Swarm and Evolutionary Computation xxx (xxxx) xxx

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization

Ali W. Mohamed [a,*], Anas A. Hadi [b], Kamal M. Jambi [b]

[a] Operations Research Department, Institute of Statistical Studies and Research, Cairo University, Giza, 12613, Egypt
[b] Faculty of Computing and Information Technology, King Abdulaziz University, P. O. Box 80200, Jeddah 21589, Saudi Arabia

ARTICLE INFO

Keywords:
Evolutionary computation
Global optimization
Differential evolution
Novel ordered mutation

ABSTRACT

Proposing new mutation strategies to improve the optimization performance of differential evolution (DE) is an important research study. Therefore, the main contribution of this paper goes in three directions: The first direction is introducing a less greedy mutation strategy with enhanced exploration capability, named *DE/current-to-ord_best/1* (*ord* stands for *ordered*) or *ord_best* for short. In the second direction, we introduce a more greedy mutation strategy with enhanced exploitation capability, named *DE/current-to-ord_pbest/1* (*ord_pbest* for short). Both of the proposed mutation strategies are based on ordering three selected vectors from the current generation to perturb the target vector, where the directed differences are used to mimic the gradient decent behavior to direct the search toward better solutions. In *ord_best*, the three vectors are selected randomly to enhance the exploration capability of the algorithm. On the other hand, *ord_pbest* is designed to enhance the exploitation capability where two vectors are selected randomly and the third is selected from the global *p* best vectors. Based on the proposed mutation strategies, *ord_best* and *ord_pbest*, two DE variants are introduced as EDE and EBDE, respectively. The third direction of our work is a hybridization framework. The proposed mutations can be combined with DE family algorithms to enhance their search capabilities on difficult and complicated optimization problems. Thus, the proposed mutations are incorporated into SHADE and LSHADE to enhance their performance. Finally, in order to verify and analyze the performance of the proposed mutation strategies, numerical experiments were conducted using CEC2013 and CEC2017 benchmarks. The performance was also evaluated using CEC2010 designed for Large-Scale Global Optimization. Experimental results indicate that in terms of robustness, stability, and quality of the solution obtained, both mutation strategies are highly competitive, especially as the dimension increases.

1. Introduction

Differential Evolution (DE), proposed by Storn and Price [38,39], is a stochastic population-based search method. It exhibits excellent capability in solving a wide range of optimization problems with different characteristics from several fields and many real-world application problems [34]. Similar to all other Evolutionary algorithms (EAs), the evolutionary process of DE uses mutations, crossover, and selection operators at each generation to reach the global optimum. Besides, it is one of the most efficient evolutionary algorithms (EAs) currently in use [8]. It has been widely applied in solving many real-world applications of science and engineering, such as flow shop scheduling [31], financial markets dynamic modeling [16], feature selection [33], neural networks [10], Robot control [37], and economic dispatch [23]. For more applications, interested readers can refer to [8].

In DE, each individual in the population is called a target vector. Mutation is used to generate a mutant vector, which perturbs a target vector using the difference vector of other individuals in the population. After that, crossover operation generates the trial vector by combining the parameters of the mutation vector with the parameters of a parent vector selected from the population. Finally, the fitness value and selection operation determine which of the vectors should be chosen for the next generation by implementing a one-to-one competition between the generated trail vectors and the corresponding parent vectors [19,43]. The performance of DE depends on the mutation strategy and the crossover operator. Besides, the intrinsic control parameters (population size *NP*, scaling factor *F*, and crossover rate *Cr*) play a vital role in balancing the diversity of population and convergence speed of the algorithm.

* Corresponding author.
E-mail addresses: aliwagdy@gmail.com (A.W. Mohamed), anas1401@gmail.com (A.A. Hadi), kjambi@kau.edu.sa (K.M. Jambi).

https://doi.org/10.1016/j.swevo.2018.10.006
Received 1 March 2018; Received in revised form 10 October 2018; Accepted 15 October 2018
Available online xxxx

DE has many advantages such as simplicity of implementation, reliability, speed, and robustness [34]. However, DE has many weaknesses, as all other evolutionary search techniques do with respect to the "no free lunch" theorem [60]. Generally, DE has a good global exploration ability that can reach the region of global optimum, but it is slow at the exploitation of the solution [28]. Additionally, the parameters of DE are problem dependent and it is difficult to adjust them for different problems. Moreover, DE performance decreases as search space dimensionality increases [6]. Finally, the performance of DE deteriorates significantly when the problems of premature convergence and/or stagnation occur [6,27]. Consequently, researchers have suggested many techniques to improve the basic DE. From the literature [7,8,53], these proposed modifications, improvements, and developments on DE focus on adjusting control parameters in an adaptive or self-adaptive manner while there are a few attempts in developing new mutations rule.

Therefore, the main originality of this paper goes in three directions: The first direction is introducing a new DE algorithm, named EDE, for solving global numerical optimization problems over continuous space. In EDE, a less greedy mutation strategy with advanced exploration capability is introduced, named *DE/current-to-ord_best/1* (*ord* stands for *ordered*) or *ord_best* for short. Classical *DE/current-to-best/1* uses the best individual vector in the population from the current generation to perturb the target vector. On the other hand, *ord_best* is based on ordering the three randomly selected vectors for each target vector according to their fitness values. The best one of them is used as *best* and the remaining two vectors are *median* and *worst*. The directed differences from the *worst* to the *best* are used to mimic the gradient decent behavior to direct the search toward better solutions.

In the second direction, another DE algorithm, named EBDE, is introduced. EBDE introduces a more greedy mutation strategy with advanced exploitation capability named *DE/current-to-ord_pbest/1* (*ord_pbest* for short). This mutation is based on incorporating the global *p* best vectors within *ord_best* mutation strategy. For each target vector, one vector will be selected form the best *p* vectors, while the other two are randomly selected vectors. Again, the selected three vectors are ordered according to their fitness values, the best one of them is used as *best* and the remaining two vectors are *median* and *worst*. This mutation strategy is more effective when the population size reduction is used.

The third direction of our work is a hybridization framework. The proposed mutations can be combined with DE family algorithms to enhance their search capabilities on difficult and complicated optimization problems. Thus, the proposed mutations were incorporated into SHADE and LSHADE to enhance their performance, as SHADE and LSHADE are highly competitive DE variants with self-adaptive parameters.

In order to verify and analyze the performance of both mutation strategies, numerical experiments were conducted using CEC2013 [21] and CEC2017 [58] benchmarks. The performance was also evaluated using CEC2010 designed for Large-Scale Global Optimization (LSGO) [59]. Experimental results indicate that in terms of robustness, stability, and quality of the solution obtained, both mutation strategies are highly competitive, especially as the dimension increases.

The rest of the paper is organized as follows: Section 2 introduces the canonical DE algorithm, including its typical mutation operators, crossover, and selection operators. Section 3 reviews the related work. Next, in section 4, the proposed EDE, EBDE, and the hybridization framework are introduced. The effectiveness of the proposed algorithms is reported and discussed in section 5 according to the computational results, and on the comparison with other state-of-the-art techniques. Finally, conclusions and future works are drawn in section 6.

2. Differential evolution

This section provides a brief summary of the basic Differential Evolution (DE) algorithm. In simple DE, generally known as *DE/rand/1/bin* [11,39], an initial random population, denoted by *P*, consists of *NP* individual. Each individual is represented by the vector X_i using $X_i = (x_{1,i}, x_{2,i}, \ldots, x_{D,i})$, where D is the number of dimensions in solution space. Since the population will be varied with the running of the evolutionary process, the generation times in DE are expressed by $G = 0, 1, \ldots, G_{max}$, where G_{max} is the maximal times of generations. For the ith individual of P at the G generation, it is denoted by $X_i^G = (x_{1,i}^G, x_{2,i}^G, \ldots, x_{D,i}^G)$. The lower and upper bounds in each dimension of search space are respectively recorded by X_L and X_U, where $X_L = (x_{1,L}, x_{2,L}, \ldots, x_{D,L})$ and $X_U = (x_{1,U}, x_{2,U}, \ldots, x_{D,U})$. The initial population P^0 is generated randomly according to a uniform distribution within the boundaries (X_L, X_U). After initialization, DE operators (mutation and crossover) evolve these individuals to generate a trial vector. Then, a comparison between the parent and its trial vector is done to select the vector that should survive to the next generation [7,22]. Detailed DE steps are discussed below:

2.1. Initialization

In order to establish a starting point for the optimization process, an initial population P^0 must be created. Typically, in P^0, each jth component ($j = 1, 2, \ldots, D$) of the ith individuals ($i = 1, 2, \ldots, NP$) is obtained as the following:

$$x_{i,j}^0 = x_{j,L} + rand.(x_{j,U} - x_{j,L}) \tag{1}$$

where *rand* returns a uniformly distributed random number in the interval $[0, 1]$.

2.2. Mutation

At generation G, for each target vector x_i^G, a mutant vector v_i^G is generated according to the following:

$$v_i^G = x_{r_1}^G + F.\left(x_{r_2}^G - x_{r_3}^G\right), r_1 \neq r_2 \neq r_3 \neq i \tag{2}$$

where randomly chosen indices $r_1, r_2, r_3 \in \{1, 2, \ldots NP\}$. F is a real number to control the amplification of the difference vector $(x_{r_2}^G - x_{r_3}^G)$. In this work, if a component of a mutant vector violates search space, then a new value of this component is generated using Eq. (1). The mutation strategy described in Eq. (2) is known as *DE/rand/1/bin*. The other most frequently used mutations strategies are:

"*DE/best/1/bin*" [22]: $$v_i^G = x_{best}^G + F.\left(x_{r_1}^G - x_{r_2}^G\right) \tag{3}$$

"*DE/best/2/bin*" [22]: $$v_i^G = x_{best}^G + F.\left(x_{r_1}^G - x_{r_2}^G\right) + F.\left(x_{r_3}^G - x_{r_4}^G\right) \tag{4}$$

"*DE/rand/2/bin*" [22]: $$v_i^G = x_{r_1}^G + F.\left(x_{r_2}^G - x_{r_3}^G\right) + F.\left(x_{r_4}^G - x_{r_5}^G\right) \tag{5}$$

"*DE/current − to − best/1/bin*" [22]: $$v_i^G = x_i^G + F.\left(x_{best}^G - x_i^G\right) + F.\left(x_{r_1}^G - x_{r_2}^G\right) \tag{6}$$

"*DE/current − to − rand/1/bin*" [22]: $$v_i^G = x_i^G + F.\left(x_{r_1}^G - x_i^G\right) + F.\left(x_{r_2}^G - x_{r_3}^G\right) \tag{7}$$

where r_1, r_2, r_3, r_4, and r_5 are mutually different integers randomly generated within the range $(1, 2, \ldots, NP)$, which are also different from the index i. These indices are randomly generated once for each mutant vector. The scale factor F is a positive control parameter for scaling the difference vector. x_{best}^G is the best individual vector with the best fitness value in the population at generation G. The condition of selecting mutually different indices has been commonly adopted in DE. According

to the observation of the extensive experiments, Wang et al. [55] conclude that relaxing this condition may also be useful in designing better DE algorithms.

2.3. Crossover

There are two main crossover types used in DE, binomial and exponential. We here elaborate the binomial crossover. In the binomial crossover, the target vector is mixed with the mutated vector, using the following scheme, to yield the trial vector u_i^G:

$$u_{j,i}^G = \begin{cases} v_{j,i}^G, & if\ (rand < Cr\ or\ j = j_{rand}) \\ x_{j,i}^G, & otherwise \end{cases} \tag{8}$$

where *rand* is a uniformly distributed random number in [0,1], $Cr \in [0,1]$ called the crossover rate that controls how many components are inherited from the mutant vector, j_{rand} is a uniformly distributed random integer in $[1,\ D]$ that makes sure that at least one component of the trial vector is inherited from the mutant vector.

2.4. Selection

DE adapts a greedy selection strategy [46]. The trial vector u_i^G is set to x_i^{G+1} if and only if fitness function value of u_i^G is equal to or better than x_i^G. Otherwise, the old vector x_i^G is retained. The selection scheme is as follows (for a minimization problem):

$$x_i^{G+1} = \begin{cases} u_i^G, f(u_i^G) \le f(x_i^G) \\ x_i^G, otherwise \end{cases} \tag{9}$$

3. Related work

Virtually, the performance of the canonical DE algorithm mainly depends on the chosen mutation/crossover strategies and the associated control parameters. Due to DE limitations as previously aforementioned in the introduction section, many researchers have been working on the improvement of DE. Thus, many researchers have proposed novel techniques to overcome its problems and improve its performance [7]. A brief overview of these techniques is discussed in this section.

Firstly, many trial-and-errors experiments have been conducted to adjust the control parameters. Storn and Price [39] suggested that population size *NP* to be between 5*D* and 10*D* and 0.5 as a good initial value of mutation scaling factor *F*, they also stated that "*Values of F smaller than 0.4, like those greater than 1, are only occasionally effective*". The crossover rate $Cr = 0.1$ is an initial good choice; however, since a large *Cr* often accelerates convergence, it is appropriate to first try *Cr* as 0.9 or 1 in order to check if a quick solution is possible. Gamperle at al. [13] recommended that a good choice for *NP* is between 3*D* and 8*D*, with $F = 0.6$ and *Cr* lies in [0.3, 0.9]. However, Ronkkonen et al. [36] concluded that $F = 0.9$ is a good compromise between convergence speed and convergence rate. Additionally, *Cr* depends on the nature of the problem, so *Cr* with a value between 0.9 and 1 is suitable for non-separable and multimodal objective functions, while a value of *Cr* between 0 and 0.2 is suitable when the objective function is separable.

In general, scaling factor *F* is an important parameter that controls the evolving rate of the population i.e. it is closely related to the convergence speed [35]. A small *F* value encourages the exploitation tendency of the algorithm that makes the search focus on the neighborhood of the current solutions; hence it can enhance the convergence speed. However, it may also lead to premature convergence [12]. On the other hand, large *F* value improves the exploration capability of the algorithm that can make the mutant vectors distribute widely in the search space and can increase the diversity of the population [44]. However, it may slow down the search [12].

The constant crossover *Cr* practically controls the diversity of the population [44]. In fact, large values of *Cr* will increase the population diversity. Nevertheless, this may reduce the stability of the algorithm. On the other hand, small values of *Cr* increase the possibility of stagnation that may weaken the exploration ability of the algorithm to open up new search space. Additionally, *Cr* is usually more sensitive to problems with different characteristics such as unimodality and multimodality, separable and non-separable problems.

Instead of manual tuning procedure of the control parameters, some techniques have been designed with a view of adjusting control parameters in an adaptive or self-adaptive manner. Most of these methods are based on generating random values from uniform, normal, or Cauchy distributions. On the other hand, some methods are based on generating different values from a pre-defined parameter candidate pool, beside the previous experience of generating better solutions as offspring to guide the adaptation of these parameters [2,5,18,22,28,30,32,35,44,49,50]. Other studies make use of full information about the distribution conditions of each vector-individual in successive populations dynamically through generations to generate suitable control parameters [1,15,46,51].

Wu et al. [48] proposed a multi-population-based framework to realize an adapted ensemble of three mutation strategies into a novel DE variant, named MPEDE. The population of MPEDE is dynamically partitioned into several sub-populations including three indicator sub-populations and one reward sub-population. Each indicator sub-population with a relatively smaller size is assigned to a constituent mutation strategy and the reward sub-population with a relatively larger size is assigned to the currently best-performed mutation strategy as an extra reward. This way, dynamically computational resource allocation among the mutation strategies is realized and the best mutation strategy is expected to obtain the most computational resources. To non-linearly changing both the values of *F* and *Cr* along with their direction of changing with the progress of iterations, Draa et al. [9] proposed six adaptation schemes based on sinusoidal functions. In these schemes, the *F* and *Cr* values can be obtained from scaled and shifted sinusoidal terms. It is experimentally shown that both *F* and *Cr* values are sampled from sinusoidal terms with increasing coefficients, outperforms other proposed schemes. Likewise, Brest et al. [2] proposed an efficient technique, named jDE, for self-adapting control parameter settings. This technique encodes the parameters into each individual and adapts them by means of evolution. The results showed that jDE is better than, or at least comparable to, the standard DE algorithm and other state-of-the-art algorithms when considering the quality of the solutions obtained. In the same context, controlling the degree of randomization of *F* and *Cr* and investigating their effect on the performance of DE is an important research direction in DE. Zamuda and Brest [49] published an interesting work in this direction. The authors introduced a randomness level parameter, which influences the dynamics of the control parameter values and their propagation through "suitable individuals' improvement contributions during elitist selection" in an extended framework of the jDE algorithm. Qin et al. [35] introduced a self-adaptive differential evolution (SaDE). The main idea of SaDE is to implement four mutation strategies simultaneously: *DE/rand/1/bin*, *DE/rand/2/bin*, *DE/rand-to-best/2/bin*, and *DE/current-to-rand/1/bin*, as well as adapt mutation and crossover parameters. The performance of SaDE was evaluated and compared with the conventional DE and three adaptive DE variants. The experimental results demonstrated that SaDE yielded better quality solutions and had a higher success rate. Inspired by SaDE and motivated by the success of diverse self-adaptive DE approaches, Mallipeddi et al. developed a self-adaptive DE, called EPSDE [22]. In EPSDE, a pool of distinct mutation strategies along with a pool of values for each control parameter coexists throughout the evolution process and competes to produce offspring. The resulting comparisons showed that EPSDE algorithm outperformed conventional DE and other state-of-the-art parameter adaptive DE variants in terms of solution quality and robustness. Similarly, motivated by the important results obtained by other researchers, Wang et al. [44] proposed a novel method, called composite

DE (CoDE). This method used three mutation strategies and three control parameter settings. It randomly combines them to generate trial vectors. The performance of CoDE has been evaluated on IEEE CEC2005 benchmark test functions and it was very competitive to other compared algorithms. In the same direction, Caraffini et al. [4] proposed Super-fir Multi-criteria Adaptive DE (SMADE). SMADE is a Memetic approach based on the hybridization of the Covariance Matrix Adaptive Evolutionary Strategy (CMAES) [17] with modified DE, namely Multi-criteria Adaptive DE (MADE). MADE makes use of four mutation/crossover strategies in an adaptive manner which are *rand/1/bin*, *rand/2/bin*, *rand-to-best/2/bin,* and *current-to-rand/1*. The control parameters *F* and *Cr* are adaptively adjusted during the evolution. At the beginning of the optimization process, CMAES is used to generate a solution with a high quality, which is then injected into the population of MADE. The performance of SMADE has been evaluated on IEEE CEC2013 benchmark test functions and the experimental results were very promising. Recently, Wu et al. [66] proposed EDEV as a higher-level ensemble of multiple DE variants rather than an ensemble of mutation strategies. EDEV consists of JADE [50], CoDE [44], and EPSDE [22], where the most efficient algorithm obtains more computational resources.

On the other hand, improving the mutation strategy has attracted many researchers. In order to improve the convergence velocity of DE, Fan and Lampinen [11] proposed a trigonometric mutation strategy, which is considered a local search operator, and combined it with *DE/rand/1/bin* mutation operator to design TDE algorithm. Zhang and Sanderson [50] introduced a new DE algorithm, named JADE, to improve optimization performance by implementing a new mutation strategy *DE/current-to-pbest* with optional external archive and by updating control parameters in an adaptive manner. Simulation results show that JADE was better than, or at least competitive to, other classic or adaptive DE in terms of convergence performance. Tanabe and Fukunaga [40] proposed Success History based DE (SHADE) as an improved variant of JADE algorithm [50]. Instead of sampling *F* and *Cr* values from gradually adapted probability distributions, SHADE used historical memory archives M_{Cr} and M_F. Historical memory archives store a set of *Cr* and *F* values, respectively that have performed well in the recent past. The algorithm generates new *Cr*, *F* pairs by directly sampling the parameter space close to one of the stored pairs. Out of the 21 algorithms that participated in the IEEE CEC2013 competition [21], SHADE ranked 3rd, the first two ranks being taken by non-DE-based algorithms. Tanabe and Fukunaga [41] further improved the SHADE algorithm by using the linear population size reduction (L-SHADE). In L-SHADE, the population size of SHADE is continually reduced by means of a linear function [8]. In IEEE CEC2014 competition, L-SHADE was the first ranked algorithm. To overcome the premature convergence and stagnation problems observed in the classical DE, Islam et al. [18] proposed modified DE with *p*-best crossover, named MDE_*p*BX. This mutation strategy is similar to *DE/current-to-best/1* strategy. It selects the best vector from a dynamic group of *q*% of the randomly selected population members. Moreover, their crossover strategy uses a vector that is randomly selected from the *p* top ranking vectors according to their objective values in the current population (instead of the target vector) to enhance the inclusion of generic information from the elite individuals in the current generation. The parameter *p* is linearly reduced with generations to favor the exploration at the beginning of the search and exploitation during the later stages by gradually downsizing the elitist portion of the population. Das et al. [5] proposed two kinds of topological neighborhood models for DE in order to achieve a better balance between its explorative and exploitative tendencies. In this method, called DEGL, two trial vectors are created by the global and local neighborhood-based mutation. These two trial vectors are combined to form the actual trial vector using a weight factor. DEGL showed very competitive results according to the conducted experimental results. Ranking-based mutation operators for the DE algorithm was proposed by Gong and Cai [61]. In this framework, some of the parents in the mutation operators are proportionally selected according to their rankings. The higher ranking a parent obtains, the more opportunity it will be selected. According to the experimental results, this framework is able to enhance the performance of the DE-based algorithms. In order to solve unconstrained and constrained optimization problems, Mohamed et al. [25,26] proposed a directed mutation based on the weighted difference vector between the best and the worst individuals at a particular generation. The new directed mutation rule is combined with the modified basic mutation strategy *DE/rand/1/bin*, where only one of the two mutation rules is applied with a probability of 0.5. The proposed mutation rule is shown to enhance the local search ability of the basic DE and to get a better trade-off between convergence rate and robustness. Numerical experiments on well-known unconstrained and constrained benchmark test functions and five engineering design problems have shown that the new approach is efficient, effective and robust. Similarly, to enhance global and local search capabilities and simultaneously increases the convergence speed of DE, Mohamed [24] introduced a new triangular mutation rule based on the convex combination vector of the triplet defined by the three randomly chosen vectors and the difference vector between the best and the worst individuals among the three randomly selected vectors. In this algorithm, called IDE, the mutation rule is combined with the basic mutation strategy through a non-linear decreasing probability rule. A restart mechanism is also proposed to avoid premature convergence. IDE is tested on a well-known set of unconstrained problems and shows its superiority to state-of-the-art differential evolution variants. Recently, Mohamed et al. [62] proposed LSHADE-SPACMA as a hybridization framework between LSHADE [41] and CMAES [17]. LSHADE-SPACMA introduced a Semi Parameter Adaptation (SPA) for *F* and *Cr*. In SPA, the adaptation goes into two phases. In the first phase, the adaptation is concentrated on *Cr* values while *F* parameter is generated randomly within the range [0.4, 0.6] according to a uniform distribution. During the second phase, *Cr* values are frozen to the adopted values and *F* adaptation is activated. LSHADE-SPACMA was ranked 4th according to IEE CEC2017 competition.

Conventional DE depends on initiating the population according to the uniform distribution. Poikolainen et al. [54] proposed Cluster-Based Population Initialization (CBPI) as an intelligent initialization pre-processing procedure. CBPI implicitly performs an initial screening of the problem features in order to roughly estimate the most interesting regions of the decision space. In order to exploit distribution information provided by the population, Wang et al. [56] proposed a cumulative population distribution information-based DE framework called CPI-DE. In this framework, an eigen coordinate system is established using covariance matrix adaptation. In the same direction, Liu et al. [57] introduced an adaptive framework to tune the coordinate systems ACoS based on Eigen coordinate system. Eigen coordinate system is coupled with the original coordinate system, and they are selected according to a probability vector.

Practically, it can be observed that the main modifications, improvements, and developments on DE focus on adjusting control parameters in an adaptive or self-adaptive manner. However, a few enhancements have been implemented to modify the structure and/or mechanism of basic DE algorithm or to propose new mutation rules to enhance the local and global search ability of DE and to overcome the problems of stagnation or premature convergence.

4. Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization

In this section, the proposed mutation strategies are presented. After that, the procedures of the proposed algorithms are presented in detail.

DE/rand/1 is the fundamental mutation strategy developed by Storn and Price [34,39]; and it was widely used in the literature [7]. In this strategy, the three vectors are chosen from the population at random. One of them is selected to be the base vector while the other two vectors form the difference vector that is added to the base vector. Thus, it is able to maintain population diversity and global search capability with no

bias to any specific search direction, but it slows down the convergence speed of DE algorithms [35]. *DE/rand/2* strategy, is as the former strategy with extra two vectors that form another difference vector, which might lead to better perturbation than one-difference-vector-based strategies [35]. Furthermore, it can provide more various differential trial vectors than the *DE/rand/1* strategy that increases its exploration ability of the search space. On the other hand, greedy strategies like *DE/best/1*, *DE/best/2* and *DE/current-to-best/1* incorporate the information of the best solution found so far in the evolutionary process to increase the local search tendency that leads to fast convergence speed of the algorithm. However, the diversity of the population and exploration capability of the algorithm may deteriorate or may be completely lost through a very small number of generations, which cause problems such as stagnation and/or premature convergence. Consequently, in order to overcome the shortcomings of both types of mutation strategies, most of the recent successful algorithms utilize the strategy candidate pool that combines different mutation strategies with different control parameter settings [35,22,and44]]. Contrarily, taking into consideration the weakness of existing greedy strategies, Zhang and Sanderson [50] introduced a new differential evolution (DE) algorithm, named JADE, to improve optimization performance by implementing a new mutation strategy *DE/current-to-pbest* with optional external archive and updating control parameters in an adaptive manner.

Consequently, proposing new mutations strategies that can greatly improve the search capability of DE algorithms and increase the possibility of achieving promising and successful results in complex and large-scale optimization problems is still an open challenge for the evolutionary computation research.

4.1. DE/current-to-ord_best/1 mutation strategy (ord_best)

In order to overcome the limitations of fast but less reliable convergence performance of *DE/current-to-best/1* strategy, this research introduces a less greedy and more explorative variant of the *DE/current-to-best/1* mutation rule named *DE/current-to-ord_best/1* or *ord_best* for short. *ord_best* mutation is designed with a view of balancing the global exploration ability and the local exploitation tendency and enhancing the convergence rate of the algorithm. It based on replacing x^G_{best}, $x^G_{r_1}$ and $x^G_{r_2}$ vectors in Eq. (6) with $x^G_{ord\_best}$, $x^G_{ord\_median}$, and $x^G_{ord\_worst}$. According to *ord_best* mutation, the three vectors that are selected for each individual are ordered according to their fitness values. The best one will be referred to as $x^G_{ord\_best}$, the median will be $x^G_{ord\_median}$, and the worst will be $x^G_{ord\_worst}$. Thus, the trail vector will be generated according to the following equation:

$$v_i^{G+1} = x_i^G + F_i.\left(x^G_{ord\_best} - x_i^G\right) + F_i.\left(x^G_{ord\_median} - x^G_{ord\_worst}\right) \tag{10}$$

where x_i^G is the target vector, F_i is the scaling factor parameter used by individual x_i^G. and $x^G_{ord\_best}$, $x^G_{ord\_median}$ and $x^G_{ord\_worst}$ are the ordered best, median, and worst of the three randomly selected vectors, respectively.

We can observe from mutation Eq. (10) that the incorporation of the objective function value in the mutation strategy has two benefits. Firstly, the first perturbation part is formed by the difference vector in the direction of the best vector among the three randomly selected vectors. Thus, the target vectors are not always attracted toward the same best position found so far by the entire population which helps significantly in avoiding premature convergence at local optima while it accelerates the convergence speed of the proposed algorithm by following the directions of best vectors. Secondly, the second perturbation part of the mutation is formed by the difference vector in the direction of the better vector among the three randomly selected vectors. Therefore, the directed perturbations in the proposed mutation resemble the concept of the gradient as the difference vectors are directed from the worst to the better vectors [12]. Thus, it is used to explore the landscape of the objective function being optimized in different sub-region around the target vectors within search-space through the optimization process. Therefore, the global solution can be reached if all vectors follow the direction of the best vectors besides they also follow the opposite direction of the worst vectors among the randomly selected vectors.

4.2. DE/current-to-ord_pbest/1 mutation strategy (ord_pbest)

In conventional DE, the population size *NP* is suggested to be between 5*D* and 10*D* according to Storn and Price [39] and between 3*D* and 8*D* according to Gamperle at al. [13]. When population size reduction techniques are used to enhance DE performance, the initial population size is increased to be 18*D* as in LSHADE [41]. This increment in population size will affect *ord_best* mutation strategy where the probability of $x^G_{ord\_best}$ vector to be among the global best solutions will be decreased. Thus, the behavior of *ord_best* mutation strategy will approximate *DE/rand/1* mutation strategy.

In order to overcome this drawback, a modified version of our mutation strategy, named *DE/current-to-ord_pbest/1* or *ord_pbest* for short, is proposed to be appropriate for large population size. The proposed modification is to include one vector from the top *p* best vectors according to their fitness values. The other two vectors are randomly selected from the entire population. Then, the three vectors are ordered according to their fitness values. The best one will be referred to as $x^G_{ord\_pbest}$, the median will be $x^G_{ord\_pmedian}$, and the worst will be $x^G_{ord\_pworst}$. Thus, the trail vector will be generated according to the following equation:

$$v_i^{G+1} = x_i^G + F.\left(x^G_{ord\_pbest} - x_i^G\right) + F.\left(x^G_{ord\_pmedian} - x^G_{ord\_pworst}\right) \tag{11}$$

where x_i^G is the target vector, F is the scaling factor parameter used by individual x_i^G.

4.3. The enhanced DE algorithms (EDE and EBDE)

The successful performance of DE algorithm is significantly dependent upon the choice of the scaling factor F and crossover rate Cr [7,34,49]. They play a vital role because they greatly influence the effectiveness, efficiency, and robustness of the algorithm. Furthermore, it is difficult to determine the optimal values of the control parameters for a variety of problems with different characteristics at different stages of evolution. Therefore, current population influences best future control parameters. Hence, for achieving good performance, the proposed algorithm EDE consists of the proposed novel mutation and parameter adaptation method used in SHADE [40].

SHADE maintains a historical memory of length H for both F and Cr. The parameter values in the memory is denoted as $M_{F,i}$ and $M_{Cr,i}$, $(i = 1, 2, \ldots, H)$. Initially, these values are all set to 0.5. In each generation, F and Cr for individual x_i are created by choosing a random index r_i from $[1,\ H]$ and apply the following equations:

$$F_i = randc(M_{F,r_i},\ 0.1) \tag{12}$$

$$Cr_i = randn(M_{Cr,r_i},\ 0.1) \tag{13}$$

where *randc* and *randn* are values sampled according to Cauchy and normal distributions, respectively. If the values generation using Eq. (12) and Eq. (13) are outside the range $[0,\ 1]$, they are repaired. If Cr_i is out of the range, it is replaced by the limit value (0 or 1) closest to the generated value. If $F_i > 1$, it is truncated to 1, and if $F_i \leq 0$, we repeat the execution of Eq. (13) until a valid value is generated.

In the gth generation, if a trial vector u_i^G survives after competing with x_i^G as in Eq. (9), the associated F_i^G and Cr_i^G values are considered as successful and recorded as S_F and S_{Cr}, respectively. Then, at the end of the generation, the content of F memory ($M_{F,k}^{G+1}$) is updated as follows:

$$M_{F,k}^{G+1} = \begin{cases} mean_{WL}(S_F) \text{ if } S_F \neq \emptyset \\ M_{F,k}^{G}, \text{ otherwise} \end{cases} \quad (14)$$

where $mean_{WL}(S_F)$ is the weighted Lehmer mean calculated by:

$$mean_{WL}(S_F) = \frac{\sum_{k=1}^{|S_F|} w_k S_{F,k}^2}{\sum_{k=1}^{|S_F|} w_k S_{F,k}} \quad (15)$$

where $w_k = \Delta f_k / \sum_{k=1}^{|S_F|} \Delta f_k$ and $\Delta f_k = |f(u_k^G) - f(x_k^G)|$.

On the other hand, the memory for Cr ($M_{Cr,k}^{G+1}$) is updated as follows:

$$M_{Cr,k}^{G+1} = \begin{cases} mean_{WA}(S_{Cr}) \text{ if } S_{Cr} \neq \emptyset \\ M_{Cr,k}^{G}, \text{ otherwise} \end{cases} \quad (16)$$

where $mean_{WA}(S_{Cr})$ is the weighted arithmetic mean calculated by:

$$mean_{WA}(S_{Cr}) = \sum_{k=1}^{|S_{Cr}|} w_k S_{Cr,k} \quad (17)$$

Index $k (1 \leq k \leq H)$ determines the position in the memory to be updated. Initially, the value of k is set to 1. It is incremented whenever a new element is inserted into the memory. If $k > H$, it is set to 1. In Eq. (14) and Eq. (16), note that when all individuals in generation G fail to generate a trial vector, the memory is not updated.

The pseudo code of EDE is presented in Fig. 1. EDE depends by default on *ord_best* mutation strategy as illustrated in line 11 in the algorithm. By updating this line, we can use *ord_pbest* to generate new individuals. To distinguish between the two algorithms, we name the latter one as EBDE.

4.4. ESHADE and ELSHADE frameworks

LSHADE is an improved version of SHADE where linear population size reduction (LPSR) is used. In LPSR the population size is decreased according to a linear function according to:

$$N_{G+1} = round\left[\left(\frac{N^{min} - N^{init}}{MAX_{NFE}}\right) * NFE + N^{init}\right] \quad (18)$$

where NFE is the current number of fitness evaluations, MAX_{NFE} is the maximum number of fitness evaluations, N^{init} is the initial population size, and $N^{min} = 4$ which is the minimum number of individuals that DE can work with.

In order to improve the performance of SHADE and LSHADE, we implement a hybridization framework between *pbest* mutation and the proposed mutations strategies. Fig. 2 shows ELSHADE pseudo code.

The framework starts with a mutual population P. Each individual x_i^G in P will generate offspring individual u_i^G using either *pbest* or *ord_best* mutations. This assignment is done according to a class probability variable (FCP). FCP values are randomly selected from memory slots M_{FCP}. By the end of each generation, one memory slot M_{FCP} is updated according to the performance of each mutation. In other words, more individuals will be generated by the better performing mutation. In our

Algorithm: EDE algorithm

```
1  G = 1; N_G = N^init; Archive A = ∅; nfes=0;
2  Initialize population P_G = (x_1,G, ..., x_N,G) randomly;
3  Set all values in M_CR, M_F to 0.5;
4  while termination criteria is not met do
5      S_CR = ∅; S_F = ∅;
6      for i = 1 to N do
7          r_i = Select from [1,H] randomly;
8          CR_i,G = randn_i(M_CR,ri, 0.1);
9          F_i,G = randc_i(M_F,ri, 0.1);
10     end
11     V_G = Generate donor vectors using ord_best mutation;
12     U_G = Generate trial vectors( V_G , CR_G);
13     Evaluate U_G;
14     Update nfes;
15     Update P_G according to the evaluation of U_G;
16     Store successful F_G, and CR_G;
17     Update archive A;
18     If (archive size> |A|)
19         delete randomly selected individuals from archive;
20     end
21     Update memory M_CR, and M_F;
22     G = G +1;
23 end
```

Fig. 1. EDE pseudo code.

Algorithm: ELSHADE algorithm

```
1  G = 1; N_G = N^init; Archive A = ∅; nfes=0;
2  Initialize population P_G = (x_1,G, ..., x_N,G) randomly;
3  Set all values in M_CR, M_F, M_FCP to 0.5;
4  while termination criteria is not met do
5      S_CR = ∅; S_F = ∅; S_FCP = ∅;
6      for i = 1 to N do
7          r_i = Select from [1,H] randomly;
8          CR_i,G = randn_i(M_CR,ri, 0.1);
9          FCP_i,G = M_FCP,ri;
10         F_i,G = randc_i(M_F,ri, 0.1);
11     end
12     [P_G,pbest , P_G,tr_best]= Split(P_G , FCP_G);
13     V_G,pbest = Generate donor vectors using pbest mutation ;
14     V_G,ord_best = Generate donor vectors using ord_best mutation;
15     V_G = Concatenate( V_G,pbest , V_G,ord_best);
16     U_G = Generate trial vectors( V_G , CR_G);
17     Evaluate U_G;
18     Update nfes;
19     Update P_G according to the evaluation of U_G;
20     Store successful FCP_G , F_G, and CR_G ;
21     Update archive A;
22     If (archive size > |A|)
23         delete randomly selected individuals from archive;
24     end
25     Update memory M_CR, M_FCP, and M_F;
26     Calculate N_G+1 according to Eq. (18);
27     if N_G+1 < N_G then
28         Sort individuals in P based on their fitness
29         values and delete lowest N_G − N_G+1 members;
30         Resize archive size |A| according to new |P|;
31     end
32     G = G+1;
33 end
```

Fig. 2. Description of ELSHADE algorithm.

framework, mutation performance is determined by the amount of improvement it achieves for each of its assigned individuals. This is accomplished according to:

$$\omega_{m1} = \sum_{i=1}^{n} f\left(x_i^G\right) - f\left(u_i^G\right) \tag{19}$$

where ω_m is the summation of differences between old and new fitness values for each individual belongs to mutation $m1$, f is the fitness function, x_i^G is the old individual, u_i^G is the offspring individual, and n is the number of individuals belonging to the mutation $m1$. Then, we can calculate the improvement rate (Δ_m) for each mutation using:

$$\Delta_{m1} = \min\left(0.8,\ \max\left(0.2, \frac{\omega_{m1}}{\omega_{m1} + \omega_{m2}}\right)\right) \tag{20}$$

where 0.2 and 0.8 values are the minimum and maximum probabilities assigned to each mutation. Thus, the improvement rate (Δ_m) will be always kept in the range (0.2, 0.8) to maintain the execution of both mutations simultaneously. Finally, the improvement rate (Δ_m) for each mutation will be used to update the memory slot M_{FCP} according to:

$$M_{FCP,g+1} = (1 - c)M_{FCP,g} + c\Delta_{m1} \tag{21}$$

where c is the learning rate. Learning rate constant c is used to benefit from the accumulated knowledge about the performance of each mutation strategy.

The population size reduction is the main difference between ESHADE and ELSHADE which is implemented in the lines 26–31 from Fig. 2. For both of them, *ord_best* mutation is used by default in this algorithm as illustrated in line 14. *ord_best* mutation could be replaced by *ord_pbest* by updating this line in the pseudocode.

Table 1 illustrates a list of the proposed algorithms. EDE is the main algorithm, illustrated in Fig. 1; where *ord_best* mutation strategy is used. EBDE is the modified algorithm, that includes *ord_pbest* rather than *ord_best* mutation strategy. ESHADE, as illustrated in Fig. 2, is the hybridization framework between *ord_best* and SHADE's *pbest* mutation strategies. EBSHADE is the hybridization framework between *ord_pbest* and SHADE's *pbest* mutation strategies. ELSHADE is the hybridization framework between *ord_best* and SHADE's *pbest* mutation strategies along with using population size reduction. Finally, EBLSHADE is ELSHADE with replacing *ord_best* mutation with *ord_pbest*.

5. Numerical experiments and comparisons

In this section, the computational results of the proposed algorithms are discussed along with comparisons with other state-of-the-art algorithms.

To perform a comprehensive evaluation, the presentation of the experimental results is divided into three subsections. First, the setup of the experiments is described in details. Second, the performances of the proposed algorithms according to the numerical results are discussed. Besides, an overall performance comparison between the proposed algorithms is provided. Finally, a comparative study against state-of-the-art algorithms is provided.

Table 1
List of the proposed algorithms.

#	Algorithm
1	EDE
2	EBDE
3	ESHADE
4	EBSHADE
5	ELSHADE
6	EBLSHADE

5.1. Experiments setup

The performance of the proposed algorithms was tested on three different CEC benchmarks. CEC2013, CEC2017, and CEC2010. The latter one is designed for large-scale global optimization. CEC2013 consists of 28 benchmark functions proposed in the CEC2013 special session on real-parameter optimization. A detailed description of these test functions can be found in Ref. [21]. The dimensions (D) of the search spaces are 10, 30, and 50. CEC2017 consists of 29 benchmark functions proposed in the CEC2017 special session on real-parameter optimization [58]. The dimensions (D) of the search spaces are 10, 30, 50, and 100. CEC2010 consists of 20 scalable optimization functions for the CEC2010 special session and competition on large-scale global optimization. A detailed description of these test functions can be found in Ref. [42]. The dimension (D) of the search space was 1000.

To evaluate the performance of algorithms, we use the solution error measure ($f(x)-f(x^*)$), where x is the best solution obtained by algorithms in one run, and x^* is the well-known global optimum of each benchmark function. The obtained best error values and standard deviations smaller than 10^{-8} are taken as zero [21]. The maximum number of function evaluations (FEs), the terminal criteria, is set to $10000 \times D$ for CEC2013 and CEC2017. For CEC2010, the maximum number of function evaluations (FEs) is set to $3000 \times D$. All experiments for each function and each algorithm are repeated 51 times independently for CEC2013 and CEC2017, and 25 times for CEC2010. If the population size reduction is omitted, the population size NP and memory size are set to 100 for all dimensions as recommended in Tanabe and Fukunaga [40]. On the other hand, if the population size reduction is included, the population size NP is set to $18 \times D$ and memory size is set to 5 as it is used in LSHADE.

To compare and analyze the solution quality from a statistical point of view [14], the results were compared using two non-parametric statistical hypothesis tests. The first is the Friedman test that obtains the final rankings of different algorithms for all functions. The null hypothesis for this test is that "*there is no difference among the performance of all algorithms*" and that alternative hypothesis states "*there is a difference among the performance of all algorithms*".

The second test is the multi-problem Wilcoxon signed-rank test that checks the differences between all algorithms for all functions. Wilcoxon signed-rank test was used with 0.05 significance level [14]. According to Wilcoxon signed-rank test, R^+ is the sum of ranks for the functions in which the first algorithm outperforms the second algorithm in the row and R^- is the sum of ranks for the opposite. Larger ranks indicate larger performance discrepancy. For each of the competitive functions, the numbers in better, equal, and worse denote the number of problems in which the first algorithm is better than, equal to, or worse than the second algorithm. Based on the result of the test, one of three signs ($+$, $-$, and $\approx$) is assigned for the comparison of any two algorithms. Plus ($+$) sign means the first algorithm is significantly better than the second, minus ($-$) sign means the first algorithm is significantly worse than the second, and approximation ($\approx$) sign means that there is no significant difference between the two algorithms. As a null hypothesis, it is assumed that "*there is no difference between the mean results of the two samples*". Whereas the alternative hypothesis is "*there is a difference in the mean results of the two samples*". The *p-value* was used and compared with the significance level. The null hypothesis was rejected if the *p-value* is less than or equal the significance level. The *p-values* under the significance level are shown in bold.

MATLAB R2014 was used to conduct all experiments, and SPSS 20.00 was used to compute the statistical tests in this paper. The source code is available upon request.

5.2. Numerical results and statistical comparisons of the proposed algorithms

The detailed statistical results of the proposed algorithms on CEC2013, CEC2017, and CEC2010 benchmarks are illustrated in

Tables S2–S10 in the supplementary material file. It includes the obtained mean and standard deviation values of error from optimum solution overall runs for all benchmark functions and dimensions. The statistical results of SHADE, LSHADE are also included. In this section, the statistical analysis results of applying Friedman and multiple-problem Wilcoxon tests between the proposed algorithms will be discussed.

Table 2 lists the average ranks according to Friedman test for the proposed algorithms using CEC2013. SHADE and LSHADE are also included as base algorithms. The best ranks are shown in bold and the second ranks are underlined. From this table, we can see that *p-value* computed through Friedman test for all dimensions are less than 0.05. Thus, it can be concluded that there is a significant difference between the performances of the algorithms.

Besides, we can clearly see that LSHADE family algorithms outperform SHADE family algorithms in all dimensions, which demonstrate the efficiency of applying population size reduction. ELSHADE obtains the best ranking in 10 dimensions. In 30 and 50 dimensions. EBLSHADE obtains the best ranking. LSHADE ranked best in 100 dimensions. Regarding mean ranking, EBLSHADE gets the first ranking followed by LSHADE and ELSHADE. Both of ESHADE and EBSHADE were better than SHADE in all dimensions. This observation confirms the positive effect of *ord_best* and *ord_pbest* mutation strategies on SHADE algorithm. When population size reduction was applied, the performance of *ord_best* mutation was deteriorated. This behavior is due to the increment in population size, where the increment in population size will decrease the probability of $x^G_{ord\_best}$ vector to be among the global best solutions. Thus, the behavior of *ord_best* mutation strategy will approximate *DE/rand/1* mutation strategy. On the other hand, *ord_pbest* mutation provides a suitable alternative for this case. In general, the performance EBLSHADE that use this mutation is better than LSHADE.

Using CEC2017 benchmark test functions, Table 3 lists the average ranks according to Friedman test. We can see that *p-value* computed through Friedman test for all dimensions are less than 0.05. Thus, it can be concluded that there is a significant difference between the performances of the algorithms. Actually, the observations are consistent here. The ranking of all algorithms was the same, except for EBDE where its rank improved from fourth to second. Again, we can clearly observe the positive effect of *ord_best* and *ord_pbest* mutation strategies. Regarding mean ranking, EBLSHADE again was the best algorithm.

Starting with a large population size when solving large-scale optimization problems increases the memory complexity. Thus, we will limit our study to SHADE family and the effect of our proposed mutation strategy using CEC2010 benchmark. Table 4 lists the average ranks according to Friedman test. We can see that *p-value* computed through Friedman test is less than 0.05. Thus, we can conclude that there is a significant difference between the performances of the algorithms. The best rank was for ESHADE followed by SHADE and EDE respectively.

Table 5 summarizes the statistical analysis results of applying Wilcoxon's test between EBLSHADE and other proposed algorithms using CEC2013 for all dimensions. According to this test, EBLSHADE is significantly better than EDE in both 10 and 100 dimensions. For 30 dimensions, it is significantly better than five algorithms named SHADE, EDE, ESHADE, EBSHADE, and LSHADE. It was also significantly better than five algorithms named SHADE, EDE, EBDE, ESHADE, and EBSHADE in 50 dimensions. On the other hand, in the cases of EBLSHADE versus LSHADE, EBLSHADE gets higher R^+ values than R^- in all dimensions except for 100 dimensions.

The statistical analysis results of applying Wilcoxon's test between EBLSHADE and other proposed algorithms using CEC2017 for all dimensions are summarized in Table 6. In general, the performance of EBLSHADE is better than other algorithms in this benchmark. Except for EBDE, EBLSHADE was significantly better than all other algorithms in 30, 50, and 100 dimensions. For 10 dimensions, it is significantly better than EDE, ESHADE, and EBSHADE algorithms. In the cases of EBLSHADE versus LSHADE, EBLSHADE gets higher R^+ values than R^- in all dimensions except for 10 dimensions.

Table 7 gives the statistical analysis results of applying Wilcoxon's test between ESHADE and other algorithms using CEC2010. We can observe that ESHADE is significantly better SHADE and EDE. ESHADE provides higher R^+ values than R^- in both cases.

Fig. 3 illustrates the parameter adaptation process, sample functions (2, 7, 11, and 28 in 30 dimensions) were selected here, and the rest figures could be found in the supplementary material file. During the optimization process, adopted *F* and *Cr* values were recorded. Each curve represents one run. Generally, we can observe that the behavior of the adaptation process for each repeated run follows the same pattern. On the other hand, we can notice that this behavior is problem dependent. The adaptation behavior for function #11 demonstrates the capability of the adaptation process to pick the optimal *F* and *Cr* values during the optimization process. Depending on the characteristics of the optimized problem, *F* and *Cr* parameters adaptively converge to different sets of values.

As previously mentioned, a hybridization framework between and SHADE's *pbest* mutation strategies is proposed in this paper. The hybridization process is performed according to a class probability variable *FCP* which is randomly selected from memory slots M_{FCP}. By the end of each generation, one memory slot is updated according to the performance of each mutation. Fig. 4 shows the hybridization process according to M_{FCP} values for functions 4, 13, 20, and 26 in 30 dimensions. The blue curve represents the probability of selecting *pbest* mutation strategy to generate new individuals, while the red line represents the probability of selecting *ord_pbest* mutation strategy. We can observe that the hybridization behavior is problem dependent. Additionally, Fig. 4 function #26 demonstrate the capability of the hybridization process to change the probability of each mutation strategy according to the

Table 2
Average ranks for all algorithms across all problems and all dimensions using CEC2013.

Algorithm	10*D*	30*D*	50*D*	100*D*	Mean Ranking	Rank
EBLSHADE	4.07	**2.96**	**3.05**	3.82	**3.48**	**1**
LSHADE	3.63	3.48	3.30	**3.59**	3.50	2
ELSHADE	**3.38**	3.77	3.57	4.46	3.79	3
EBDE	3.77	3.32	3.80	4.73	3.91	4
ESHADE	5.21	5.45	5.02	4.21	4.97	5
EBSHADE	4.75	5.23	5.48	4.75	5.05	6
SHADE	5.27	6.07	6.02	5.00	5.59	7
EDE	5.93	5.71	5.75	5.43	5.71	8
Friedman-P-value	0.000	0.000	0.000	0.000		

Table 3
Average ranks for all algorithms across all problems and all dimensions using CEC2017.

Algorithm	10*D*	30*D*	50*D*	100*D*	Mean Ranking	Rank
EBLSHADE	3.59	**2.62**	2.38	**2.84**	**2.86**	1
EBDE	3.84	**2.62**	**2.05**	3.34	2.97	2
LSHADE	**3.40**	3.60	3.43	3.12	3.39	3
ELSHADE	3.57	3.26	3.86	3.84	3.63	4
ESHADE	5.53	5.78	5.52	4.81	5.41	5
EBSHADE	5.41	5.72	5.95	5.40	5.62	6
SHADE	4.43	6.53	6.74	6.03	5.94	7
EDE	6.22	5.86	6.07	6.60	6.19	8
Friedman-P-value	0.000	0.000	0.000	0.000		

Table 4
Average ranks for EDE, SHADE, and ESHADE across all problems using CEC2010.

Algorithm	1000*D*	Rank
ESHADE	**1.35**	1
SHADE	2.30	2
EDE	2.35	3
Friedman-P-value	0.001	

Table 5
Wilcoxon's test between EBLSHADE and other algorithms using CEC2013 for $D = 10$, 30, 50 and 100.

D	Algorithms	R^+	R^-	p-value	+	≈	−	Dec.
10	EBLSHADE vs SHADE	131.5	85.5	0.142	14	9	5	≈
	EBLSHADE vs LSHADE	119.5	90.5	0.588	10	8	10	≈
	EBLSHADE vs EDE	147	43	**0.036**	15	9	4	+
	EBLSHADE vs EBDE	37	99	0.109	5	12	11	≈
	EBLSHADE vs ESHADE	129	61	0.171	14	9	5	≈
	EBLSHADE vs EBSHADE	121	69	0.295	11	9	8	≈
	EBLSHADE vs ELSHADE	92	118	0.627	9	8	11	≈
30	EBLSHADE vs SHADE	252	24	**0.001**	19	5	4	+
	EBLSHADE vs LSHADE	94	42	0.179	11	12	5	≈
	EBLSHADE vs EDE	226	27	**0.001**	19	6	3	+
	EBLSHADE vs EBDE	100	53	0.266	12	11	5	≈
	EBLSHADE vs ESHADE	238	15	**0.000**	20	6	2	+
	EBLSHADE vs EBSHADE	209	22	**0.001**	18	7	3	+
	EBLSHADE vs ELSHADE	118.5	34.5	**0.047**	13	11	4	+
50	EBLSHADE vs SHADE	247	26	**0.000**	20	4	4	+
	EBLSHADE vs LSHADE	131.5	78.5	0.322	12	8	8	≈
	EBLSHADE vs EDE	257	19	**0.000**	20	5	3	+
	EBLSHADE vs EBDE	144	46	**0.049**	16	9	3	+
	EBLSHADE vs ESHADE	211	20	**0.001**	17	7	4	+
	EBLSHADE vs EBSHADE	259	17	**0.000**	19	5	4	+
	EBLSHADE vs ELSHADE	92	61	0.463	10	11	7	≈
100	EBLSHADE vs SHADE	199	126	0.326	15	3	10	≈
	EBLSHADE vs LSHADE	78	153	0.192	7	7	14	≈
	EBLSHADE vs EDE	226	74	**0.030**	18	4	8	+
	EBLSHADE vs EBDE	186	90	0.144	17	5	6	≈
	EBLSHADE vs ESHADE	202	123	0.288	15	3	10	≈
	EBLSHADE vs EBSHADE	198	127	0.339	16	3	9	≈
	EBLSHADE vs ELSHADE	180.5	95.5	0.196	16	5	7	≈

characteristics of the optimized problem.

According to the previous statistical analysis, we can conclude that the proposed mutation strategies are very effective, and they are able to enhance the performance of SHADE and LSHADE significantly.

5.3. Comparison against state-of-the-art algorithms

The proposed algorithms were compared with state-of-the-art optimization algorithms using CEC2013. These algorithms are:

- Success History based parameter adaptation for DE (SHADE) [40]
- SHADE with linear population size reduction (LSHADE) [41]
- Super-fit Multi-criteria Adaptive Differential Evolution (SMADE) [4]
- Covariance Matrix Adaptive Evolution Strategy (CMAES) [17]
- Self-adaptive Differential Evolution with $_p$BX cross over (MDE_$_p$BX) [18]
- Cooperatively Coevolving Particle Swarms Optimizers (CCPSO2) [20]
- CMA-ES with increasing population size (NBIPOPaCMAES) [63]

Table 6
Wilcoxon's test between EBLSHADE and other algorithms using CEC2017 for $D = 10$, 30, 50 and 100.

D	Algorithms	R^+	R^-	p-value	+	≈	−	Dec.
10	EBLSHADE vs SHADE	169.5	61.5	0.061	13	8	8	≈
	EBLSHADE vs LSHADE	66	105	0.396	7	11	11	≈
	EBLSHADE vs EDE	258	18	**0.000**	21	6	2	+
	EBLSHADE vs EBDE	166.5	64.5	0.076	14	8	7	≈
	EBLSHADE vs ESHADE	230.5	45.5	**0.005**	18	6	5	+
	EBLSHADE vs EBSHADE	211	20	**0.001**	16	8	5	+
	EBLSHADE vs ELSHADE	90	81	0.845	10	11	8	≈
30	EBLSHADE vs SHADE	308	17	**0.000**	24	4	1	+
	EBLSHADE vs LSHADE	243	10	**0.000**	18	7	4	+
	EBLSHADE vs EDE	274	26	**0.000**	22	5	2	+
	EBLSHADE vs EBDE	113.5	117.5	0.945	11	8	10	≈
	EBLSHADE vs ESHADE	284	16	**0.000**	23	5	1	+
	EBLSHADE vs EBSHADE	305	20	**0.000**	23	4	2	+
	EBLSHADE vs ELSHADE	166	44	**0.000**	14	9	6	+
50	EBLSHADE vs SHADE	346	32	**0.000**	25	2	2	+
	EBLSHADE vs LSHADE	254	46	**0.003**	19	5	5	+
	EBLSHADE vs EDE	375	31	**0.000**	26	1	2	+
	EBLSHADE vs EBDE	133	218	0.280	11	3	15	≈
	EBLSHADE vs ESHADE	340	38	**0.000**	25	2	2	+
	EBLSHADE vs EBSHADE	341	37	**0.000**	25	2	2	+
	EBLSHADE vs ELSHADE	270	30	**0.001**	22	5	2	+
100	EBLSHADE vs SHADE	341	94	**0.008**	22	0	7	+
	EBLSHADE vs LSHADE	294	112	**0.038**	16	1	12	+
	EBLSHADE vs EDE	391	44	**0.000**	27	0	2	+
	EBLSHADE vs EBDE	207.5	198.5	0.918	17	1	11	≈
	EBLSHADE vs ESHADE	315	91	**0.011**	22	1	6	+
	EBLSHADE vs EBSHADE	311	95	**0.014**	21	1	7	+
	EBLSHADE vs ELSHADE	301.5	49.5	**0.011**	21	3	5	+

Table 7
Wilcoxon's test between ESHADE and other algorithms using CEC2010 for $D = 1000$.

Algorithms	R^+	R^-	p-value	+	≈	−	Dec.
ESHADE vs SHADE	181	29	**0.005**	18	0	2	+
ESHADE vs EDE	151	20	**0.004**	14	2	4	+

- CMAES with an iterated local search (iCMAESILS) [64]

SHADE, LSHADE, SMADE, and MDE_$_p$BX have been briefly introduced in section 3. CMAES proposed by Hansen and Ostermeier [17], is an evolution strategy (ES) based on completely de-randomized self-adaptation. CCPSO2, proposed by Li and Yao [20], is an improved version of particle swarm optimization (PSO). CCPSO2 adopts a new PSO position update rule that relies on Cauchy and Gaussian distributions to

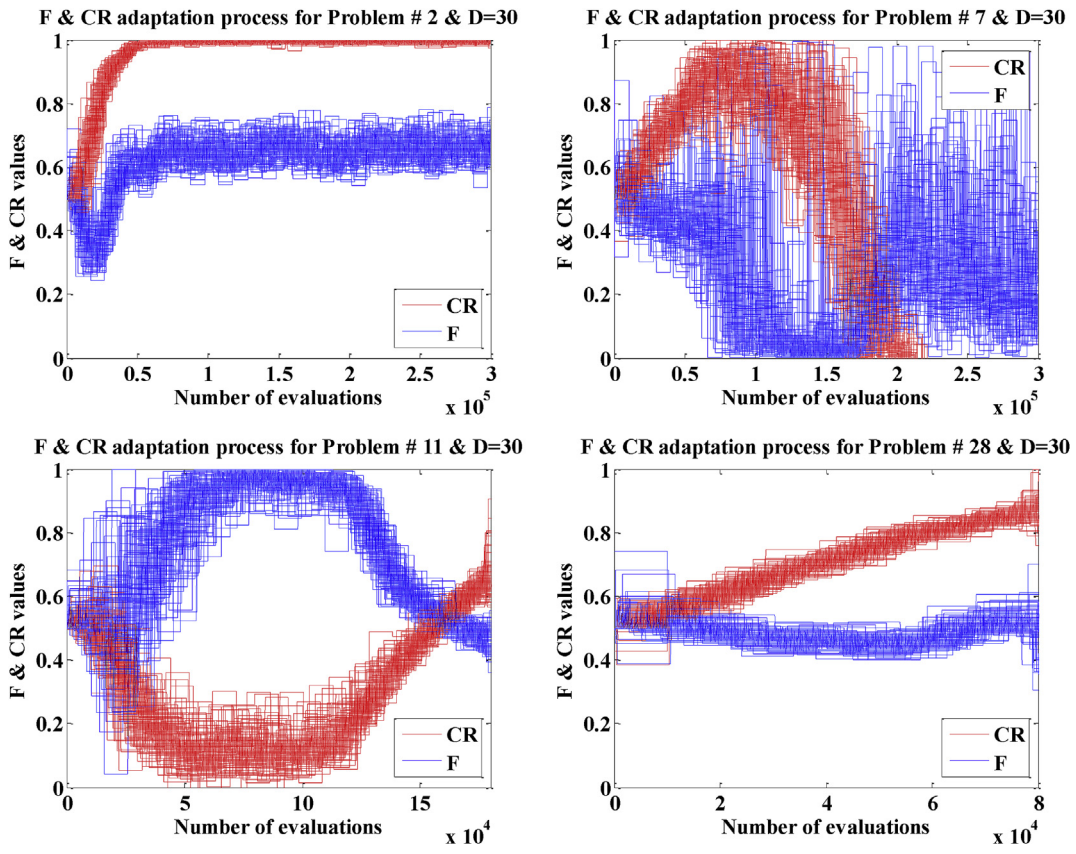

Fig. 3. *F* and *Cr* Adaptation process for functions 2, 7, 11, and 28 in 30 dimensions.

sample new points in the search space, to determine the coevolving subcomponent sizes of the variables dynamically. NBIPOPaCMAES is one of six versions of CMAES proposed by Loshchilov [63]. This version includes a restart strategy with increasing population size (IPOP) in addition to decreasing initial step-size. NBIPOPaCMAES was the best performing algorithms among 21 other algorithms according to CEC2013 competition. iCMAESILS, proposed by Liao and Stutzle [64], is a CMAES version with an iterated local search and increasing population size. It was ranked second in CEC2013 competition.

Although CEC2013 has a definition for 100 dimensions, it was not included in CEC2013 competition and the performance of these algorithms was not reported for 100 dimensions. Thus, another comparison with four state-of-the-art and advanced DE variants has been carried out with D = 30,50 and 100. These DE variants are: EPSDE [22], CoDE [44], CoBiDE [45] and CIPDE [52]. EPSDE [22], CoDE [44] have been briefly introduced in section 3, These DE variants are selected due to their well-known competitive performance and high popularity. Moreover, they represent different types of improvements that were discussed in section 3. CoBiDE proposed by Wang et al. [45], is an advanced DE with covariance matrix learning crossover and control parameter adaptation. On the other hand, CIPDE proposed by Zheng et al. [52], is a DE variant powered by collective information. Actually, it is among the most recent DE-based algorithm for global numerical optimization with outstanding performance. The detailed statistical results of the compared algorithms on CEC2013, CEC2017, and CEC2010 benchmarks are illustrated in Tables S11–S21 in the supplementary material file.

Table 8 lists the average ranks of the compared algorithms according to Friedman test using CEC2013 for $D = 10$, 30 and 50, respectively. From this table, we can see that *p-value* computed through Friedman test for all dimensions are less than 0.05. Thus, it can be concluded that there is a significant difference between the performances of the algorithms.

EBLSHADE was ranked second for 10 and 30 dimensions. NBIPOPaCMAES obtains the best ranking in all dimensions. Regarding mean ranking, NBIPOPaCMAES gets the first ranking followed by EBLSHADE and iCMAESILS.

According to Wilcoxon's test illustrated in Table 9, EBLSHADE was significantly better than SMADE, MDE-*p*BX, CMAES, and CCPSO2 for all dimensions. On the other hand, there was no significant difference between EBLSHADE and both of NBIPOPaCMAES and iCMAESILS for all dimensions (except for iCMAESILS in 50 dimensions).

The Comparison with state-of-the-art-and advanced DE Variants on the CEC 2013 is illustrated in Tables 10 and 11. Table 10 lists the average ranks of the five algorithms according to Friedman test for $D = 30$, 50 and 100, respectively. From this table, it can be noted that *p-value* computed through Friedman test for all dimensions are less than 0.05. Thus, it can be concluded that there is a significant difference between the performances of the algorithms.

Moreover, it can be clearly observed from Table 10 that, EBLSHADE obtains the best ranking in all dimensions. Regarding mean ranking, EBLSHADE gets the first ranking followed by CIPDE and CoBiDE respectively.

Table 11 summarizes the statistical analysis results of applying Wilcoxon's test between EBLSHADE and other compared. EBLSHADE was significantly better than all the compared algorithms for all dimensions (except for CIPDE in 100 dimensions). From the results shown in Table 11, we can see that EBLSHADE provides higher R^+ values than R^- in all cases.

For CEC2017 benchmark, we will study the effect of integrating the proposed mutation strategies within two of the top 5 algorithms according to CEC2017. These algorithms are LSHADE-SPACMA [62] and LSHADE-cnEpSin [65]. LSHADE-SPACMA have been briefly introduced in section 3. LSHADE-cnEpSin is an LSHADE variant proposed by Awad et al. [65]. It is an extension of LSHADE-EpSin that was ranked as the joint winner in CEC2016. LSHADE-cnEpSin is based on an ensemble of sinusoidal approaches and covariance matrix learning for the crossover operator.

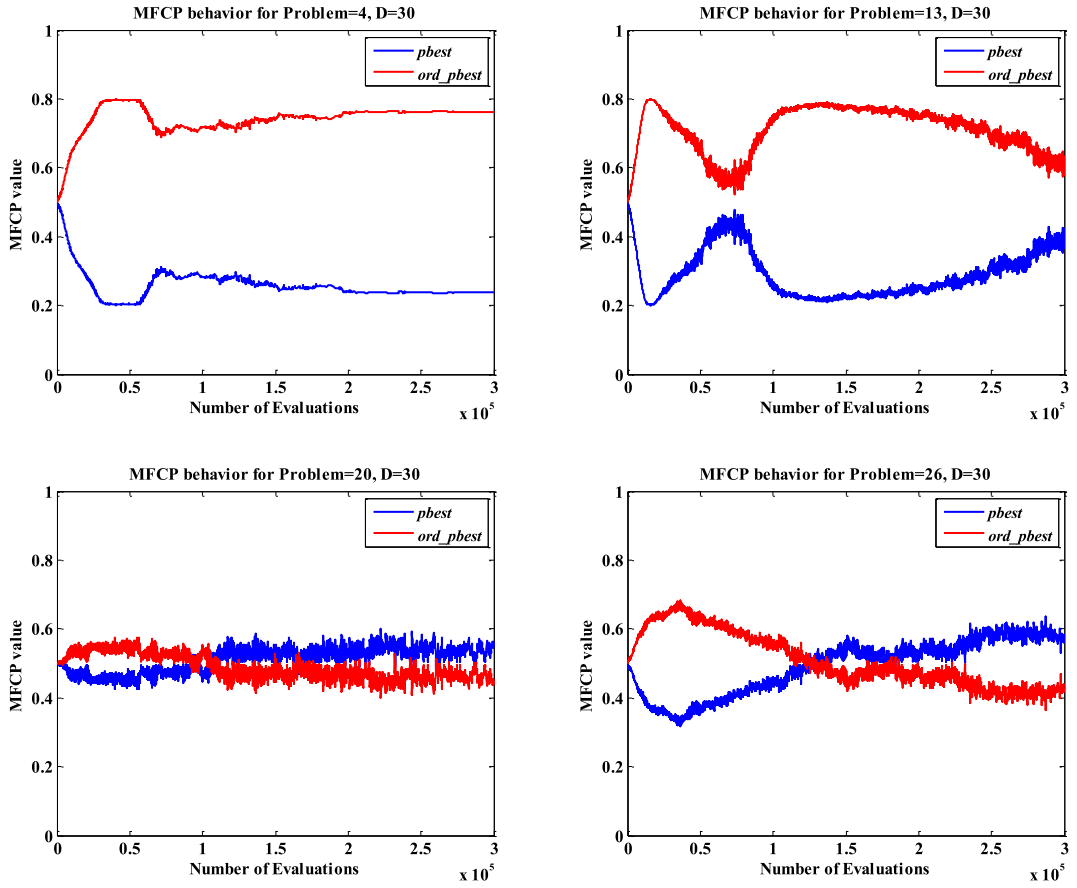

Fig. 4. Hybridization process according to MFCP values for functions 4, 13, 20, and 26 in 30 dimensions.

Table 8
Average ranks for all algorithms across all problems for D = 10, 30, and 50 using CEC2013.

Algorithm	10*D*	30*D*	50*D*	Mean Ranking	Rank
NBIPOPaCMAES	**2.29**	**2.59**	**2.57**	**2.48**	1
EBLSHADE	2.59	2.57	2.86	2.67	2
iCMAESILS	3.02	2.93	2.84	2.93	3
SMADE	3.68	4.09	4.18	3.98	4
CCPSO2	5.66	4.89	4.84	5.13	5
MDE-*p*BX	4.95	5.29	5.25	5.16	6
CMAES	5.82	5.64	5.46	5.64	7
Friedman-P-value	0.000	0.000	0.000		

In Table 12, the effect of replacing *pbest* with *ord_pbest* mutation is illustrated using the average ranks according to Friedman for all dimensions. LSHADE-cnEpSin variant with *ord_pbest* mutation has the best rank in 30 dimensions and 50 dimensions, and the second rank in 10 dimensions. Besides, LSHADE-SPACMA variant with *ord_pbest* mutation is ranked third in 30 and 100 dimensions followed by LSHADE-SPACMA. Although both algorithms were not constantly improved overall dimensions, *ord_pbest* mutation is considered as a very competitive mutation comparing with *pbest* mutation. This observation is supported by Friedman's *p-value* which is higher than 0.05 for all dimensions. This means that there is no significant difference in the performance of the algorithms. We note the same observation on the statistical comparison using Wilcoxon's test. All *p-values* were higher than 0.05 for all cases. This means that there is no significant difference between the four algorithms (see Table 13).

Regarding CEC'2010, the proposed algorithms were compared to DE-based algorithms that were all tested on this test suite in this competition. These algorithms are:

- Self-adaptive Differential Evolution Algorithm (jDElsgo) [3].
- DE Enhanced by Neighborhood Search (SDENS) [47].
- Cooperative Co-evolution with Delta Grouping (DECCDML) [29].

Self-adaptive differential evolution algorithm (jDElsgo) was proposed by Brest et al. [3]. Crossover and scaling factor control parameters were adopted during the optimization process. jDElsgo also includes a population size reduction mechanism. SDENS introduced by Wang et al. [47], is a Sequential Differential Evolution (DE) enhanced by neighborhood search. SDENS depends on the local and global neighborhood to create new individuals. DE is used as a population-based optimizer to accelerate the convergence speed. Omidvar et al. [29] proposed cooperative co-evolution with delta grouping (DECCDML). Delta grouping as a new decomposition strategy was introduced in DECCDML where averaged difference in a certain variable across the entire population is measured and used for identifying interacting variables.

Table 14 lists the average ranks of the six algorithms for D = 1000. The *p-value* computed through Friedman test is less than 0.05. Thus, it can be concluded that there is a significant difference between the performances of the algorithms.

It can be observed from this table that, jDElsgo obtains the best mean ranking while ESHADE and SHADE obtain the second and third mean ranking among all algorithms.

Table 15 summarizes the statistical analysis results of applying Wilcoxon's test between ESHADE and the compared algorithms on the functions with dimension 1000. From the results shown in this table, we can see that ESHADE provides higher R^+ values than R^- in all the cases with exception to jDElsgo. ESHADE outperforms SDENS, DECC-DML, SHADE and EDE algorithms significantly while it is comparable with jDElsgo algorithm. It is noteworthy to mention that this is the first time that SHADE has been used to solve large-scale optimization problems

Table 9
Wilcoxon's test between EBLSHADE and other algorithms for $D = 10$, 30, and 50 using CEC2013.

D	Algorithms	R^+	R^-	*p-value*	+	≈	−	Dec.
10	EBLSHADE vs SMADE	242.5	33.5	**0.000**	21	5	2	+
	EBLSHADE vs MDE-*p*BX	292	8	**0.000**	23	4	1	+
	EBLSHADE vs CMAES	291	9	**0.000**	23	4	1	+
	EBLSHADE vs CCPSO2	372.5	33.5	**0.000**	25	0	3	+
	EBLSHADE vs NBIPOPaCMAES	92	208	0.097	8	4	16	≈
	EBLSHADE vs iCMAESILS	160	246	0.327	15	0	13	≈
30	EBLSHADE vs SMADE	287	13	**0.000**	21	4	3	+
	EBLSHADE vs MDE-*p*BX	346	5	**0.000**	25	2	1	+
	EBLSHADE vs CMAES	297	3	**0.000**	23	4	1	+
	EBLSHADE vs CCPSO2	298	27	**0.000**	22	3	3	+
	EBLSHADE vs NBIPOPaCMAES	116	160	0.503	10	5	13	≈
	EBLSHADE vs iCMAESILS	145	206	0.438	13	2	13	≈
50	EBLSHADE vs SMADE	267	33	**0.000**	20	4	4	+
	EBLSHADE vs MDE-*p*BX	346	5	**0.000**	25	2	1	+
	EBLSHADE vs CMAES	263	37	**0.000**	20	4	4	+
	EBLSHADE vs CCPSO2	293.5	31.5	**0.000**	22	3	3	+
	EBLSHADE vs NBIPOPaCMAES	101.5	174.5	0.267	10	5	13	≈
	EBLSHADE vs iCMAESILS	88	263	**0.026**	9	2	17	−

Table 10
Average ranks for DE-based algorithms across all problems for $D = 30$, 50, and 100 using CEC2013.

Algorithm	30*D*	50*D*	100*D*	Mean Ranking	Rank
EBLSHADE	**1.71**	**1.80**	**2.04**	**1.85**	1
CIPDE	2.57	2.66	2.45	2.56	2
CoBiDE	3.29	2.89	2.89	3.02	3
CoDE	3.14	3.21	3.23	3.20	4
EPSDE	4.29	4.43	4.39	4.37	5
Friedman-P-value	0.000	0.000	0.000		

with $D = 1000$.

For the purpose of analyzing the convergence behavior and parameter adaptation of EDE algorithm as it contains the proposed mutation only, the convergence characteristics are illustrated in the supplementary material file (Fig. S1). The behavior of the adaptive parameters *F* and *Cr* for 28 functions is illustrated in the supplementary material file (Fig. S2). Additionally, it can be deduced from Fig. S3 that the proposed mutation *ord_best* converges faster than the basic *DE/current-to-best/1* mutation in most cases. The convergence speed of the EDE algorithm is fast at the early stage of the optimization process for all functions with different shapes, complexity, and dimensions. Furthermore, the convergence speed is dramatically decreased and its improvement is found to be significant in the middle and later stages of the optimization process. Additionally, the convergence figure suggests that the EDE algorithm can reach the global solution or better solution in most problems in a fewer number of generations less than the maximum predetermined number of generations. In general, EDE is scalable enough and can balance greatly the exploration and exploitation abilities until the maximum FEs is reached. Therefore, the proposed algorithms are proven as effective and powerful approaches for solving unconstrained global optimization problems within a limited number of function evaluations, which is a very important issue when dealing with real-world problems.

Table 11
Wilcoxon's test between EBLSHADE and DE algorithms for $D = 30$, 50, and 100 using CEC2013.

D	Algorithms	R^+	R^-	*p-value*	+	≈	−	Dec.
30	EBLSHADE vs EPSDE	300	0	**0.000**	24	4	0	+
	EBLSHADE vs CoDE	231	22	**0.001**	19	6	3	+
	EBLSHADE vs CoBiDE	282	18	**0.000**	22	4	2	+
	EBLSHADE vs CIPDE	208.5	44.5	**0.008**	17	6	5	+
50	EBLSHADE vs EPSDE	290	10	**0.000**	22	4	2	+
	EBLSHADE vs CoDE	312	39	**0.001**	21	2	5	+
	EBLSHADE vs CoBiDE	257	43	**0.002**	20	4	4	+
	EBLSHADE vs CIPDE	238	38	**0.002**	19	5	4	+
100	EBLSHADE vs EPSDE	292	8	**0.000**	23	4	1	+
	EBLSHADE vs CoDE	298.5	52.5	**0.002**	20	2	6	+
	EBLSHADE vs CoBiDE	272	53	**0.003**	19	3	6	+
	EBLSHADE vs CIPDE	223	102	0.104	15	3	10	≈

Table 12
Average ranks for the compared algorithms across all problems and dimensions using CEC2017.

Algorithm	10*D*	30*D*	50*D*	100*D*	Mean Ranking	Rank
LSHADE-cnEpSin	**2.24**	2.52	2.41	**2.29**	**2.37**	1
EBLSHAD-cnEpSin	2.52	**2.12**	**2.33**	2.79	2.44	2
LSHADE-SPACMA	2.52	2.74	2.59	2.48	2.58	3
EBLSHADE-SPACMA	2.72	2.62	2.67	2.43	2.61	4
Friedman-P-value	0.410	0.187	0.689	0.474		

From the above results, comparisons, and discussion through this section, we can conclude that the proposed mutation strategies are of better searching quality, efficiency, and robustness for solving small, moderate, and large-scale unconstrained global optimization problems. It is clear that the proposed mutation strategies keep the balance between the local optimization speed and the global optimization diversity in challenging optimization environment with invariant performance. Note that the main contribution of this study is to propose a new mutation strategy that could be integrated with other DE-based algorithms, and not to propose a "Best" algorithm or competitor to defeat other state-of-the-art algorithms.

6. Conclusion and future work

In order to enhance the overall performance of the basic DE algorithm, introducing new mutation rules combined with self-adaptive schemes for scaling factor and crossover rate is a must although it is a challenging task. In this paper, two new mutation strategies (*ord_best* and *ord_pbest*) were proposed. The proposed mutation strategies are shown to enhance the global and local search capabilities and to increase the convergence speed. Furthermore, the proposed mutation strategies were incorporated into SHADE and LSHADE to enhance their performance. In the proposed framework, both mutations *DE/current-to-pbest/1* and *ord_best* (or *ord_pbest*) work simultaneously on the same population, but more individuals will be assigned gradually to the better performance mutation. In order to test the effectiveness of the proposed algorithms, they were applied to solve CEC2013, CEC2017, and CEC2010 benchmarks. The Friedman test and Wilcoxon's test were used to analyze the performance of the proposed algorithms statistically.

Table 13
Wilcoxon's test between LSHADE-cnEpSin and LSHADESPACMA, EBLSHADE-cnEpSin, and EBLSHADE-cnEpSin algorithms for $D = 10$, 30, 50, and 100 respectively.

D	Algorithms	R^+	R^-	*p-value*	+	≈	−	Dec.
10	LSHADE-cnEpSin vs LSHADESPACMA	108	123	0.794	13	8	8	≈
	LSHADE-cnEpSin vs EBLSHADESPACMA	129	102	0.639	13	8	8	≈
	LSHADE-cnEpSin vs EBLSHADE-cnEpSin	123	67	0.260	12	10	7	≈
30	LSHADE-cnEpSin vs LSHADESPACMA	192.5	83.5	0.097	16	6	7	≈
	LSHADE-cnEpSin vs EBLSHADESPACMA	145	131	0.831	11	6	12	≈
	LSHADE-cnEpSin vs EBLSHADE-cnEpSin	93.5	182.5	0.176	7	6	16	≈
50	LSHADE-cnEpSin vs LSHADESPACMA	200.5	124.5	0.307	16	4	9	≈
	LSHADE-cnEpSin vs EBLSHADESPACMA	182	169	0.869	13	3	13	≈
	LSHADE-cnEpSin vs EBLSHADE-cnEpSin	163.5	187.5	0.760	12	3	14	≈
100	LSHADE-cnEpSin vs LSHADESPACMA	225	126	0.902	18	3	8	≈
	LSHADE-cnEpSin vs EBLSHADESPACMA	198.5	152.5	0.559	13	3	13	≈
	LSHADE-cnEpSin vs EBLSHADE-cnEpSin	223.5	182.5	0.641	15	1	13	≈

Table 14
Average ranking of all algorithms according to Friedman test for $D = 1000$ using CEC2010.

Algorithm	Mean Ranking	Rank
jDElsgo	**2.15**	1
ESHADE	2.80	2
SHADE	3.75	3
DECC-DML	3.95	4
SDENS	4.15	5
EDE	4.20	6
Friedman-P-value	0.000	

Table 15
Wilcoxon's test between ESHADE and other algorithms on $D = 1000$ using CEC2010.

Algorithms	R^+	R^-	*p-value*	+	≈	−	Dec.
ESHADE vs jDElsgo	87	123	0.502	6	0	14	≈
ESHADE vs SDENS	164	46	**0.033**	13	0	7	+
ESHADE vs DECC-DML	163	47	**0.030**	12	0	8	+
ESHADE vs SHADE	181	29	**0.005**	18	0	2	+
ESHADE vs EDE	151	20	**0.004**	14	2	4	+

As a summary of results, the performance of the proposed algorithms was statistically superior to and competitive with other recent and well-known state-of-the-art algorithms in the majority of functions and for different dimensions. Moreover, they are easily implemented and have been proven to be reliable approaches for real parameter optimization with small, moderate and large dimensions.

Several current and future works can be developed from this study. First, current research efforts focus on how to modify the proposed algorithms for handling constrained and multi-objective optimization problems. Secondly, it would be very interesting to propose other adaptive versions from the proposed algorithms by combining other mutation strategies and other adaptive mechanisms. Future research studies may focus on integrating the proposed mutation strategy with other self-adaptive DE variants like CIPDE, CoBiDE, SMADE, MDE-pBX, and many others. Finally, hybridizing the proposed algorithms with other powerful metaheuristics, such as CMAES, CCPSO2 are thought to be a promising direction. The Matlab source code of the proposed algorithms can be downloaded from https://sites.google.com/view/optimization-project/files.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.swevo.2018.10.006.

References

[1] M.M. Ali, A. Törn, Population set based global optimization algorithms: some modifications and numerical studies, Comput. Oper. Res. 31 (2004) 1703–1725.
[2] J. Brest, S. Greiner, B. Bošković, M. Mernik, V. žumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (6) (2006) 646–657.
[3] J. Brest, A. Zamuda, I. Fister, M.S. Maučec, Large scale global optimization using self-adaptive differential evolution algorithm, in: Evolutionary Computation (CEC), IEEE Congress, 2010, pp. 1–8.
[4] F. Caraffini, F. Neri, J. Cheng, G. Zhang, L. Picinail, G. Iacca, E. Mininno, Super-fit multicriteria adaptive differential evolution, in: 2013 IEEE Congress on Evolutionary Computation (CEC), IEEE, New York, 2013, pp. 1678–1685.
[5] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood based mutation operator, IEEE Trans. Evol. Comput. 13 (3) (2009) 526–553.
[6] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood based mutation operator, IEEE Trans. Evol. Comput. 13 (3) (2009) 526–553.
[7] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 4–31.
[8] S. Das, S.S. Mullick, P.N. Suganthan, Recent advances in differential evolution-an updated survey, Swarm Evolut. Comput. 27 (2016) 1–30.
[9] A. Draa, S. Bouzoubia, I. Boukhalfa, A sinusoidal differential evolution algorithm for numerical optimization, Appl. Soft Comput. 27 (2015) 99–126.
[10] E.N. Dragoi, S. Curteanu, A.I. Galaction, D. Cascaval, Optimization methodology based on neural networks and self-adaptive differential evolution algorithm applied to an aerobic fermentation process, Appl. Soft Comput. 13 (2013) 222–238.
[11] H.Y. Fan, J. Lampinen, A trigonometric mutation operation to differential evolution, J. Global Optim. 27 (1) (2003) 105–129.
[12] V. Feoktistov, Differential Evolution: in Search of Solutions, Springer-verlag, Berlin, Germany, 2006.
[13] R. Gamperle, S.D. Muller, P. Koumoutsakos, A parameter study for differential evolution, Adv. Intell. Syst. Fuzzy Syst., Evolut. Comput. 10 (2002) 293–298.
[14] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behavior: a case study on the CEC'2005 special session on real parameter optimization, J. Heuristics 15 (2009) 617–644.
[15] A. Ghosh, S. Das, A. Chowdhury, R. Giri, An improved differential evolution algorithm with fitness-based adaptation of the control parameters, Inf. Sci. 181 (18) (2011) 3749–3765.
[16] N. Hachicha, B. Jarboui, P. Siarry, A fuzzy logic control using a differential evolution algorithm aimed at modeling the financial market dynamics, Inf. Sci. 181 (1) (2011) 79–91.
[17] N. Hansen, S.D. Műller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), Evol. Comput. 11 (1) (2003) 1–18.
[18] S. Islam, S. Das, S. Ghosh, S. Roy, P.N. Suganthan, An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization, Syst., Man, Cybern., Part B: Cybern., IEEE Trans. 42 (2) (2012) 482–500.
[19] X. Li, M. Yin, Modified differential evolution with self-adaptive parameters method, J. Combin. Optim. 31 (2) (2014) 546–576.
[20] X. Li, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Trans. Evol. Comput. 16 (2) (2012) 210–224.
[21] J.J. Liang, B.Y. Qin, P.N. Suganthan, A.G. Hernández-Díaz, Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-parameter Optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report 201212, 2013.
[22] R. Mallipeddi, P.N. Suganthan, Q.K. Pan, M.F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2) (2011) 1679–1696.

[23] R. Mallipeddi, S. Jeyadevi, P.N. Suganthan, S. Baskar, Efficient constraint handling for optimal reactive power dispatch problems, Swarm Evolut. Comput. 5 (2012) 28–36.
[24] A.W. Mohamed, An improved differential evolution algorithm with triangular mutation for global numerical optimization, Comput. Ind. Eng. 85 (2015) 359–375.
[25] A.W. Mohamed, H.Z. Sabry, Constrained optimization based on modified differential evolution algorithm, Inf. Sci. 194 (2012) 171–208.
[26] A.W. Mohamed, H.Z. Sabry, M. Khorshid, An alternative differential evolution algorithm for global optimization, J. Adv. Res. 3 (2) (2012) 149–165.
[27] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artif. Intell. Rev. 33 (1–2) (2010) 61–106.
[28] N. Noman, H. Iba, Accelerating differential evolution using an adaptive local search, IEEE Trans. Evol. Comput. 12 (1) (2008) 107–125.
[29] M.N. Omidvar, X. Li, X. Yao, Cooperative co-evolution with delta grouping for large scale non-separable function optimization, in: Evolutionary Computation (CEC), 2010 IEEE Congress, 2010, pp. 1–8.
[30] M.G.H. Omran, A. Salman, A.P. Engelbrecht, Self-adaptive differential evolution, in: Computational Intelligence and Security, Proceedings Lecture Notes in Artificial Intelligence, 2005, pp. 192–199.
[31] Q.K. Pan, L. Wang, L. Gao, W.D. Li, An effective hybrid discrete differential evolution algorithm for the flow shop scheduling with intermediate buffers, Inf. Sci. 181 (3) (2011) 668–685.
[32] Q.K. Pan, P.N. Suganthan, L. Wang, L. Gao, R. Mallipeddi, A differential evolution algorithm with self-adapting strategy and control parameters, Comput. Oper. Res. 38 (2011) 394–408.
[33] S. Paul, S. Das, Simultaneous feature selection and weighting - an evolutionary multi-objective optimization approach, Pattern Recogn. Lett. 65 (2015) 51–59.
[34] K. Price, R. Storn, J. Lampinen, Differential Evolution: a Practical Approach to Global Optimization, Springer, Heidelberg, 2005.
[35] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398–417.
[36] J. Ronkkonen, S. Kukkonen, K. V. Price, Real parameter optimization with differential evolution, in: Proc. IEEE Congr. Evol. Comput. (CEC-2005), vol. 1. Piscataway, NJ, IEEE Press, pp. 506–513.
[37] S. Sarkar, S. Das, S.S. Chaudhuri, A multilevel color image thresholding scheme based on minimum cross entropy and differential evolution, Pattern Recogn. Lett. 54 (2015) 27–35.
[38] R. Storn, K. Price, Differential Evolution – a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, Technical Report TR-95-012, ICSI, 1995, http://.icsi.berkeley.edu/~storn/litera.html.
[39] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[40] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: Proceedings of the IEEE Congresson Evolutionary Computation 2013, June20–23, Cancún,México, 2013, pp. 71–78.
[41] R. Tanabe, A.S. Fukunaga, Improving the search performance of shade using linear population size reduction, in: Proceedings of IEEE Congress on Evolutionary Computation2014, July6–11, Beijing, China, 2014, pp. 1658–1665.
[42] K. Tang, Xiaodong Li, P.N. Suganthan, Z. Yang, T. Weise, Benchmark Functions for the CEC'2010 Special Session and Competition on Large Scale Global Optimization, Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2009.
[43] Y. Wang, H. Li X, T. Huang, L. Li, Differential evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18 (2014) 232–247.
[44] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (1) (2011) 55–66.
[45] Y. Wang, H.-X. Li, T. Huang, L. Li, Differential evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18 (2014) 232–247.
[46] Y. Wang, B.C. Wang, H.X. Li, G.G. Yen, Incorporating objective function information into the feasibility rule for constrained evolutionary optimization, IEEE Trans. Cybern. 46 (12) (2016) 2938–2952.
[47] H. Wang, Z. Wu, S. Rahnamayan, D. Jiang, Sequential DE enhanced by neighborhood search for large scale global optimization, in: Evolutionary Computation (CEC), 2010 IEEE Congress, 2010, pp. 1–7.
[48] G.H. Wu, R. Mallipeddi, P.N. Suganthan, R. Wang, H. Chen, Differential evolution with multi population based ensemble of mutation strategies, Inf. Sci. (2015). https://doi.org/10.1016/j.ins.2015.09.009.
[49] A. Zamuda, J. Brest, Self-adaptive control parameters: randomization frequency and propagations in differential evolution, Swarm Evolut. Comput. 25 (2015) 72–99.
[50] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (5) (2009) 945–958.
[51] X. Zhang, W. Chen, C. Dai, W. Cai, Dynamic multi-group self-adaptive differential evolution algorithm for reactive power optimization, Int. J. Electrical Power 32 (2010) 351–357.
[52] L.M. Zheng, S.X. Zhang, K.S. Tang, S.Y. Zheng, Differential evolution powered by collective information, Inf. Sci. 399 (2017) 13–29.
[53] R.D. Al-Dabbagh, F. Neri, N. Idris, M.S. Baba, Algorithm design issues in adaptive differential evolution: review and taxonomy, Swarm Evolut. Comput. (2018), https://doi.org/10.1016/j.swevo.2018.03.008.
[54] I. Poikolainen, F. Neri, F. Caraffini, Cluster-based population initialization for differential evolution frameworks, Inf. Sci., Elsevier 297 (March 2015) 216–235.
[55] Y. Wang, Z.Z. Liu, J. Li, H.X. Li, J. Wang, On the selection of solutions for mutation in differential evolution, Front. Comput. Sci. 12 (2) (2018) 297–315.
[56] Y. Wang, Z.Z. Liu, J. Li, H.X. Li, G.G. Yen, Utilizing cumulative population distribution information in differential evolution, Appl. Soft Comput. 48 (2016) 329–346.
[57] Z.Z. Liu, Y. Wang, S. Yang, K. Tang, An adaptive framework to tune the coordinate systems in nature-inspired optimization algorithms, IEEE Trans. Cybern. (2018) 1–14, https://doi.org/10.1109/TCYB.2018.2802912.
[58] N.H. Awad, M.Z. Ali, J.J. Liang, B.Y. Qu, P.N. Suganthan, Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained real-parameter numerical optimization, in: Technical Report, Nanyang Technological University Singapore, 2016.
[59] K. Tang, X. Li, P.N. Suganthan, Z. Yang, T. Weise, Benchmark Functions for the CEC 2010 Special Session and Competition on Large-scale Global Optimization, Nature Inspired Computation and Applications Laboratory, University of Science and Technology of China, 2009. Technical Report.
[60] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[61] W. Gong, Z. Cai, Differential evolution with ranking-based mutation operators, IEEE Trans. Cybern. 43 (6) (2013) 2066–2081.
[62] A.W. Mohamed, A.A. Hadi, A.M. Fattouh, K.M. Jambi, LSHADE with semi-parameter adaptation hybrid with CMA-ES for solving CEC 2017 benchmark problems, in: Evolutionary Computation (CEC), 2017 IEEE Congress on, 2017, pp. 145–152.
[63] I. Loshchilov, CMA-ES with restarts for solving CEC 2013 benchmark problems, in: IEEE Congress on Evolutionary Computation, 2013, pp. 369–376.
[64] T. Liao, T. Stutzle, Benchmark results for a simple hybrid algorithm on the CEC 2013 benchmark set for real-parameter optimization, in: IEEE Congress on Evolutionary Computation, 2013, pp. 1938–1944.
[65] N.H. Awad, M.Z. Ali, P.N. Suganthan, Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems, in: Evolutionary Computation (CEC), 2017 IEEE Congress on, 2017, pp. 372–379.
[66] G. Wu, X. Shen, H. Li, H. Chen, A. Lin, P.N. Suganthan, Ensemble of differential evolution variants, Inf. Sci. 423 (2018) 172–186.