Swarm and Evolutionary Computation 80 (2023) 101322

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Differential evolution with objective and dimension knowledge utilization

Sheng Xin Zhang [a,*], Shao Yong Zheng [b], Li Ming Zheng [a]

[a] College of Information Science and Technology, Jinan University, Guangzhou, China
[b] School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China

ARTICLE INFO

Keywords:
Differential evolution (DE)
Global numerical optimization
Objective space knowledge
Dimension space knowledge
Collective dimensional learning
Single dimensional learning

ABSTRACT

Performance of differential evolution, which is one of the most competitive evolutionary algorithms, heavily depends on the utilization of feedback information. The feedback information can be from objective, solution and dimension spaces. To facilitate the better utilization of feedbacks for performance enhancements, this paper proposes an objective-dimension feedback (ODF) method with two novel mechanisms to, respectively take advantages of dimension and objective space knowledge. The first mechanism, named “Small diversity dimensions exploit, Large diversity dimensions explore” classifies dimensions into exploitation and exploration dimensions according to their diversity rankings and assigns them with collective and single dimensional learning strategies, respectively. The second mechanism, named “number of dimensions automatic configuring” automatically configures the number of dimensions performing exploitation and exploration in each solution according to its fitness ranking. Experiments on 29 benchmark functions confirm the effectiveness of ODF by performance comparisons with single utilization of objective and dimension space knowledge, single utilization of dimensional learning strategies and several objective, solution and dimension space knowledge-based methods from literatures.

1. Introduction

The No Free Lunch theorem [1] states that on average, all algorithms perform similarly on all problems. However, in practice, we usually consider one specific or a specific kind of problems. A target real-world problem usually has certain structures for an algorithm to learn. The better an algorithm learns to fit the problem, the better performance it might achieve. The quality of learning depends on the sufficiency of learning sources and the design of learning methods.

As an efficient global optimizer, differential evolution (DE) [2–7] evolves a group of solutions with three typical genetic operations, including differential mutation, dimension-wise crossover and one-to-one selection. In the past two decades, researchers have improved its performance by designing various learning methods for mutation strategies [8–14] as well as control parameters [15–23]. For learning, the source data can be objective space knowledge (OSK), solution space knowledge (SSK) and dimension space knowledge (DSK). OSK describes the information collected based on fitness, such as the fitness comparison of parent and offspring and the fitness ranking of solutions. SSK estimates the distribution of solutions with the consideration of all dimensions as a unity, such as the distance among solutions. While DSK refers to the information derived from the dimension level of solutions, such as the dimension correlation and dimension difference.

In DE, the success/fail of mutation strategy and control parameters can be observed from the parent-offspring one-to-one fitness comparison. This kind of OSK has been widely applied for constructing DE algorithms, such as SaDE (strategy adaptive DE) [9], jDE (Janez’s DE) [15], JADE (Jingqiao and Arthur ’s DE) [8], SHADE (success-history adaptive DE) [16], GLCDE (global-local cooperative DE) [24], CSDE (cooperative strategy-based DE) [25], L-SHADE (SHADE with linear population size reduction) [17] and its improved variants [26–29]. With the population feature, OSK in DE can also be the fitness ranking. Fitness ranking has also been used for improving DE, such as rank-DE (ranking-based mutation for DE) [10], IDE (individual dependent DE) [20], CIPDE (collective information powered DE) [12] and DSNDE (dynamic scale-free network-based DE) [30]. SSK has been used for offspring generation [31] and mutation [32]. DSK has been utilized for improving DE’s crossover by considering the dimension correlation [33] and the enhancement of population diversity by considering the dimension convergence [34]. It is observed that in OSK-based methods, DSK is usually neglected and vice versa. Fully utilizing OSK and DSK may be beneficial to learn problem characteristics and improve the

* Corresponding author.
E-mail address: zhangsx@jnu.edu.cn (S.X. Zhang).

https://doi.org/10.1016/j.swevo.2023.101322
Received 1 October 2022; Received in revised form 15 April 2023; Accepted 25 April 2023
Available online 2 May 2023

Nomenclature

OSK	Objective space knowledge
SSK	Solution space knowledge
DSK	Dimension space knowledge
ODF	Objective-dimension feedback
SEiLEr	Small diversity dimensions exploit, Large diversity dimensions explore
NDAC	Number of dimensions automatic configuring
CDL	Collective dimensional learning
SDL	Single dimensional learning

performance.

This paper proposes an objective-dimension feedback (ODF) method with two novel mechanisms for better utilization of DSK and OSK, respectively. The first mechanism, named "Small diversity dimensions exploit, Large diversity dimensions explore" (SEiLEr) evaluates the evolution differences of dimensions and assigns the diversity rankings. Dimensions with small diversity perform an exploitative search by collectively learning from the corresponding dimensions of multiple fittest solutions while dimensions with large diversity perform an explorative search by single learning from dimension of fittest solutions. The second mechanism, named "number of dimensions automatic configuring" (NDAC) takes advantages of OSK to determine the number of dimensions performing exploitative search to improve convergence or explorative search to alleviate local minima in each solution by utilizing its fitness ranking.

Fig. 1 illustrates the differences in strategy assignment among existing OSK-, DSK-based methods and the ODF method. In the OSK-based method [9,20] (subplot (a)), the same strategies are distributed to all the D dimensions without considering the dimension differences (i.e. DSK is neglected). In the DSK-based method [34] (subplot (b)), an exploitative strategy is first performed for all the dimensions, then some of the dimensions will perform a diversity enhanced explorative strategy when they converge or stagnate. OSK is not considered to make strategy assignment differences among individuals. While in the proposed ODF-based method (subplot (c)), the assignment of strategies depends on both the fitness ranking of individuals and the diversity ranking of dimensions (Details will be given in Section III). Consequently, strategies are distinguishable among both dimensions and individuals and the amount of exploitation and exploration could be adjusted in different dimensions and individuals. Moreover, ODF includes new mechanisms for better utilization of DSK and OSK, as will be introduced in Section III.

The main contributions of this paper can be summarized as follows:

(1) For the utilization of DSK and OSK, SEiLEr and NDAC mechanisms are, respectively designed. ODF thus distributes strategies by simultaneously taking advantage of DSK and OSK, which is significantly different from existing methods.
(2) Collective and single dimensional learning strategies are proposed for the cooperation of exploitation and exploration in a single solution but different dimensions.
(3) Benefits of the proposed SEiLEr and NDAC mechanisms, the dimensional learning strategies and the simultaneous consideration of DSK and OSK are verified by experiments conducted on 29 CEC2017 benchmark functions [35]. The working mechanisms are also investigated and explained.

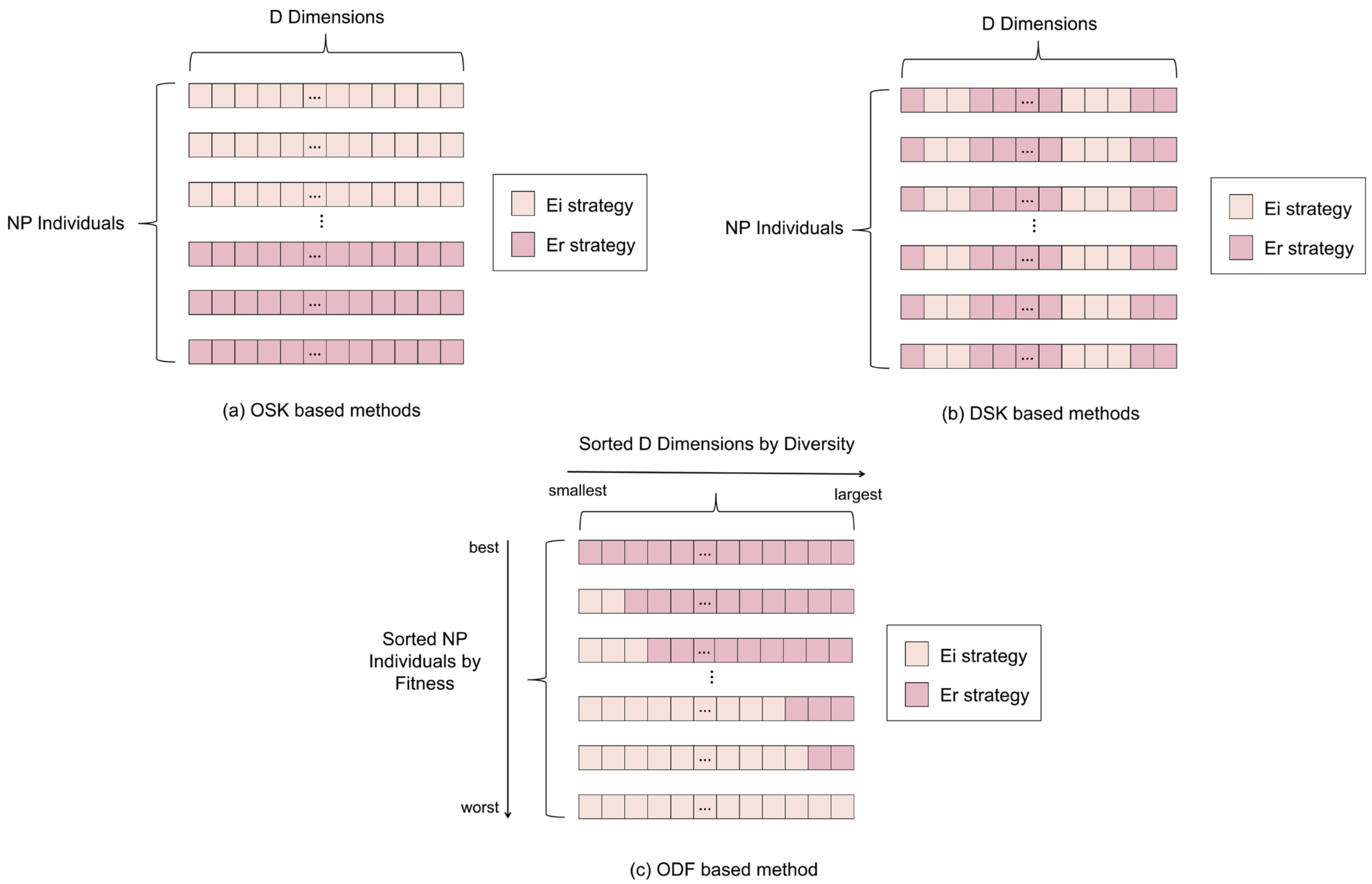

Fig. 1. Illustration of strategy assignment in OSK-, DSK- and ODF-based methods to a population with NP D-dimensional individuals.
Ei: Exploitative, Er: Explorative.

The rest of this paper is organized in order. Section 2 presents a review of related works with the utilization of different kinds of knowledge. Section 3 describes the proposed method in details. The experimental results and discussions are presented in Section 4. Finally, Section 5 concludes this paper.

2. Literature review

In this section, we first describe the classic DE algorithm. Following, related works on multi-strategy DE with utilization of OSK, SSK and DSK are reviewed with a summary presented in Table 1.

2.1. DE

Constructed based on the population concept, DE explores a search space by iteratively performing three genetic operations, i.e. mutation, crossover and selection.

At the beginning, i.e. generation $G = 0$, a population $\mathbf{P}_G = \{\vec{x}_{1,G}, \vec{x}_{2,G}, \cdots, \vec{x}_{NP,G}\}$with NP D-dimensional individuals is randomly sampled from the search space.

Mutation: DE mutates an individual by adding one or more differential vectors to a basic vector. Some widely used mutation strategies include:

DE/rand/1:

$$\vec{v}_{i,G} = \vec{x}_{r_1,G} + F\left(\vec{x}_{r_2,G} - \vec{x}_{r_3,G}\right) \tag{1}$$

DE/best/1:

$$\vec{v}_{i,G} = \vec{x}_{best,G} + F\left(\vec{x}_{r_1,G} - \vec{x}_{r_2,G}\right) \tag{2}$$

DE/current-to-best/1:

$$\vec{v}_{i,G} = \vec{x}_{i,G} + F\left(\vec{x}_{best,G} - \vec{x}_{i,G}\right) + F\left(\vec{x}_{r_1,G} - \vec{x}_{r_2,G}\right) \tag{3}$$

where r_1, r_2 and r_3 are three random integers from$\{1, 2, \cdots, \mathrm{NP}\}$and are mutually different. $\vec{x}_{best,G}$ is the best solution in the current population and F is a scaling factor within (0,1].

Crossover: Following mutation, crossover is performed on each mutant individual $\vec{v}_{i,G}$and its corresponding target individual $\vec{x}_{i,G}$ to generate a trial individual $\vec{u}_{i,G}$. The classic crossover operation is as follows:

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & \textit{if } rand_j(0,1) \le CR \text{ or } j = jrand \\ x_{i,j,G} & \textit{otherwise} \end{cases} \tag{4}$$

where $rand_j$ (0,1) is a uniformly sampled random number within (0,1), *jrand* is a random integer from $\{1, 2, \cdots, D\}$ and *CR* is a crossover factor within [0,1].

Selection: DE selects offspring by comparing the fitness of each pair of trial and target individuals. The fitter one (for minimization problem is the one with smaller fitness value) is selected into the next generation, as follows:

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G} & \textit{if } f\left(\vec{u}_{i,G}\right) \le f\left(\vec{x}_{i,G}\right) \\ \vec{x}_{i,G} & \textit{otherwise} \end{cases} \tag{5}$$

2.2. Multi-strategy DE with utilization of OSK

With the parent-offspring competition feature of DE, OSK can be the success/fail replacement of parent (denoted as SFR). This knowledge has been widely applied to strategy combination [36], strategy selection [37] and strategy adaptation [38–48]. In DEGL (DE with global and local strategies) [36], global and local mutation strategies are combined using a weighting factor. Settings of the weighting factor are adaptive based on SFR. In SaDE (strategy adaptive DE) [9], selection probabilities of mutation strategies are dynamically adjusted based on their success/fail history. In SaM (strategy adaptive mechanism) [38], index of mutation strategies is treated as a control parameter and adaptively adjusted using

Table 1
Literature review of multi-strategy DEs with utilization of OSK, SSK and DSK.

Year	Method	For component(s) of DE	Utilized knowledge	Year	Method	For component(s) of DE	Utilized knowledge
2009	DEGL [36]	mutation strategy	OSK/SFR	2017	IDEI [49]	mutation strategy and control parameter	OSK/FR
2009	SaDE [9]	mutation strategy	OSK/SFR	2018	HHDE [50]	mutation strategy and control parameter	OSK/SFR+FR
2010	SaM [38]	mutation strategy	OSK/SFR	2018	MTDE [54]	offspring generation strategy	OSK/FR
2011	Adap_SS [39]	mutation strategy	OSK/SFR	2018	L-SHADE-RSP [27]	mutation strategy	OSK/FR
2011	EPSDE [43]	mutation strategy and control parameter	OSK/SFR	2019	NDE [53]	mutation strategy	OSK/FR
2011	CoDE [37]	mutation strategy and control parameter	OSK/SFR	2022	DSNDE [30]	mutation strategy	OSK/FR
2013	ISAMODECMA [42]	mutation and crossover strategies	OSK/SFR	2019	MLCC [51]	mutation strategy and control parameter	OSK/SFR+FR
2013	FRRMAB [40]	mutation strategy	OSK/SFR	2008	ODE [31]	offspring generation strategy	SSK
2015	SPS [45]	parent selection	OSK/SFR	2012	N-DE [32]	mutation strategy	SSK
2016	ZEPDE [44]	mutation strategy and control parameter	OSK/SFR	2015	EIG [33]	crossover strategy	DSK
2016	MPEDE [41]	mutation strategy	OSK/SFR	2015	AEPD [34]	offspring generation strategy	DSK
2017	ETI [46]	offspring generation strategy	OSK/SFR+FR	2015	CSM [55]	mutation strategy	OSK+SSK
2017	MVC [47]	mutation strategy and control parameter	OSK/SFR	2016	MPADE [56]	mutation strategy	OSK+SSK
2018	ACoS [48]	crossover strategy	OSK/SFR	2017	UMDE [57]	mutation strategy	OSK+SSK
2021	CSDE [25]	mutation strategy and control parameter	OSK/SFR	2018	UMS [58]	mutation strategy	OSK+SSK
2015	IDE [20]	mutation strategy and control parameter	OSK/FR	2019	TOAs [59]	mutation strategy	OSK+SSK
2016	TS [52]	mutation strategy	OSK/FR	2020	SCSS [60]	mutation strategy and control parameter	OSK+SSK

SFR: Success/fail replacement of parent, FR: Fitness ranking.

parameter control methods in [8,15]. These parameter control methods are commonly based on SFR. In Adap_SS (adaptive strategy) [39], probabilities of mutation strategies are calculated by probability matching and adaptive pursuit techniques. In FRRMAB (fitness-rate-rank-based multiarmed bandit) [40], mutation strategies compete based on multi-armed bandits. In MPEDE (multi-subpopulation ensemble DE) [41], SFR is used to adjust the sizes of subpopulations for mutation strategies. In ISAMODECMA (improved self-adaptive multi-operator DE) [42], SFR is used for allocating mutation and crossover strategies. EPSDE (DE with ensemble of parameter and strategy) [43] uses SFR to distribute mutation strategies and control parameters for each individual. ZEPDE (DE with zoning evolution of parameters) [44] employs SFR to assign appropriate mutation strategies and control the zoning evolution of parameters. CoDE (composite DE) [37] uses fitness information to select the fittest offspring from three candidates generated by three pairs of mutation strategies and parameters. SPS (successful-parent-selecting) [45] utilizes SFR to identify the stagnation of solutions and choose different parent selection methods. With SFR, ETI (event-triggered impulsive) [46] calculates the success rate of population. When success rate decreases to a small value, two kinds of impulses are triggered on superior and inferior individuals, respectively. With SFR, MVC (multi-variant coordination) [47] calculates the contributions of different DE algorithms and adaptively assigns DE at different evolution stages. With SFR, ACoS (adaptive coordinate system) [48] estimates the suitability of two coordinate systems.

With the population structure of DE, OSK can also be the fitness ranking (FR) of solutions. FR has been used to distribute mutation strategies and control parameters [49–54]. In IDE (individual dependent DE) [20], IDEI (improved individual dependent DE) [49], HHDE (historical and heuristic-based DE) [50] and MLCCDE (multi-layer competitive-cooperative DE) [51], individuals are evolved with strategies and parameters determined by their FR. In TSDE (two-step strategy DE) [52] and NDE (adaptive neighborhood DE) [53], strategies are assigned to subpopulations according to FR. In MTDE (multi-topology DE) [54], offspring is generated by individual fitness dependent topology. In DSNDE (dynamic scale-free network-based DE) [30], FR is used to construct a scale-free network, which determines the selection of solutions for mutation.

2.3. Multi-strategy DE with utilization of SSK or DSK

SSK originates from the distribution of NP D-dimensional solutions, which usually treats the D dimensions as a whole. Difference among dimensions is not considered. In ODE (opposition-based DE) [31], a new offspring generation method, named opposition-based learning (OBL) was proposed for population initialization and generation jumping. OBL generates opposite solutions by using the current solutions and the bounds of search space. In N-DE (neighborhood mutation DE) [32], mutation is performed within each Euclidean neighborhood, which is constructed based on distances among solutions.

While DSK takes the correlation or difference among dimensions into consideration. In EIG (eigenvector-based crossover operator) [33], covariance matrix is computed by using the dimensional information of population. In AEPD (auto-enhanced population diversity) [34], distribution of each dimension is calculated at each generation. Dimensions will be diversified when they converge or stagnate.

2.4. Multi-strategy DE with utilization of OSK and SSK

In CSM (cheap surrogate model-based multi-operator search) [55], density is computed by using distances among individuals. In order to make fitter solutions contribute more, fitness ranking information is also used for the calculation of density. In MPADE (multiple sub-populations adaptive DE) [56], solutions for mutation are selected with the consideration of their solution space distances. Besides, different mutation strategies are assigned to sub-populations with different fitness values. In UMDE (underestimation assisted multistage DE) [57], abstract convex underestimation (ACUM) which is used to estimate the evolution stage, is constructed based on fitness and the distribution of solutions in solution space. In UMS (underestimation-based multimutation strategy) [58], cheap ACUM is employed to filter an offspring from multiple candidates. In TOAs (team of optimization algorithms) [59], solution quality is measured by combining fitness difference with solution space distance to the optimal solution. In SCSS (selective candidate with similarity selection rule) [60], multiple candidates are sampled for each current solution with different strategies and control parameters. The final candidate is selected based on the fitness ranking of the current solution and its solution space distance to the corresponding candidates.

To summarize, the above methods are tabulated in Table 1.

3. Proposed method

3.1. Motivation

DE, as a classic evolutionary algorithm, evolves a group of NP candidate solutions. Each solution could prefer a certain evolution direction, i.e. to exploit or to explore based on its current status. Further, for each solution, exploitation and exploration needs may be different for the D dimensions. Therefore, it is necessary to detect the characteristic of dimensions, classify and assign them with appropriate exploitation and exploration search strategies, which is the focus of this paper.

For black-box optimization, an appropriate assignment of strategies requires sufficient utilization of the feedbacks from both objective and decision spaces. On the one hand, strategies have been assigned to solutions at the individual level [20, 49–54]. However, since the optimization of D dimensions may be asynchronous, a single strategy may not be suitable for all the D dimensions. A typical class of such kind of functions is the hybrid function [35], in which variables are randomly divided into several groups with each group associated with a different basic function. Since the basic functions have different degree of difficulties and thus some dimensions of the hybrid function are relatively easy to optimize while the others are relatively difficult. Therefore, to measure differences at the dimension level and allocate appropriate exploitation and exploration strategies for various dimensions could improve the performance. On the other hand, at the population level, the amount of exploitation and exploration needs in solutions might vary according to their status (e.g. fitness) in the population. Thus, the number of dimensions performing exploitation and exploration in each solution should be configured.

With the above considerations, this paper proposes the objective-dimension feedback (ODF) method with two novel mechanisms, namely SEiLEr and NDAC which simultaneously merges DSK with OSK. Further, to meet different search requirements in different dimensions, exploitative collective dimensional learning strategy which merges the dimension information of multiple promising solutions and explorative single dimensional learning strategy which learns from single dimension are proposed to implement a dimension level search strategy adaptation. They are described in the following subsections step by step.

3.2. Utilization of DSK in ODF: SEiLEr mechanism

In real-world optimization problems, different subcomponents of variables may have different properties [35]. To detect their properties and better assign exploitative or explorative search strategies, the **SEiLEr** (“Small diversity dimensions exploit, Large diversity dimensions explore”) mechanism is proposed. The working flows are as follows: From the beginning, diversity of each dimension j at each generation G is calculated as

$$d_{j,G} = \frac{1}{\mathrm{NP}}\sum_{i=1}^{\mathrm{NP}}\left(x_{i,j,G} - \overline{x}_{j,G}\right)^{2} \tag{6}$$

where $\overline{x}_{j,G} = (1/\mathrm{NP})\sum_{i=1}^{\mathrm{NP}} x_{i,j,G}$ is the center of the j-th dimension. Afterwards, $d_{j,G}$ $\{j = 1, 2,\ldots, \mathrm{D}\}$ is sorted in ascending order and assigned a diversity ranking value $Rd_{j,\ G}$ from$\{1, 2, \cdots, \mathrm{D}\}$, as shown in Eq. (7),

$$Rd_{j,G} \leftarrow sort(d_{j,G}) \tag{7}$$

Consequently, $Rd_{j,\ G} = 1$ is for the dimension with the smallest diversity while $Rd_{j,\ G} = \mathrm{D}$ is for the one with the largest diversity, as illustrated in Fig. 2.

In principle, dimensions with Small $Rd_{j,\ G}$ values will perform an exploitative dimensional search while those with Large $Rd_{j,\ G}$ values will perform an explorative dimensional search. It is based on the following considerations. Small diversity dimensions converge faster, implying that they may be relatively easy to optimize. For hybrid functions, these dimensions may come from the relatively simple functions, thus, they are consistently assigned exploitation task to benefit convergence. While for dimensions with large diversity, the large diversity indicates that they are relatively hard to converge, therefore they are assigned exploration task to help maintain population diversity. Consequently, SEiLEr mechanism classifies dimensions into exploitation and exploration dimensions.

For demonstration, an illustrative experiment was conducted on the SHADE [16] algorithm with the SEiLEr mechanism on ten 30-D CEC2017 hybrid functions F11-F20 [35]. Each hybrid function consists of several groups of variables with each group associated with one basic function. The number of groups and the number of variables in each group are shown in Table 2 while the basic function in each group is shown in Table 3. In our methodology, $Rd_{j,\ G}$ of each dimension j at each generation was recorded and finally the average ranking value aRd_j of each dimension j over the entire evolution process was obtained. According to SEiLEr (i.e. small ranking value dimensions assigned an exploitative task while large ranking value dimensions assigned an explorative task) and assuming that the exploitative and explorative percentages are both 50%, i.e. half of the dimensions with ranking values smaller than D/2 performs an exploitative task while the rest performs an explorative task. Then, statistically, dimension j with $aRd_j < \mathrm{D}/2$ is mainly assigned an exploitative task while dimension j with $aRd_j > \mathrm{D}/2$ is mainly assigned an explorative task. Therefore, in each group, we know which dimensions perform an exploitative task and which dimensions perform an explorative task. The exploitative task ratio R_{Ei} and explorative task ratio R_{Er} within each group can be, respectively calculated as the number of dimensions performing exploitative and explorative tasks divided by the total number of dimensions within the group. The larger value L of R_{Ei} and R_{Er} is defined as the homogeneity in the group. L value and the corresponding task are

Table 2
The ten 30-D cec2017 hybrid functions F11-F20 [35].

Fun.	Number of groups	Number of variables in each group	Fun.	Number of groups	Number of variables in each group
F11	3	(6, 12, 12)	F16	4	(6, 6, 9, 9)
F12	3	(9, 9, 12)	F17	5	(3, 6, 6, 6, 9)
F13	3	(9, 9, 12)	F18	5	(6, 6, 6, 6, 6)
F14	4	(6, 6, 6, 12)	F19	5	(6, 6, 6, 6, 6)
F15	4	(6, 6, 9, 9)	F20	6	(3, 3, 6, 6, 6, 6)

Table 3
Basic function in each group of the hybrid functions [35].

Fun.	Group index					
	1	2	3	4	5	6
F11	f2	f3	f4	NA	NA	NA
F12	f8	f7	f1	NA	NA	NA
F13	f1	f3	f6	NA	NA	NA
F14	f8	f10	f16	f4	NA	NA
F15	f1	f14	f4	f3	NA	NA
F16	f5	f14	f3	f7	NA	NA
F17	f12	f10	f15	f7	f4	NA
F18	f8	f10	f4	f14	f9	NA
F19	f1	f4	f15	f11	f5	NA
F20	f13	f12	f10	f4	f7	f16

Unimodal basic functions f1: Bent Cigar; f2: Zakharov; f8: High Conditioned Elliptic; f9: Discus; **Multimodal basic functions** f3: Rosenbrock; f4: Rastrigin; f5: Expanded Schaffer's F6; f6: Lunacek bi-Rastrigin; f7: Modified Schwefel; f10: Ackley; f11: Weierstrass; f12: Katsuura; f13: HappyCat; f14: HGBat; f15: Expanded Griewank plus Rosenbrock; f16: Schaffer's F7; NA: Not available.

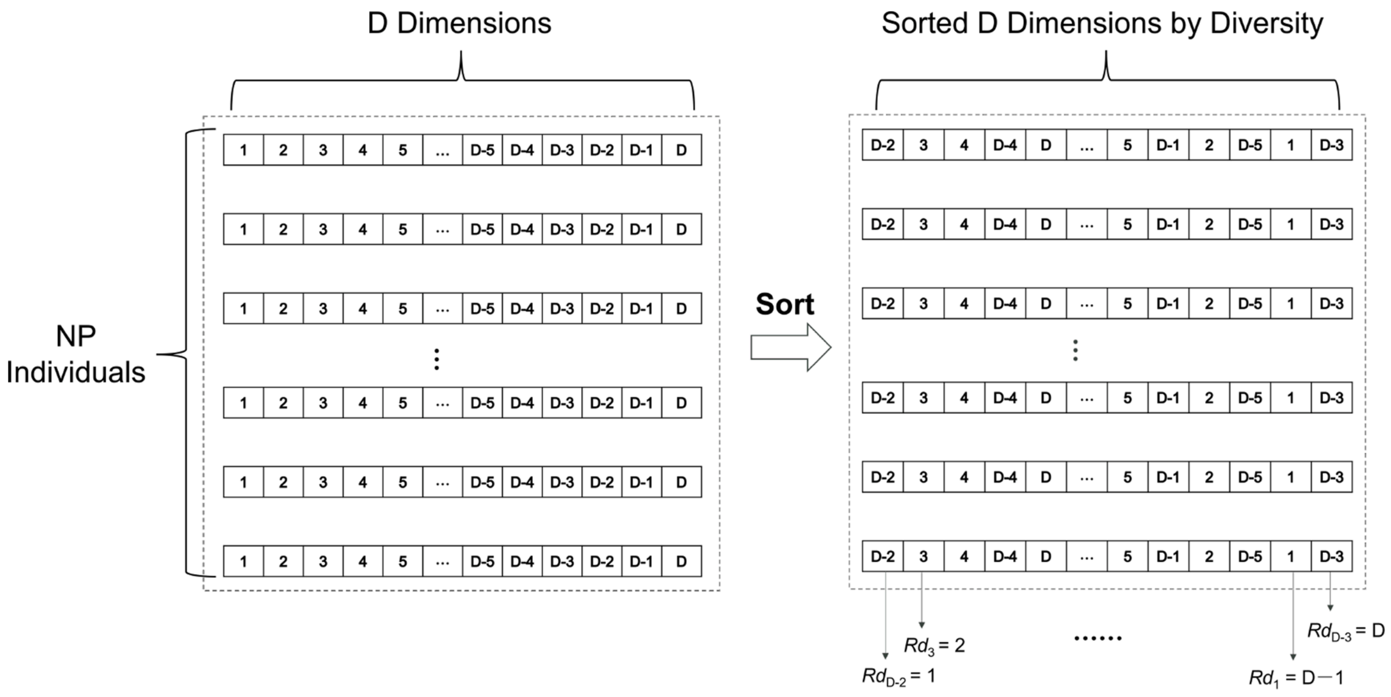
D Dimensions
Sorted D Dimensions by Diversity
NP Individuals
Sort
1 2 3 4 5 ... D-5 D-4 D-3 D-2 D-1 D
D-2 3 4 D-4 D ... 5 D-1 2 D-5 1 D-3
$Rd_3 = 2$
$Rd_{D-2} = 1$
$Rd_{D-3} = D$
$Rd_1 = D-1$

Fig. 2. An example of assigning ranking to dimension according to its relative diversity.

Table 4
Homogeneity and corresponding task in each group with SEiLEr method.

Fun.	Group index					
	1	2	3	4	5	6
F11	100% (Ei)	100% (Er)	75% (Ei)	NA	NA	NA
F12	56% (Ei)	100% (Er)	83% (Ei)	NA	NA	NA
F13	100% (Ei)	56% (Er)	83% (Er)	NA	NA	NA
F14	100% (Ei)	67% (Er)	100% (Er)	58% (Ei)	NA	NA
F15	100% (Ei)	100% (Ei)	89% (Er)	89% (Er)	NA	NA
F16	100% (Er)	83% (Ei)	100% (Ei)	89% (Er)	NA	NA
F17	100% (Er)	100% (Er)	100% (Ei)	100% (Er)	67% (Ei)	NA
F18	100% (Ei)	83% (Er)	100% (Er)	67% (Er)	100% (Ei)	NA
F19	100% (Ei)	83% (Er)	100% (Ei)	67% (Ei)	100% (Er)	NA
F20	100% (Ei)	67% (Ei)	100% (Er)	67% (Ei)	100% (Er)	100% (Er)

Ei: Exploitative, Er: Explorative, NA: Not available.

reported in Table 4.

With respect to the L value, it can be seen from Table 4 that in most (more than half) of the groups, L reaches 100%. We know that if exploitative and explorative tasks are randomly assigned to the dimensions, then, take a group with 12 dimensions (F11, Group 2) for example, L value is likely to be near 50%. The probability to be 100% is $(1+1)/2^{12} = 1/2048$, which is quite small. However, with SEiLEr, from Table 4, L is 100%. Thus, it is concluded that SEiLEr significantly increases the homogeneity of exploitative and explorative tasks within each group.

With respect to the type of functions, from Tables 3 and 4, it is observed that for the group with relatively simple basic function, including Bent Cigar (f1), Zakharov (f2), High Conditioned Elliptic (f8) and Discus (f9) functions with unimodal feature, they are consistently assigned with mainly the exploitation task (highlighted with gray background in Table 4), indicating that SEiLEr correctly detects the dimension difficulty. The optimization performance of SEiLEr on hybrid functions will be further demonstrated by comparative experiments in Section 4.3.

The pseudo-code of **SEiLEr** is shown in Algorithm 1. The parameter NoD (Number of Dimensions) in line 4 is calculated according to OSK, as will be introduced in Section 3.3. The exploitative collective dimensional learning (line 5) and explorative single dimensional learning (line 7) will be described in Section 3.4.

From the above descriptions, it is seen that SEiLEr is significantly different from AEPD [34], as summarized in the following three aspects:

(1) The working mechanism and trigger condition are different. AEPD triggers exploration when dimensions converge or stagnate. While SEiLEr performs exploitation on small diversity dimensions and exploration on large diversity dimensions and works from the beginning of the optimization. AEPD acts in a passive way while SEiLEr is an active way.
(2) AEPD measures the diversity of dimensions in isolation with the relative diversity among dimensions being neglected. Specifically, a threshold parameter was introduced in AEPD to decide whether a dimension converges/stagnates or not. While SEiLEr evaluates the relative diversity among dimensions by the sorting and ranking operations.
(3) SEiLEr takes advantages of OSK to adjust the amount of exploitation and exploration in dimension while AEPD does not as AEPD merely utilizes the convergence information of the dimension value.

Algorithm 1
SEiLEr.

```
1:  Sort dimensions according to diversity and store the ranking values in $Rd_{j,\ G}$ {$j$ = 1, 2, …, D};
2:  For $i$ = 1: NP
3:    For $j$ = 1: D
4:      If $Rd_{i,j,G}$ < NoD    // NoD: Number of Dimensions
5:        Perform exploitative collective dimensional learning (CDL) on dimension $j$;
6:      Else
7:        Perform explorative single dimensional learning (SDL) on dimension $j$;
8:      End If
9:    End For
10: End For
```

3.3. Utilization of OSK in ODF: NDAC mechanism

The proposed number of dimensions automatic configuring (NDAC) mechanism works as follows: From the beginning, solutions in the population $P_{i,\ G}$ $\{i = 1, 2, \ldots, \text{NP}\}$ are sorted from the best to the worst and assigned fitness ranking values $Rf_{i,\ G}$ from 1 to NP. $Rf_{i,\ G}$ is used to determine the number of dimensions performing CDL/SDL, i.e. the parameter NoD in Algorithm 1 (line 4), as calculated in the followings:

$$NoD = min\left(floor\left(Rf_{i,G} \times D^{\alpha} / NP\right), D\right) \tag{8}$$

$$\alpha = 1 + 2 \times FES/Max\_FES \tag{9}$$

where floor $(\cdot)$ denotes a floor value, min (a, b) is a smaller value of a and b, FES is the currently consumed function evaluations while Max_FES is the maximum number of evaluations. From these equations, the following principles can be observed:

(1) The setting of NoD is associated with the dimensionality D and population size NP to distribute the D dimensions to the NP individuals.
(2) Inferior solutions with large $Rf_{i,\ G}$ values are assigned more dimensions encouraging exploitation to help them jump out from inferior dimension values.
(3) With α, at the early evolution stage, NoD is relatively small to encourage exploration, but as the evolution goes on, more and more dimensions will perform an exploitative strategy. Fig. 3 shows NoD against Rf_i with different FES values when NP=100 and $D = 30$. It can be seen that at the beginning (blue line), NoD

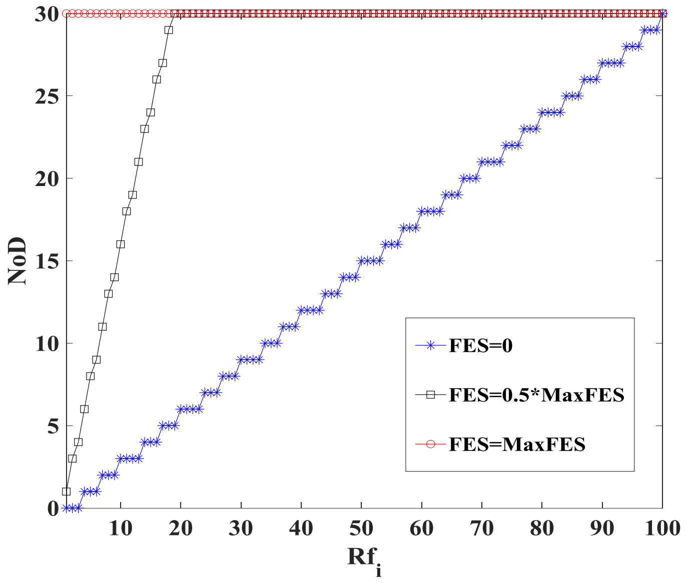

Fig. 3. NoD against Rf_i with different FES values.

Algorithm 2
ODF.

```
1:  Sort the population according to fitness and store the ranking values in $Rf_{i,G}$ {i = 1, 2, …, NP};
2:  Sort dimensions according to diversity and store the ranking values in $Rd_{j,G}$ {j = 1, 2, …, D};
3:  For i = 1: NP
4:    NoD = min (floor ($Rf_{i,G} \times D^{\alpha}/NP$), D);
5:      For j = 1: D
6:        If $Rd_{i,j,G}$ < NoD
7:          Perform exploitative collective dimensional learning (CDL), i.e. Eqs. (10), (12) and (13) as will be introduced in Section 3.4 on dimension j;
8:        Else
9:          Perform explorative single dimensional learning (SDL) i.e. Eqs. (14) and (15) as will be introduced in Section 3.4 on dimension j;
10:       End If
11:     End For
12: End For
```

increases with Rf_i while at the final generation (red line), NoD is D (i.e. 30) for all the solutions.

Combining SEiLEr with NDAC, pseudo-code of the proposed ODF method is shown in Algorithm 2. ODF is further illustrated in Fig. 4. From Figs. 3 and 4, it is seen that the choice of an exploitative/explorative strategy for each dimension is cooperatively determined by the dimension diversity ranking, the fitness ranking and the current evolution stage. The individual contribution of DSK and OSK will be identified by experiments in Section 4.2.

3.4. Collective and single dimensional strategies

The collective and single dimensional learning strategies included in Algorithm 2 (lines 7 and 9) are described as follows.

Collective dimensional learning (CDL): CDL consists of collective mutation and collective crossover. The population $\boldsymbol{P}_G$ is first sorted according to fitness from best to worst. Thus, $\vec{x}_{1,G}$ and $\vec{x}_{NP,G}$ are the best and worst solutions, respectively. Assuming that o is the dimension to perform CDL:

Collective mutation:

$$v_{i,o,G} = x_{i,o,G} + F\cdot\left(x_{ci\_mbest^i,o,G} - x_{i,o,G}\right) + F\cdot\left(x_{r1,o,G} - \overline{x}_{r2,o,G}\right) \tag{10}$$

where $x_{r1,o,G}$ is randomly selected from $\boldsymbol{P}_G$ and $\overline{x}_{r2,o,G}$ is randomly selected from the union population of $\boldsymbol{P}_G$ and recently replaced NP solutions.$x_{ci_mbest^i,o,G}$is the weighted average of the top-m fittest solutions for the o-th dimension, as calculated in Eq. (11)

$$x_{ci\_mbest^i,o,G} = \sum_{k=1}^{m} w_k \times x_{k,o,G} \tag{11}$$

where m is a random integer from$\{1, 2, \cdots, i\}$ andw_k is a normalized value as $w_k = \frac{(m-k+1)}{(1+2+\cdots+m)}$, for$k = 1, 2, \ldots, m$. The smaller the k value, the fitter the solution, and consequently the larger w_k. Therefore, the dimension of superior solutions has more contribution to $x_{ci\_mbest^i,o,G}$. The collective mutation is similar to the classic "current-to-best" mutation with the o-th dimension of the best vector replaced by $x_{ci\_mbest^i,o,G}$.

Collective crossover:

For the collective crossover, we employ UN_UP (i) as the unsuccessful update counter for each individual i. At the beginning, $UN\_UP(i)$ is 0. Then, in the selection of DE, if offspring $\vec{u}_{i,G}$ is better than parent $\vec{x}_{i,G}$, UN_UP (i) is set to 0, otherwise, UN_UP (i) increases by 1. From the collective crossover operation shown in Eqs. (12) and (13), it is seen that if UN_UP (i) is smaller than the threshold value T (T = 90 [12]), then classic crossover (i.e. Eq. (12)) is used. Otherwise, collective crossover (i.e. Eq. (13)) is performed. The difference between collective and classic crossover is that in collective crossover, collective vector $x_{ci\_mbest^i,o,G}$ is used to help stagnated solutions escape from stagnation.

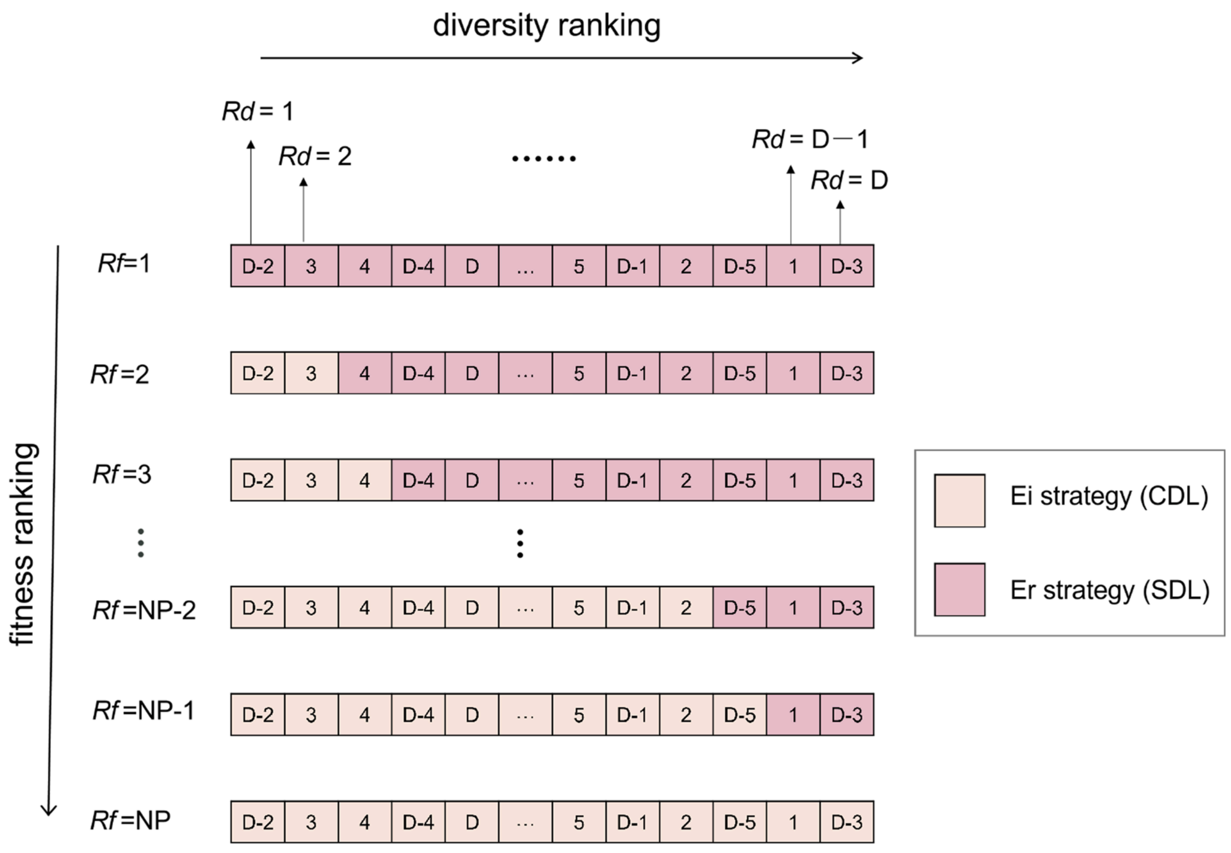

Fig. 4. Illustration of the ODF method.

If $UN\_UP(i) \leq T$

$$u_{i,o,G} = \begin{cases} v_{i,o,G} & \text{if } rand_o(0,1) \leq CR \text{ or } o = o_{rand} \\ x_{i,o,G} & \text{otherwise} \end{cases} \quad (12)$$

Else

$$u_{i,o,G} = \begin{cases} v_{i,o,G} & \text{if } rand_o(0,1) \leq CR \text{ or } o = o_{rand} \\ x_{ci\_mbest^i,o,G} & \text{otherwise} \end{cases} \quad (13)$$

End If

Single dimensional learning (SDL): SDL consists of single mutation and single crossover. Assuming that q is the dimension to perform SDL:

Single mutation:

$$v_{i,q,G} = x_{i,q,G} + F\cdot\left(x_{pbest,q,G} - x_{i,q,G}\right) + F\cdot\left(x_{r1,q,G} - \overline{x}_{r2,q,G}\right) \quad (14)$$

where $x_{r1,q,G}$is randomly selected from $\boldsymbol{P}_G$ and $\overline{x}_{r2,q,G}$ is randomly selected from the union population of $\boldsymbol{P}_G$ and recently replaced NP solutions. $x_{pbest,q,G}$is the q-th dimension of one of the top-p fittest solutions.

Single crossover:

$$u_{i,q,G} = \begin{cases} v_{i,q,G} & \text{if } rand_q(0,1) \leq CR \text{ or } q = q_{rand} \\ x_{i,q,G} & \text{otherwise} \end{cases} \quad (15)$$

Although the principles of these mutation and crossover operations inherit from CIPDE [12] and JADE [8], a significant difference is that in this proposal, these operations operate at the same individual but different dimensions. Fig. 5 illustrates how CDL and SDL operations at an individual. For CDL, dimensions, such as “D−2″ and “3″ learn from collective dimensions of top-fittest solutions. While for SDL, dimensions, such as “D−3″ and “1″ learn from single dimension of top-fittest individual. In this way, CDL and SDL cooperate at an individual to generate a new offspring. Since CDL collectively utilizes the multiple dimensions of promising solutions, it accelerates the information flow among individuals and exhibits a more exploitative characteristic when compared with SDL. Moreover, since CDL and SDL are implemented at the dimension level, it facilitates the adaptation and cooperation of exploitative and explorative search strategies for dimensions. Exploitation and exploration capabilities of CDL and SDL as well as the benefit of the cooperative mechanism will be verified by experiments in Section 4.1.

3.5. *Time complexity of the proposed mechanisms*

The time complexity of diversity calculation and diversity ranking in the SEiLEr mechanism is $O(\mathrm{NP} \times \mathrm{D})$ and $O(D \times \log_2 \mathrm{D})$, respectively at each generation. The time complexity of fitness ranking in the NDAC mechanism is $O(\mathrm{NP} \times \log_2 \mathrm{NP})$ at each generation. Thus, the overhead of the proposed mechanisms at each generation is $O(\mathrm{NP} \times D + D \times \log_2 \mathrm{D} + \mathrm{NP} \times \log_2 \mathrm{NP})$.

4. Experimental results

In this section, experiments are performed to verify the effectiveness of the proposed ODF method. The CEC2017 [35] test suite, which consists of 30 functions, is considered. Note that function F2 has been removed [35] because of its instability, so there are 29 functions tested. Performance of algorithm is measured using error value, defined as $f(x)$ - $f(x^*)$, where x is the best solution obtained with the maximum function evaluations of 10,000 × D while x^* is the optimal solution of the function. For each function, 51 trials [35] are performed and the Wilcoxon’s signed rank test [61] with a significance level of 5% is used to compare the performance of two algorithms. When the considered algorithm is significantly better than (i.e. win, W), similar to (i.e. tie, T) or worse than (i.e. lose, L) the compared algorithm, we mark it using “+”, “=” and “-”, respectively. For the setting of control parameters F and CR, parameter control method in SHADE [16] is adopted. Besides, the population size NP is set as 100. Pseudo-code of ODF-based DE, i.e. ODFDE is shown in Algorithm 3.

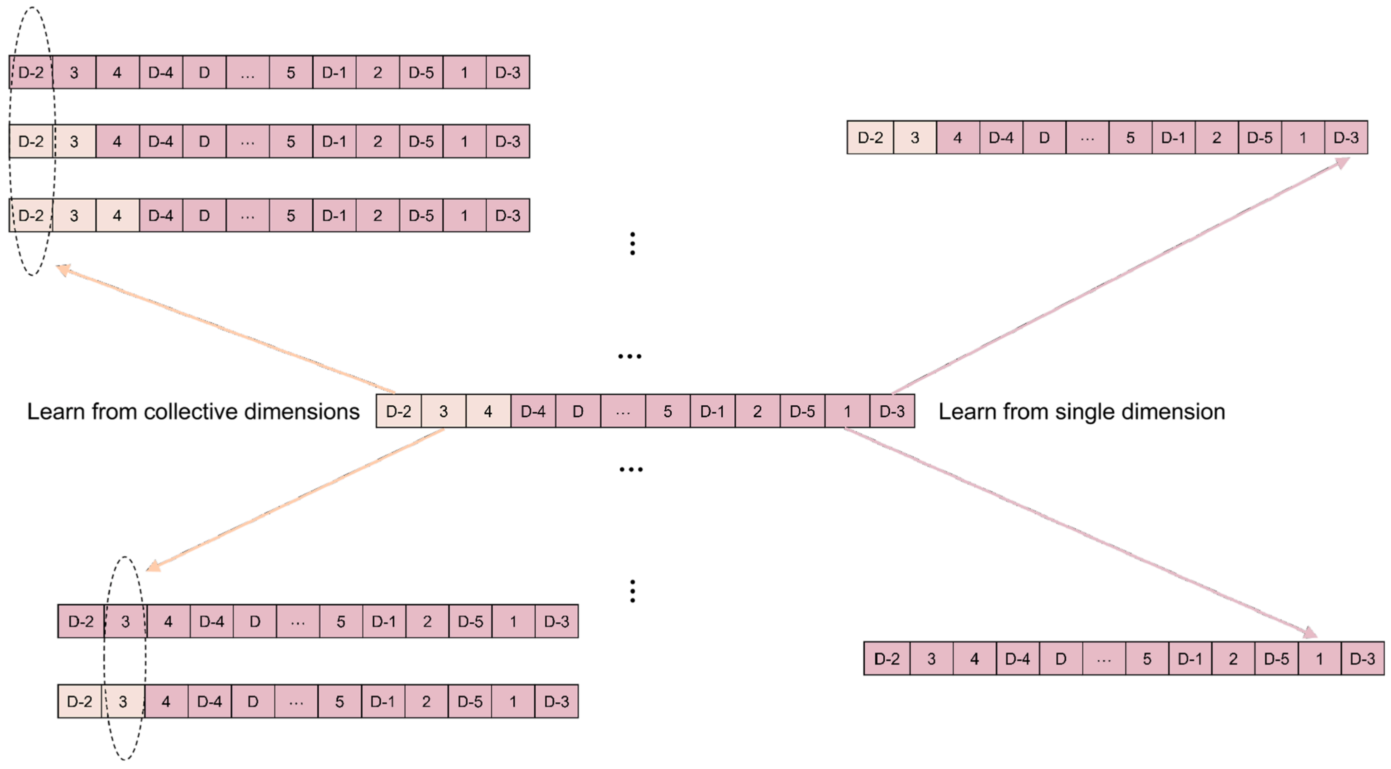

Fig. 5. An example that CDL and SDL cooperate at an individual.

Algorithm 3
ODFDE.

```
1:  Initialize a population $P_0 = \{\vec{x}_{i,0}, i \in \{1,2,\cdots NP\}\}$;
2:  Set memory $M_F = 0.5$, $M_{CR} = 0.5$, history length $H = D$, initialize history index
    $k = 1$, initialize external archive $A = \varnothing$, set generation count $G = 0$;
3:  While the stopping criteria are not satisfied, Do
4:    Sort the population according to fitness and store the ranking values in $Rf_{i,G}$
      $\{i = 1, 2, \ldots, NP\}$;
5:    Sort dimensions according to diversity and store the ranking values in $Rd_{j,G}$
      $\{j = 1, 2, \ldots, D\}$;
6:    Set $S_F = \varnothing$, $S_{CR} = \varnothing$;
7:    For $i = 1$: NP
8:      NoD = min (floor ($Rf_{i,G} \times D^{\alpha}/NP$), D);
9:      Generate the scaling factor and crossover factor for SDL and CDL,
        respectively, i.e. $r_i = randint[1,H]$, $F_{i,G} = randc_i(M_{F,r_i}, 0.1)$, $CR_{i,G} =$
        $randn_i(M_{CR,r_i}, 0.1)$, where $randc(a,b)$ and $randn(a,b)$ are Cauchy distribution
        and normal distribution with location parameter $a$ and scale parameter $b$,
        respectively.
10:     For $j = 1$: D
11:       If $Rd_{i,j,G} <$ NoD
12:         Perform exploitative collective dimensional learning (CDL), i.e. Eqs.
            (10), (12) and (13) to generate trial vector $u_{i,j,G}$;
13:       Else
14:         Perform explorative single dimensional learning (SDL) i.e. Eqs. (14) and
            (15) to generate trial vector $u_{i,j,G}$;
15:       End If
16:     End For
17:     If $f(\vec{u}_{i,G}) \le f(\vec{x}_{i,G})$
18:       $\vec{x}_{i,G+1} = \vec{u}_{i,G}$, $A \leftarrow \vec{x}_{i,G}$, $S_F \leftarrow F_{i,G}$, $S_{CR} \leftarrow CR_{i,G}$;
19:     Else
20:       $\vec{x}_{i,G+1} = \vec{x}_{i,G}$;
21:     End If
22:   End For
23:   If $|A| >$ NP
24:     Randomly remove $|A| -$ NP individuals from $A$;
25:   End If
26:   Update $M_F$ and $M_{CR}$ based on $S_F$ and $S_{CR}$, respectively according to
      Procedure 1;
27:   $G = G + 1$;
28: End While
```

Procedure 1
Update M_F and M_{CR}

```
1:  $M_{F,k,G+1} = \begin{cases} mean_{W_L}(S_F) & \text{if } S_F \neq \varnothing \\ M_{F,k,G} & \text{otherwise} \end{cases}$
    $M_{CR,k,G+1} = \begin{cases} mean_{W_A}(S_{CR}) & \text{if } S_{CR} \neq \varnothing \\ M_{CR,k,G} & \text{otherwise} \end{cases}$
    where $mean_{W_A}(S_{CR}) = \sum_{m=1}^{|S_{CR}|} w_m \cdot S_{CR,m}$, $mean_{W_L}(S_F) = \frac{\sum_{m=1}^{|S_F|} w_m \cdot S_{F,m}^2}{\sum_{m=1}^{|S_F|} w_m \cdot S_{F,m}}$, $w_m =$
    $\frac{\Delta f_m}{\sum_{m=1}^{|S_{CR}|} \Delta f_m}$, where $\Delta f_m = \left| f(\vec{u}_{m,G}) - f(\vec{x}_{m,G}) \right|$
2:  If $S_F \neq \varnothing$ and $S_{CR} \neq \varnothing$
3:    $k = k + 1$;
4:    If $k > H$
5:      $k = 1$;
6:    End If
7:  End If
```

4.1. Comparison with single SDL and CDL utilization

In the ODF method, SDL and CDL are cooperatively used for offspring generation. To show the benefit, it is compared with single methods, in which SDL and CDL are, respectively adopted for all the dimensions at all time. Table 5 shows the comparison results on 30-D and 50-D functions. From Table 5, ODF performs significantly better than both single methods. In the total of 116 cases, it outperforms in 78 and underperforms in 6 cases. More specifically, it performs better than SDL and CDL in 42 (=25+17), 36 (=16+20) cases and loses in 2 (=1 + 1), 4 (=2 + 2) cases, respectively. Compared with the baselines, ODFDE mainly loses on the unimodal function F3, which indicates that the proposed dimension-level strategy allocation is not suitable for solving this problem.

CDL exhibits a general more exploitative characteristic than SDL on all the twenty-nine functions. As an example, Fig. 6 plots the achieved final diversity of dimensions by SDL and CDL on 30-D simple multi-modal function F5 and hybrid function F15. It is observed that on most of the 30 dimensions, CDL achieves smaller diversity than SDL. Fig. 7 shows the advantages of CDL and SDL on the twenty-nine 30-D CEC2017 functions. As seen, CDL is better than SDL on 8 functions (i.e. F4, F5, F7, F8, F10, F20, F21, F29) while SDL is better than CDL on other 8 functions (i.e. F11, F13-F15, F18, F19, F25, F30). Nevertheless, according to Table 5, the advantage of ODF on these 16 functions is clear with the observation that ODFDE performs better than SDL and CDL in 16 and 9 cases and worse in 0 and 1 case, respectively.

Fig. 8 shows the convergence plots of SDL, CDL and ODF over generations on 30-D simple multi-modal function F5 and hybrid function F15. It is seen that CDL has advantages over SDL on function F5 while SDL is more suitable for solving function F15. However, ODF achieves better results on both functions, meaning that ODF could coordinate both methods for a better performance.

For the total 58 cases, Table 6 shows the comparison results according to Holm, Hochberg and Hommel procedures [62]. As shown, ODF is statistically better when compared with both SDL and CDL.

The effectiveness of ODF is also investigated with fixed F and CR settings of $F = 0.5$ and $CR = 0.5$. As shown in Table S1 in the supplementary file, ODF still outperforms SDL and CDL with the "win/lose" metrics of "19/6" and "15/1" in the 30-D case and "20/6" and "19/1" in the 50-D case.

4.2. Comparison with single OSK and DSK utilization

ODF method is characterized by the simultaneous utilization of DSK and OSK. Herein, we verify the effectiveness by comparing with the following two variants:

OSK_only: In this variant, only OSK but no DSK is utilized. Specifically, using OSK, the inferior solutions are assigned CDL while the superior solutions are assigned SDL. Without DSK, CDL and SDL are not assigned at the dimension level.

DSK_only: In this variant, only DSK but no OSK is utilized. Specifically, utilizing DSK, SEiLEr (Algorithm 1) is preserved but without OSK, the number of dimension NoD for each solution is set as a uniformly distributed random integer within [1, D]. According to our preliminary experiment, this random setting performs better than the other eleven fixed NoD settings, including 0.1 × D to 0.9 × D with a step of 0.1 × D, 0.95 × D and 0.99 × D, as shown in Table 7.

Table 8 presents the comparison and summarizes the "win/tie/lose" results. It can be observed that ODF shows advantages over OSK_only and DSK_only with the win count much larger than the lose count i.e. 25 (=16+9) against 3 (2 + 1). This result indicates that appropriately merging the knowledge from both objective and dimension spaces could improve the performance.

4.3. Effectiveness of SEiLEr

In the utilization of DSK, SEiLEr mechanism encourages small diversity dimensions to exploit while large diversity dimensions to explore. Section 3.2 has illustrated the reasons. To further confirm the

Table 5
Performance comparisons of ODFDE with SDL and CDL on 30-D and 50-D CEC2017 benchmark set over 51 independent runs.

	30-D								50-D							
	SDL			CDL			ODFDE		SDL			CDL			ODFDE	
	mean	std	sig	mean	std	sig	mean	std	mean	std	sig	mean	std	sig	mean	std
F1	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00
F3	0.00E+00	0.00E+00	-	0.00E+00	0.00E+00	-	6.15E+03	1.37E+04	0.00E+00	0.00E+00	-	0.00E+00	0.00E+00	-	2.34E+04	4.54E+04
F4	5.91E+01	1.67E+00	+	4.80E+01	2.35E+01	-	5.90E+01	1.51E+00	5.45E+01	4.46E+01	=	5.09E+01	4.86E+01	=	5.79E+01	5.06E+01
F5	1.53E+01	3.14E+00	+	1.24E+01	4.98E+00	=	1.10E+01	4.85E+00	3.21E+01	5.68E+00	+	4.02E+01	1.17E+01	+	2.67E+01	5.95E+00
F6	3.87E-05	5.78E-05	+	5.63E-05	8.58E-05	+	1.75E-07	3.78E-07	8.10E-04	1.28E-03	+	3.47E-03	7.73E-03	+	1.37E-05	1.76E-05
F7	4.51E+01	2.38E+00	+	4.20E+01	4.53E+00	=	4.12E+01	4.94E+00	8.09E+01	3.51E+00	+	8.47E+01	1.04E+01	+	7.51E+01	7.98E+00
F8	1.57E+01	2.93E+00	+	1.29E+01	2.93E+00	=	1.24E+01	4.84E+00	3.39E+01	5.11E+00	+	3.99E+01	9.81E+00	+	2.70E+01	7.50E+00
F9	4.95E-02	1.11E-01	+	1.05E-01	1.73E-01	+	0.00E+00	0.00E+00	9.73E-01	9.34E-01	+	1.13E+00	1.24E+00	+	2.13E-02	9.01E-02
F10	1.67E+03	2.38E+02	+	1.47E+03	3.59E+02	=	1.49E+03	4.19E+02	3.40E+03	2.72E+02	=	3.05E+03	5.67E+02	-	3.38E+03	4.80E+02
F11	3.20E+01	2.62E+01	+	4.64E+01	3.27E+01	+	2.36E+01	2.78E+01	1.27E+02	3.11E+01	+	1.50E+02	2.80E+01	+	3.73E+01	8.59E+00
F12	1.20E+03	4.31E+02	+	1.51E+03	1.55E+03	+	5.00E+02	2.70E+02	5.96E+03	3.57E+03	=	8.20E+03	1.23E+04	=	6.83E+03	7.60E+03
F13	3.60E+01	1.63E+01	+	5.72E+01	3.33E+01	+	2.22E+01	1.56E+01	3.74E+02	4.27E+02	+	5.45E+02	4.56E+02	+	6.40E+01	5.19E+01
F14	3.04E+01	5.88E+00	+	4.95E+01	1.80E+01	+	2.37E+01	3.60E+00	2.15E+02	6.32E+01	+	2.64E+02	7.41E+01	+	3.89E+01	1.45E+01
F15	2.22E+01	1.29E+01	+	6.82E+01	5.23E+01	+	3.57E+00	1.95E+00	3.06E+02	1.39E+02	+	3.96E+02	1.19E+02	+	5.87E+01	4.01E+01
F16	2.55E+02	1.50E+02	+	3.10E+02	1.38E+02	+	1.82E+02	1.61E+02	7.44E+02	1.88E+02	+	7.06E+02	2.13E+02	+	6.07E+02	2.10E+02
F17	5.09E+01	2.68E+01	+	4.80E+01	3.07E+01	+	3.45E+01	2.81E+01	5.27E+02	1.28E+02	=	5.89E+02	1.86E+02	=	5.83E+02	2.29E+02
F18	6.06E+01	4.71E+01	+	1.35E+02	7.60E+01	+	2.44E+01	1.15E+01	1.93E+02	1.15E+02	+	1.96E+02	1.04E+02	+	7.42E+01	3.96E+01
F19	1.55E+01	1.72E+01	+	5.36E+01	3.44E+01	+	5.98E+00	2.79E+00	1.47E+02	4.60E+01	+	1.55E+02	5.01E+01	+	2.73E+01	1.22E+01
F20	7.30E+01	4.97E+01	+	7.72E+01	6.07E+01	=	7.76E+01	6.15E+01	3.29E+02	1.15E+02	=	3.45E+02	1.83E+02	=	3.73E+02	1.88E+02
F21	2.17E+02	3.80E+00	+	2.13E+02	4.82E+00	=	2.13E+02	4.56E+00	2.35E+02	4.59E+00	+	2.40E+02	9.77E+00	+	2.30E+02	7.59E+00
F22	1.00E+02	1.49E-13	+	1.00E+02	1.14E-13	=	1.00E+02	1.21E-13	3.19E+03	1.56E+03	=	3.52E+03	1.46E+03	=	3.35E+03	1.26E+03
F23	3.64E+02	4.83E+00	+	3.64E+02	8.29E+00	=	3.61E+02	7.07E+00	4.59E+02	9.53E+00	=	4.75E+02	1.63E+01	+	4.59E+02	1.57E+01
F24	4.35E+02	3.18E+00	=	4.37E+02	6.61E+00	=	4.36E+02	5.78E+00	5.30E+02	5.58E+00	=	5.49E+02	1.17E+01	+	5.30E+02	8.13E+00
F25	3.87E+02	8.04E-02	+	3.87E+02	2.57E-01	+	3.87E+02	1.92E-02	5.14E+02	3.79E+01	=	5.21E+02	4.22E+01	=	5.10E+02	3.58E+01
F26	1.08E+03	6.11E+01	=	1.08E+03	1.81E+02	=	1.07E+03	6.58E+01	1.40E+03	8.54E+01	=	1.58E+03	1.45E+02	+	1.40E+03	1.08E+02
F27	5.06E+02	6.75E+00	+	5.09E+02	8.10E+00	+	5.03E+02	5.63E+00	5.45E+02	2.64E+01	+	5.61E+02	3.05E+01	+	5.21E+02	9.76E+00
F28	3.43E+02	5.82E+01	+	3.48E+02	6.18E+01	+	3.24E+02	4.68E+01	4.89E+02	2.50E+01	+	4.92E+02	2.24E+01	+	4.67E+02	1.84E+01
F29	4.65E+02	3.52E+01	+	4.49E+02	3.08E+01	+	4.35E+02	1.55E+01	4.61E+02	9.22E+01	+	5.25E+02	1.18E+02	+	3.86E+02	6.73E+01
F30	2.09E+03	1.27E+02	+	2.21E+03	2.02E+02	+	2.02E+03	9.06E+01	7.23E+05	1.28E+05	+	6.63E+05	6.18E+04	+	6.11E+05	3.57E+04
W	**25**			**16**					**17**			**20**				
T	**3**			**11**					**11**			**7**				
L	**1**			**2**					**1**			**2**				

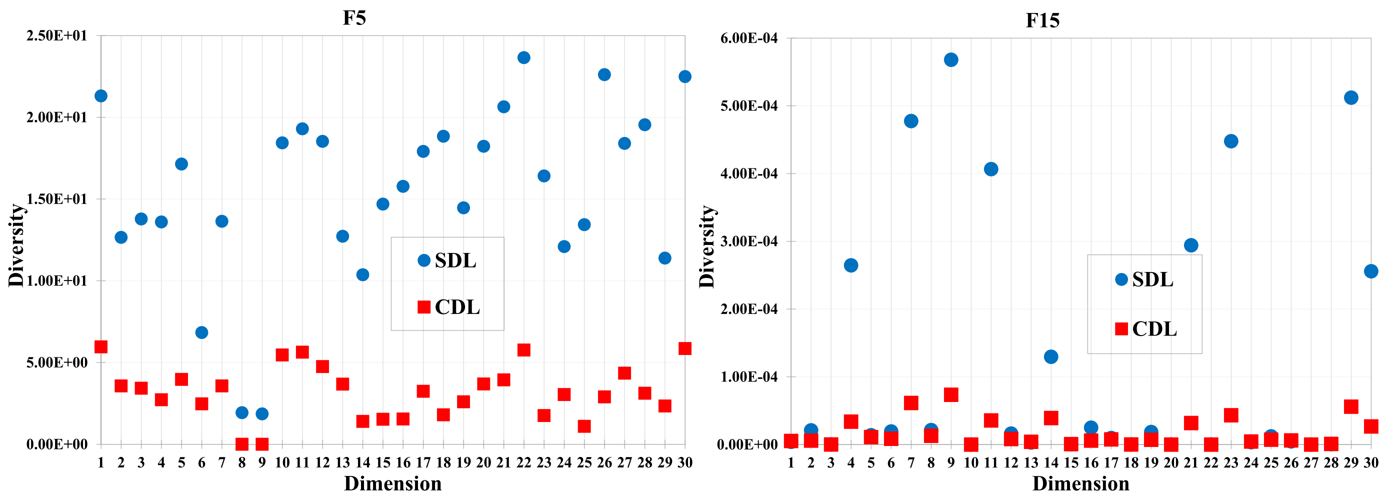

Fig. 6. Achieved final diversity of dimensions by SDL and CDL on 30-D simple multi-modal function F5 and hybrid function F15.

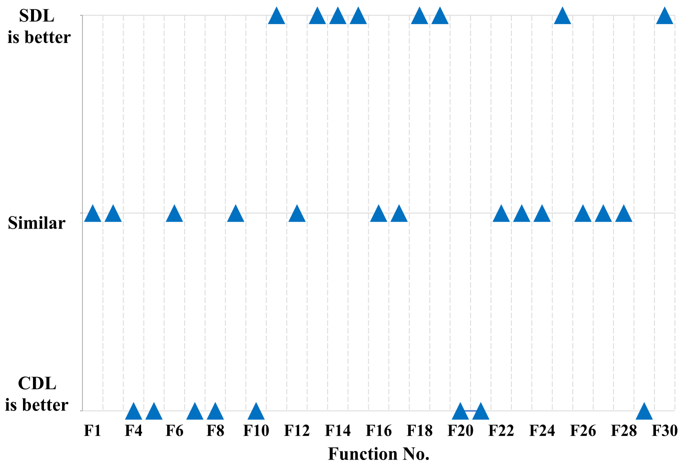

Fig. 7. Performance comparison of SDL and CDL on the twenty-nine 30-D CEC2017 functions.

effectiveness from the perspective of experiment, it is compared with the following two variants.

reverse: In this variant, small diversity dimensions perform exploration while large diversity dimensions perform exploitation, termed as SErLEi.

random: In this variant, dimensions to exploit or to explore is random, regardless of their diversity.

Except the above differences, other settings are kept the same as ODFDE. Table 9 shows the comparison results on 30-D functions. From Table 9, ODFDE is significantly better than the reverse variant, winning in 21 functions and losing in 1 function. On the ten hybrid functions F11-F20, ODFDE performs better than the reverse variant on 9 functions (F11, F12 and F14-F20) and similarly on 1 function (F13). Compared with the random variant, ODFDE is better on 5 functions. Again, from Table 9, four (F12, F15, F18 and F19) out of these five functions are hybrid functions. Performance superiority to *reverse* is much more significant because this variant always assigns exploitative and explorative tasks against ODFDE.

To have a deeper insight into the working process, the average number of exploitative operations in dimension within each group over the entire evolution process is compared and the rankings among groups

Table 6
Comparison results of ODFDE with single strategy according to Holm, Hochberg and Hommel procedures.

vs.	unadjusted p	p_{Holm}	$p_{Hochberg}$	p_{Hommel}
SDL	**0.000351**	**0.000351**	**0.000351**	**0.000351**
CDL	**<1E-6**	**<1E-6**	**<1E-6**	**<1E-6**

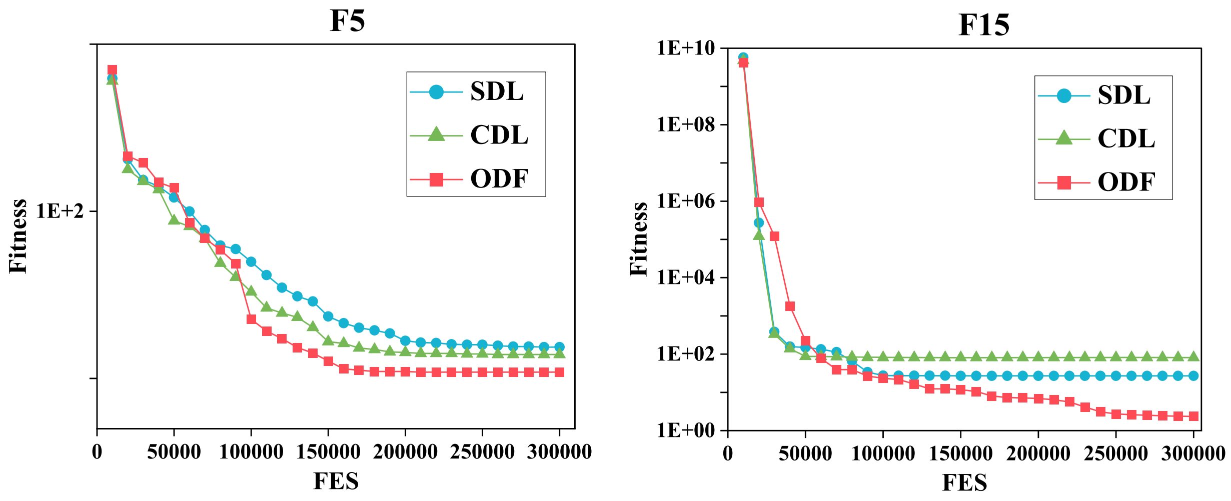

Fig. 8. Convergence plots of SDL, CDL and ODF on 30-D simple multi-modal function F5 and hybrid function F15 in the trial with the median error value.

Table 7
Performance comparison results of DSK_only with other NoD values.

vs.	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	0.95	0.99
win	19	18	20	19	17	20	15	15	6	10	7
tie	10	11	8	9	11	9	14	14	21	19	21
lose	0	0	1	1	1	0	0	0	2	0	1

Table 8
Performance comparisons of ODFDE with single OSK and DSK on 30-D CEC2017 benchmark set over 51 independent runs.

	OSK_only			DSK_only			ODFDE	
	mean	std	sig	mean	std	sig	mean	std
F1	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00
F3	0.00E+00	0.00E+00	-	1.40E+04	1.56E+04	+	6.15E+03	1.37E+04
F4	4.33E+01	2.68E+01	-	5.76E+01	8.30E+00	-	5.90E+01	1.51E+00
F5	9.99E+00	2.83E+00	=	1.16E+01	3.54E+00	=	1.10E+01	4.85E+00
F6	4.55E-05	6.70E-05	+	4.50E-07	1.07E-06	=	1.75E-07	3.78E-07
F7	3.93E+01	2.41E+00	=	4.23E+01	3.34E+00	+	4.12E+01	4.94E+00
F8	1.20E+01	2.86E+00	=	1.24E+01	3.78E+00	=	1.24E+01	4.84E+00
F9	1.03E-01	2.23E-01	+	1.42E-02	6.64E-02	+	0.00E+00	0.00E+00
F10	1.51E+03	3.77E+02	=	1.43E+03	3.15E+02	=	1.49E+03	4.19E+02
F11	4.07E+01	2.56E+01	+	3.79E+01	2.93E+01	+	2.36E+01	2.78E+01
F12	1.27E+03	5.38E+02	+	4.85E+03	3.90E+03	+	5.00E+02	2.70E+02
F13	5.87E+01	3.61E+01	+	4.13E+01	2.99E+01	+	2.22E+01	1.56E+01
F14	5.26E+01	1.95E+01	+	1.92E+01	9.03E+00	=	2.37E+01	3.60E+00
F15	6.34E+01	6.47E+01	+	4.11E+00	3.14E+00	=	3.57E+00	1.95E+00
F16	3.07E+02	1.35E+02	+	2.21E+02	1.29E+02	=	1.82E+02	1.61E+02
F17	5.64E+01	4.71E+01	+	4.23E+01	2.94E+01	+	3.45E+01	2.81E+01
F18	1.28E+02	7.46E+01	+	9.22E+03	4.07E+04	=	2.44E+01	1.15E+01
F19	5.38E+01	3.34E+01	+	7.16E+00	3.08E+00	=	5.98E+00	2.79E+00
F20	8.27E+01	6.05E+01	=	5.48E+01	5.20E+01	=	7.76E+01	6.15E+01
F21	2.12E+02	3.04E+00	=	2.14E+02	3.84E+00	=	2.13E+02	4.56E+00
F22	1.00E+02	1.14E-13	=	1.00E+02	1.66E-13	=	1.00E+02	1.21E-13
F23	3.64E+02	4.67E+00	+	3.64E+02	6.62E+00	=	3.61E+02	7.07E+00
F24	4.36E+02	5.55E+00	=	4.37E+02	4.95E+00	=	4.36E+02	5.78E+00
F25	3.87E+02	2.30E-01	+	3.87E+02	2.95E-02	+	3.87E+02	1.92E-02
F26	1.06E+03	6.48E+01	=	1.06E+03	6.61E+01	=	1.07E+03	6.58E+01
F27	5.08E+02	6.85E+00	+	5.05E+02	4.95E+00	=	5.03E+02	5.63E+00
F28	3.48E+02	5.90E+01	+	3.34E+02	5.47E+01	=	3.24E+02	4.68E+01
F29	4.36E+02	2.04E+01	=	4.40E+02	3.17E+01	=	4.35E+02	1.55E+01
F30	2.18E+03	1.92E+02	+	2.09E+03	8.67E+01	+	2.02E+03	9.06E+01
W	**16**			**9**				
T	**11**			**19**				
L	**2**			**1**				

in descending order are shown in Table 10. The unimodal basic functions are highlighted with gray background. From Table 10, these unimodal basic functions have the highest rankings among the groups. Among the associated hybrid functions, ODF performs better than *reverse* and *random* on {F11, F12, F14, F15, F18, F19} and {F12, F15, F18, F19}, respectively. The above observations demonstrate the advantages of SEiLEr for solving hybrid functions by correctly detecting the relative function difficulties and confirm the illustration given previously in Section 3.2.

Fig. 9 plots the best fitness achieved by the reverse variant and SEiLEr over generations on functions F5 and F15. It was known from Fig. 6 that F5 requires relatively more exploitation while F15 needs relatively more exploration. Fig. 9 implies that on both functions, SEiLEr obtains better results, indicating that SEiLEr maintains a better exploitation and exploration balance than the reverse variant and exhibits a more reliable performance for different types of optimization tasks.

To show the differences in decision space, Fig. 10 presents the diversity comparison. From this figure, SEiLEr has a faster diversity decrease than the reverse variant on F5 while a relatively slower diversity decrease on F15. This correctly matches the exploitation/exploration needs of F5 and F15.

4.4. Comparison with OSK-based multi-strategy DEs

In literature, many OSK-based multi-strategy methods have been proposed. The following six methods are considered for performance comparisons with ODF:

ETI-SHADE: Event-triggered impulsive control-based SHADE [46];
SaM-SHADE: Strategy adaptive [38] SHADE;
SaDE: Strategy adaptive DE [9];
IDE: Individual-dependent DE [20];
ACoS-SHADE: Adaptive coordinate system [48] improved SHADE;
MLCCDE: Multi-layer competitive-cooperative DE [51].

To tune the comparison algorithms, which did not use the CEC2017 functions in the original paper for the best performance, the key parameters that mainly affect the performance of the algorithms are

Table 9
Performance comparisons of ODFDE with the variants of SEiLEr on 30-D CEC2017 benchmark set over 51 independent runs.

	reverse			random			ODFDE	
	mean	std	sig	mean	std	sig	mean	std
F1	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00
F3	2.12E+03	8.61E+03	-	2.16E+03	8.84E+03	=	6.15E+03	1.37E+04
F4	5.85E+01	8.68E+00	+	5.80E+01	7.93E+00	=	5.90E+01	1.51E+00
F5	1.46E+01	2.96E+00	+	1.17E+01	5.20E+00	=	1.10E+01	4.85E+00
F6	9.66E-06	1.18E-05	+	7.26E-07	2.30E-06	=	1.75E-07	3.78E-07
F7	4.55E+01	3.32E+00	+	4.29E+01	4.87E+00	+	4.12E+01	4.94E+00
F8	1.67E+01	2.50E+00	+	1.33E+01	5.58E+00	=	1.24E+01	4.84E+00
F9	2.82E-02	7.35E-02	+	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00
F10	1.62E+03	2.57E+02	=	1.54E+03	3.78E+02	=	1.49E+03	4.19E+02
F11	2.83E+01	2.80E+01	+	2.18E+01	2.63E+01	=	2.36E+01	2.78E+01
F12	1.19E+03	3.75E+02	+	8.72E+02	2.98E+02	+	5.00E+02	2.70E+02
F13	2.39E+01	1.40E+01	=	2.35E+01	1.57E+01	=	2.22E+01	1.56E+01
F14	2.60E+01	3.48E+00	+	2.42E+01	2.24E+00	=	2.37E+01	3.60E+00
F15	1.23E+01	1.78E+01	+	5.18E+00	2.55E+00	+	3.57E+00	1.95E+00
F16	2.62E+02	1.27E+02	+	2.35E+02	1.67E+02	=	1.82E+02	1.61E+02
F17	4.95E+01	1.89E+01	+	3.47E+01	2.42E+01	=	3.45E+01	2.81E+01
F18	3.92E+01	2.77E+01	+	2.43E+01	3.08E+00	+	2.44E+01	1.15E+01
F19	7.81E+00	1.80E+00	+	8.49E+00	3.17E+00	+	5.98E+00	2.79E+00
F20	8.10E+01	5.67E+01	+	7.11E+01	6.08E+01	=	7.76E+01	6.15E+01
F21	2.19E+02	3.41E+00	+	2.13E+02	4.68E+00	=	2.13E+02	4.56E+00
F22	1.00E+02	1.14E-13	=	1.00E+02	1.11E-13	=	1.00E+02	1.21E-13
F23	3.64E+02	4.96E+00	+	3.60E+02	7.04E+00	=	3.61E+02	7.07E+00
F24	4.36E+02	3.87E+00	=	4.35E+02	5.24E+00	=	4.36E+02	5.78E+00
F25	3.87E+02	5.37E-02	+	3.87E+02	1.72E-02	=	3.87E+02	1.92E-02
F26	1.05E+03	1.23E+02	=	1.05E+03	7.34E+01	=	1.07E+03	6.58E+01
F27	5.05E+02	7.15E+00	=	5.04E+02	6.48E+00	=	5.03E+02	5.63E+00
F28	3.37E+02	5.67E+01	+	3.19E+02	4.12E+01	=	3.24E+02	4.68E+01
F29	4.61E+02	2.45E+01	+	4.38E+02	2.70E+01	=	4.35E+02	1.55E+01
F30	2.08E+03	1.10E+02	+	2.02E+03	9.12E+01	=	2.02E+03	9.06E+01
W	**21**			**5**				
T	**7**			**24**				
L	**1**			**0**				

Table 10
Ranking (in descending order) of the average number of Ei operations in dimension within each group (1 denotes the highest ranking).

Fun.	Group index					
	1	2	3	4	5	6
F11	1	3	2	NA	NA	NA
F12	2	3	1	NA	NA	NA
F13	1	2	3	NA	NA	NA
F14	1	3	4	2	NA	NA
F15	1	4	3	2	NA	NA
F16	4	2	1	3	NA	NA
F17	5	4	1	3	2	NA
F18	1	4	5	3	2	NA
F19	1	3	2	4	5	NA
F20	1	2	6	3	4	5

NA: Not available

identified from the original literature. Several values of the parameters are tested, and the best-performing parameters are obtained by the Friedman test with the best ranking. The considered parameters, range and the tuned values are summarized in Table S2 in the supplementary file.

Table S3 in the supplementary file shows the comparisons on 10-, 30-, 50- and 100-D functions while Table 11 summarizes the results. From these tables, ODFDE performs significantly better in 30-, 50- and 100-D cases, winning on no less than 15 functions and losing on no more than 6 functions. For instance, in the 30-D case, the "win/lose" count compared with ETI-SHADE, SaM-SHADE, SaDE, IDE, ACoS-SHADE and MLCCDE is "20/3", "24/1", "24/3", "20/2", "21/3" and "19/4", respectively. On the 10-D functions, IDE and MLCCDE perform much better while ODFDE is superior to SaM-SHADE, SaDE and competitive to ETI-SHADE and ACoS-SHADE.

Table 12 reports the achieved p valves according to Holm, Hochberg and Hommel procedures [62]. It can be observed that ODFDE performs statistically better than all the six DEs in all the cases except MLCCDE in

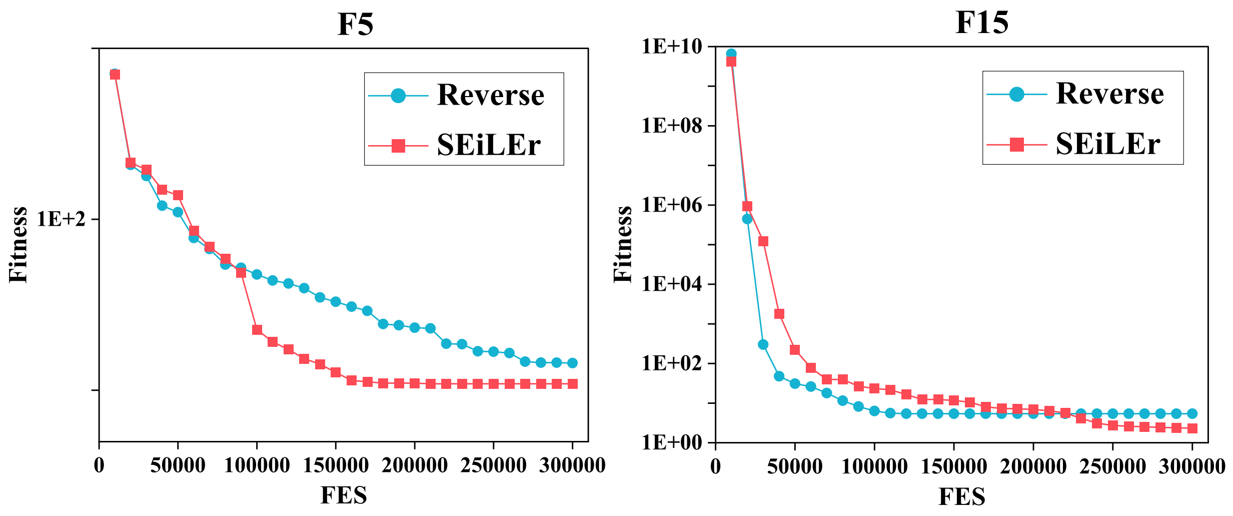

Fig. 9. Best fitness achieved by the reverse variant and SEiLEr over function evaluations (FES) on 30-D simple multi-modal function F5 and hybrid function F15 in the trial with the median error value.

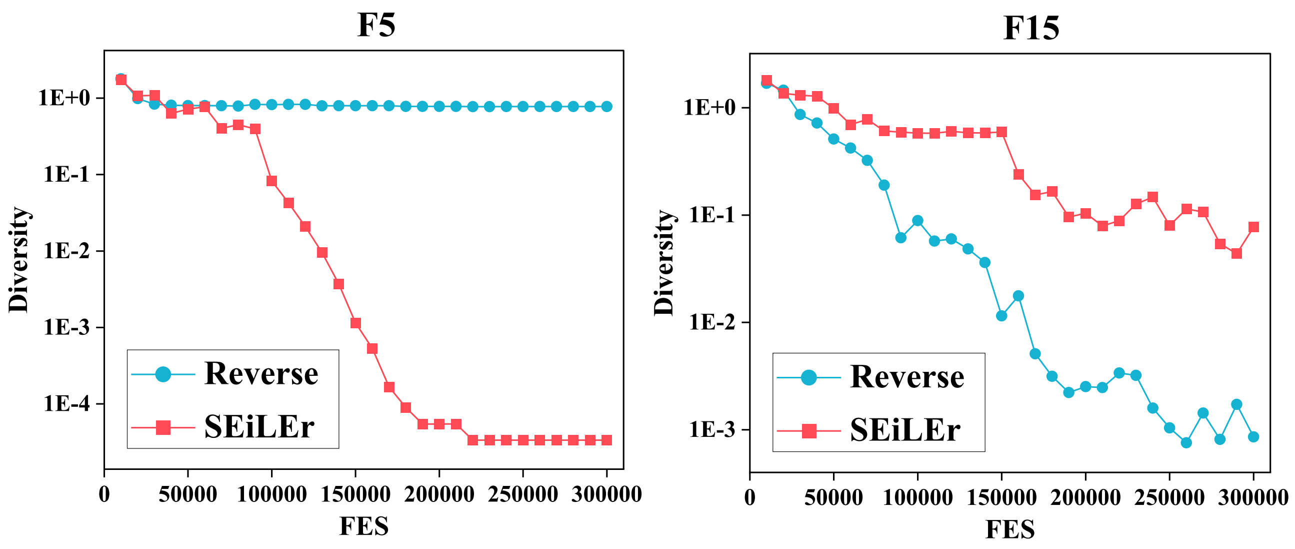

Fig. 10. Diversity achieved by the reverse variant and SEiLEr over function evaluations (FES) on 30-D simple multi-modal function F5 and hybrid function F15 in the trial with the median error value.

Table 11
Comparison results of ODFDE with OSK-based multi-strategy DEs.

vs.	win	tie	lose	win	tie	lose
	10-D			30-D		
ETI-SHADE	**8**	13	8	**20**	6	3
SaM-SHADE	**8**	19	2	**24**	4	1
SaDE	**9**	13	7	**24**	2	3
IDE	**5**	12	12	**20**	7	2
ACoS-SHADE	**7**	14	8	**21**	5	3
MLCCDE	**5**	12	12	**19**	6	4
	50-D			100-D		
ETI-SHADE	**20**	5	4	**15**	10	4
SaM-SHADE	**20**	8	1	**18**	6	5
SaDE	**24**	3	2	**26**	3	0
IDE	**20**	6	3	**23**	2	4
ACoS-SHADE	**21**	5	3	**19**	4	6
MLCCDE	**20**	5	4	**22**	4	3

Table 12
Overall performance comparisons of ODFDE with all the compared algorithms according to Holm, Hochberg and Hommel procedures.

vs.	unadjusted p	p_{Holm}	$p_{Hochberg}$	p_{Hommel}
ETI-SHADE	**6E-6**	**1.2E-5**	**1.2E-5**	**1.2E-5**
SaM-SHADE	**<1E-6**	**0.000001**	**0.000001**	**0.000001**
SaDE	**<1E-6**	**<1E-6**	**<1E-6**	**<1E-6**
IDE	**0.000007**	**0.000028**	**0.000028**	**0.000028**
ACoS-SHADE	**0.000123**	**0.000123**	**0.000123**	**0.000123**
MLCCDE	**0.043476**	0.086952	**0.048882**	**0.048882**
AEPD-SHADE	**<1E-6**	**<1E-6**	**<1E-6**	**<1E-6**
SHADE/eig	**0.001**	**0.001**	**0.001**	**0.001**
CSM-SHADE	**<1E-6**	**<1E-6**	**<1E-6**	**<1E-6**
SCSS-SHADE	**0.048882**	0.086952	**0.048882**	**0.048882**

Table 13
Comparison results of ODFDE with DSK-based multi-strategy DEs.

vs.	win	tie	lose	win	tie	lose
	10-D			30-D		
AEPD-SHADE	**15**	10	4	**20**	8	1
SHADE/eig	**9**	12	8	**19**	8	2
	50-D			100-D		
AEPD-SHADE	**15**	11	3	**16**	6	7
SHADE/eig	**21**	6	2	**22**	4	3

Table 14
Comparison results of ODFDE with OSK and SSK-based multi-strategy DEs.

vs.	win	tie	lose	win	tie	lose
	10-D			30-D		
CSM-SHADE	**18**	10	1	**26**	1	2
SCSS-SHADE	**6**	14	9	**21**	7	1
	50-D			100-D		
CSM-SHADE	**23**	4	2	**23**	3	3
SCSS-SHADE	**16**	11	2	**13**	12	4

terms of Holm test.

4.5. Comparison with DSK-based multi-strategy DEs

ODFDE is further compared with two DSK-based multi-strategy DE algorithms. They are:

AEPD-SHADE: Auto-enhanced population diversity [34] -based SHADE;

SHADE/eig: Eigenvector-based crossover operator [33] enhanced SHADE.

Similarly, the key parameters of these algorithms are tuned, as shown in Table S2. From Tables S4 and 13, it is seen that ODFDE exhibits better performance than AEPD-SHADE and SHADE/eig on all the considered dimensionalities, outperforming on 66 (15+20+15+16) and 71 (9+19+21+22) and underperforming on 15 (4+1+3+7) and 15 (8+2+2+3) functions, respectively. This is also confirmed by Holm, Hochberg and Hommel procedures shown in Table 12, in which ODFDE is statistically better with the p value < 0.05.

4.6. Comparison with OSK and SSK-based multi-strategy DEs

ODFDE is also compared with OSK and SSK-based multi-strategy methods. We consider the following two algorithms:

CSM-SHADE: Cheap surrogate model-based SHADE [55];

SCSS-SHADE: Selective candidate with similarity selection rule-based SHADE [60].

As seen from Tables S5 and 14, ODFDE achieves better results than CSM-SHADE and SCSS-SHADE on 90 and 56 functions and loses on 8 and 16 functions, respectively. With respect to the function dimensionality, it is seen that ODFDE is superior to CSM-SHADE in all the cases. Compared with SCSS-SHADE, ODFDE exhibits better performance in 30-, 50- and 100-D cases while on the 10-D functions, SCSS-SHADE performs slightly better with the “win/lose” metric of “9/6″. Table 12 shows that ODFDE performs statistically better than CSM-SHADE according to Holm, Hochberg and Hommel procedures and SCSS-SHADE according to Hochberg and Hommel procedures.

Fig. 11 shows the convergence graphics of the eleven DEs from Subsections 4–6 on twelve selected 10-, 30-, 50- and 100-D CEC2017 functions. As seen, ODFDE converges to the best fitness on 10 functions and achieves similar performance on 1 function.

4.7. Comparison with state-of-the-art multi-strategy utilization schemes

Sections 4.4, 4.5 and 4.6 have compared ODFDE with OSK, DSK and OSK+SSK based multi-strategy DE algorithms. It is also interesting to test the competitiveness of ODF against other multi-strategy utilization schemes. To this end, the following three methods are considered for comparisons.

Sa: Strategy adaptation method proposed in [9];

CSM: Cheap surrogate model presented in [55];

SCSS: Selective candidate with similarity selection rule proposed in [60].

Each of the three methods is, respectively implemented with SDL and CDL and compared with ODFDE. As seen from Tables S6 and 15, ODF outperforms Sa, CSM and SCSS on 54, 74, 50 and loses on 13, 11, 17 functions, respectively. Similar to

SCSS-SHADE, SCSS exhibits better performance in the 10-D case. From the multi-problem comparison results shown in Table 16, ODF consistently achieves superior performance in all the three cases. In the case of Sa and CSM, the difference is statistically significant.

4.8. Application in real-world problems

To further evaluate the performance of ODF method, it is also compared on ten CEC2011 [63] real-world problems, as shown in Table S7. Thirty trials were performed for each problem with each trial assigned 10,000 × D function evaluations. From the results summarized in Table 17, it can be observed that ODF also performs better on the real-world problems, winning in 6, 6 and 4 and losing in none, none and one function when compared with Sa, CSM and SCSS, respectively.

As pointed out in [35], in real-world optimization problems, different subcomponents of variables may have different properties. The proposed SEiLEr mechanism optimizes subcomponents separately with different strategies. To demonstrate its contribution in the real-world optimization, the two variants *reverse* and *random* constructed previously in Section 4.3 are compared with ODF. As seen from Tables S8 and 18, ODF performs better than *reverse* and *random* in 6 and 4 cases and loses in none case, confirming the effectiveness of the assignment of exploitative and explorative tasks to subcomponents.

4.9. Time complexity of the multi-strategy utilization schemes

To study the time complexity, we follow the method suggested in [35], as described in the followings. T_0 is the time for computing the test program below:

for i = 1:1,000,000

x = 0.55 + (double) i;

*x = x + x; x = x/2; x = x*x; x=sqrt(x); x=log(x); x=exp(x); x = x/(x + 2);* end

T_1 represents the time consumed by 200,000 evaluations on 30-D F18; T_2 is average computing time over 5 trials for an algorithm to optimize the 30-D F18 with 200,000 evaluations. Consequently, $(T_2-T_1)/T_0$ gives the complexity of the algorithm.

All the considered Sa, CSM, SCSS and ODF methods were implemented using MATLAB language for fair comparisons. Table 19 shows the experimental results. It shows that the time complexity of ODF is lower than SCSS and CSM but higher than Sa. The reason is that compared with SCSS and CSM, ODF does not require multiple offspring generation and distance calculations. While compared with Sa, ODF

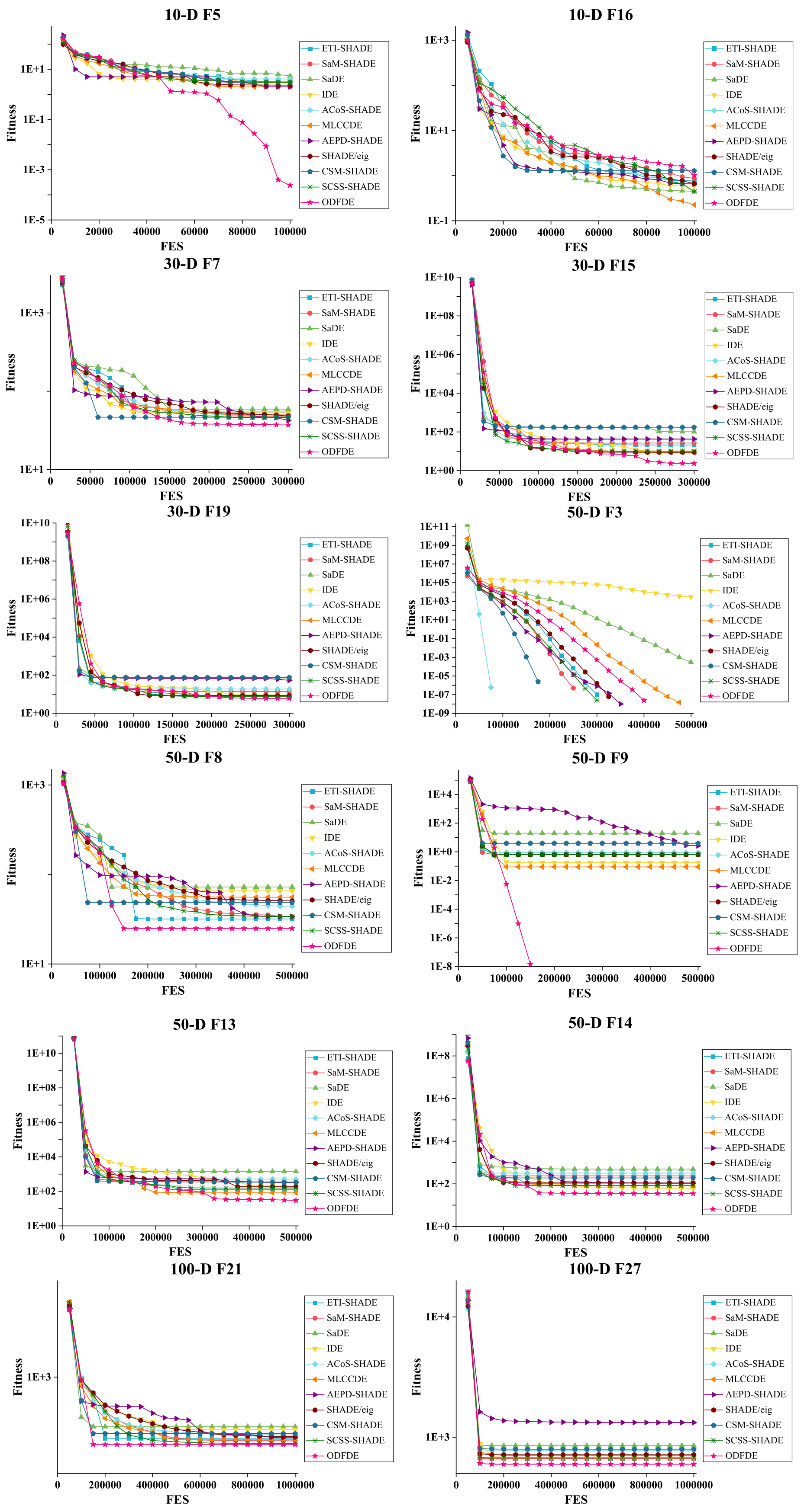

Fig. 11. Convergence graphics of the best fitness by the compared DEs on twelve selected 10-D, 30-D, 50-D and 100-D CEC2017 functions in the trial with the median error value. On 50-D F3 and F9, algorithms terminate when the best fitness reach 1E-08.

Table 15
Comparison results of ODF with state-of-the-art multi-strategy utilization schemes.

vs.	win	tie	lose	win	tie	lose
	10-D			30-D		
Sa	**5**	19	5	**17**	10	2
CSM	**8**	18	3	**23**	4	2
SCSS	**2**	19	8	**12**	13	4
	50-D			100-D		
Sa	**15**	12	2	**17**	8	4
CSM	**22**	5	2	**21**	4	4
SCSS	**17**	10	2	**19**	7	3

Table 16
Overall performance comparisons of ODF with state-of-the-art multi-strategy utilization schemes according to Holm, Hochberg and Hommel procedures.

vs.	unadjusted p	p_{Holm}	$p_{Hochberg}$	p_{Hommel}
Sa	**<1E-6**	**<1E-6**	**<1E-6**	**<1E-6**
CSM	**0.000042**	**0.000085**	**0.000085**	**0.000085**
SCSS	0.127099	0.127099	0.127099	0.127099

Table 17
Comparison results of ODF with state-of-the-art multi-strategy utilization schemes on real-world problems.

vs.	win	tie	lose
Sa	**6**	4	0
CSM	**6**	4	0
SCSS	**4**	5	1

Table 18
Contribution of SEiLEr mechanism in the real-world optimization.

vs.	win	tie	lose
reverse	**6**	4	0
random	**4**	6	0

Table 19
Time complexity comparisons of the strategy utilization schemes (in second).

	T_0	T_1	T_2	$(T_2-T_1)/T_0$
Sa	0.1073	0.7561	1.4706	6.65
CSM			15.0184	132.91
SCSS			2.795	19.00
ODF			1.6758	8.57

Table 20
Comparison results of ODF with state-of-the-art DEs.

	vs.	win	tie	lose	win	tie	lose
		10-D			30-D		
ODF-SCSS-L-SHADE	L-SHADE	**3**	24	2	**11**	15	3
	SCSS-L-SHADE	**5**	19	5	**13**	12	4
ODF-SCSS-jSO	jSO	**7**	15	7	**11**	15	3
	SCSS-jSO	**6**	16	7	**11**	15	3
		50-D			100-D		
ODF-SCSS-L-SHADE	L-SHADE	**13**	13	3	**18**	8	3
	SCSS-L-SHADE	**9**	17	3	**12**	14	3
ODF-SCSS-jSO	jSO	**15**	13	1	**25**	4	0
	SCSS-jSO	**14**	11	4	**16**	12	1

needs to: (1) measure the dimension diversity and (2) sort the fitness and dimension diversity. Thus, the superior performance of ODF does not come for free.

Table 21
Overall performance comparisons of ODF with state-of-the-art DEs according to Holm, Hochberg and Hommel procedures.

ODF-SCSS-L-SHADE vs.	unadjusted p	p_{Holm}	$p_{Hochberg}$	p_{Hommel}
L-SHADE	**0.000008**	**0.000023**	**0.000023**	**0.000023**
SCSS-L-SHADE	**0.009498**	**0.018996**	**0.018996**	**0.018996**
ODF-SCSS- jSO vs.	unadjusted p	p_{Holm}	$p_{Hochberg}$	p_{Hommel}
jSO	**<1E-6**	**<1E-6**	**<1E-6**	**<1E-6**
SCSS-jSO	**0.01977**	**0.01977**	**0.01977**	**0.01977**

Table 22
Comparison results of ODF-SCSS-jSO with other state-of-the-art DEs.

vs.	win	tie	lose	win	tie	lose
	10-D			30-D		
PaDE	**12**	10	7	**13**	8	8
L-SHADE-SP	**8**	12	9	**12**	17	0
EBL-SHADE	**8**	13	8	**13**	13	3
EaDE	**7**	12	10	**11**	16	2
	50-D			100-D		
PaDE	**19**	6	4	**24**	2	3
L-SHADE-SP	**13**	15	1	**24**	5	0
EBL-SHADE	**20**	6	3	**23**	5	1
EaDE	**15**	12	2	**24**	5	0

4.10. Flexibility of the ODF method

(1) **Flexibility on state-of-the-art DEs**

To demonstrate the flexibility of the ODF method, it is further incorporated with two state-of-the-art DE algorithms, namely SCSS-L-SHADE [60] algorithm with linear population size reduction scheme and SCSS-jSO [60], leading to two new variants named ODF-SCSS-L-SHADE and ODF-SCSS-jSO, respectively. To fit the scheme, α in Eq. (9) is set as 1+FES/Max_FES. Parameters for the comparison algorithms are from the corresponding literature since the CEC2017 test suite was originally used.

Performance comparisons of ODF-SCSS-L-SHADE and ODF-SCSS-jSO with the original DEs are shown in Tables S9 and 20. From these tables, ODF variants perform significantly better than the original baselines in 30-, 50- and 100-D cases and are competitive in 10-D case. For instance, when comparing ODF-SCSS-jSO with SCSS-jSO, the "win/lose" metric in 10-, 30-, 50- and 100-D cases is "6/7", "11/3", "14/4" and "16/1", respectively. According to the comparison results given by Holm, Hochberg and Hommel procedures in Table 21, ODF methods perform better in all the cases and are statistically significant.

It is also interesting to investigate the performance of the constructed ODF-SCSS-jSO against other state-of-the-art DE variants. To this end, we consider the following four algorithms:

PaDE [22]: It is an improved L-SHADE algorithm with a new parameter adaptation scheme;

L-SHADE-SP [64]: It is an improved jSO algorithm with a selective pressure strategy;

EBL-SHADE [65]: It is an improved L-SHADE algorithm with novel mutation strategies;

EaDE [66]: It is an explicitly adaptative DE based on two competitive

Table 23
Overall performance comparisons of ODF-SCSS-jSO with other state-of-the-art DEs according to Holm, Hochberg and Hommel procedures.

	unadjusted p	p_{Holm}	$p_{Hochberg}$	p_{Hommel}
PaDE	**<1E-6**	**<1E-6**	**<1E-6**	**<1E-6**
L-SHADE-SP	**7.1E-4**	**7.1E-4**	**7.1E-4**	**7.1E-4**
EBL-SHADE	**<1E-6**	**<1E-6**	**<1E-6**	**<1E-6**
EaDE	**1.9E-3**	**1.9E-3**	**1.9E-3**	**1.9E-3**

baselines.

Parameters for these state-of-the-art DEs are from the corresponding literature since the CEC2017 test suite was originally used. The best, worst, median, mean and standard deviation values obtained by ODF-SCSS-jSO are presented in Table S10 and the detailed results in the format required by the CEC competition are provided as supplementary data. From the comparison results in Tables S11 and 22, the ODF variant performs better than the four compared DEs in the 30-, 50- and 100-D cases and comparably in the 10-D case. It is also observed that the superiority becomes more significant as the dimensionality increases. The reason lies in that ODF considers the dimensional level difference and might be more suitable for handling many variables. From Table 23, the ODF variant is statistically better in all the cases compared with all the considered DEs.

(2) **Flexibility on the CEC2022 test suite**

The previous experiments employed the CEC2017 test suite as the standard benchmark for verifying the performance of ODF. To further examine the flexibility on a wider variety of functions, we consider the CEC2022 test suite, which consists of twelve functions with the dimensionalities of 10 and 20, respectively. Following the requirement in [67], the maximum number of function evaluations is set as 200,000 and 1000,000 for the 10-D and 20-D functions, respectively. For a comparison, we consider the NL-SHADE-LBC (Non-linear population size reduction success-history adaptive DE with linear bias change) [68] algorithm, which is the best performing pure DE algorithm and the second-best competitor in the CEC2022 competition.

Table S12 shows the best, worst, median, mean and standard deviation values obtained by ODF-SCSS-jSO and the detailed results in the format required by the CEC competition are provided as supplementary data. The comparison results of ODF-SCSS-jSO with NL-SHADE-LBC are collected in Table 24, from which NL-SHADE-LBC performs better on the 10-D functions with better and worse performance on 4 and 2 functions, respectively. In the 20-D case, ODF-SCSS-jSO is better with the “win/lose” metric of “3/1″. Overall, it is seen that the advantage of ODF is more significant in the high- dimensional case, which is in consistent with the previous observation on the CEC2017 functions.

Table 24
Comparison results of ODF-SCSS-jSO with NL-SHADE-LBC on the CEC2022 test suite.

		NL-SHADE-LBC			ODF-SCSS-jSO	
		mean	std	sig.	mean	std
10-D	F1	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00
	F2	1.33E-01	7.28E-01	-	7.27E+00	2.36E+00
	F3	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00
	F4	1.30E+00	7.91E-01	=	1.43E+00	1.13E+00
	F5	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00
	F6	1.24E-01	1.27E-01	-	2.76E-01	1.59E-01
	F7	0.00E+00	0.00E+00	-	3.95E-01	3.06E-01
	F8	4.60E-02	3.87E-02	-	1.48E+00	3.77E+00
	F9	2.29E+02	8.67E-14	=	2.29E+02	0.00E+00
	F10	1.00E+02	3.00E-02	+	1.00E+02	4.22E-02
	F11	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00
	F12	1.65E+02	4.11E-01	+	1.64E+02	1.12E+00
	W/T/L	**2/6/4**				
20-D	F1	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00
	F2	4.73E+01	8.97E+00	+	4.84E+01	1.59E+00
	F3	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00
	F4	4.45E+00	1.42E+00	=	4.34E+00	1.68E+00
	F5	0.00E+00	0.00E+00	=	0.00E+00	0.00E+00
	F6	6.36E-01	5.69E-01	=	4.90E-01	1.38E-02
	F7	2.58E+00	5.84E+00	-	4.51E+00	6.18E+00
	F8	1.65E+01	6.43E+00	=	1.87E+01	3.67E+00
	F9	1.81E+02	2.89E-14	=	1.81E+02	8.67E-14
	F10	1.00E+02	2.33E-02	+	1.00E+02	1.61E-02
	F11	3.03E+02	1.83E+01	=	3.00E+02	8.44E-14
	F12	2.39E+02	4.20E+00	+	2.32E+02	8.97E-01
	W/T/L	**3/8/1**				

5. Conclusion

In this paper, an objective-dimension feedback (ODF) based method has been proposed to fully utilize the objective and dimension space knowledge for performance enhancement of DE. In ODF, diversity ranking of each dimension and fitness ranking of individuals are simultaneously used to assign the tasks of exploitation and exploration (with the SEiLEr mechanism) and the amount of exploitation and exploration capabilities (with the NDAC mechanism), respectively. In response to the dimension difference, as suggested by SEiLEr, collective dimensional learning (CDL) and single dimensional learning (SDL) are cooperatively implemented in each solution to generate offspring.

Experiments have been conducted mainly on the 29 CEC2017 benchmark functions. The results show that with SEiLEr, performance improvements on hybrid functions in which dimensions are with different properties have been observed. And with OSK, exploitation and exploration capabilities could be distributed among dimensions for general performance enhancements. ODF method has also been compared with several OSK-, SSK- and DSK-based multi-strategy utilization schemes from literature. Studies also show that ODF exhibits significantly superior performance.

With respect to the utilization of OSK in ODF, FR rather than SFR was adopted. The reason is that credit assignments for strategies become difficult when assigned at the dimension level. Further investigations for possible solutions would be considered as future works. Another potential research direction is to apply the proposed method to other kinds of optimization, such as constrained and multi-objective optimization problems [69].

CRediT authorship contribution statement

Sheng Xin Zhang: Conceptualization, Methodology, Software, Writing – original draft, Funding acquisition. **Shao Yong Zheng:** Writing – review & editing, Funding acquisition. **Li Ming Zheng:** Writing – review & editing, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

The work described in this paper was supported by the National Natural Science Foundation of China (No. 62201227; No. 61671485), the Applied Science and Technology Research and Development Special Fund Project of Guangdong Province (No. 2016B010126004), the National Special Project Number for International Cooperation (No. 2015DFR11050) and the Guangzhou Basic and Applied Basic Research (No. SL2022A04J01366).

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.swevo.2023.101322.

References

[1] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1997) 67–82.
[2] R. Storn, K. Price, Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (1997) 341–359.
[3] K. Price, R. Storn, J. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer-Verlag, Berlin, Germany, 2005.
[4] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (2011) 4–31.
[5] S. Das, S.S. Mullick, P.N. Suganthan, Recent advances in differential evolution—an updated survey, Swarm Evol. Comput. 27 (2016) 1–30.
[6] S.X. Zhang, Y.N. Wen, Y.H. Liu, L.M. Zheng, S.Y. Zheng, Differential evolution with domain transform, IEEE Trans. Evol. Comput. (2022), https://doi.org/10.1109/TEVC.2022.3220424.
[7] S. Gao, Y. Yu, Y. Wang, J. Wang, J. Cheng, M. Zhou, Chaotic local search-based differential evolution algorithms for optimization, IEEE Trans. Syst. Man Cybern. Syst. 51 (2021) 3954–3967.
[8] J. Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evol. Comput. 13 (2009) 945–958.
[9] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2009) 398–417.
[10] W. Gong, Z. Cai, Differential evolution with ranking-based mutation operators, IEEE Trans. Cybern. 43 (2013) 2066–2081.
[11] X. Zhang, S.Y. Yuen, A directional mutation operator for differential evolution algorithms, Appl. Soft Comput. 30 (2015) 529–548.
[12] L.M. Zheng, S.X. Zhang, K.S. Tang, S.Y. Zheng, Differential evolution powered by collective information, Inf. Sci. 399 (2017) 13–29.
[13] L.M. Zheng, et al., Enhancing differential evolution with interactive information, Soft Comput. 22 (2018) 7919–7938.
[14] G. Wu, R. Mallipeddi, P.N. Suganthan, Ensemble strategies for population-based optimization algorithms—a survey, Swarm Evol. Comput. 44 (2019) 695–711.
[15] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (2006) 646–657.
[16] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, in: Proceedings of the IEEE Congress on Evolutionary Computation, Cancun, Mexico, 2013, pp. 71–78. Jun.
[17] R. Tanabe, A.S. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in: Proceedings of the IEEE Congress on Evolutionary Computation, Beijing, China, 2014, pp. 1658–1665.
[18] R.A. Sarker, S.M. Elsayed, T. Ray, Differential evolution with dynamic parameters selection for optimization problems, IEEE Trans. Evol. Comput. 18 (2014) 689–707.
[19] W.-.J. Yu, et al., Differential evolution with two-level parameter adaptation, IEEE Trans. Cybern. 44 (2014) 1080–1099.
[20] L. Tang, Y. Dong, J. Liu, Differential evolution with an individual-dependent mechanism, IEEE Trans. Evol. Comput. 19 (2015) 560–574.
[21] N.H. Awad, M.Z. Ali, P.N. Suganthan, Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2017, pp. 372–379. Jun.
[22] Z. Meng, J.-.S. Pan, K.-.K. Tseng, PaDE: an enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization, Knowl. Based Syst. 168 (2019) 80–99.
[23] R. Tanabe, A. Fukunaga, Reviewing and benchmarking parameter control methods in differential evolution, IEEE Trans. Cybern. 50 (2020) 1170–1184.
[24] X.-G. Zhou, C.-X. Peng, J. Liu, Y. Zhang, G.-J. Zhang, Underestimation-assisted global-local cooperative differential evolution and the application to protein structure prediction, IEEE Trans. Evol. Comput. 24 (2020) 536–550.
[25] Z. Meng, Y. Zhong, C. Yang, CS-DE: cooperative strategy based differential evolution with population diversity enhancement, Inf. Sci. 577 (2021) 663–696.
[26] J. Brest, M.S. Maučec, B. Bošković, Single objective real-parameter optimization: algorithm jSO, in: Proceedings of the IEEE Congress on Evolutionary Computation, San Sebastian, 2017, pp. 1311–1318.
[27] V. Stanovov, S. Akhmedova, E. Semenkin, LSHADE algorithm with rank-based selective pressure strategy for solving CEC 2017 benchmark problems, in: Proceedings of the IEEE Congress on Evolutionary Computation, Rio de Janeiro, 2018, pp. 1–8.
[28] Z. Meng, C. Yang, Hip-DE: historical population based mutation strategy in differential evolution with parameter adaptive mechanism, Inf. Sci. 562 (2021) 44–77.
[29] X. Wang, C. Li, J. Zhu, Q. Meng, L-SHADE-E: ensemble of two differential evolution algorithms originating from L-SHADE, Inf. Sci. 552 (2021) 201–219.
[30] Y. Yu, Z. Lei, Y. Wang, T. Zhang, C. Peng, S. Gao, Improving dendritic neuron model with dynamic scale-free network-based differential evolution, IEEE/CAA J. Autom. Sin. 9 (2022) 99–110. Jan.
[31] S. Rahnamayan, H.R. Tizhoosh, M. MA Salama, Opposition-based differential evolution, IEEE Trans. Evol. Comput. 12 (2008) 64–79.
[32] B.-.Y. Qu, P.N. Suganthan, J.-.J. Liang, Differential evolution with neighborhood mutation for multimodal optimization, IEEE Trans. Evol. Comput. 16 (2012) 601–614.
[33] S.-.M. Guo, C.-.C. Yang, Enhancing differential evolution utilizing eigenvector-based crossover operator, IEEE Trans. Evol. Comput. 19 (2015) 31–49.
[34] M. Yang, et al., Differential evolution with auto-enhanced population diversity, IEEE Trans. Cybern. 45 (2015) 302–315.
[35] 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 real-parameter numerical optimization, Nanyang Technol. Univ., Singapore, Nov, 2016.
[36] S. Das, et al., Differential evolution using a neighborhood-based mutation operator, IEEE Trans. Evol. Comput. 13 (2009) 526–553.
[37] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (2011) 55–66.
[38] W. Gong, Z. Cai, C.X. Ling, H. Li, Enhanced differential evolution with adaptive strategies for numerical optimization, IEEE Trans. Syst. Man Cybern. Cybern. 41 (2011) 397–413.
[39] W. Gong, Á. Fialho, Z. Cai, H. Li, Adaptive strategy selection in differential evolution for numerical optimization: an empirical study, Inf. Sci. 181 (2011) 5364–5386.
[40] K. Li, Á. Fialho, S. Kwong, Q. Zhang, Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 18 (2013) 114–130.
[41] G. Wu, R. Mallipeddi, P.N. Suganthan, et al., Differential evolution with multi-population based ensemble of mutation strategies, Inf. Sci. 329 (2016) 329–345.
[42] S.M. Elsayed, R.A. Sarker, D.L. Essam, An improved self-adaptive differential evolution algorithm for optimization problems, IEEE Trans. Ind. Inform. 9 (2013) 89–99.
[43] R. Mallipeddi, P. Suganthan, Q. Pan, M. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Appl. Soft Comput. 11 (2011) 1679–1696.
[44] Q. Fan, X. Yan, Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies, IEEE Trans. Cybern. 46 (2016) 219–232.
[45] S.-M. Guo, C.-C. Yang, P.-H. Hsu, J.S.-H. Tsai, Improving differential evolution with successful-parent-selecting framework, IEEE Trans. Evol. Comput. 19 (2015) 717–730.
[46] W. Du, et al., Differential evolution with event-triggered impulsive control, IEEE Trans. Cybern. 47 (2017) 244–257.
[47] S.X. Zhang, S.Y. Zheng, L.M. Zheng, An efficient multiple variants coordination framework for differential evolution, IEEE Trans. Cybern. 47 (2017) 2780–2793.
[48] Z.-.Z. Liu, et al., An adaptive framework to tune the coordinate systems in nature-inspired optimization algorithms, IEEE Trans. Cybern. 49 (2018) 1403–1416.
[49] M. Tian, X. Gao, C. Dai, Differential evolution with improved individual-based parameter setting and selection strategy, Appl. Soft Comput. 56 (2017) 286–297.
[50] X.-.F. Liu, et al., Historical and heuristic-based adaptive differential evolution, IEEE Trans. Syst. Man Cybern. Syst. 49 (2018) 2623–2635.
[51] S.X. Zhang, L.M. Zheng, K.S. Tang, S.Y. Zheng, W.S. Chan, Multi-layer competitive-cooperative framework for performance enhancement of differential evolution, Inf. Sci. 482 (2019) 86–104.
[52] L.M. Zheng, S.X. Zhang, S.Y. Zheng, Y.M. Pan, Differential evolution algorithm with two-step subpopulation strategy and its application in microwave circuit designs, IEEE Trans. Ind. Inform. 12 (2016) 911–923.
[53] M. Tian, X. Gao, Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization, Inf. Sci. 478 (2019) 422–448.
[54] G. Sun, et al., Differential evolution with individual-dependent topology adaptation, Inf. Sci. 450 (2018) 1–38.
[55] W. Gong, A. Zhou, Z. Cai, A multi-operator search strategy based on cheap surrogate models for evolutionary optimization, IEEE Trans. Evol. Comput. 19 (2015) 746–758.
[56] L. Cui, G. Li, Q. Lin, J. Chen, N. Lu, Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations, Comput. Oper. Res. 67 (2016) 155–173.
[57] X. Zhou, G. Zhang, Differential evolution with underestimation-based multimutation strategy, IEEE Trans. Cybern. 49 (2018) 1353–1364.
[58] X. Zhou, G. Zhang, Abstract convex underestimation assisted multistage differential evolution, IEEE Trans. Cybern. 47 (2017) 2730–2741.
[59] S. Elsayed, R. Sarker, C.A. Coello Coello, Fuzzy rule-based design of evolutionary algorithm for optimization, IEEE Trans. Cybern. 49 (2019) 301–314.
[60] S.X. Zhang, W.S. Chan, Z.K. Peng, S.Y. Zheng, K.S. Tang, Selective-candidate framework with similarity selection rule for evolutionary optimization, Swarm Evol. Comput. 56 (2020), 100696.
[61] D.J. Sheskin. Handbook of Parametric and Nonparametric Statistical Procedure, 5th Ed., CRC Press, Boca Raton, FL, 2011.
[62] J. Derrac, S. Garcia, 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.
[63] S. Das, P.N. Suganthan, Problem Definitions and Evaluation Criteria For CEC 2011 Competition On Testing Evolutionary Algorithms On Real World Optimization Problems, Jadavpur University, Nanyang Technological University, 2010. Technical Report.
[64] V. Stanovov, A. Shakhnaz, S. Eugene, Selective pressure strategy in differential evolution: exploitation improvement in solving global optimization problems, Swarm Evol. Comput. 50 (2019), 100463.
[65] A.W. Mohamed, A.A. Hadi, K.M. Jambi, Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization, Swarm Evol. Comput. 50 (2019), 100455.

[66] S.X. Zhang, W.S. Chan, K.S. Tang, S.Y. Zheng, Adaptive strategy in differential evolution via explicit exploitation and exploration controls, Appl. Soft Comput. 107 (2021), 107494.

[67] A. Kumar, K.V. Price, A.W. Mohamed, A.A. Hadi, P.N. Suganthan, Problem Definitions and Evaluation Criteria For the CEC 2022 Special Session and Competition On Single Objective Bound Constrained Numerical Optimization, Nanyang Technol. Univ., Singapore, 2022. Tech. Rep.

[68] V. Stanovov, S. Akhmedova, E. Semenkin, NL-SHADE-LBC algorithm with linear parameter adaptation bias change for CEC 2022 numerical optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, Padua, Italy, 2022, pp. 01–08.

[69] J.D. Ser, et al., Bio-inspired computation: where we stand and what's next, Swarm Evol. Comput. 48 (2019) 220–250.