1} following eq. (16);
29. $\mathbf{A} \leftarrow \mathbf{A} \cup \mathbf{A_{temp}}$;
30. reduce the archive size to $NA_{g+1} = \lceil 2.6 \cdot NP_{g+1} \rceil$ by discarding random individuals from **A**;
31. update **MF**, **MCR** following eqs (12), (13);
32. **end while;**

Fig. 1. The pseudocode of L-SHADE-PWI algorithm.

of various dimensionality from different fields of science (CEC'2011 [8]). Note that technically in [8] the first ten CEC'2011 problems were called F1-F10, the next ten problems were called F11.1-F11.10 and the last two problems were called F12-F13; for simplicity we call such 22 problems F1-F22 (problems F11.1-F11.10 are in this study named F11-F20), to have similar nomenclature as in CEC'2017 and CEC'2014. The control parameters of all four L-SHADE variants are set as in the source papers (details are given in section 3). The maximum number of function calls for particular test sets are set as in the source papers, to $10{,}000 \cdot D$ (where D is the problem dimensionality) for CEC'2017 [3] and CEC'2014 [24] problems, and 150,000 for CEC'2011 [8] ones. 51 runs of each algorithm on every problem are performed.

5.1. Impact of PWI-based mutation strategy on L-SHADE algorithms

To find the impact of PWI-term on the performance of L-SHADE algorithms, first the classical versions of all four L-SHADE variants are tested on CEC'2017, CEC'2014 and CEC'2011 problems, and then the same algorithms with PWI-based mutation defined by Eq. (41) are applied for the same problems. To mark the difference between the original and the PWI-based version of each algorithm, we add PWI to the end of the name when PWI-based mutation is used (e.g. we have the basic version L-SHADE-50 and PWI-added version L-SHADE-50-PWI). Separately for each among four L-SHADE-based variants, the statistical significance of the differences in performance between the classical variant and the related PWI-based variant is computed for each problem by Wilcoxon signed-rank test [11] at $\alpha = 0.05$. For each set of problems (CEC'2017, CEC'2014 and CEC'2011) the number of cases when PWI-based version was significantly better (+) than the classical version of particular algorithm, the number of cases when PWI-based version was significantly poorer (−) than the classical algorithm, and the

L-SHADE-cnEpSin-PWI pseudocode:

---

1. set generation counter $g = 0$ and problem-related information: D (problem dimensionality), $MNFC$ (maximum number of function calls);
2. set control parameters: $NP_{g=0} = 18 \cdot D$, $NP_{min} = 4$, $H = 5$, $p\% = 11\%$, $freq = 0.5$, $ps = 0.5$, $pc = 0.4$, $LP = 20$, $mi_{g=0} = 0.01 \cdot NP_{g=0}$;
3. create memory sets: **MF**, **MCR, MFQ** ($|\mathbf{MF}| = |\mathbf{MCR}| = |\mathbf{MFQ}| = H$, $MF_i = MCR_i = MFQ_i = 0.5$, $i = 1{:}H$), **A** ($|\mathbf{A}| = 0$), **SF, SCR, SFQ**, $\mathbf{PWI}_{g=0} = \mathbf{0}$;
4. set L^d and U^d as lower and upper bounds of d-th variable ($d = 1,\ldots,D$);
5. initialize population $\mathbf{P_g}$ of individuals $\mathbf{x}_{i,g}$ according to eq. (2);
6. find objective function values $f(\mathbf{x}_{i,g})$ for all individuals in $\mathbf{P_g}$;
7. set the number of function calls used $NFC = NP_0$;
8. find the best solution found so far $\mathbf{x}_{best,0}$;
9. in the first generation the initial population is used, hence $NP_1 = NP_0$;
10. **while** $NFC < MNFC$
11. $g = g + 1$;
12. $|\mathbf{SF}| = |\mathbf{SCR}| = |\mathbf{SFQ}| = 0$, $|\mathbf{A_{temp}}| = 0$;
13. **if** $g \le LP$
14. *randtemp1* = rand(0,1); % *all F_i values will be generated by the same method in particular g;*
15. **end if;**
16. *randtemp2* = rand(0,1); % *all individuals will follow the same crossover approach in particular g;*
14. **for** $i = 1{:}\ NP_g$
14. generate ri according to eq. (6);
15. generate CR_i according to eq. (8);
16. **if** $NFC < 0.5 \cdot MNFC$
17. generate fq_i according to eq. (19);
17. **if** $g \le LP$
18. **if** *randtemp1* > 0.5
19. generate F_i according to eq. (17);
20. **else**
21. generate F_i according to eq. (18);
22. **end if;**
19. **else**
20. compute pf_1,g and pf_2,g according to eqs (21)-(24);
18. **if** $pf_{1,g} > pf_{2,g}$
19. generate F_i according to eq. (17);
20. **else**
21. generate F_i according to eq. (18);
22. **end if;**
23. **end if;**
23. **else**
24. generate F_i according to eq. (7);
25. **end if;**
26. choose randomly $\mathbf{x}_{pbest,g}$ for i among round($NP_g \cdot p\%$) best individuals in $\mathbf{P_g}$;
27. perform mutation following eq. (41);
28. **if** *randtemp2* < pc
19. perform eigenvector-based crossover for the whole population according to eqs (25)-(29);
20. **else**
21. perform classical crossover for the whole population according to eq. (10)
22. **end if;**
29. perform bounds handling following eq. (11) and selection following eq. (5);
30. **if** $f(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{i,g})$
31. $\mathbf{x}_{i,g} \rightarrow \mathbf{A_{temp}}$, $CR_i \rightarrow \mathbf{SCR}$, $F_i \rightarrow \mathbf{SF}$, $fq_i \rightarrow \mathbf{SFQ}$;
32. **end if;**
33. $NFC = NFC + 1$;
34. **end for;**
35. **if** $NFC < 0.5 \cdot MNFC$
36. compute the number of successful and unsuccessful moves performed by each among sinusoidal functions in generation g ($ns_{1,g}$ and $ns_{2,g}$) that will be needed for eqs (23)-(24);
37. **end if;**
30. compute mi_g according to eq. (42);
31. compute $\mathbf{PWI_g}$ according to eqs (38)-(40);
35. reduce population size to NP_{g+1} following eq. (16);
36. $\mathbf{A} \leftarrow \mathbf{A} \cup \mathbf{A_{temp}}$;
37. reduce the archive size to $NA_{g+1} = \lceil 1.4 \cdot NP_{g+1} \rceil$ by discarding random individuals from **A**;
38. update **MF**, **MCR, MFQ** following eqs (12), (13), (19);
39. **end while;**

Fig. 2. The pseudocode of L-SHADE-cnEpSin-PWI algorithm.

L-SHADE-SPACMA-PWI pseudocode:

---

1. set generation counter $g = 0$ and problem-related information: D (problem dimensionality), $MNFC$ (maximum number of function calls);
2. set control parameters: $NP_{g=0} = 18{\cdot}D$, $NP_{min} = 4$, $H = 5$, $p\% = 11\%$, $cp = 0.8$, $mi_{g=0} = 0.01{\cdot}NP_{g=0}$;
3. create memory sets: **MF**, **MCR, MFCP** ($|\mathbf{MF}| = |\mathbf{MCR}| = |\mathbf{MFCP}| = H$, $MF_i = MCR_i = MFCP_i = 0.5$, $i = 1{:}H$), **A** ($|\mathbf{A}| = 0$), **SF, SCR, PWI**$_{g=0} = \mathbf{0}$;
4. initialize CMA-ES-based parameters (see [19] for details), including: $\sigma_0 = 0.5$, $\mathbf{C}_0 = \mathbf{I}$, $\boldsymbol{m}_{g=0}$ composed of D random values rand(0,1);
4. set L^d and U^d as lower and upper bounds of d-th variable ($d = 1,\ldots,D$);
5. initialize population $\mathbf{P_g}$ of individuals $\mathbf{x}_{i,g}$ according to eq. (2);
6. find objective function values $f(\mathbf{x}_{i,g})$ for all individuals in $\mathbf{P_g}$;
7. set the number of function calls used $NFC = NP_0$;
8. find the best solution found so far $\mathbf{x}_{best,0}$;
9. in the first generation the initial population is used, hence $NP_1 = NP_0$;
10. **while** $NFC < MNFC$
11. $g = g + 1$;
12. $|\mathbf{SF}| = |\mathbf{SCR}| = 0$, $|\mathbf{A}_{\mathbf{temp}}| = 0$;
13. **for** $i = 1{:}\ NP_g$
14. generate ri according to eq. (6);
15. set $FCP_{i,g} = MFCP(ri)$;
15. generate CR_i according to eq. (8);
16. **if** $NFC < 0.5 \cdot MNFC$
17. $F_i = 0.45 + 0.1 \cdot \text{rand}(0,1)$;
18. **else**
19. generate F_i according to eq. (7);
20. **end if;**
21. choose randomly $\mathbf{x}_{pbest,g}$ for i among round($NP_g \cdot p\%$) best individuals in $\mathbf{P_g}$**;**
22. **if** $FCP_{i,g} \geq rand(0,1)$
23. perform DE/current-to-pbest/1-PWI mutation following eq. (41);
24. **else**
25. generate offspring by CMA-ES-based equation following eq. (30);
26. **end if;**
23. perform crossover following eq. (10); % *all individuals follow crossover*
24. perform bounds handling following eq. (11) and selection following eq. (5);
25. **if** $f(\mathbf{u}_{i,g}) \leq f(\mathbf{x}_{i,g})$
26. $\mathbf{x}_{i,g} \rightarrow \mathbf{A_{temp}}$, $CR_i \rightarrow \mathbf{SCR}$, $F_i \rightarrow$ **SF;**
27. **end if;**
28. $NFC = NFC + 1$;
29. **end for;**
30. compute mi_g according to eq. (42);
31. compute $\mathbf{PWI}_g$ according to eqs (38)-(40);
32. reduce population size to NP_{g+1} following eq. (16);
33. $\mathbf{A} \leftarrow \mathbf{A} \cup \mathbf{A}_{\mathbf{temp}}$;
34. reduce the archive size to $NA_{g+1} = \lceil 1.4 \cdot NP_{g+1} \rceil$ by discarding random individuals from **A**;
35. update **MF**, **MCR, MFCP** following eqs (12), (13) and (35);
36. update CMA-ES parameters (see section 3.4.1) based on the whole population (not just individuals that followed CMA-ES-based mutation);
36. **end while;**

Fig. 3. The pseudocode of L-SHADE-SPACMA-PWI algorithm.

L-SHADE-50-PWI pseudocode:

---

1. set generation counter $g = 0$ and problem-related information: D (problem dimensionality), $MNFC$ (maximum number of function calls);
2. set control parameters: $NP_{g=0} = 18 \cdot D$, $NP_{min} = 4$, $H = 5$, $p\% = 11\%$, $mi_{g=0} = 0.01 \cdot NP_{g=0}$;
3. create memory sets: **MF**, **MCR** ($|\mathbf{MF}| = |\mathbf{MCR}| = H$, $MF_i = MCR_i = 0.5$, $i = 1{:}H$), **A** ($|\mathbf{A}| = 0$), **SF, SCR, PWI**$_{g=0}$ **= 0**;
4. set L^d and U^d as lower and upper bounds of d-th variable ($d = 1,\ldots,D$);
5. initialize population $\mathbf{P_g}$ of individuals $\mathbf{x}_{i,g}$ according to eq. (2);
6. find objective function values $f(\mathbf{x}_{i,g})$ for all individuals in $\mathbf{P_g}$;
7. set the number of function calls used $NFC = NP_0$;
8. find the best solution found so far $\mathbf{x}_{best,0}$;
9. in the first generation the initial population is used, hence $NP_1 = NP_0$;
10. **while** $NFC < MNFC$
11. $g = g + 1$;
12. $|\mathbf{SF}| = |\mathbf{SCR}| = 0$, $|\mathbf{A_{temp}}| = 0$;
13. **for** $i = 1{:}\ NP_g$
14. generate ri according to eq. (6);
15. generate CR_i according to eq. (8);
16. **if** $NFC < 0.5 \cdot MNFC$
17. $F_i = 0.5$;
18. **else**
19. generate F_i according to eq. (7);
20. **end if;**
21. choose randomly $\mathbf{x}_{pbest,g}$ for i among round($NP_g \cdot p\%$) best individuals in $\mathbf{P_g}$**;**
22. perform mutation following eq. (41);
23. perform crossover following eq. (10);
24. perform bounds handling following eq. (11) and selection following eq. (5);
25. **if** $f(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{i,g})$
26. $\mathbf{x}_{i,g} \rightarrow \mathbf{A_{temp}}$, $CR_i \rightarrow \mathbf{SCR}$, $F_i \rightarrow$ **SF;**
27. **end if;**
28. $NFC = NFC + 1$;
29. **end for;**
30. compute mi_g according to eq. (42);
31. compute $\mathbf{PWI_g}$ according to eqs (38)-(40);
32. reduce population size to NP_{g+1} following eq. (16);
33. $\mathbf{A} \leftarrow \mathbf{A} \cup \mathbf{A_{temp}}$;
34. reduce the archive size to $NA_{g+1} = \lceil 1.4 \cdot NP_{g+1} \rceil$ by discarding random individuals from **A**;
35. update **MF**, **MCR** following eqs (12) and (13);
36. **end while;**

Fig. 4. The pseudocode of L-SHADE-50-PWI algorithm.

number of cases where differences were statistically not significant (≈), is computed. Mean performances from 51 runs on each problem are summarised in Tables 1–4. Respective standard deviations are given in Suppl. Tables 1–4, available as Supplementary Material to the online version of the paper. In Tables 1–4 it is also shown for how many problems of each test set (CEC'2017, CEC'2014 and CEC'2011) the PWI-based version is statistically significantly worse (−), equal (≈) and better (+) than the basic version of particular L-SHADE-based algorithm. Together 12 different comparisons are presented (for four L-SHADE variants and three benchmark sets).

Table 1
Mean performance of L-SHADE-PWI versus L-SHADE on CEC'2017, CEC'2014 and CEC'2011 problems. + means that L-SHADE-PWI is statistically significantly better at $\alpha = 0.05$ than L-SHADE; − means that L-SHADE-PWI is statistically significantly worse at $\alpha = 0.05$ than L-SHADE; ≈ means that there are no statistically significant differences between the two algorithms; sum − / ≈ / + is the number of problems on which L-SHADE-PWI is statistically worse/equal/better than L-SHADE. Note that technically in [8] the CEC'2011 problems were called F1-F10, the next ten problems were called F11.1-F11.10 and the last two problems F12-F13; for simplicity we call such 22 problems F1-F22, like in CEC'2017 and CEC'2014 cases.

	CEC'2017			CEC'2014			CEC'2011		
Algorithm	L-SHADE	L-SHADE-PWI	PWI is −/≈/+	L-SHADE	L-SHADE-PWI	PWI is −/≈/+	L-SHADE	L-SHADE-PWI	PWI is −/≈/+
F1	2.034E−14	2.201E−14	≈	1.529E+03	4.021E+03	−	3.085E−04	1.116E−05	≈
F2	1.000E+00	2.142E+04	−	4.012E−14	3.845E−14	≈	−2.607E+01	−2.607E+01	≈
F3	1.605E−13	1.928E−13	−	5.573E−14	4.904E−14	≈	1.151E−05	1.151E−05	≈
F4	7.461E+01	6.042E+01	≈	6.578E+01	9.352E+01	−	1.744E+01	1.609E+01	+
F5	1.127E+01	1.273E+01	−	2.025E+01	2.025E+01	≈	−3.636E+01	−3.615E+01	≈
F6	5.051E−08	1.702E−07	−	1.436E−01	1.293E−01	≈	−2.916E+01	−2.915E+01	≈
F7	6.351E+01	6.438E+01	−	4.235E−14	8.917E−15	+	1.157E+00	1.140E+00	≈
F8	1.155E+01	1.339E+01	−	9.718E−11	7.246E−11	+	2.200E+02	2.200E+02	≈
F9	3.121E−14	2.229E−15	+	1.191E+01	1.294E+01	−	2.496E+03	2.367E+03	≈
F10	3.114E+03	2.991E+03	+	4.971E−02	6.501E−02	−	−2.161E+01	−2.165E+01	≈
F11	4.858E+01	4.235E+01	+	3.237E+03	3.093E+03	≈	5.205E+04	5.220E+04	≈
F12	2.434E+03	3.036E+03	−	2.119E−01	2.170E−01	≈	1.774E+07	1.767E+07	+
F13	6.734E+01	6.692E+01	≈	1.555E−01	1.463E−01	≈	1.544E+04	1.544E+04	+
F14	3.063E+01	2.839E+01	+	3.131E−01	3.667E−01	≈	1.810E+04	1.812E+04	≈
F15	5.078E+01	4.577E+01	≈	5.068E+00	4.954E+00	≈	3.274E+04	3.274E+04	−
F16	3.638E+02	3.071E+02	+	1.680E+01	1.655E+01	+	1.240E+05	1.241E+05	≈
F17	2.379E+02	1.652E+02	+	1.718E+03	1.554E+03	+	1.863E+06	1.865E+06	≈
F18	4.930E+01	4.837E+01	≈	1.066E+02	1.405E+02	−	9.318E+05	9.320E+05	≈
F19	3.381E+01	2.778E+01	+	7.962E+00	7.566E+00	≈	9.396E+05	9.392E+05	≈
F20	1.458E+02	1.283E+02	+	1.486E+01	1.075E+01	+	9.317E+05	9.323E+05	≈
F21	2.143E+02	2.156E+02	−	6.318E+02	6.875E+02	≈	1.612E+01	1.605E+01	≈
F22	2.647E+03	2.785E+03	≈	1.008E+02	6.875E+01	+	1.403E+01	1.292E+01	+
F23	4.313E+02	4.282E+02	+	3.440E+02	3.440E+02	≈			
F24	5.118E+02	5.107E+02	≈	2.749E+02	2.737E+02	+			
F25	4.812E+02	4.843E+02	≈	2.052E+02	2.052E+02	≈			
F26	1.201E+03	1.203E+03	≈	1.002E+02	1.002E+02	≈			
F27	5.443E+02	5.481E+02	≈	3.441E+02	3.412E+02	≈			
F28	4.716E+02	4.604E+02	+	1.119E+03	1.126E+03	≈			
F29	3.471E+02	3.473E+02	≈	8.031E+02	8.331E+02	−			
F30	6.684E+05	7.355E+05	−	8.666E+03	8.783E+03	≈			
sum −/≈/+	9 / 11 / 10			6 / 17 / 7			1 / 17 / 4		

From Tables 1–4 we find that in each among 12 comparisons the number of statistically significant wins (+) by the PWI-based variant of specific L-SHADE-based algorithm over the classical L-SHADE-based version is larger than the number of statistically significant loses (−). Hence, PWI-based term improves the performance of each L-SHADE-based variant tested on every considered set of problems.

However, the improvement obtained when using PWI-based term is uneven for different L-SHADE-based algorithms. PWI-based mutation leads to the largest improvements for L-SHADE-cnEpSin and L-SHADE-50 algorithms, slightly lower for L-SHADE-SPACMA, and barely noticeable improvements for the basic L-SHADE. It must also be noted that for each algorithm PWI-term improves the results on a number of problems, but also leads to deterioration of the results on some others. Pointing out problems for which PWI-based mutation does not work turns out very difficult – it simply depends on the specific L-SHADE-based algorithm. Nonetheless, on average the PWI-based term improves the performance of each L-SHADE variant on every set of problems.

5.2. Impact of the control parameter $\mathrm{mi_g}$

The performance of PWI-based variants of L-SHADE algorithms depends on the minimum number of successful moves (mi_g) in the current generation g that are required to compute $\mathbf{PWI}_g$. mi_g is the only new control parameter introduced in this study. To show how the choice of mi_g affects the performance of PWI-based L-SHADE variants, we consider two possibilities: that mi_g is fixed for the whole run (independent of NP_g; in such a case index g could be skipped), or that it is related to the current population size NP_g.

If the value of mi_g is to be fixed, it cannot be higher than 3 to allow PWI-term to be effective during the whole run. This is because the final (minimum) population size is set to 4 in L-SHADE-based algorithms, hence the number of possible improvements during late parts of the run is limited to 4. However, it is un-realistic to expect that all individuals may improve at the end of the search, hence fixed mi_g should de facto not exceed 2. Of course, mi_g cannot be lower than 0, as at least one successful move is necessary to compute $\mathbf{PWI}_g$.

Table 2
Mean performance of L-SHADE-cnEpSin-PWI versus L-SHADE-cnEpSin on CEC'2017, CEC'2014 and CEC'2011 problems. + means that L-SHADE-cnEpSin-PWI is statistically significantly better at $\alpha = 0.05$ than L-SHADE-cnEpSin; − means that L-SHADE-cnEpSin-PWI is statistically significantly worse at $\alpha = 0.05$ than L-SHADE-cnEpSin; ≈ means that there are no statistically significant differences between the two algorithms; sum − / ≈ / + is the number of problems on which L-SHADE-cnEpSin-PWI is statistically worse/equal/better than L-SHADE-cnEpSin. Note that technically in [8] the CEC'2011 problems were called F1-F10, the next ten problems were called F11.1-F11.10 and the last two problems F12-F13; for simplicity we call such 22 problems F1-F22, like in CEC'2017 and CEC'2014 cases.

	CEC'2017			CEC'2014			CEC'2011		
Algorithm	L-SHADE-cnEpSin	L-SHADE-cnEpSin-PWI	PWI is - / ≈ / +	L-SHADE-cnEpSin	L-SHADE-cnEpSin-PWI	PWI is - / ≈ / +	L-SHADE-cnEpSin	L-SHADE-cnEpSin-PWI	PWI is - / ≈ / +
F1	1.951E−14	1.644E−14	+	1.413E−11	1.210E−09	≈	2.402E+00	1.831E+00	≈
F2	1.608E+00	1.157E+00	≈	3.344E−14	2.954E−14	≈	−2.602E+01	−2.579E+01	≈
F3	1.148E−13	1.271E−13	≈	5.684E−14	5.461E−14	≈	1.151E−05	1.151E−05	≈
F4	5.389E+01	5.182E+01	≈	5.423E+01	7.899E+01	≈	1.866E+01	1.742E+01	+
F5	2.675E+01	2.755E+01	≈	2.028E+01	2.029E+01	≈	−3.624E+01	−3.614E+01	≈
F6	7.576E−07	8.906E−07	≈	1.040E−02	1.024E−02	+	−2.909E+01	−2.912E+01	≈
F7	7.850E+01	7.933E+01	≈	9.362E−14	3.790E−14	+	1.061E+00	1.036E+00	+
F8	2.673E+01	2.617E+01	≈	4.676E−08	3.433E−08	≈	2.200E+02	2.200E+02	≈
F9	8.248E−14	1.783E−14	+	2.908E+01	2.532E+01	+	1.720E+03	1.258E+03	+
F10	3.164E+03	3.029E+03	+	1.103E+00	1.336E+00	≈	−2.164E+01	−2.167E+01	≈
F11	2.209E+01	2.697E+01	−	3.277E+03	3.132E+03	+	5.208E+04	5.203E+04	≈
F12	1.244E+03	1.399E+03	≈	2.720E−01	2.659E−01	≈	1.791E+07	1.769E+07	+
F13	6.594E+01	7.615E+01	≈	2.089E−01	1.934E−01	+	1.545E+04	1.545E+04	≈
F14	2.647E+01	2.474E+01	+	1.907E−01	1.726E−01	+	1.810E+04	1.811E+04	−
F15	2.604E+01	2.382E+01	+	5.701E+00	5.697E+00	≈	3.274E+04	3.274E+04	≈
F16	3.200E+02	2.491E+02	+	1.690E+01	1.653E+01	+	1.232E+05	1.229E+05	+
F17	2.114E+02	1.584E+02	+	3.924E+02	3.354E+02	≈	1.834E+06	1.829E+06	≈
F18	2.487E+01	2.354E+01	+	2.462E+01	1.497E+01	+	9.291E+05	9.288E+05	≈
F19	1.726E+01	1.636E+01	≈	9.851E+00	9.906E+00	≈	9.379E+05	9.371E+05	+
F20	1.115E+02	9.786E+01	+	6.470E+00	5.694E+00	+	9.288E+05	9.292E+05	≈
F21	2.267E+02	2.273E+02	≈	3.263E+02	3.293E+02	≈	1.533E+01	1.570E+01	−
F22	9.663E+02	1.715E+03	≈	1.133E+02	5.868E+01	+	1.669E+01	1.483E+01	+
F23	4.395E+02	4.393E+02	≈	3.440E+02	3.440E+02	≈			
F24	5.137E+02	5.152E+02	≈	2.681E+02	2.664E+02	+			
F25	4.803E+02	4.805E+02	−	2.049E+02	2.050E+02	≈			
F26	1.211E+03	1.218E+03	≈	1.002E+02	1.002E+02	+			
F27	5.264E+02	5.308E+02	≈	3.102E+02	3.200E+02	≈			
F28	4.582E+02	4.588E+02	−	1.158E+03	1.174E+03	≈			
F29	3.531E+02	3.500E+02	≈	8.142E+02	8.082E+02	≈			
F30	6.483E+05	6.580E+05	≈	8.622E+03	8.617E+03	≈			
sum − / ≈ / +	3 / 18 / 9			0 / 18 / 12			2 / 13 / 7		

Table 3
Mean performance of L-SHADE-SPACMA-PWI versus L-SHADE-SPACMA on CEC'2017, CEC'2014 and CEC'2011 problems. + means that L-SHADE-SPACMA-PWI is statistically significantly better at $\alpha = 0.05$ than L-SHADE-SPACMA; − means that L-SHADE-SPACMA-PWI is statistically significantly worse at $\alpha = 0.05$ than L-SHADE-SPACMA; ≈ means that there are no statistically significant differences between the two algorithms; sum − / ≈ / + is the number of problems on which L-SHADE-SPACMA-PWI is statistically worse/equal/better than L-SHADE-SPACMA. Note that technically in [8] the CEC'2011 problems were called F1-F10, the next ten problems were called F11.1-F11.10 and the last two problems F12-F13; for simplicity we call such 22 problems F1-F22, like in CEC'2017 and CEC'2014 cases.

	CEC'2017			CEC'2014			CEC'2011		
Algorithm	L-SHADE-SPACMA	L-SHADE-SPACMA-PWI	PWI is - /≈/ +	L-SHADE-SPACMA	L-SHADE-SPACMA-PWI	PWI is - /≈/ +	L-SHADE-SPACMA	L-SHADE-SPACMA-PWI	PWI is - /≈/ +
F1	0.000E+00	0.000E+00	≈	2.842E−14	1.737E−07	≈	1,014E+01	1,015E+01	≈
F2	3.567E−14	3.678E−14	≈	0.000E+00	0.000E+00	≈	−2,654E+01	−2,643E+01	≈
F3	0.000E+00	0.000E+00	≈	0.000E+00	0.000E+00	≈	1,151E−05	1,151E−05	≈
F4	3.613E+01	3.249E+01	≈	7.118E+01	8.080E+01	≈	1,941E+01	1,914E+01	≈
F5	5.541E+00	6.594E+00	−	2.025E+01	2.026E+01	≈	−3,632E+01	−3,645E+01	≈
F6	1.070E−13	2.351E−10	≈	0.000E+00	1.018E−02	≈	−2,916E+01	−2,912E+01	≈
F7	5.664E+01	5.832E+01	−	0.000E+00	0.000E+00	≈	1,127E+00	1,140E+00	≈
F8	5.404E+00	6.926E+00	−	6.090E−12	2.878E−12	≈	2,200E+02	2,200E+02	≈
F9	0.000E+00	0.000E+00	≈	4.916E+00	6.126E+00	−	2,176E+03	2,022E+03	+
F10	3.493E+03	3.336E+03	≈	1.035E−01	1.276E−01	−	−2,156E+01	−2,158E+01	≈
F11	3.197E+01	2.862E+01	+	3.167E+03	3.069E+03	≈	5,224E+04	5,223E+04	≈
F12	1.604E+03	1.499E+03	≈	2.195E−01	2.162E−01	≈	1,785E+07	1,770E+07	+
F13	3.605E+01	3.161E+01	≈	1.898E−01	1.779E−01	+	1,544E+04	1,545E+04	≈
F14	2.881E+01	2.586E+01	+	1.912E−01	2.135E−01	−	1,810E+04	1,810E+04	≈
F15	3.067E+01	2.604E+01	+	4.706E+00	4.593E+00	≈	3,274E+04	3,274E+04	≈
F16	4.463E+02	3.029E+02	+	1.632E+01	1.547E+01	+	1,237E+05	1,237E+05	≈
F17	2.921E+02	2.116E+02	+	6.576E+02	5.463E+02	+	1,864E+06	1,857E+06	+
F18	3.163E+01	2.748E+01	+	2.645E+01	1.821E+01	+	9,320E+05	9,320E+05	≈
F19	2.062E+01	1.861E+01	+	5.348E+00	5.342E+00	≈	9,400E+05	9,393E+05	+
F20	1.531E+02	1.339E+02	+	7.285E+00	5.370E+00	+	9,319E+05	9,321E+05	≈
F21	2.148E+02	2.165E+02	≈	4.873E+02	4.461E+02	≈	1,581E+01	1,593E+01	≈
F22	1.516E+03	2.193E+03	≈	1.294E+02	1.001E+02	+	1,561E+01	1,429E+01	+
F23	4.389E+02	4.382E+02	≈	3.440E+02	3.440E+02	≈			
F24	5.132E+02	5.145E+02	≈	2.684E+02	2.664E+02	+			
F25	4.809E+02	4.820E+02	≈	2.049E+02	2.050E+02	≈			
F26	1.129E+03	1.142E+03	≈	1.002E+02	1.002E+02	+			
F27	5.307E+02	5.418E+02	−	3.159E+02	3.268E+02	−			
F28	4.588E+02	4.588E+02	≈	1.155E+03	1.159E+03	≈			
F29	3.716E+02	3.602E+02	+	8.088E+02	8.281E+02	−			
F30	6.415E+05	6.907E+05	−	8.578E+03	8.641E+03	≈			
sum − /≈/ +	5 / 16 / 9			5 / 17 / 8			0 / 17 / 5		

Table 4
Mean performance of L-SHADE-50-PWI versus L-SHADE-50 on CEC'2017, CEC'2014 and CEC'2011 problems. + means that L-SHADE-50-PWI is statistically significantly better at $\alpha = 0.05$ than L-SHADE-50; − means that L-SHADE-50-PWI is statistically significantly worse at $\alpha = 0.05$ than L-SHADE-50; ≈ means that there are no statistically significant differences between the two algorithms; sum − / ≈ / + is the number of problems on which L-SHADE-50-PWI is statistically worse/equal/better than L-SHADE-50. Note that technically in [8] the CEC'2011 problems were called F1-F10, the next ten problems were called F11.1-F11.10 and the last two problems F12-F13; for simplicity we call such 22 problems F1-F22, like in CEC'2017 and CEC'2014 cases.

	CEC'2017			CEC'2014			CEC'2011		
Algorithm	L-SHADE-50	L-SHADE-50-PWI	PWI is − / ≈ / +	L-SHADE-50	L-SHADE-50-PWI	PWI is − / ≈ / +	L-SHADE-50	L-SHADE-50-PWI	PWI is − / ≈ / +
F1	1.700E−14	1.560E−14	≈	1.253E−06	7.720E−02	≈	5.639E−01	7.643E−01	≈
F2	8.824E−01	9.216E−01	≈	3.511E−14	2.954E−14	+	−2.641E+01	−2.622E+01	≈
F3	1.560E−13	1.817E−13	≈	5.684E−14	5.461E−14	≈	1.151E−05	1.151E−05	≈
F4	4.621E+01	5.599E+01	≈	5.871E+01	7.169E+01	≈	1.883E+01	1.860E+01	≈
F5	2.905E+01	2.993E+01	≈	2.024E+01	2.025E+01	≈	−3.657E+01	−3.658E+01	≈
F6	3.114E−07	4.197E−07	≈	1.918E−04	2.936E−05	+	−2.912E+01	−2.916E+01	≈
F7	7.827E+01	7.945E+01	≈	1.025E−13	4.012E−14	+	1.176E+00	1.143E+00	+
F8	2.901E+01	2.778E+01	≈	6.983E−11	4.165E−11	≈	2.200E+02	2.200E+02	≈
F9	6.242E−14	1.115E−14	+	2.794E+01	3.022E+01	−	1.767E+03	1.388E+03	+
F10	2.981E+03	2.864E+03	+	3.006E−02	4.938E−02	−	−2.162E+01	−2.168E+01	≈
F11	2.719E+01	2.757E+01	≈	3.039E+03	2.962E+03	≈	5.208E+04	5.215E+04	≈
F12	1.301E+03	1.428E+03	≈	2.105E−01	2.027E−01	≈	1.774E+07	1.757E+07	+
F13	4.257E+01	4.528E+01	≈	2.005E−01	1.940E−01	≈	1.545E+04	1.545E+04	≈
F14	2.716E+01	2.546E+01	+	1.920E−01	1.914E−01	+	1.811E+04	1.810E+04	≈
F15	2.384E+01	2.243E+01	+	5.370E+00	5.339E+00	≈	3.274E+04	3.274E+04	≈
F16	3.336E+02	3.093E+02	≈	1.644E+01	1.607E+01	+	1.234E+05	1.234E+05	≈
F17	2.429E+02	1.611E+02	+	3.605E+02	4.130E+02	−	1.848E+06	1.845E+06	≈
F18	2.463E+01	2.382E+01	≈	1.878E+01	1.263E+01	+	9.300E+05	9.302E+05	≈
F19	1.650E+01	1.655E+01	≈	9.369E+00	9.869E+00	≈	9.386E+05	9.381E+05	≈
F20	1.225E+02	1.148E+02	≈	6.150E+00	4.961E+00	+	9.298E+05	9.302E+05	−
F21	2.277E+02	2.282E+02	≈	3.317E+02	3.382E+02	≈	1.546E+01	1.576E+01	−
F22	1.227E+03	2.535E+03	−	1.150E+02	7.849E+01	+	1.638E+01	1.439E+01	+
F23	4.431E+02	4.411E+02	≈	3.440E+02	3.440E+02	+			
F24	5.135E+02	5.148E+02	≈	2.678E+02	2.660E+02	+			
F25	4.811E+02	4.815E+02	≈	2.049E+02	2.050E+02	≈			
F26	1.266E+03	1.289E+03	−	1.002E+02	1.002E+02	+			
F27	5.300E+02	5.350E+02	−	3.076E+02	3.268E+02	−			
F28	4.616E+02	4.588E+02	+	1.140E+03	1.145E+03	≈			
F29	3.438E+02	3.430E+02	≈	8.095E+02	8.217E+02	−			
F30	6.597E+05	6.948E+05	≈	8.501E+03	8.458E+03	≈			
sum − / ≈ / +	3 / 21 / 6			5 / 14 / 11			2 / 16 / 4		

If mi_g will vary with NP_g, the only natural limits are 0 and NP_g. However, too large value of mi_g means that we need many successful moves in particular generation, hence in practice it implies that $\mathbf{PWI}_g \equiv \mathbf{0}$ for almost all generations, making PWI-based term useless. Due to that we have tested the values of mi_g related to NP_g within the interval $[0.2 \cdot NP_g, 0.005 \cdot NP_g]$. Note that technically mi_g do not need to be an integer, but the impact of $mi_g = 1.0$ and $mi_g = 1.9$ on the search is equal, as both simply set the requirement of minimum 2 successful moves per generation.

Results obtained with different values of mi_g for all four L-SHADE-based variants on three benchmark sets are given in Table 5. The differences between the number of times the PWI-based L-SHADE-variants are statistically significantly better (+), and the number of times they are statistically significantly worse (−), than the basic L-SHADE variants depend on the choice of mi_g. The differences also depend on the benchmark set. In the case of CEC'2014 and CEC'2017 problems, too large mi_g (roughly $mi_g > 0.025 \cdot NP_g$) leads to poorer results than the original L-SHADE variants. Setting mi_g fixed to 0 is also not advised. There are two good competitive choices: either relating mi_g to NP_g by a relatively small value ($0.01 \cdot NP_g$ or $0.005 \cdot NP_g$), or keeping mi_g fixed to 1 or 2. Among these options, setting $mi_g = 0.01 \cdot NP_g$ leads to the best results. Interestingly, in contradiction to CEC'2014 and CEC'2017 benchmarks, for real-world problems (CEC'2011) PWI-based mutation almost always improves the performance of all original L-SHADE variants, irrespectively which value of mi_g is set (there is just one draw, when $mi_g = 0.2 \cdot NP_g$ for L-SHADE-SPACMA-PWI). This suggests that the proposed PWI term may be more robust for real-world problems than for artificial benchmarks.

When summing up the overall (for all L-SHADE-based variants and problem sets) differences between the numbers of significant improvements (+) and loses (−) for each mi_g value, we confirm that there are two almost equally good choices of mi_g (see column "+ versus − difference" in Table 5): relating it with NP_g by $mi_g = 0.01 \cdot NP_g$, or setting it fixed to 2. We also finds that for tested values of mi_g the summed up differences between statistically significant wins and loses are positive, apart from the one case ($mi_g = 0.2 \cdot NP_g$).

5.3. Comparison against other metaheuristics

The performance of four L-SHADE algorithms with PWI-based mutation has been tested against 16 other metaheuristics, discussed in Table 6. The competing methods were proposed between 2009 and 2018. Among 16 metaheuristics there are seven DE algorithms not based on L-SHADE (but including recently introduced version of SHADE without linear population size reduction, called ETI-SHADE [12]), three recent PSO variants, two Genetic Algorithms (GA), two multialgorithms (UMOEA-II and AMALGAM) and two other kinds of recently proposed metaheuristics. After reading [38], we tried to avoid algorithms which originality has been considered doubtful, but the choice of competitors is, unavoidably, subjective. Two among competing methods were considered among the winners of IEEE CEC competitions (GA-MPC and UMOEA-II).

The benchmark-set average ranks obtained by each algorithm for each test set (CEC'2017, CEC'2014 or CEC'2011) are given in Table 7. Average ranks for each benchmark set are obtained as follows. First, for each specific problem the mean performance from 51-runs is computed for each among 20 algorithms. Such mean performances of algorithms (accompanied by the respective 51-runs based standard deviations) on each specific problem are given in Suppl. Tables 5–10. Then algorithms are ranked from the best (rank 1, the algorithm with the lowest 51-run averaged function value) to the poorest (rank 20, the algorithm with the highest 51-run averaged function value). Finally, ranks achieved by each algorithm for all problems (30 or 22) in particular set (CEC'2017, CEC'2014 or CEC'2011) are averaged. Such averaged ranks are given in Table 7 (the lower averaged rank, the better algorithm). To facilitate reading, in Table 7 also the final position (position 1 means the best, position 20 – the worst) of each algorithm for every benchmark set is given. If the averaged ranks of two algorithms are equal, both algorithms share the same final position (for example CoBiDE and MPEDE are both 11.5th best methods for CEC'2017 problems, consult Table 7).

From Table 7 we find that all four PWI-based L-SHADE algorithms are always located in positions 1–5 for each benchmark test. Moreover, for each set of problems the two best positions are always occupied by PWI-based L-SHADE variants. The only methods that are competitive to PWI-based L-SHADE algorithms are CoBiDE, which is ranked 3rd best for CEC'2011 real-world problems, and UMOEA-II, which is 4th and 3rd best on CEC'2017 and CEC'2014 benchmarks, respectively. All four PWI-based L-SHADE variants are, on each benchmark set, better than all other 14 algorithms.

Choosing the best among PWI-based L-SHADE variants is difficult. We may note that L-SHADE-PWI is inferior to three other variants, but competition among L-SHADE-50-PWI, L-SHADE-SPACMA-PWI and L-SHADE-cnEpSin-PWI is tense. L-SHADE-50-PWI is the winner for real-world problems (CEC'2011) and benchmarks from CEC'2014 set (together with L-SHADE-SPACMA-PWI), but is only the 3rd best for CEC'2017 set. L-SHADE-SPACMA-PWI is the winner for CEC'2017 and CEC'2014 (jointly with L-SHADE-50-PWI), but is only the 5th best for real-world problems from CEC'2011. L-SHADE-cnEpSin-PWI is twice the second best and once (for CEC'2014) the 4th best method.

In this section multiple comparisons among various algorithms on many problems are considered. Hence, the statistical significance of the pair-wise comparisons among 20 algorithms is tested by means of the Friedman's test with post-hoc Shaffer's static procedure at $\alpha = 0.05$, as suggested in [16,43]. The results are given in Tables 8–10, where 1 means that the difference between the performances of particular pair of algorithms is statistically significant, and empty cell means that it is not. The respective codes of statistical tests were obtained from www.cmpe.boun.edu.tr/~ulas/m2test/details.php [43].

As in the case of other applications of multiple inter-comparisons, the number of cases when differences are statistically significant is relatively low. However, L-SHADE-50-PWI and L-SHADE-SPACMA-PWI are statistically significantly better than

Table 5
Impact of the control parameter mi_g on the performance on L-SHADE-based variants with PWI mutation strategy. sum $-$ / $\approx$ / $+$ is the number of problems on which specific PWI-based version is statistically significantly worse/equal/better at $\alpha = 0.05$ than the original variant on specific benchmark set; "$+$ versus $-$ difference" means the difference between the total number of statistically significant (at $\alpha = 0.05$) wins ($+$) and loses ($-$) achieved by all L-SHADE variants with specific mi_g on all test sets. Results for the chosen $mi_g = 0.01 \cdot NP_g$ are bolded.

		CEC'2017				CEC'2014				CEC'2011			
mi_g	$+$ versus $-$ difference	L-SHADE-PWI	L-SHADE-50-PWI	L-SHADE-cnEpSin-PWI	L-SHADE-SPACMA-PWI	L-SHADE-PWI	L-SHADE-50-PWI	L-SHADE-cnEpSin-PWI	L-SHADE-SPACMA-PWI	L-SHADE-PWI	L-SHADE-50-PWI	L-SHADE-cnEpSin-PWI	L-SHADE-SPACMA-PWI
		sum - / $\approx$ / +	sum - / $\approx$ / +	sum - / $\approx$ / +	sum - / $\approx$ / +	sum - / $\approx$ / +	sum - / $\approx$ / +	sum - / $\approx$ / +	sum - / $\approx$ / +	sum - / $\approx$ / +	sum - / $\approx$ / +	sum - / $\approx$ / +	sum - / $\approx$ / +
2	47	6/16/8	5/19/6	6/13/11	7/15/8	7/14/9	2/17/11	4/16/10	4/17/9	0/20/2	1/16/5	0/14/8	0/18/4
1	29	8/12/10	6/17/7	10/10/10	6/15/9	7/17/6	8/12/10	5/15/10	4/19/7	2/16/4	1/15/6	2/13/7	0/18/4
0	28	6/17/7	8/15/7	12/9/9	8/13/9	6/15/9	6/15/9	4/16/10	3/19/8	1/18/3	2/14/6	1/15/6	0/18/4
$0.2 \cdot NP_g$	-14	6/22/2	3/25/2	5/23/2	6/23/1	8/20/2	7/20/3	2/23/5	2/26/2	0/17/5	1/19/2	0/17/5	1/20/1
$0.1 \cdot NP_g$	8	7/20/3	0/28/2	4/23/3	4/23/3	10/16/4	6/18/6	3/20/7	3/24/3	0/21/1	1/15/6	0/16/6	0/19/3
$0.05 \cdot NP_g$	18	6/15/9	3/23/4	5/19/6	3/22/5	8/18/4	5/19/6	3/21/6	5/22/3	1/19/2	1/15/6	0/15/7	0/21/1
$0.025 \cdot NP_g$	38	5/17/8	1/22/7	6/18/6	4/19/7	7/16/7	5/20/5	2/18/10	2/19/9	1/18/3	2/15/5	2/14/6	0/18/4
$0.01 \cdot NP_g$	**48**	**9/11/10**	**3/21/6**	**3/18/9**	**5/16/9**	**6/17/7**	**5/14/11**	**0/18/12**	**5/17/8**	**1/17/4**	**2/16/4**	**2/13/7**	**0/17/5**
$0.005 \cdot NP_g$	41	8/11/11	7/15/8	7/15/8	8/13/9	7/16/7	3/18/9	3/16/11	3/19/8	1/17/4	0/17/5	1/14/7	0/17/5

Table 6
Sixteen algorithms used for the comparison with L-SHADE-based variants with PWI-term. 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	[44]	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 and CEC2'017 initialized with 20	AMALGAM variant that merges CMA-ES [19], GA and PSO [13] is used, as suggested and described in [44]. AMALGAM makes a number of sub-runs within time budget; for the conditions of commencing sub-run, see [44]. 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 [44].
AM-DEGL	Adaptive memetic DE with global and local neighborhood-based mutation	[32]	2013	5D (within [10,500] range)	Classical Adaptive Memetic DE variant. Note that the number of local nearest neighbours is assumed to be not smaller than 6 in this study.
ATPS-DE	JADE with adaptive population tuning scheme	[50]	2013	variable within [50,200], initialized with 5·D within bounds	Variant of JADE [49] with population size variable within [50,200] as default limit. 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·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.
cNrGA [*]	Non-revisiting GA with adaptive mutation using constant memory	[25]	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	[45]	2014	60	The first DE variant that put attention to the problem of coordinate system rotation.
DSO-DRONES [*]	Drone Squadron Optimization	[10]	2018	100 (divided into 4 sub-populations of equal size)	A specific kind of metaheuristics based on engineering devices (drone squads). The code of the algorithm has been obtained from https://github.com/melovv/DSO-MATLAB.
EFADE [*]	Enhanced fitness-adaptive DE algorithm with novel mutation	[28]	2018	50	Recently proposed adaptive variant of DE with triangular mutation operator. The MATLAB code has been obtained from prof. Suganthan's page http://www.ntu.edu.sg/home/epnsugan/.
EPSO [*]	Ensemble PSO	[26]	2017	40, divided into small and large sub-populations composed of 15 and 25	This ensemble algorithm is composed of five different PSO variants that are used with probabilities that are adaptively modified during run. The MATLAB code of EPSO has been obtained from prof. Suganthan's page http://www.ntu.edu.sg/home/epnsugan/.
ETI-SHADE	Successful history based DE with event-triggered impulsive control scheme	[12]	2017	150	Event-trigged impulsive control (ETI) scheme has been tested within various DE variants in [12]. The best results were obtained by authors of [12] when ETI is coupled with SHADE (ETI-SHADE), hence such variant is tested in this paper. Authors of [12] 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.
GA-MPC [*]	Genetic algorithm with multi-parent crossover	[14]	2011	90	GA that was the winner of the CEC'2011 competition. The MATLAB code of GA-MPC has been submitted from CEC'2011 web page (http://www3.ntu.edu.sg/home/epnsugan/index_files/CEC11-RWP/CEC11-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	[17]	2016	50	A recent PSO variant that mixes various elements from PSO and GA.

(*continued on next page*)

Table 6 (*continued*)

short name	description	reference	year	Population size	Comments
HIV-BBO [*]	Hybrid invasive weed / biogeography-based optimization	[22]	2017	variable, initialized with 100, maximum popululation size set to 200	This hybrid approach has large number of control parameters that were used as suggested in the source paper [22]. The MATLAB code has been obtained from prof. Simon's web page http://academic.csuohio.edu/simond/.
IDE [*]	DE with an Individual-independent mechanism	[42]	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 and CEC'2017 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 [42] paper.
IILPSO	PSO with Interswarm Interactive Learning Strategy	[36]	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.
MPEDE [*]	Multi-population ensemble DE	[47]	2016	250	MPEDE is a new self-adaptive DE variant that brings together the benefits from both adaptive and distributed DE methods. In the original paper [47] population size was set fixed to 250, what is justified by the need to manage sub-populations of uneven size. The code has been submitted from dr Wu's web page (http://guohuawunudt.gotoip2.com/publications.html).
UMOEA-II [*]	United multi-operator Evolutionary Algorithms-II with structurally biased term removed	[15,34]	2016	linearly decreasing from: 18·D to 4	UMOEA-II, a multialgorithm composed of CMA-ES [19] and DE [39], is a co-winner of IEEE CEC'2016 competition. However, one of UMOEA-II elements based on DE mutation is structurally biased, as shown in [34]. Following [34], we use UMOEA-II with corrected version of the biased mutation strategy. The code has been obtained from IEEE CEC web page http://www3.ntu.edu.sg/home/epnsugan/index_files.

Table 7
Comparison of 4 PWI-based L-SHADE variants and 16 other metaheuristics by means of average ranks (ranks averaged over all problems from particular benchmark set – CEC'2017, CEC'2014 or CEC'2011). Final position is the order of algorithms according to their average ranks (the lower average rank, the better final position).

	CEC'2017		CEC'2014		CEC'2011	
Algorithm	Average ranks	Final position	Average ranks	Final position	Average ranks	Final position
AMALGAM	5.87	6	7.42	7	12.14	14
AM-DEGL	10.20	9	10.42	9	10.14	9
ATPS-DE	11.03	13	10.22	8	8.98	7
cNrGA	18.07	19	16.57	19	16.50	20
CoBiDE	10.97	11.5	10.50	10	6.64	3
DSO-DRONES	18.80	20	14.17	16	15.14	19
EFADE	10.80	10	12.48	13	9.64	8
EPSO	14.80	16	14.37	17	10.84	12
ETI-SHADE	7.77	7	7.07	6	11.64	13
GA-MPC	12.00	14	12.60	14	10.32	10
GLPSO	16.63	17	15.90	18	12.86	16
HIV-BBO	12.37	15	12.70	15	12.36	15
IDE	9.67	8	11.17	12	10.55	11
IILPSO	17.83	18	16.87	20	15.05	18
L-SHADE-50-PWI	4.38	3	4.72	1.5	5.77	1
L-SHADE-SPACMA-PWI	3.70	1	4.72	1.5	7.73	5
L-SHADE-PWI	5.47	5	6.52	5	7.23	4
L-SHADE-cnEpSin_PWI	3.75	2	5.65	4	5.82	2
MPEDE	10.97	11.5	10.58	11	7.77	6
UMOEA-II	4.93	4	5.38	3	12.91	17

13 competitive algorithms (among 16) on CEC'2017 and CEC'2014 sets (see Tables 8 and 9); respective numbers for L-SHADE-cnEpSin-PWI and L-SHADE-PWI are lower, but still not smaller than 8 (50% of competitors). For real-world problems, the differences of multiple inter-comparisons are rather rarely statistically significant (see Table 10).

Table 8
The statistical significance of pair-wise comparisons among all algorithms on 50-dimensional CEC'2017 problems by means of Friedman's test with post-hoc Shaffer's procedure at $\alpha = 0.05$ significance level. 1 – the difference between two algorithms is statistically significant; empty cell – the difference between two algorithms is not statistically significant.

CEC'2017	AMALGAM	AM-DEGL	ATPS-DE	cNrGA	CoBiDE	DSO-DRONES	EFADE	EPSO	ETI-SHADE	GA-MPC	GLPSO	HIV-BBO	IDE	IILPSO	L-SHADE-50-PWI	L-SHADE-SPACMA-PWI	L-SHADE-PWI	L-SHADE-cnEpSin-PWI	MPEDE	UMOEA-II
AMALGAM				1		1		1		1	1	1		1						
AM-DEGL				1		1					1			1	1	1		1		1
ATPS-DE				1		1					1			1	1	1	1	1		1
cNrGA	1	1	1		1		1		1	1		1	1		1	1	1	1	1	1
CoBiDE				1		1					1			1	1	1	1	1		1
DSO-DRONES	1	1	1		1		1		1	1		1	1		1	1	1	1	1	1
EFADE				1		1					1			1	1	1	1	1		1
EPSO	1								1						1	1	1	1		1
ETI-SHADE				1		1		1			1			1						
GA-MPC	1			1		1								1	1	1	1	1		1
GLPSO	1	1	1		1		1		1				1		1	1	1	1	1	1
HBV-BBO	1			1		1								1	1	1	1	1		1
IDE				1		1					1			1	1	1		1		
IILPSO	1	1	1		1		1		1	1		1	1		1	1	1	1	1	1
L-SHADE-50-PWI		1	1	1	1	1	1	1		1	1	1	1	1					1	
L-SHADE-SPACMA-PWI		1	1	1	1	1	1	1		1	1	1	1	1					1	
L-SHADE-PWI			1	1	1	1	1	1		1	1	1		1					1	
L-SHADE-cnEpSin-PWI		1	1	1	1	1	1	1		1	1	1	1	1					1	
MPEDE				1		1					1			1	1	1	1	1		1
UMOEA-II		1	1	1	1	1	1	1		1	1	1		1					1	

Table 9
The statistical significance of pair-wise comparisons among all algorithms on 50-dimensional CEC'2014 problems by means of Friedman's test with post-hoc Shaffer's procedure at $\alpha = 0.05$ significance level. 1 – the difference between two algorithms is statistically significant; empty cell – the difference between two algorithms is not statistically significant.

CEC'2014	AMALGAM	AM-DEGL	ATPS-DE	cNrGA	CoBiDE	DSO-DRONES	EFADE	EPSO	ETI-SHADE	GA-MPC	GLPSO	HIV-BBO	IDE	IILPSO	L-SHADE-50-PWI	L-SHADE-SPACMA-PWI	L-SHADE-PWI	L-SHADE-cnEpSin-PWI	MPEDE	UMOEA-II
AMALGAM				1		1		1			1			1						
AM-DEGL				1							1			1	1	1				
ATPS-DE				1							1			1	1	1				
cNrGA	1		1	1					1				1		1	1	1	1	1	1
CoBiDE				1							1			1	1	1				
DSO-DRONES	1								1						1	1	1	1		1
EFADE									1						1	1	1	1		1
EPSO	1								1						1	1	1	1		1
ETI-SHADE				1		1	1	1		1	1	1		1						
GA-MPC									1						1	1	1	1		1
GLPSO	1		1	1		1			1						1	1	1	1		1
HBV-BBO									1						1	1	1	1		1
IDE				1										1	1	1		1		1
IILPSO	1		1	1		1			1				1		1	1	1	1	1	1
L-SHADE-50-PWI			1	1		1	1	1		1	1	1	1	1					1	
L-SHADE-SPACMA-PWI			1	1		1	1	1		1	1	1	1	1					1	
L-SHADE-PWI				1		1	1	1		1	1	1		1						
L-SHADE-cnEpSin-PWI				1		1	1	1		1	1	1	1	1						
MPEDE				1										1	1	1				
UMOEA-II				1		1	1	1		1	1	1	1	1						

Table 10
The statistical significance of pair-wise comparisons among all algorithms on real-world CEC'2011 problems by means of Friedman's test with post-hoc Shaffer's procedure at $\alpha = 0.05$ significance level. 1 – the difference between two algorithms is statistically significant; empty cell – the difference between two algorithms is not statistically significant.

CEC'2011	AMALGAM	AM-DEGL	ATPS-DE	cNrGA	CoBiDE	DSO-DRONES	EFADE	EPSO	ETI-SHADE	GA-MPC	GLPSO	HIV-BBO	IDE	IILPSO	L-SHADE-50-PWI	L-SHADE-SPACMA-PWI	L-SHADE-PWI	L-SHADE-cnEpSin-PWI	MPEDE	UMOEA-II
AMALGAM																				
AM-DEGL																				
ATPS-DE				1																
cNrGA			1		1		1								1		1		1	
CoBiDE				1		1								1						
DSO-DRONES					1										1		1		1	
EFADE				1																
EPSO																				
ETI-SHADE																				
GA-MPC																				
GLPSO															1				1	
HBV-BBO															1				1	
IDE																				
IILPSO					1										1		1		1	
L-SHADE-50-PWI				1		1					1	1		1						1
L-SHADE-SPACMA-PWI				1		1								1						
L-SHADE-PWI				1		1								1						
L-SHADE-cnEpSin-PWI				1		1					1	1		1						1
MPEDE				1		1								1						
UMOEA-II															1				1	

6. Conclusions

In this study we introduce the population-wide inertia (PWI) term into DE/current-to-*p*best/1 mutation strategy that is used by L-SHADE-based variants. The aim of the PWI term is to keep some information on the averaged direction and size of moves that were successful in the previous generation, and use it during the mutation phase of the next generation.

The use of PWI-based mutation is straightforward, but needs one additional control parameter – the minimum number of successes in the recent generation that is needed to compute the population-averaged direction of successful moves. If in particular generation the number of successes is too small, all coordinates of PWI vector are set to 0 and the algorithm proceeds in this generation as classical L-SHADE-based variants.

PWI-based mutation has been introduced into four L-SHADE-based variants, namely: L-SHADE [41], L-SHADE-cnEpSin [4], L-SHADE-SPACMA [27] and L-SHADE-50 [34]. The idea has been tested against classical versions of these four algorithms on three sets of problems: 50-dimensional CEC'2017 [3], 50-dimensional CEC'2014 [24] and 22 real-world problems of various dimensionalities from different fields of science (CEC'2011 [8]).

It was shown that the PWI-based version of each considered L-SHADE-based variant perform better on every set of problems than the version of respective L-SHADE-based variant without PWI-based mutation: L-SHADE-PWI is better than L-SHADE, L-SHADE-50-PWI is better than L-SHADE-50, L-SHADE-SPACMA-PWI is better than L-SHADE-SPACMA and L-SHADE-cnEpSin-PWI is better than L-SHADE-cnEpSin.

The performance of four L-SHADE-based variants with PWI term has been compared against 16 other metaheuristics proposed during the last 10 years (between 2009 and 2018). All four PWI-based L-SHADE variants were ranked among five best methods on each set of problems (CEC'2017, CEC'2014 and CEC'2011), and for each benchmark set one among L-SHADE variants with PWI-based mutation become the winner. Only CoBiDE [45] and UMOEA-II [15] algorithms were found to be competitive methods (CoBiDE for CEC'2011, UMOEA-II for CEC'2014 and CEC'2017 problems).

Acknowledgments

This work was supported within statutory activities No 3841/E-41/S/2018 of the Ministry of Science and Higher Education of Poland.

We would like to thank Prof. Jasper A. Vrugt for providing the MATLAB code of AMALGAM, Dr Tang for providing the MATLAB code of IDE, and all other authors for making their codes widely available on respective web pages.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ins.2018.08.030.

References

[1] R.W. Al-Dabbagh, F. Neri, N. Idris, M.S. Baba, Algorithmic design issues in adaptive differential evolution schemes: Review review and taxonomy, Swarm Evol. Comput. (in press) DOI: 10.1016/j.swevo.2018.03.008.
[2] 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, Proc of the IEEE Congress on Evolutionary Computation, Vancouver, 2016, doi:10.1109/CEC.2016.7744163.
[3] 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.
[4] N.H. Awad, M.Z. Ali, P.N. Suganthan, Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems, 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, doi:10.1109/CEC.2017.7969336.
[5] P.P. Biswas, P.N. Suganthan, G.A.J. Amaratunga, Minimizing harmonic distortion in power system with optimal design of hybrid active power filter using differential evolution, Appl. Soft Comput. 61 (2017) 486–496.
[6] P.P. Biswas, P.N. Suganthan, G.A.J. Amaratunga, Optimal placement of wind turbines in a windfarm using L-SHADE algorithm, 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, doi:10.1109/CEC.2017.7969299.
[7] J. Brest, M.S. Maucec, B. Boskovic, Single objective real-parameter optimization: Algorithm jSO, 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, doi:10.1109/CEC.2017.7969456.
[8] 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, 2010 Dec.
[9] S. Das, S.S. Mullick, P.N. Suganthan, Recent advances in differential evolution – an updated survey, Swarm Evol. Comput. 27 (2016) 1–30.
[10] V.V. de Melo, W. Banzhaf, Drone Squadron Optimization: a novel self-adaptive algorithm for global numerical optimization, Neural Comput. Appl. (in press) doi: 10.1007/s00521-017-2881-3.
[11] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (2011) 3–18.
[12] 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.
[13] 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.
[14] S.M. Elsayed, R.A. Sarker, D.L. Essam, GA with a new multi-parent crossover for constrained optimization, in: IEEE Congress on Evolutionary Computation, 2011, pp. 857–864.
[15] S. Elsayed, N. Hamza, R. Sarker, Testing united multi-operator evolutionary algorithms-II on single objective optimization problems, Proc of the IEEE Congress on Evolutionary Computation, 2016, doi:10.1109/CEC.2016.7744164.
[16] 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.
[17] 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.

[18] 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.
[19] N. Hansen, The CMA Evolution Strategy: A Comparing Review, in: J.A. Lozano, P. Larranaga, I. Inza, E. Bengoetxea (Eds.), Towards a New Evolutionary Computation, Springer, 2006 Studies in Fuzziness and Soft Computing series.
[20] M.R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Nation. Bur. Stand. 49 (6) (1952) 409–436.
[21] C. Igel, M. Toussaint, A no-free lunch theorem for non-uniform distributions of target functions, J. Math. Modell. Algorithms 3 (2004) 313–322.
[22] G. Khademi, H. Mohammadi, D. Simon, Hybrid invasive weed/biogeography-based optimization, Eng. Appl. Artif. Intell. 64 (2017) 213–231.
[23] 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.
[24] 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, China and Nanyang Technological University, Singapore, 2013.
[25] Y. Lou, S.Y. Yuen, Non-revisiting genetic algorithm with adaptive mutation using constant memory, Memetic Comput. 8 (2016) 189–210.
[26] N. Lynn, P.N. Suganthan, Ensemble particle swarm optimizer, Appl. Soft Comput. 55 (2017) 533–548.
[27] 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, 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, doi:10.1109/CEC.2017.7969307.
[28] A.W. Mohamed, P.N. Suganthan, Real-parameter unconstrained optimization based on enhanced fitness-adaptive differential evolution algorithm with novel mutation, Soft Comput. (in press) doi: 10.1007/s00500-017-2777-2.
[29] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artif. Intell. Rev. 33 (1–2) (2010) 61–106.
[30] K.R. Opara, J. Arabas, Differential Evolution: A survey of theoretical analyses, Swarm Evol. Comput. (in press) doi: 10.1016/j.swevo.2018.06.010.
[31] 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, No r way, 2009, pp. 1889–1895.
[32] A.P. Piotrowski, Adaptive memetic differential evolution with global and local neighborhood-based mutation operators, Inform. Sci. 241 (2013) 164–194.
[33] A.P. Piotrowski, M.J. Napiorkowski, J.J. Napiorkowski, P.M. Rowinski, Swarm intelligence and evolutionary algorithms: performance versus speed, Inform. Sci. 384 (2017) 34–85.
[34] A.P. Piotrowski, J.J. Napiorkowski, Some metaheuristics should be simplified, Inform. Sci. 427 (2018) 32–62.
[35] A.P. Piotrowski, J.J. Napiorkowski, Step-by-step improvement of JADE and SHADE-based algorithms: success or failure? Swarm Evol. Comput. (in press) doi: 10.1016/j.swevo.2018.03.007.
[36] Q. Qin, S. Cheng, Q. Zhang, L. Li, Y. Shi, PSO with interswarm interactive learning strategy, IEEE Trans. Cybern. 46 (10) (2016) 2238–2251.
[37] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceeding in IEEE Congress on Evolutionary Computation (CEC), 1998, pp. 69–73.
[38] K. Sorensen, Metaheuristics—the metaphor exposed, Int. Trans. Oper. Res. 22 (2015) 3–18.
[39] R. Storn, K.V. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
[40] 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.
[41] R. Tanabe, A. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in: Proc. IEEE Congress on Evolutionary Computation, Bejing, Ch i na, 2014, pp. 1658–1665.
[42] L. Tang, Y. Dong, J. Liu, Differential evolution with an individual-dependent mechanism, IEEE Trans. Evol. Comput. 19 (4) (2015) 560–574.
[43] A. Ulas, O.T. Yildiz, E. Alpaydin, Cost-conscious comparison of supervised learning algorithms over multiple data sets, Pattern Recognit. 45 (2012) 1772–1781.
[44] 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.
[45] 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.
[46] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[47] 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.
[48] A. Zamuda, Adaptive constraint handling and success history differential evolution for CEC 2017 constrained real-parameter optimization, 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, doi:10.1109/CEC.2017.7969601.
[49] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (5) (2009) 945–958.
[50] W. Zhu, Y. Tang, J.A. Fang, W.B. Zhang, Adaptive population tuning scheme for differential evolution, Inf. Sci. 223 (2013) 164–191.