Enhancing Differential Evolution on Continuous Optimization Problems by Detecting Promising Leaders

YIQIAO CAI, DUANWEI WU, SHAOPEG LIU, SHUNKAI FU, AND PEIZHONG LIU

1College of Computer Science and Technology, Huaqiao University, Xiamen 361021, China
2Department of Computer Science, Guangdong Polytechnic Normal University, Guangzhou 510006, China
3College of Engineering, Huaqiao University, Quanzhou 362000, China

Corresponding authors: Yiqiao Cai (caiyq@hqu.edu.cn) and Shaopeng Liu (149265005@qq.com)

This work was supported in part by the Natural Science Foundation of Fujian Province of China under Grant 2018J01091, in part by the Fujian Provincial Science and Technology Major Project under Grant 2020HZ02014, and in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2019A1515012048.

ABSTRACT Due to that the performance of differential evolution (DE) significantly depends on offspring generation strategies, various DE variants have been reported with the improved mutation operators. However, on the one hand, the mutation operators in most DE variants are guided by the elites in terms of the fitness value, without considering their distribution information in the fitness landscape. It may lead to the population being evolving towards the unpromising areas more frequently if these elites are clustered in a locally optimal region. On the other hand, in most DE variants, the evolutionary information of the potential trial vectors is not fully utilized to guide the search, which will hamper the local exploitation in the promising regions that they are located in. To overcome these weaknesses, this article proposes an enhanced DE framework (DELDG) with a leaders-detection-and-guidance mechanism that consists of an adaptive leaders detection (ALD) and a neighborhood-based tournament selection (NTS). With these two novel operators, DELDG can not only guide the mutation process of each individual with multiple promising leaders detected by ALD, but also accelerate the convergence speed with the competition among the potential trial vectors by NTS. Therefore, DELDG is characterized by the explicit detection of the promising leaders according to their fitness values and distribution information and the effective use of the potential trial vectors in the neighborhood of each leader. Compared with 36 excellent DE variants and evolutionary algorithms (EAs), the experimental results on 28 IEEE CEC2013 real-parameter functions and 17 IEEE CEC2011 real-world problems have demonstrated the competitive performance of DELDG.

INDEX TERMS Differential evolution, guiding mechanism, adaptive leaders detection, neighborhood-based tournament selection, continuous optimization.

I. INTRODUCTION

Since its invention in 1997, differential evolution (DE) has received rapid popularization and developed as a powerful and successful optimization tool for the optimization problems [1], [2]. Due to its simple structure, easy to implement, and strong robustness, DE has been successfully applied to solve different complicated optimization problems, such as multi-objective, multi-modal, large-scale, and dynamic problems, and various real-world applications in the field of science and engineering [3], [4].

Like other population-based evolutionary algorithms (EAs), DE manipulates a population of candidate individuals by three evolutionary operators, i.e., mutation, crossover, and selection [1]. However, as shown in [3], [4], the performance of DE significantly depends on offspring generation strategies (i.e., mutation and crossover) and the control parameters (i.e., the population size $NP$, the scale factor for mutation $F$, and the crossover rate $Cr$). Therefore, to further enhance the performance of DE, much research effort has been devoted to the advanced search techniques to overcome the weaknesses of DE, which results in the emergence of many DE variants. For the mutation operator of DE, the techniques in the existing DE variants mainly focus on the following aspects:

...
developing new mutation strategies, integrating multiple strategies, and hybridize with other EAs and swarm intelligence algorithms [2]–[4]. For the crossover operator of DE, several sophisticated techniques, e.g., multiple exponential recombination technique [5], linkage learning technique [6], and the Eigen coordinate system technique [7], are introduced into DE to enhance its search ability. For the parameter control in DE, adaptive or self-adaptive techniques are designed for DE based on the feedback information from the whole population or/and different individuals [8]–[10]. A detailed review of the enhanced DE variants with the above techniques will be given in Section III.

Although the above-mentioned techniques can alleviate the shortages of DE to some extent, the search ability of the mutation strategy still suffers from a few limitations. Generally, the mutation process of most DE variants is mainly guided by the elites with the best fitness values. Typically, several original mutation strategies, such as “DE/best/1”, “DE/current-to-best/1” and “DE/rand-to-best/1”, use the best individual of the current population to guide the search. In [11], a local and global neighborhood-based mutation operator was presented by guiding the population with the best neighbor of the target individual and the best individual of the current population concurrently. In [9], a new mutation strategy “DE/current-to-pbest” was proposed in JADE by using the top 100p% individuals as the leaders to generate the trial vector for each individual. Due to its competitive performance, many modifications of JADE have been developed to further improve its robustness, such as SHADE [12], L-SHADE [13], jSO [14], and so on. In [15], the strategy “DE/current-to-pbest” with the best neighbor of the target individual and the strategy “DE/current-to-pbest” were used independently to generate two trial vectors for each individual. In these mutation strategies, the leaders for guiding the population are mostly selected based on their fitness values, which may lead to the population being evolved towards the unpromising areas more frequently when they are clustered in a locally optimal region. Therefore, it is necessary to develop an efficient strategy to detect the promising leaders at different stages of evolution by considering both their fitness values and distribution information. Further, in most DE variants, the trial vectors lost in comparison with their parent rivals are directly discarded during the selection process and reuse these failed trial vectors to compete against the stagnated individuals belonging to the same neighborhood. In NTS, the individuals will be regarded as stagnation if they have not been improved within a certain number of times, and each defeated trial vector will have a chance to replace these stagnated individuals if it performs better than the them.

In general, the characteristics of DELDG can be summarized as follows:

- Two novel operators, i.e., adaptive leaders detection (ALD) and neighborhood-based tournament selection (NTS), are introduced into DE to enhance its performance for global optimization. With these two operators, DELDG can provide an effective guidance for each individual with multiple promising leaders and further promote convergence of DE by strengthening the search within the neighboring region of each leader.
- An adaptive leaders detection (ALD) is proposed to avoid the guidance towards the unpromising areas. Other than the mechanism in most DE variants that only uses the top best individuals as leaders to guide the search, ALD elaborately detects the promising leaders for guidance by considering their fitness values and distribution information simultaneously. Further, by arming with an adaptive leaders sizing strategy, ALD can dynamically adjust the number of leaders as the optimization proceeds.
- A neighborhood-based tournament selection (NTS) is presented to make use of the potential trial vectors for the local exploitation. Different from directly discarding the failed trial vectors in most DE variants, NTS performs the tournament selection among the failed trial vectors and the stagnated individuals of the same neighborhood to further exploit the neighboring region of each leader.

The remaining of this article is organized as follows. Section II briefly introduces the basics of DE. Section III reviews the related works on the techniques for offspring generation strategies and control parameters in DE. The proposed DELDG framework is presented in detail in Section IV. In Section V, the experimental study on two sets of benchmark problems are carried out to evaluate the performance of DELDG. Finally, the conclusions and future works of this study are presented in Section VI.

II. BASICS OF DE

DE is a population-based evolutionary algorithm for solving continuous optimization problems. In DE, an initial population ($P^0$) of NP individuals is randomly sampled from the pre-specified search space at the beginning of evolution. Each individual in $P^0$ is denoted as $x^0_i = (x^0_{i,1}, x^0_{i,2}, \ldots, x^0_{i,D})$. 

226558
where $D$ is the dimension of the optimization problem, and $i = 1, 2, \ldots, NP$.

Then, the $j$th variable of $x_{ij}^0$ is generated as follows:

$$x_{ij}^0 = L_j + \text{rndreal}(0, 1) \cdot (U_j - L_j), \quad j = 1, 2, \ldots, D \quad (1)$$

where $\text{rndreal}(0, 1)$ represents a uniformly distributed random number in the interval [0,1], and $U_j (L_j)$ is the upper (lower) bound of the $j$th variable.

Followed that, three key operators of DE, i.e., mutation, crossover, and selection, are carried out iteratively to search for the optimal solution of the problem being optimized.

1) Mutation operator (MO)

In the mutation operator, each individual of the current population ($\overrightarrow{x}_j^g$, also called the target vector) employs a mutation strategy to generate a mutant vector ($\overrightarrow{v}_j^g$). In the DE community, several mutation strategies are used frequently, which are shown below:

- DE/rand/1

$$\overrightarrow{v}_j^g = x_{r1}^g + F \cdot (x_{r2}^g - x_{r3}^g) \quad (2)$$

- DE/rand/2

$$\overrightarrow{v}_j^g = x_{r1}^g + F \cdot (x_{r2}^g - x_{r3}^g) + F \cdot (x_{r4}^g - x_{r5}^g) \quad (3)$$

- DE/best/1

$$\overrightarrow{v}_j^g = x_{\text{best}}^g + F \cdot (x_{r1}^g - x_{r2}^g) \quad (4)$$

- DE/current-to-best/1

$$\overrightarrow{v}_j^g = x_j^g + F \cdot (x_{\text{best}}^g - x_j^g) + F \cdot (x_{r1}^g - x_{r2}^g) \quad (5)$$

- DE/rand-to-best/1

$$\overrightarrow{v}_j^g = x_{r1}^g + F \cdot (x_{\text{best}}^g - x_{r1}^g) + F \cdot (x_{r2}^g - x_{r3}^g) \quad (6)$$

where $r1, r2, r3, r4$, and $r5 \in \{1, 2, \ldots, NP\} \setminus \{i\}$ are mutually different random indices, $x_{\text{best}}^g$ is the best solution at generation $g$. More details of them can refer to [1], [2].

2) Crossover operator (XO)

After the mutation operator, the crossover operator is applied to each pair of $\overrightarrow{x}_j^g$ and $\overrightarrow{v}_j^g$ to generate a trial vector ($\overrightarrow{u}_j^g$). The binomial crossover used in this article is expressed as follows:

$$u_{ij}^g = \begin{cases} \overrightarrow{v}_{ij}^g, & \text{if } \text{rndreal}(0, 1) \leq Cr \\ x_{ij}^g, & \text{otherwise} \end{cases} \quad (7)$$

where $Cr \in [0, 1]$ is the crossover rate and $j_{\text{rand}} \in [1, D]$ is a randomly selected integer. If the variable of the trial vector is out of the pre-specified interval, it will be re-initialized by Eq. (1).

3) Selection operator (SO)

After the crossover operator, the selection operator is carried out to choose the better solution for the next generation from each pair of $\overrightarrow{x}_j^g$ and $\overrightarrow{u}_j^g$, which is executed as follows:

$$x_{ij}^{g+1} = \begin{cases} u_{ij}^g, & \text{if } f(u_{ij}^g) \leq f(x_{ij}^g) \\ x_{ij}^g, & \text{otherwise} \end{cases} \quad (8)$$

where $f(\overrightarrow{x})$ is the fitness value of $\overrightarrow{x}$ for the optimization problem to be minimized. As shown in Eq. (8), the trial vector will successfully replace its parent if it has better performance; otherwise, it will be abandoned directly.

III. RELATED WORK

In recent decades, much research effort has been devoted to further improving the performance of DE on complex optimization problems, leading to the coming of many advanced DE variants. As pointed out in [2]–[4], the offspring generation strategies, i.e., mutation and crossover operators, and the control parameters greatly affect the performance of DE. Therefore, a large body of research has focused on investigating the techniques to overcome the weaknesses of DE from the following aspects.

For the studies on setting the control parameters, numerous adaptive or self-adaptive techniques have been introduced into DE. For example, a self-adaptive DE (SaDE) was proposed in [8] by using a control parameter strategy based on the historical successful and failure experience in generating offspring. In [10], an individual-dependent parameter setting was proposed by the parameter values for each individual based on its fitness value. In [16], the control parameters technique based on zoning evolution was designed for DE by considering its performance with different combinations of control parameters in different search zones. In [17], a new historical and heuristic DE (HHIDE) was presented with a historical-heuristic-based parameter adaptation mechanism. In [18], a self-adaptive method for the parameters control was proposed by learning from the information on the superior individuals. In [19], a novel parameter adaptive DE was proposed by employing a grouping strategy with an adaptation scheme for both $F$ and $Cr$ and a parabolic population size reduction scheme. In [20], a decreasing mechanism for the $J_{\text{rand}}$ number in the crossover operator was proposed based on a feedback guidance technique, and a new strategy for updating the control parameters was designed by the fixed orientation and the Levy distribution.

For the studies on crossover operators, some researchers have focused on investigating the strategy that can effectively handle different types of variable interrelations. In [6], a novel DE with hybrid linkage crossover was proposed by extracting and incorporating the linkage information of the problem to be optimized into the crossover process. In [5], a multiple exponential recombination for DE was presented, where multiple segments of mutant vector and target vector will be exchanged to obtain the trial vector. In [7], an adaptive DE framework to tune the coordinate system was proposed.
and thus the crossover operator was performed based on both the original and Eigen coordinate systems.

In addition to the works on setting the control parameters and enhancing the crossover operator, a great deal of research is focused on developing the high-efficiency mutation operator for DE. In general, the studies on mutation operators can be broadly classified under the following four categories.

The first category is to propose new mutation strategies. In [21], a novel interactive information scheme (IIN) was proposed by incorporating a directional vector that is constructed with the ranking information into the mutation operator for providing promising directional information. In [22], an enhanced fitness-adaptive DE (EFADE) with a new triangular mutation scheme was presented, in which three solutions are selected randomly and their difference vectors are used to define the convex combination vector of the triplet. In [23], an adaptive guided DE (AGDE) with a novel mutation scheme was proposed by randomly selecting two vectors from the top and the bottom 100p% (p ∈ (0, 1)) individuals respectively and selecting the third vector from the middle individuals of the current population. In [9], an important and powerful DE variant, JADE, was proposed with a novel mutation strategy “current-to-pbest” in which the top 100p% best solutions are used to guide the mutation process. Due to its encouraging performance, plenty of improvements have been introduced into JADE. For example, in [12], an enhanced JADE variant with a success-history based parameter adaptation (SHADE) was proposed by using a historical memory of recently successful parameter sets to generate new control parameter values. Followed that, an extended version of SHADE with linear population size reduction (L-SHADE) was presented to further improve its performance by continually decreasing the population size from a larger value (i.e., 18 × D) to a smaller value (i.e., 4) related to the number of fitness evaluations. Recently, an improved version of L-SHADE, termed jSO [14], was proposed with a novel mutation strategy “DE/current-to-pbest/w-1” where a weighted factor $F_w$ is used to control the influence of the guided vector. More details of the JADE and SHADE-based variants can be referred to [24].

The second category is to employ multiple mutation strategies simultaneously to generate new mutant vectors. In [25], a composite DE (CoDE) was proposed by using three different mutation schemes simultaneously to generate trial vectors. In [26], a cheap surrogate model-based multi-operator search strategy was proposed for DE, where multiple mutant vectors are generated by multiple reproduction operators, and the best one evaluated by the surrogate model is selected as the final offspring. Similar to this, a DE variant with an underestimation-based multi-mutation strategy was presented in [27] by selecting the offspring with a cheap abstract convex underestimation model. In [28], a multi-population ensemble DE (MPDE) was presented, where three mutation strategies are used concurrently with three equally sized smaller indicator subpopulations, and an additional larger reward subpopulation is allocated to the best performing mutation strategy. In [18], a DE variant based on the individual difference information (DI-DE) was proposed by selecting a suitable strategy from three different mutation strategies for the superior and inferior individuals respectively based on an adaptive method. In [29], an adaptive DE was proposed by combining two mutation operators with different characteristics, and a cooperative rule was employed to adaptively select a suitable mutation operator based on their historical successful information. In [30], a fitness-based adaptive DE algorithm (FADE) was proposed by adaptively selecting one of three specified mutation strategies to generate offspring for different individuals in the same swarm based on their fitness values.

The third category is to hybridize with other techniques. In [31], a comprehensive review of the existing hybrids on DE and particle swarm optimization (PSO) was given by considering five hybridization factors and building a systematic taxonomy of hybridization strategies. In [32], a hybrid artificial bee colony with DE (HABCDE) was proposed by modifying the scout bee phase with the search mechanisms of DE. In [33], a hybridization of cultural algorithm and DE was proposed where the two meta-heuristics are performed in parallel. In [34], an estimation of distribution algorithm (EDA) was hybridized with DE for generating the offspring together by using the evolutionary operators of DE and the Gaussian probabilistic model of EDA. In [35], a learning-enhanced DE (LeDE) was proposed by combining DE with a clustering-based learning strategy (CLS). In LeDE, the one-step K-means clustering technique was used to divide the population into several clusters and the information of population was exchanged between within the same cluster and between different clusters [35]. In [36], the K-means clustering technique was also incorporated into DE to present a cluster-based population initialization operator where the initial population is composed of the best solution in each cluster. In [37], a novel DE framework, termed self-organizing neighborhood-based DE (SON-DE), was proposed by incorporating a two-dimensional self-organizing map (SOM) into DE to learn the neighborhood relationship of the population for guiding the search of DE. In [38], an adaptive dimension level adjustment framework was designed for DE to address the problem of stagnation through two re-initialization strategies with different search characteristics. In [39], a double layer search mechanism with a species-based clustering partition method was introduced into DE to deal with the multi-modal optimization problems, with the purpose of enhancing the population’s diversity and locating more optima.

The fourth category is to design novel algorithmic frameworks for the parents selection. In [40], a proximity-based DE framework (ProDE) was proposed by selecting the parents based on the probability that is inversely proportional to their Euclidean distances from the target individual. In [41], a DE framework with the ranking-based mutation operators (RankDE) was presented by selecting the individuals as the parents for mutation based on their ranking values.
in the current population. In [42], an enhanced DE with multi-objective sorting-based mutation operators (MSDE) was proposed, in which the non-dominated sorting is used to sort all the individuals based on their fitness and diversity contribution, and then the parents selection is executed based on their ranking values. In [43], a neighborhood-adaptive DE (NaDE) was proposed by using different index-based topologies to define multiple neighborhood relationships for each individual and adaptively selecting a suitable neighborhood for it based on the historical successful and failure experiences of these topologies. In [44], a social learning DE (SL-DE) was presented, in which the social influence of each individual is adopted to construct the neighborhood relationships between individuals and the parents are selected from the neighborhood of each individual.

Issues existed in the previous DE variants: As reviewed above, much research effort has been devoted into studying the mutation operator of DE. Although the promising results have been achieved by these DE variants, the matter of selecting appropriate parents for mutation has not yet been solved properly. For example, JADE and its variants (e.g., SHADE, L-SHADE, and jSO) selects a set of the best individuals to guide the process of mutation only based on their fitness values in the objective space, regardless of their distribution information in the decision space. Accordingly, the evolution of the population might be misguided towards the unpromising areas if these leaders are clustered in a locally optimal region. Therefore, how to choose appropriate parents to guide the mutation process is still an issue that is worth studying. On the other hand, the failed trial vectors generated by the mutation operator will be directly discarded during the selection process in most DE variants. In general, each trial vector can provide the useful feedback information about the problem being optimized, which is beneficial in describing the characteristics of its local fitness landscape. Hence, how to effectively utilize the information of the failed trial vectors is also an important issue to the local exploitation for the problems being optimized, especially for the computationally expensive problems.

IV. PROPOSED ALGORITHM: DELDG

To overcome the above weaknesses in the existing DE variants, this study proposes an enhanced DE framework (DELDG) with a leaders-detection-and-guidance mechanism. In DELDG, an adaptive leaders detection (ALD) is used to dynamically and elaborately detect the promising leaders to build a leadership group (LG) for guiding the population, while a neighborhood-based tournament selection (NTS) is designed to enhance the local exploitation within the neighborhood of each leader by executing the tournament selection among the failed trial vectors and the stagnated individuals.

To present the proposed algorithm more clearly, several notations used in DELDG are listed below.

- \( LG = (\overrightarrow{l_1}, \overrightarrow{l_2}, \ldots, \overrightarrow{l_p}) \): the constructed leadership group with \( p \) leaders;
- \( \overrightarrow{l_i} \): the \( i \)th leader in \( LG \);
- \( p \): the size of \( LG \);
- \( N_i \): the neighborhood for \( \overrightarrow{l_i} \);
- \( Q_i \): the set of the failed trial vectors generated by the individuals in \( N_i \);
- \( t \): the parameter to control the initial value of \( p \);
- \( DIV \): the used division method in ALD;
- \( CC \): the crowding clustering method;
- \( SC \): the speciation clustering method;
- \( KC \): the K-means clustering method;
- \( C_i \): the \( i \)th cluster obtained by applying \( DIV \) to \( PE \);
- \( S_i \): the indicator of search state for \( x_i^0 \);
- \( TH \): the pre-defined threshold for detecting stagnation;
- \( SN_i \): the number of consecutive generations that \( x_i^0 \) cannot be updated successfully.

A. THE GENERAL FRAMEWORK OF DELDG

The general framework of DELDG is summarized in Algorithm 1. It is clear that DELDG consists of three main components: adaptive leaders detection (ALD), offspring generation operators (i.e., the DE operators), and neighborhood-based tournament selection (NTS). In the initialization phase of Algorithm 1, the population \( P^0 = \{x_1^0, x_2^0, \ldots, x_{NP}^0\} \) is initialized randomly from the specified search space by using Eq. (1). Then, the fitness value of each individual \( x_i^0 \in P^0 \) is evaluated with \( f(x_i^0), i = 1, 2, \ldots, NP \). During the search phase, ALD firstly builds a leadership group (LG) with a set of promising individuals detected by a division method (DIV). Here, the size of \( LG \) (denoted as \( p \)) is determined by an adaptive leaders sizing strategy. Then, for each leader \( \overrightarrow{l_j} \in LG \), its neighborhood \( N_j \) is constructed by the members of the cluster that it belongs to. Afterward, with the built \( N_1, \ldots, N_p \), and \( LG \), the DE operators (i.e., \( MO \), \( XO \), and \( SO \)) are performed to update the neighborhood \( N_j \) of \( \overrightarrow{l_j} \) under the guidance of \( LG \), and the failed trial vectors

Algorithm 1 General Framework of DELDG

1: INPUT: the problem to be optimized \( f(\overrightarrow{x}) \);
2: OUTPUT: the optimal solution \( \overrightarrow{x}^* \);
3: Initialize \( P^0 = \{x_1^0, x_2^0, \ldots, x_{NP}^0\} \), and evaluate the fitness value of \( x_i^0 \in P^0, i = 1, \ldots, NP \);
4: While the termination criterion is not satisfied Do
5: /*Adaptive leaders detection*/
6: \( (LG, N_1, \ldots, N_p) = ALD(g, PE, DIV) \);
7: /*Offspring generation operators*/
8: \( (LG, N_1, \ldots, N_p, Q_1, \ldots, Q_p) = DE(LG, \)
9: \( N_1, \ldots, N_p, MO, XO, SO) \);
10: /*Neighborhood-based tournament selection*/
11: \( P_{t+1} = NTS(LG, N_1, \ldots, N_p, Q_1, \ldots, Q_p) \);
12: End While
which are generated by the individuals in $N_j$ are stored in $Q_j$. Finally, NTS is executed on the failed trial vectors in $Q_j$ and the stagnated neighbors in $N_j$ for $\overrightarrow{l_j}$ to obtain the next population.

Further, the flowcharts of the original DE algorithm and the DELDG framework are depicted in Figure 1. Clearly, compared with the original DE algorithm in Figure 1 (a), DELDG is armed with ALD to detect promising leaders for guiding the mutation process and NTS to introduce the competition into the neighborhood of each leader for exploiting the local areas. In the following subsection, these two main components of DELDG, i.e., ALD and NTS, will be detailed in the following subsections.

**B. ADAPTIVE LEADERS DETECTION (ALD)**

To detect a group of promising leaders for guiding, ALD needs to deal with the following three problems:

1) How many promising leaders should be included in $LG$ during the process of evolution?
2) How to detect the promising leaders for $LG$ from current population?
3) How to construct the neighborhood for each leader?

**1) SETTING THE SIZE OF LG**

For the first problem, ALD uses an adaptive leaders sizing strategy to set the value of $p$ dynamically. To be specific, $p$ is determined as follows:

$$ p = \max\{\lfloor NP \times (1 - \frac{g}{G_{\text{max}}}) \times t^{-1} \rfloor, 2\} \quad (9) $$

where $t$ is used to control the initial size of $LG$, $\lfloor \cdot \rfloor$ is the function for rounding down, $g$ is the number of current generation, and $G_{\text{max}}$ is the maximal number of generations. Note that the termination criterion is set based on the maximum number of fitness evaluations ($Max_{\text{FE}}$) in this article, and thus $G_{\text{max}}$ can be obtained by $G_{\text{max}} = Max_{\text{FE}}/NP$.

It can be seen that the value of $p$ is gradually decreased from $NP \times t^{-1}$ to 2 as the iteration proceeds. The rationality of adaptively setting $p$ is as follows. First, as shown in [40], the individuals of population distribute sparsely at the beginning of the iteration and then will gradually cluster densely along with the evolution. Therefore, the distribution information of population at different evolutionary stages can be effectively used by setting $p$ through Eq. (9). Second, the $LG$ with more leaders will be beneficial to the population diversity in the early phase of evolution, while the $LG$ with fewer leaders will focus on convergence in the late phase of
optimization. Hence, the balance between convergence and diversity of population can be dynamically adjusted with the value of $p$ related to the number of generations.

2) DETECTING THE LEADERS FOR $LG$

For the second problem, a division method ($DIV$) is first introduced into ALD to partition the current population into a number of non-overlapping clusters, and then the best individual in each cluster will be identified and selected as a promising leader for $LG$.

Concretely, after $p$ is determined by Eq. (9), a $DIV$ is employed to partition the whole population into $p$ clusters. In this article, three clustering methods are considered as the $DIV$.

- Crowding clustering (CC) [45]
  In CC, a reference point ($\overrightarrow{r}$) is randomly generated first in the search space. Then, the individual $\overrightarrow{x}$ that is closest to $\overrightarrow{r}$ is determined. Afterward, the $NP/p$ individuals nearest to $\overrightarrow{x}$ are selected and combined with $\overrightarrow{x}$ to form a cluster. Once the cluster is constructed, the individuals within it will be eliminated from the population. Other $p-1$ clusters will also be constructed in the same way. The procedure of CC is presented in Algorithm 2.

Algorithm 2 Crowding Clustering (CC) [45]

1: INPUT: $P^g$ and $p$;
2: OUTPUT: the clusters $C_1, C_2, \ldots, C_p$;
3: Randomly produce a reference point $\overrightarrow{r}$;
4: While $P^g \neq \emptyset$ Do
5:   Determine $\overrightarrow{x}$ that is closest to $\overrightarrow{r}$;
6:   Select $NP/p-1$ individuals those are nearest to $\overrightarrow{x}$;
7:   Combine the selected individuals with $\overrightarrow{x}$ to form a cluster;
8:   Eliminate these $NP/p$ from $P^g$;
9: End While

- Speciation clustering (SC) [45]
  In SC, all the individuals of $P^g$ are first sorted based on their fitness values. Then, the best individual of the population ($\overrightarrow{x}_{best}^g$) is chosen as a seed. Next, the $NP/p-1$ individuals nearest to $\overrightarrow{x}_{best}$ are identified and combined with $\overrightarrow{x}_{best}$ to form a cluster. After the cluster is formed, the individuals that belong to it will be removed from $P^g$. As an analogy, the construction of other $p-1$ clusters is the same. The procedure of SC is given in Algorithm 3.

Algorithm 3 Speciation Clustering (SC) [45]

1: INPUT: $P^g$ and $p$;
2: OUTPUT: the clusters $C_1, C_2, \ldots, C_p$;
3: Sort all the individuals in $P^g$ based on their fitness values;
4: While $P^g \neq \emptyset$ Do
5:   Choose the best individual $\overrightarrow{x}_{best} \in P^g$ as a seed;
6:   Select $NP/p-1$ individuals nearest to $\overrightarrow{x}_{best}$;
7:   Combine the selected individuals with $\overrightarrow{x}_{best}$ to form a cluster;
8:   Remove these $NP/p$ from $P^g$;
9: End While

- K-means clustering (KC) [46]
  In KC, $p$ individuals are randomly selected from the current population as the initial centroids of the $p$ clusters. Then, each individual is assigned to the cluster (centroid) that is closer than other clusters. After that, the centroids of the clusters are recalculated by taking the average of all the individuals of that cluster. The above steps will be repeated until the assignment of all the individuals to the clusters does not change anymore. Finally, the $p$ clusters have been constructed. The procedure of KC is shown in Algorithm 4.

Algorithm 4 K-Means Clustering (KC) [46]

1: INPUT: $P^g$ and $p$;
2: OUTPUT: the clusters $C_1, C_2, \ldots, C_p$;
3: Randomly select $p$ individuals as the centroids of the clusters;
4: While the termination criterion is not met Do
5:   Assign each individual to the cluster with the closest centroid;
6:   Recalculate the centroid of each cluster by taking the mean of all the individuals of that cluster;
7: End While

Once the $p$ clusters have been constructed by a $DIV$, the $LG$ will be built as follows:

$$LG = \{ \overrightarrow{l}_i \mid \overrightarrow{l}_i \text{ is the best individual in } C_i, 1 \leq i \leq p \}$$ (10)

From Eq. (10), it can be found that the promising leaders in $LG$ is detected by considering both their fitness information in the objective space and their distribution information in the decision space. In this way, the problem of misguidance with the clustered leaders will be alleviated, and thus the guidance of the population towards the promising search regions could be more effective.

3) CONSTRUCTING THE NEIGHBORHOOD FOR EACH LEADER

For the third problem, based on the partition of current population by using a $DIV$, the members of the cluster that the leader belongs to will be treated as its neighbors directly. Therefore, the neighborhood of $\overrightarrow{l}_i$ is constructed as follows:

$$N_i = C_i - \{ \overrightarrow{l}_i \}$$ (11)

4) SUMMARIZING THE PROCEDURE OF ALD

By solving the above three problems, the procedure of ALD is summarized in Algorithm 5, where we can see that ALD includes four key steps. To be specific, by using an adaptive leaders sizing strategy, the first step of ALD is to determine
the value of $p$ related to the number of the current generation $g$. Next, the $p$ clusters are obtained by applying a $DIV$ method (i.e., CC, SC, or KC) to the current population. At the third step, the $LG$ is built with $p$ promising leaders, each of which is selected from an obtained cluster based on its fitness value. Finally, the neighborhood of each leader is constructed with the individuals of the same cluster.

C. NEIGHBORHOOD-BASED TOURNAMENT SELECTION (NTS)

To effectively utilize evolutionary information of the failed trial vectors for the local exploitation, the following two issues will be addressed by NTS.

1) How to detect the individuals of the current population being stagnant?
2) How to perform the tournament selection among the failed trial vectors and the detected stagnant individuals?

1) DETECTING THE STAGNANT INDIVIDUALS

As expressed in [47], stagnation is used to describe the phenomenon that the population cannot generate any better solutions. It means that all the individuals of a population might get trapped in the locally optimal regions and cannot be updated with improved solutions. Therefore, in this study, the individual being located in an unpromising or locally optimal area will be referred to as a stagnant individual.

To detect the stagnant individuals of the current population, the following rule is used [47]:

\[
S_i = \begin{cases} 
1, & \text{if } SN_i \geq TH \\
0, & \text{otherwise} 
\end{cases} 
\]  

where $S_i$ is an indicator of the search state of $x_i$. $S_i = 1$ means $x_i$ is in stagnation. $SN_i$ is used to record the number of consecutive generations that $x_i$ cannot be updated successfully. If $x_i$ is replaced by its trial vector, $SN_i = 0$; otherwise, $SN_i = SN_i + 1$. $TH$ is a pre-defined threshold for detecting stagnation.

2) TOURNAMENT SELECTION BASED ON THE NEIGHBORHOOD OF EACH LEADER

As discussed above, the trial vectors generated during the iterations can provide the useful feedback information about the local fitness landscape of the optimization problem. Based on this consideration, the tournament selection is executed based on the neighborhood of each detected leader in $LG$.

In each generation, each individual in $\{l_i\} \cup N_i$ is updated first via the DE operators, and the trial vector lost in comparison with its parent vector will be stored in $Q_i$.

After that, if $Q_i \neq \emptyset$, the tournament selection will be executed among the failed trial vectors in $Q_i$ and the stagnant individuals in $N_i$. Specifically, the tournament selection with regard to $l_i$ is given as follows:

\[
x_k^{g+1} = \begin{cases} 
\overrightarrow{u_q}, & \text{if } \overrightarrow{x_k} \in N_i \land S_k = 1 \land \overrightarrow{u_q} \in Q_i \\
\land f(\overrightarrow{u_q}) < f(\overrightarrow{x_k}) \\
\overrightarrow{x_k}, & \text{otherwise}.
\end{cases} 
\]  

As presented in Eq. (13), each failed trial vector in $Q_i$ competes against all the stagnant individuals in $N_i$, and it will replace the competitor that is worse than it. In this study, to keep the diversity of the population, each failed trial vector will only replace the worse competitor that it first encounters.

3) SUMMARIZING THE PROCEDURE OF NTS

With the above two operators, the procedure of NTS is given in Algorithm 6. From Algorithm 6, NTS is performed to update the population for the next generation based on the neighborhood of each leader in $LG$. Concretely, for each leader $l_i \in LG$, NTS first checks the search state ($S_i$) of each neighbor within its neighborhood $N_i$. If a neighbor cannot be improved for $TH$ consecutive generations, it will be identified as a stagnant individual, and $S_i$ will be set to 1. Otherwise, $S_i$ will be still set to 0. Next, the stagnant neighbors are detected according to their $S_i$ values. After that, the tournament selection is carried out among the failed trial vectors in $Q_i$ and the detected stagnant individuals in $N_i$. If a failed trial vector is better than a stagnant neighbor, it will replace the stagnant neighbor and be removed from $Q_i$ immediately. Finally, when the neighborhood of each leader has been updated by the tournament selection, the new population for the next generation will be obtained by combining $LG$ and the updated neighborhood of each leader.

Algorithm 5 Adaptive Leaders Detection (ALD)

1: INPUT: the number of current iteration $g$, $P^g$, the used $DIV$;
2: OUTPUT: $LG$ and $N_i, i = 1, 2, \ldots, p$;
3: Calculate $p$ using Eq. (9);
4: \{ $C_1, C_2, \ldots, C_p$ \} \leftarrow $DIV$(P^g, $p$);
5: Build $LG$ using Eq. (10);
6: Construct $N_1, N_2, \ldots, N_p$ using Eq. (11).

Algorithm 6 Neighborhood-Based Tournament Selection (NTS)

1: INPUT: $LG$, $N_i$ and $Q_i$, $i = 1, 2, \ldots, p$;
2: OUTPUT: the population for the next generation $P^{g+1}$;
3: FOR each leader $l_i \in LG$ DO
4: \hspace{1cm} Check the search states of individuals in $N_i$ by Eq. (12);
5: \hspace{1cm} Detect the stagnant individuals in $N_i$;
6: \hspace{1cm} Replace the stagnant individuals with $Q_i$ by Eq. (13);
7: END FOR
8: \hspace{1cm} Obtain the population $P^{g+1}$ for the next generation by combining $N_1, N_2, \ldots, N_p$, and $LG$. 

226564
D. APPLYING DELDG TO JADE AND SHADE

As presented in Algorithm 1 and Figure 1, DELDG has a simple structure and can be applied to most DE variants easily. When implementing a specific DE algorithm under the DELDG framework, the only thing to consider is how to generate the offspring with both the evolutionary operators and the guidance of LG. In this article, DELDG is applied to two important and representative DE variants, JADE [9] and SHADE [12]. The DELDG variants with JADE and SHADE are termed JADE-LDG and SHADE-LDG, respectively.

In JADE-LDG, an improved mutation strategy, named DE/current-to-LG, is designed under the guidance of LG, which is shown as follows:

\[
\vec{v}_i^g = \vec{x}_i^g + F_i \times (\vec{l}_r - \vec{x}_i^g) + F_i \times (\vec{x}_{r1}^g - \vec{x}_{r2}^g) \quad (14)
\]

where \(\vec{l}_r\) is a leader that is randomly selected from LG, and \(\vec{x}_{r1}^g\) and \(\vec{x}_{r2}^g\) are the same as those in the original JADE.

From Eq. (14), it is clear that the generation of the mutant vector in JADE-LDG is guided by the detected promising leaders in LG, instead of the top best solutions in the original JADE. Additionally, setting the size of LG is performed by an adaptive leaders sizing strategy (described in 4.3.1) in JADE-LDG, rather than using a fixed value to control the greediness of the mutation strategy in the original JADE.

Compared with JADE-LDG, SHADE-LDG is only different in the parameter adaptation scheme. Specifically, JADE-LDG generates the control parameters based on the distribution around a single pair of \(\mu_{Cr}\) and \(\mu_F\), while SHADE-LDG uses a historical memory to store a set of \(\mu_{Cr}\) and \(\mu_F\) pairs and then samples the control parameters based on the distribution around a randomly selected pair of \(\mu_{Cr}\) and \(\mu_F\). The details of the parameter adaptation scheme in JADE-LDG and SHADE-LDG can be referred to [9] and [12], respectively.

Algorithm 7 depicts the offspring generation operators in JADE-LDG/SHADE-LDG. For each \(\vec{l}_r \in LG\), the mutation strategy (MO) with DE/current-to-LG, the crossover operator (XO), and the selection operator (SO) will be applied sequentially to \(\vec{x}_j^g \in N_i \cup \{l_i\}\). Additionally, as shown in Steps 8 - 14 of Algorithm 7, \(\vec{x}_j^g\) will be updated if its trial vector performs better than it. Otherwise, the trial vector lost in comparison with its parent will be stored in \(Q_i\).

In summary, by integrating the three main components (i.e., ALD in Algorithm 5, NTS in Algorithm 6, and offspring generation operators in Algorithm 7) into the general framework of DELDG, the complete procedure of JADE-LDG/SHADE-LDG is built. Obviously, the main differences between JADE-LDG/SHADE-LDG and the original JADE/SHADE are the use of ALD to guide the search of mutation operator and the employ of NTS to promote local exploitation.

In addition, to show how DELDG works clearly, the detailed steps of JADE-LDG are described below:

**Algorithm 7 Offspring Generation Operators of JADE-LDG/SHADE-LDG**

1. INPUT: \(N_1, N_2, \ldots, N_p\), and LG;
2. OUTPUT: the updated LG, \(N_i\) and \(Q_i\), \(i = 1, 2, \ldots, p\);
3. For \(\vec{x}_j^g \in N_i \cup \{l_i\}\) Do
4. Generate \(F_j\) and \(Cr_j\) as those in JADE/SHADE;
5. Select \(\vec{l}_r\) from LG \(\{\vec{x}_j^g\}\) randomly;
6. Use MO with DE/current-to-LG and \(\vec{l}_r\) to generate \(\vec{v}_j^g\);
7. Use XO to generate \(\vec{u}_j^g\);
8. Use SO to select a better vector between \(\vec{u}_j^g\) and \(\vec{x}_j^g\);
9. If \(f(\vec{u}_j^g) \leq f(\vec{x}_j^g)\) Then
10. \(SN_j = 0\);
11. Else
12. \(SN_j = SN_j + 1\);
13. \(Q_i = Q_i \cup \{\vec{u}_j^g\}\);
14. End If
15. End For
16. Update the archive set, \(\mu_{F}\), and \(\mu_{Cr}\) as those in JADE/SHADE.

**Step 1:** Initialize the population \(P^0\) with \(NP\) individuals by Eq. (1), evaluate the fitness values of them, and set \(SN_j = 0\), \(j = 1, \ldots, NP\).

**Step 2:** Calculate the size of LG \((p)\) by Eq. (9).

**Step 3:** Apply a division method (e.g., CC in Algorithm 2, SC in Algorithm 3, or KC in Algorithm 4) to partition the whole population into \(p\) clusters \(C_1, \ldots, C_p\).

**Step 4:** Build the LG with the best individual in each cluster by Eq. (10).

**Step 5:** Construct the neighborhood \(N_i\) for each leader \(l_i \in LG\) by Eq. (11), and set \(Q_i = \emptyset\), \(i = 1, \ldots, p\).

**Step 6:** Generate the control parameters \(F_j\) and \(Cr_j\) for each individual \(\vec{x}_j^g \in N_i \cup \{l_i\}\) as those in JADE.

**Step 7:** Select a leader \(l_i\) from LG \(\{\vec{x}_j^g\}\) randomly, and produce a mutant vector \(\vec{v}_j^g\) with DE/current-to-LG for each individual \(\vec{x}_j^g\) by Eq. (14).

**Step 8:** Apply the crossover operator to \(\vec{x}_j^g\) and \(\vec{v}_j^g\) to produce a trial vector \(\vec{u}_j^g\) by Eq. (7).

**Step 9:** Apply the selection operator to \(\vec{x}_j^g\) and \(\vec{u}_j^g\) to determine the individual for the next generation by Eq. (8).

**Step 10:** Set \(SN_j = 0\) if \(f(\vec{u}_j^g) \leq f(\vec{x}_j^g)\); otherwise, set \(SN_j = SN_j + 1\) and \(Q_i = Q_i \cup \{\vec{u}_j^g\}\).

**Step 11:** Return to Step 6 until all the individuals of the population have been considered.

**Step 12:** Update the archive set, \(\mu_{F}\), and \(\mu_{Cr}\) as those in JADE.
Step 13: Check the search state of each individual in $N_i$ by Eq. (12), $i = 1, \ldots, p$.

Step 14: Detect the stagnated individuals in $N_i$ and replace them with $Q_i$ by Eq. (13).

Step 15: Establish the population for next generation $p_{n+1}$ by combining $N_1, \ldots, N_p$ and LG.

Step 16: Go to Step 2 for the next iteration if the termination criterion is not satisfied; Otherwise, output the best individual of the current population and terminate the process of the algorithm.

E. REMARKS

1) COMPLEXITY ANALYSIS

Take JADE-LDG as an example, the additional computational cost is caused by ALD and NTS. Specifically, for ALD in Algorithm 5, the complexity of setting the value of $p$ in line 3 is $O(1)$. In line 4, the complexity of $\text{DIV}$ operator (e.g., CC in Algorithm 2) is $O(NP^2 \times D)$. In lines 5 and 6, the complexities of building LG and constructing neighborhood for each leader are both $O(NP)$. Therefore, the total complexity of ALD is $O(NP^2 \times D)$. For NTS in Algorithm 6, the complexity of checking the search states of the neighbors in line 4 is $O(NP/p)$. In line 5, the complexity of detecting the stagnated individuals is $O(NP/p)$, while the complexity of replacing the individuals with the failed trial vectors in line 6 is $O(NP/p)^2 \times D)$. Finally, in line 8, the complexity of obtaining the new population is $O(NP \times D)$. Hence, the total complexity of NTS is $O(NP^2 \times D/p)$. Besides, the complexity of the offspring generation operators in Algorithm 7 is $O(NP \times D)$, which is the same as the original JADE. In summary, the overall complexity of JADE-DLG is $O(G_{\text{max}} \times NP^2 \times D)$.

2) COMPARISONS BETWEEN DELDG AND THE EXISTING DE VARIANTS

On the one hand, compared with the existing DE variants, the proposed DELDG is characterized by its leaders-detection-and-guidance (LDG) mechanism. As analyzed in Section III, two important issues, i.e., guiding the search with promising leaders and reusing the failed trial vectors, are still not well solved in most DE variants. Thus, the LDG mechanism with ALD and NTS is proposed to deal with these two issues. The former is used to detect multiple promising solutions by considering their information in both the objective and decision spaces, while the latter is employed to update the neighborhood of each leader with the failed trial vectors by the tournament selection. With these two components, the LDG mechanism is expected to effectively alleviate the above issues existed in the previous DE variants.

On the other hand, similar to DELDG, the DE variants for multi-modal optimization usually divide the population into several subpopulations (or species) to locate multiple optimal solutions [45], [48]. Although the multi-modal DE variants employ a similar division technique to DELDG, there are significant differences between them: (1) The division operator in the multi-modal DE variants is aimed to maintain the diversity of the population and thus the multiple potential optimal solutions can be found and preserved during the optimization process [48]. Reversely, in the proposed DELDG, the division operator is employed to detect multiple leaders for guiding the search towards the promising areas. In addition, the number of leaders is dynamically adjusted by an adaptive leaders sizing strategy. (2) The selection operator in most multi-modal DE variants is performed by comparing the trial vector with a similar individual of the current population based on the Euclidean distance. Further, the trial vector will be directly discarded if it performs worse than its competitor. On the contrary, the trial vector lost in the comparison with its parent will be archived and used to replace the stagnant individuals of the same neighborhood if it is better than the competitor.

V. EXPERIMENTAL STUDY

In this section, a suite of benchmark functions from the CEC2013 special session on real-parameter optimization [49] is used to evaluate the performance of the proposed DELDG framework. The CEC2013 test set consists of 28 functions, which can be classified into the three categories based on their properties: unimodal functions ($F_1 - F_5$), basic multimodal functions ($F_6 - F_{20}$), and composition functions ($F_{21} - F_{28}$). More details can be found in [49].

In the experiments, all the test functions are run 30 times independently for each compared algorithm. The $\text{Max}_{FE}$ for each function is set to $10^4 \times D$ as the termination criterion of a run. At the end of a run, the average value and the standard deviation of the best error got by the algorithm are recorded for comparison. In addition, the parameters for the compared DE algorithms and their augmented DELDG are set as follows if no change is mentioned: $NP = 100$, $TH = 50$, and $t = 10$. Other parameters of them under the DELDG framework are kept the same as those of their original papers.

To test the statistical significance between DELDG and the compared algorithms, single- and multiple-problem analyses by the Wilcoxon’s signed-rank test [50], [51] are executed. Here, in the single-problem analysis at $\alpha = 0.05$, “$+/1$” denotes that the considered algorithm is significantly better than, similar to and worse than its competitor on the corresponding number of functions, respectively [50]. In the multiple-problem analysis, $R+$ and $R-$ mean the sum of ranks that the considered algorithm performs significantly better than and worse than its competitor for all the test functions, respectively, and “$+$”, “$=$” and “$-$” mean the considered algorithm is better than, equal to, and worse than the compared algorithm overall at the corresponding significance level, respectively [50]. In this article, only the statistical results of the comparisons are given for brevity, and the detailed results with errors and standard deviations are shown in the supplementary file.

Based on the purpose of this study, the experiments are carried out with the following points:
TABLE 1. Results of single-problem analysis by the Wilcoxon test between the DELDG variants with different division methods for the CEC2013 functions at 30D.

| Algorithm     | ARV | Ranking | Algorithm     | ARV | Ranking |
|---------------|-----|---------|---------------|-----|---------|
| JADE-LDG-CC   | 4/24/0 |   | SHADE-LDG-CC  | 7/21/0 |   |
| JADE-LDG-SC   | -   | 6/22/0 | SHADE-LDG-SC  | -   | 9/19/0 |
| JADE-LDG-KC   | -   | -      | SHADE-LDG-KC  | -   | -      |
| JADE          | 23/9/1 |   | SHADE         | 5/17/2 |   |
| SHADE-LDG-SC  | 11/17/0 |   | JADE-LDG-KC   | 10/14/4 |   |
| SHADE-LDG-KC  | 12/13/3 |   | JADE          | 12/13/3 |   |

TABLE 2. Average Ranking values (ARV) of the DELDG variants with different division methods by the Friedman test.

| Algorithm     | ARV | Ranking | Algorithm     | ARV | Ranking |
|---------------|-----|---------|---------------|-----|---------|
| JADE-LDG-CC   | 1.77 | 1       | SHADE-LDG-CC  | 1.80 | 1       |
| JADE-LDG-SC   | 2.50 | 2       | SHADE-LDG-SC  | 2.16 | 2       |
| JADE-LDG-KC   | 2.79 | 3       | SHADE-LDG-KC  | 2.91 | 3       |
| JADE          | 2.95 | 4       | SHADE         | 3.13 | 4       |

1) Three different division methods (i.e., CC in Algorithm 2, SC in Algorithm 3, and KC in Algorithm 4) are used in DELDG respectively to analyze their effect on the performance of DELDG, which is shown in Section V-A.

2) The effectiveness of the two proposed components (i.e., ALD and NTS) is investigated to show their contributions to the performance of DELDG, which is given in Section V-B.

3) The sensitivity of the parameter settings is studied to demonstrate the effect of these parameters on the performance of DELDG, which is presented in Section V-C.

4) DELDG is applied to four JADE-based variants to verify the effectiveness of the proposed framework, which is shown in Section V-D.

5) The comparisons between DELDG and other advanced DE variants, as well as the state-of-the-art EAs, are made to display the competitive performance of the proposed framework, which are given in Section V-E to Section V-G.

6) DELDG is applied to the real-world numerical optimization problems from the CEC2011 competition [52] to evaluate its performance on the real-world problems, which is presented in Section V-H.

In the following subsections, JADE-LDG and SHADE-LDG will be used as two representative DELDG variants for comparison.

A. COMPARISON OF DIFFERENT DIVISION METHODS

To investigate the influence of the division method in ALD on the performance of DELDG, the DELDG variants with different division methods, i.e., DELDG with CC (DELDG-CC), DELDG with SC (DELDG-SC), DELDG with KC (DELDG-KC), are considered. The statistical comparison results between them are summarized in Tables 1 and 2.

As can be seen from Table 1, all the DELDG variants are capable to significantly enhance the performance of the original DE on the majority of the test functions. To be specific, when compared with the original JADE algorithm, JADE-LDG-CC, JADE-LDG-SC, and JADE-LDG-KC obtain the significantly better results on 13, 12, and 10 functions, respectively, and is outperformed by it on 2, 3, and 4 functions, respectively. When compared with the original SHADE algorithm, SHADE-LDG with CC, SC, and KC are significantly better than it on 16, 14, and 12 functions, respectively, and is worse than it on no more than 4 functions. Also, based on the Friedman test [50] in Table 2, DELDG with CC gets the first ranking for both cases, followed by that with SC and KC. The original JADE and SHADE algorithms are ranked last in their respective comparisons.

From the above results, we can draw some conclusions: 1) DELDG with CC, SC, or KC can greatly improve the performance of the considered DE algorithm, which verifies the effectiveness of different division methods in constructing LG under the proposed framework, and 2) DELDG with CC achieves the best results in terms of the average ranking among the three variants. Compared with DELDG with SC and KC, the better performance of DELDG-CC is largely attributed to the advantage of CC in finding and maintaining the detected promising leaders [45]. In the following sections, only CC is used as the division method in DELDG, and DELDG-CC will be referred to as DELDG for the sake of simplicity.

B. EFFECTIVENESS OF THE PROPOSED COMPONENTS

To study the effectiveness of the proposed two components (i.e., ALD and NTS), the following variants of DELDG are used for comparison.
C. Sensitivity Analysis of Parameters

To investigate the sensitivity of $TH$ for detecting the stagnated individuals and $t$ for setting the $LG$ size, the comparisons among the DELDG variants with different parameters are made. Here, three functions with different characteristics are selected from the CEC2013 test set, i.e., the unimodal function $F4$, the basic multimodal function $F13$, and the composition function $F25$.

1) Analysis of $TH$

For the parameter $TH$, seven different values, i.e., $TH \in \{10, 20, 30, 40, 50, 60, 70\}$, are used to study its influence on the performance of DELDG, while keeping $t = 10$. The convergence curves of JADE-LDG and SHADE-LDG with different $TH$ values on the three functions are illustrated in Figures 2 and 3.

From Figures 2 and 3, we can obtain some observations: 1) for $F13$, the DELDG variants with a larger $TH$ value (e.g., 50, 60, or 70) have a faster convergence speed than those with a smaller $TH$ value; 2) for $F13$, the DELDG variants with different $TH$ values get the similar results; 3) for $F25$ at 30D, the DELDG variants with a smaller $TH$ value (e.g., 10, 20, or 30) can obtain the better results than those with a larger $TH$ value.

Therefore, it can be concluded that the performance of DELDG is not sensitive to the setting of $TH$ in most cases. For the unimodal functions, DELDG with a larger $TH$ value can provide more robust results, while, for the composition functions, DELDG with a smaller $TH$ value might achieve the better performance. In the future, the adaptive or self-adaptive parameter control techniques will be studied to select the appropriate value of $TH$ for different types of functions.

2) Analysis of $t$

For the parameter $t$, five different values are considered in this experiment to evaluate its effect on DELDG, while $TH$ is set to 50. These $t$ values are $t \in \{2, 5, 10, 15, 20\}$. The convergence curves of JADE-LDG and SHADE-LDG with different $t$ values are shown in Figures 4 and 5.

As shown in Figures 4 and 5, the DELDG variant with a larger $t$ value is capable to obtain the slightly better results in
most cases. Specifically, for $F_4$, most DELDE variants with $t = 15$ perform better than those with the smaller $t$ values. For $F_{13}$, the DELDG variant with $t = 10$ obtains the best results overall when compared against those with other $t$ values. For $F_{25}$ at 30D, JADE-LDG with $t = 5$ performs best, while SHADE-LDG with $t = 10$ obtains the best results. For $F_{25}$ at 50D, both JADE-LDG and SHADE-LDG can achieve the best results when $t = 20$.

Overall, DELDG with $t = 10$ or 15 values can achieve a better performance for the considered unimodal and basic
multimodal functions. For the composition functions, a larger $t$ value is recommended for DELDG.

### D. DELDG FOR FOUR JADE-BASED VARIANTS

To verify the effectiveness of the proposed framework, DELDG is applied to four representative JADE-based variants. These variants are JADE, SHADE, L-SHADE, and jSO. The comparisons between the JADE-based variants and their augmented DELDG are made, and the statistical results are summarized in Table 4.

Some findings can be obtained from Table 4:

- In the case of $D = 30$, DELDG obtains better results than the corresponding JADE-based variants overall. Concretely, JADE-LDG, SHADE-LDG,
TABLE 4. Results of single- and multiple-problem analysis by the Wilcoxon test between JADE-based variants and their augmented DELDG for the CEC2013 functions.

| Algorithm at 30D | w/t/l | R+   | R−   | p-value | α = 0.05 | α = 0.1 |
|-----------------|-------|------|------|---------|--------|--------|
| JADE-LDG vs. JADE | 13/12/2 | 299.5 | 78.5 | 7.66E-03 | +      | +      |
| SHADE-LDG vs. SHADE | 16/12/0 | 330   | 76   | 3.69E-03 | +      | +      |
| L-SHADE-LDG vs. L-SHADE | 6/18/4 | 208.5 | 169.5 | 6.30E-01 | =      | =      |
| jSO-LDG vs. jSO | 10/11/7 | 221   | 157  | 4.34E-01 | =      | =      |

Algorithm at 50D
| Algorithm at 50D | w/t/l | R+   | R−   | p-value | α = 0.05 | α = 0.1 |
|-----------------|-------|------|------|---------|--------|--------|
| JADE-LDG vs. JADE | 16/12/0 | 342.5 | 63.5 | 1.43E-03 | +      | +      |
| SHADE-LDG vs. SHADE | 13/15/0 | 304   | 74   | 5.52E-03 | +      | +      |
| L-SHADE-LDG vs. L-SHADE | 12/13/3 | 229.5 | 148.5 | 3.25E-01 | =      | =      |
| jSO-LDG vs. jSO | 10/14/4 | 229.5 | 148.5 | 3.25E-01 | =      | =      |

L-SHADE-LDG, and jSO-LDG are significantly better than the corresponding competitors on 13, 16, 6, and 10 functions, respectively.

- In the case of D = 50, DELDG can achieve the consistently better results in comparisons with the original versions. Specifically, DELDG significantly outperforms the corresponding variants on 51 (= 16 + 13 + 12 + 10) functions, while it is worse than them on 7 (= 0 + 0 + 3 + 4) functions.

- According to the multiple problem statistical analysis, DELDG obtains the higher R+ values than the R− values in all the cases. Further, the p values are less than 0.05 in two cases (i.e., JADE and SHADE).

- DELDG can achieve great performance improvement toward JADE and SHADE on most test functions. For L-SHADE and jSO, although the significant improvement cannot be brought by DELDG, the proposed framework is able to obtain the better results overall.

From the above results, we can conclude that the proposed DELDG framework is beneficial to the performance enhancement of the JADE-based variants on the test functions. These improvements might derive from two aspects. First, the constructed LG in DELDG guides the mutation process of the JADE-based variants with the detected promising leaders, instead of the top best solutions in these variants. Second, the update of population via NTS can further take advantage of the evolutionary information of the potential trial vectors for the local exploitation.

E. COMPARISON WITH THE CEC2013 COMPETITORS

To evaluate the performance of the proposed framework, SHADE-LDG is used as the representative DELDG.
algorithm to compare with the advanced DE variants and the non-DE EAs participating in the CEC2013 competition [49].

1) COMPARISON WITH EIGHT ADVANCED DE VARIANTS
In this section, eight advanced DE variants participating in the CEC2013 competition [49] are used for comparison. These DE variants are: SMADE [53], b6e6rl [54], DEcfbLS [55], TLBSaDE [56], SPSRDEMMS [57], DE-APC [58], PV ADE [59], and jande [60]. Their results are directly got from the original papers, and the statistical results of the comparisons between SHADE-LDG and the DE variants are summarized in Tables 5 and 6.

From Table 5, SHADE-LDG exhibits its superiority to the other eight DE variants. Concretely, based on the multiple-problem statistical analysis, SHADE-LDG obtains the higher $R^+$ value than the $R^-$ values in all the cases. Besides, the $p$-value demonstrates that SHADE-LDG is significantly better than all the competitors for the functions both at 30D and 50D. Further, the results of the Friedman test in Table 6 also show the outstanding performance of SHADE-LDG among all the compared algorithms. SHADE-LDG is ranked first in terms of the average ranking values for the functions both at 30D and 50D.

Therefore, the above comparisons demonstrate that SHADE-LDG is capable to provide the significant better solutions for the test functions when compared with other advanced DE variants in the CEC2013 competition.

2) COMPARISON WITH TEN NON-DE EAs
In this section, the comparisons between SHADE-LDG and ten non-DE EAs participating in the CEC2013 competition [49] are made. These non-DE EAs consist of NBIPOP-ACMA-ES [61], iCMAES-ILS [62], MVMO [63], DRMA-LSch-CMA [64], CMAES-RIS [65], ABC-SPSO [66], fk-PSO [67], TPC-GA [68], PLES [69], and CDASA [70]. The results of these compared EAs are directly obtained from their original papers. The statistical comparison results are presented in Tables 7 and 8.

Tables 7 and 8 clearly show that SHADE-LDG can also obtain significantly better results in most cases. To be specific, first, based on the multiple-problem analysis, SHADE-LDG gets the higher $R^+$ values than the $R^-$ values in eight and seven cases for the functions at 30D and 50D, respectively. Second, the $p$ values are less than 0.05 in seven and four cases for the functions at 30D and 50D, respectively, which demonstrates the superiority of SHADE-LDG over these compared EAs. Third, according to the Friedman test, iCMAES-ILS and NBIPOP-ACMA-ES are ranked first and second for the functions both at 30D and 50D respectively, followed by SHADE-LDG as the third-best among all the compared EAs.
TABLE 9. Results of multiple-problem analysis by the Wilcoxon test between SHADE-LDG and the up-to-date DE for the CEC2013 functions at 30D and 50D.

| Algorithm at 30D | $R^+$ | $R^-$ | p-value | $\alpha = 0.05$ | $\alpha = 0.1$ |
|------------------|-------|-------|---------|----------------|----------------|
| DMPSADE          | 300   | 106   | 2.60E-02| +   | +             |
| MPEDE            | 283   | 123   | 6.60E-02| =   | +             |
| AGDE             | 339   | 67    | 1.88E-03| +   | +             |
| SHADE-LDG vs.    |       |       |         |     |               |
| DMPSADE          | 300   | 106   | 2.60E-02| +   | +             |
| MPEDE            | 283   | 123   | 6.60E-02| =   | +             |
| AGDE             | 339   | 67    | 1.88E-03| +   | +             |
| SHADE-LDG vs.    |       |       |         |     |               |
| DI-DE            | 322   | 84    | 6.35E-03| +   | +             |
| ZEPDE            | 273   | 105   | 4.23E-02| +   | +             |
| ZEPDE            | 273   | 105   | 4.23E-02| +   | +             |
| SHADE-LDG        | 273   | 105   | 4.23E-02| +   | +             |
| UMS-SHADE        | 176.5 | 201.5 | 7.52E-01| =   | =             |
| EBSHADE          | 194   | 184   | 8.95E-01| =   | =             |
| FADE             | 267.5 | 110.5 | 5.77E-02| =   | =             |

Based on the above observations, some conclusions can be drawn: 1) SHADE-LDG has demonstrated its outstanding performance when compared with these state-of-the-art EAs from the CEC2013 competition; 2) although iCMAES-ILS and NBiPOP-ACMA-ES are slightly better than SHADE-LDG, no significant differences can be observed between them; 3) compared with the complicated structures and sophisticated operators in iCMAES-ILS and NBiPOP-ACMA-ES, SHADE-LDG has shown its advantages in simple structure and is easy to implement; 4) the effectiveness of the proposed framework has been verified in comparison with these state-of-the-art EAs, which indicates that DELDG is a competitive algorithm for the test functions.

F. COMPARISON WITH UP-TO-DATE DE VARIANTS

To further study the effectiveness of DELDG, the comparisons of DELDG with 11 up-to-date DE variants are made. These DE variants include ZEPDE [16], DI-DE [18], IIN-SHADE [21], EFADE [22], AGDE [23], UMS-SHADE [27], MPEDE [28], DMPSADE [71], CoBiDE [72], EBSHADE [73], and FADE [30]. ZEPDE, DI-DE, IIN-SHADE, EFADE, AGDE, UMS-SHADE, MPEDE, and FADE have been briefly introduced in Section III. DMPSADE is a self-adaptive DE variant with discrete mutation control parameters where each individual has its control parameter and mutation strategy [71]. CoBiDE is a novel DE variant based on covariance matrix learning and bimodal distribution parameter setting are used [72]. EBSHADE is a DE framework by hybridizing the mutation strategy of SHADE and a new mutation strategy “DE/current-to-ord_pbest/1” [73]. These DE variants are selected for comparison due to their outstanding performance on the CEC2013 test functions. Note that three recently proposed DE frameworks with SHADE (i.e., IIN-SHADE, UMS-SHADE, and EBSHADE) are also included to show the advantages of DELDG over other excellent DE frameworks. The results of these DE variants are straightly obtained from their published papers. Tables 9 and 10 summarize the statistical comparison results between SHADE-LDG and the compared DE variants.

As shown in Tables 9 and 10, SHADE-LDG is capable to achieve the significantly better results than most compared DE variants. To be specific,

- For the functions at 30D, SHADE-LDG obtains the higher $R^+$ values than the $R^-$ values in all the cases according to the multiple-problem statistical analysis.

TABLE 10. Average ranking values of SHADE-LDG and the up-to-date DE for the CEC2013 functions by the Friedman test.

| Algorithm       | 30D | 50D | Mean | Ranking |
|-----------------|-----|-----|------|---------|
| SHADE-LDG       | 3.98| 4.59| 4.29  | 1       |
| UMS-SHADE       | 4.34| 4.41| 4.38  | 2       |
| EBSHADE         | 4.80| 4.77| 4.79  | 3       |
| DI-DE           | 5.93| 5.63| 5.78  | 4       |
| MPEDE           | 5.95| 6.46| 6.21  | 5       |
| ZEPDE           | 5.73| 7.04| 6.39  | 6       |
| SHADE-LDG       | 6.73| 6.56| 6.65  | 7       |
| CoBiDE          | 7.18| 7.17| 7.18  | 8       |
| DMPSADE         | 8.18| 6.95| 7.57  | 9       |
| EBSHADE         | 8.26| 7.14| 7.70  | 10      |
| FADE            | 8.41| 8.23| 8.32  | 11      |
| AGDE            | 8.50| 9.05| 8.78  | 12      |
Additionally, the $p$ values demonstrate that the significant differences can be observed between SHADE-LDG and the compared DE variants in eight and nine cases at $\alpha = 0.05$ and $\alpha = 0.1$, respectively. In these cases, SHADE-LDG significantly outperforms the compared DE variants overall.

- For the functions at 50D, SHADE-LDG exhibits a consistent good performance when compared with these DE variants. The results of the multiple-problem statistical analysis also indicate the superior performance of SHADE-LDG in most cases.
- Compared against the three DE frameworks with SHADE (i.e., IIN-SHADE, UMS-SHADE, and EB SHADE), SHADE-LDG can obtain better results in most cases except UMS-SHADE for the functions at 50D. UMS-SHADE can get slightly better results than SHADE-LDG for the functions at 50D, but no significant differences can be observed between them.
- According to the Friedman test, SHADE-LDG is ranked first for the functions at 30D, followed by UMS-SHADE and EB SHADE. For the functions at 50D, UMS-SHADE is ranked first, followed by SHADE-LDG. Overall, the mean aggregated ranks across all the functions and all the dimensions show that SHADE-LDG is the best among all the compared DE variants.

From the above discussions, the superior and competitive performance of SHADE-LDG has been demonstrated by comparing with these up-to-date DE variants, which further confirms the effectiveness of the proposed framework.

### G. OVERALL COMPARISON

From the results in Sections E and F, the effectiveness of DELDG has been clearly demonstrated when compared with the 18 CEC2013 competitors and 11 up-to-date DE variants on the CEC2013 test functions. To further show the advantages of the proposed framework overall, the average ranking values of all the competitors by Friedman test are shown in Figure 6.

As depicted in Figure 6, some interesting observations can be obtained:

- Among all the 30 competitors, the proposed SHADE-LDG algorithm can achieve the better results overall than most of the competitors. Specifically, SHADE-LDG obtains the second and fourth ranks for the functions at 30D and 50D, respectively.
- Compared with the 18 EAs from the CEC2013 competition, SHADE-LDG is only outperformed by iCMAES-ILS for the functions at 30D and is outperformed by iCMAES-ILS, NBPOP-ACMA-ES and UMS-SHADE for the functions at 50D.
TABLE 11. Descriptions of the real-world problems from the CEC 2011 competition on testing evolutionary algorithms on real-world optimization problems.

| Real-world Problems |
|----------------------|
| P1 Parameter Estimation for Frequency Modulated (FM) Sound Waves |
| P2 Lennard-Jones Potential Problem |
| P3 Optimal Control of a Non-Linear Stirred Tank Reactor |
| P4 Spread Spectrum Radar Polly phase Code Design |
| P5 Transmission Network Expansion Planning (TNEP) Problem |
| P6 Large Scale Transmission Pricing Problem |
| P7 Circular Antenna Array Design Problem |
| P8 The ELD Problems DED Instance 1 |
| P9 The ELD Problems DED Instance 2 |
| P10 The ELD Problems ELD Instance 1 |
| P11 The ELD Problems ELD Instance 2 |
| P12 The ELD Problems ELD Instance 3 |
| P13 The ELD Problems ELD Instance 4 |
| P14 The ELD Problems ELD Instance 5 |
| P15 The ELD Problems Hydrothermal Scheduling Instance 1 |
| P16 The ELD Problems Hydrothermal Scheduling Instance 2 |
| P17 The ELD Problems Hydrothermal Scheduling Instance 3 |

TABLE 12. Results of multiple-problem analyses by the Wilcoxon test between SHADE-LDG and the state-of-the-art EAs for the real-world problems from CEC2011.

| Problems | $R^+$ | $R^-$ | $p$-value | $\alpha = 0.05$ | $\alpha = 0.1$ |
|----------|-------|-------|------------|----------------|----------------|
| GA-MPC   | 75    | 61    | 6.98E-01   | =              | =              |
| SAMODE   | 105   | 31    | 5.23E-02   | =              | +              |
| ENSML-DE | 92.5  | 60.5  | 4.29E-01   | =              | =              |
| EA-DE-MA | 118   | 18    | 8.54E-03   | +              | +              |
| Adap.DE171 | 69   | 67    | 9.38E-01   | =              | =              |
| ED-DE    | 59    | 77    | 6.21E-01   | =              | =              |
| SHADE-LDG vs. OXCoDE | 110  | 26    | 2.80E-02   | +              | +              |
| DE-RHC   | 120   | 16    | 6.63E-03   | +              | +              |
| RGA      | 118   | 18    | 9.02E-03   | +              | +              |
| CDASA    | 110   | 26    | 2.80E-02   | +              | +              |
| mSBX-GA  | 120   | 16    | 6.29E-03   | +              | +              |
| DE-$\wedge_{Cr}$ | 22  | 114   | 1.62E-02   | -              | -              |
| WI-DE    | 106.5 | 29.5  | 4.24E-02   | +              | +              |
| Mod-DE-LS | 116  | 20    | 1.21E-02   | +              | +              |
| SHADE    | 148   | 5     | 6.07E-04   | +              | +              |

- Compared with the 11 up-to-date DE variants, SHADE-LDG can obtain the better ranking values than most of them for the functions both at 30D and 50D.
- Among all the DE-based algorithms in this comparison, SHADE-LDG can achieve the best results overall.

H. APPLICATION TO REAL-WORLD PROBLEMS

In this section, 17 problems from the CEC2011 competition [52] are used to evaluate the performance of SHADE-LDG on real-world problems. The brief descriptions of these problems are shown in Table 11. More details can be found in [52]. The comparisons between SHADE-LDG and 14 EAs participating in the CEC2011 competition (i.e., Adap.DE171 [74], DE-$\wedge_{Cr}$ [75], DE-RHC [76], EA-DE-MA [77], ED-DE [78], OXCoDE [79], ENSML-DE [80], SAMODE [81], Mod-DE-LS [82], WI-DE [83], CDASA [84], GA-MPC [85], mSBX-GA [86], and RGA [87]), as well as the original SHADE algorithm, are made. The results of these 14 EAs are directly got from [79]. Statistical comparison results are summarized in Tables 12 and 13.

From the results of Table 12, SHADE-LDG can achieve better performance in most cases. Specifically, according to the multiple problem statistical analysis, SHADE-LDG obtains the higher $R^+$ values than the $R^-$ values in 13 out of 15 cases, except ED-DE and DE-$\wedge_{Cr}$. Based on the $p$-value, SHADE-LDG significantly outperforms 9 EAs (i.e., SAMODE, EA-DE-MA, OXCoDE, DE-RHC, RGA, CDASA, mSBX-GA, WI-DE, and Mod-DE-LS) with the significance level of $\alpha = 0.1$. SHADE-LDG is only outperformed by DE-$\wedge_{Cr}$. Besides, when compared with the original SHADE, the augmented SHADE-LDG obtains the significant improvement in the majority of the problems.

From the results of the Friedman test in Table 13 show that SHADE-LDG obtains the better ranking than most compared algorithms. To be specific, DE-$\wedge_{Cr}$ and GA-MPC are ranked first and second, respectively, followed by SHADE-LDG as the third-best among all the compared algorithms. Note that the main purpose of this study is to propose a general framework for further improving the performance of DE with the promising leaders. Therefore, DE-$\wedge_{Cr}$, as a hybrid DE
TABLE 13. Average ranking values (ARV) of SHADE-LDG and the state-of-the-art EAs by Friedman test for the real-world problems from CEC2011.

| Algorithm       | ARV   | Rank |
|-----------------|-------|------|
| DE-ACCT         | 4.79  | 1    |
| GA-MPC          | 5.00  | 2    |
| SHADE-LDG       | 5.21  | 3    |
| SAMODE          | 6.65  | 4    |
| ID-DE           | 6.79  | 5    |
| AdapDE171       | 7.15  | 6    |
| ENSML-DE        | 7.50  | 7    |
| OXCoDE          | 7.97  | 8    |
| WI-DE           | 8.06  | 9    |
| SHADE           | 9.59  | 10   |
| Mod-DE-LS       | 10.09 | 11   |
| RGA             | 10.85 | 12   |
| CDASA           | 11.03 | 13   |
| mSBX-GA         | 11.38 | 14   |
| EA-DE-MA        | 11.56 | 15   |
| DE-RHC          | 12.38 | 16   |

variant, may be further enhanced by combining with DELDG for real-world problems, which will be studied in the future work.

From the above results, we can draw some conclusions. First, the proposed framework is capable to bring great benefits to enhancing the performance of SHADE for real-world problems. Second, the competitive and outstanding performance of SHADE-LDG has been verified by comparing it with the state-of-the-art EAs, which indicates that SHADE-LDG is an effective alternative for solving complex real-world problems.

VI. CONCLUSION AND FUTURE RESEARCH
To guide the search process of DE with multiple promising individuals, an enhanced DE framework with a leaders-detection-and-guidance mechanism, termed DELDG, is proposed for continuous optimization problems. In DELDG, two novel operators are designed and incorporated into DE. One is an adaptive leaders detection (ALD), which is used to adaptively select the promising individuals of the population to build a leadership group for guiding the mutation process. The other one is a neighborhood-based tournament selection (NTS), which is employed to enhance the local exploitation in the neighboring region of each leader with the potential trial vectors. To evaluate the effectiveness of the proposed framework, DELDG is applied to four JADE-based variants for solving a suite of 30 real-parameter functions from the IEEE CEC2013 and 17 real-world problems from the IEEE CEC 2011. Simulation results have clearly demonstrated the superior performance of DELDG when compared with the advanced DE variants and the state-of-the-art EAs participating in the CEC2013 competition, as well as 11 up-to-date DE variants. Further, for the real-world problems, the competitive performance of DELDG has been shown when compared with 14 state-of-the-art EAs. Also, the influence of different division methods, the effectiveness of the proposed components, and the sensitivity analysis of parameters have been investigated.

In the future, the present work will be extended in the following directions. First, other sophisticated division methods will be studied under the DELDE framework to show their potential in detecting the promising leaders. Second, the comparisons between DELDG and other representative DE variants on more test suites will be made to further investigate the effectiveness of DELDG. Third, DELDG will be extended to solve other optimization problems, such as multi-modal, multi-objective, or multi-tasking problems.

REFERENCES
[1] R. Storn and K. Price, “Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces,” J. Global Optim., vol. 11, no. 4, pp. 341–359, 1997.
[2] S. Das and P. N. Suganthan, “Differential evolution: A survey of the state-of-the-art,” IEEE Trans. Evol. Comput., vol. 15, no. 1, pp. 4–31, Feb. 2011.
[3] S. Das, S. S. Mullick, and P. N. Suganthan, “Recent advances in differential evolution—An updated survey,” Swarm Evol. Comput., vol. 27, pp. 1–30, Apr. 2016.
[4] M. Pant, H. Zaheer, L. Garcia-Hernandez, and A. Abraham, “Differential evolution: A review of more than two decades of research,” Eng. Appl. Artif. Intell., vol. 90, Apr. 2020, Art. no. 103479.
[5] X. Qiu, K. C. Tan, and J.-X. Xu, “Multiple exponential recombination for differential evolution,” IEEE Trans. Cybern., vol. 47, no. 4, pp. 995–1006, Apr. 2017.
[6] Y. Cai and J. Wang, “Differential evolution with hybrid linkage crossover,” Inf. Sci., vol. 320, pp. 244–287, Nov. 2015.
[7] Z.-Z. Liu, Y. Wang, S. Yang, and K. Tang, “An adaptive framework to tune the coordinate systems in nature-inspired optimization algorithms,” IEEE Trans. Cybern., vol. 49, no. 4, pp. 1403–1416, Apr. 2019.
[8] A. K. Qin, V. L. Huang, and P. N. Suganthan, “Differential evolution algorithm with strategy adaptation for global numerical optimization,” IEEE Trans. Evol. Comput., vol. 13, no. 2, pp. 398–417, Apr. 2009.
[9] J. Zhang and A. C. Sanderson, “JADE: Adaptive differential evolution with optional external archive,” IEEE Trans. Evol. Comput., vol. 13, no. 5, pp. 945–958, Oct. 2009.
[10] L. Tang, Y. Dong, and J. Liu, “Differential evolution with an individual-dependent mechanism,” IEEE Trans. Evol. Comput., vol. 19, no. 4, pp. 560–574, Aug. 2015.
[11] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, “Differential evolution using a neighborhood-based mutation operator,” IEEE Trans. Evol. Comput., vol. 13, no. 3, pp. 526–553, Jun. 2009.
[12] R. Tanabe and A. Fukunaga, “Evaluating the performance of SHADE on CEC 2013 benchmark problems,” in Proc. IEEE Congr. Evol. Comput., Jun. 2013, pp. 1952–1959.
[13] R. Tanabe and A. S. Fukunaga, “Improving the search performance of SHADE using linear population size reduction,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jul. 2014, pp. 1658–1665.
[14] J. Brest, M. S. Maucec, and B. Boskovic, “Single objective real-parameter optimization: AlgorithmJSO,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2017, pp. 1311–1318.
[15] L. Cui, G. Li, Z. Zhu, Q. Lin, K.-C. Wong, J. Chen, N. Lu, and J. Lu, “Adaptive multiple-elites-guided composite differential evolution algorithm with a shift mechanism,” Inf. Sci., vol. 422, pp. 122–143, Jan. 2018.
[16] Q. Fan and X. Yan, “Self-adaptive differential evolution algorithms with fuzzy control parameters and adaptive mutation strategies,” IEEE Trans. Cybern., vol. 46, no. 1, pp. 219–232, Jan. 2016.
[17] X.-F. Liu, Z.-H. Zhan, Y. Lin, W.-N. Chen, Y.-J. Gong, T.-L. Gu, H.-Q. Yuan, and J. Zhang, “Historical and heuristic-based adaptive differential evolution,” IEEE Trans. Syst., Man, Cybern. Syst., vol. 49, no. 12, pp. 2623–2635, Dec. 2019.
[18] L. Tian, Z. Li, and X. Yan, “Differential evolution algorithm directed by individual difference information between generations and current individual information,” Appl. Intell., vol. 49, no. 2, pp. 628–649, Feb. 2019.
[19] Z. Meng, J.-S. Pan, and K.-K. Tseng, “PaDE: An enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization,” Knowl.-Based Syst., vol. 168, pp. 80–99, Mar. 2019.
Y. Cai et al.: Enhancing DE on Continuous Optimization Problems by Detecting Promising Leaders

[20] L. Deng, H. Sun, and C. Li, "JDF-DE: A differential evolution with Jrand number decreasing mechanism and feedback guide technique for global numerical optimization," Appl. Intell., to be published, doi: 10.1007/s10458-020-01795-0.

[21] L. M. Zheng, L. Liu, S. X. Zhang, and S. Y. Zheng, "Enhancing differential evolution with interactive information," Soft Comput., vol. 22, no. 23, pp. 1791-1795, Dec. 2018.

[22] A. W. Mohamed and P. N. Suganthan, "Real-parameter unconstrained optimisation based on enhanced fitness-adaptive differential evolution algorithm with novel mutation," Soft Comput., vol. 22, no. 10, pp. 3215-3235, May 2018.

[23] A. W. Mohamed and A. K. Mohamed, "Adaptive guided differential evolution algorithm with novel mutation for numerical optimization," Int. J. Mach. Learn. Cybern., vol. 10, no. 2, pp. 253-277, Feb. 2019.

[24] A. P. Piotrowski and I. J. Napiorkowski, "Step-by-step improvement of evolutionary algorithm with novel mutation," Soft Comput., vol. 24, no. 1, pp. 3-18, Mar. 2011.

[25] S. Das and P. Suganthan, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization," Comput. Intell. Lab., Nanyang Technol. Univ., Zhengzhou Univ., Changsha, China, Tech. Rep. 201212, 2013.

[26] S. García, A. Fernández, J. Luengo, and F. Herrera, "A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability," Soft Comput., vol. 13, no. 10, pp. 959-977, Aug. 2009.

[27] J. Jervenc, S. García, D. Molina, and 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., vol. 1, no. 1, pp. 1-18, Mar. 2011.

[28] S. Das and P. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems," Jadavpur Univ., Nanyang Technol. Univ., Singapore, Tech. Rep., 2010.

[29] F. Caraffini, F. Neri, J. Cheng, G. Zhang, L. Piccinali, G. Iaccia, and E. Mininno, "Super-fit multicriteria adaptive differential evolution," in Proc. IEEE Congr. Evol. Comput., Cancun, Mexico, Jun. 2013, pp. 1678-1685.

[30] J. Tvrdik and R. Polakova, "Competitive differential evolution applied to CEC 2013 problems," in Proc. IEEE Congr. Evol. Comput., Cancun, Mexico, Jun. 2013, pp. 1651-1657.

[31] I. Poikolainen and F. Neri, "Differential evolution with concurrent fitness based local search," in Proc. IEEE Congr. Evol. Comput., Cancun, Mexico, Jun. 2013, pp. 384-391.

[32] S. Biswas, S. Kundu, S. Das, and A. V. Vasilakos, "Teaching and learning best differential evolution with self adaptation for real parameter optimization," in Proc. IEEE Congr. Evol. Comput., Cancun, Mexico, Jun. 2013, pp. 1115-1122.

[33] A. Zamuda, J. Breit, and E. Mezura-Montes, "Structured population size reduction differential evolution with multiple mutation strategies on CEC 2013 real parameter optimization," in Proc. IEEE Congr. Evol. Comput., Cancun, Mexico, Jun. 2013, pp. 1925-1931.

[34] S. M. Elsayed, R. A. Barker, and T. Ray, "Differential evolution with automatic parameter configuration for solving the CEC2013 competition on real-parameter optimization," in Proc. IEEE Congr. Evol. Comput., Cancun, Mexico, Jun. 2013, pp. 1932-1937.

[35] L. D. So Coelho, H. V. H. Ayala, and R. Z. Freire, "Population’s variance-based adaptive differential evolution for real parameter optimization," in Proc. IEEE Congr. Evol. Comput., Cancun, Mexico, Jun. 2013, pp. 1672-1677.

[36] J. Breit, B. Boskovic, A. Zamuda, I. Fister, and E. Mezura-Montes, "Real parameter single objective optimization using self-adaptive differential evolution algorithm with more strategies," in Proc. IEEE Congr. Evol. Comput., Cancun, Mexico, Jun. 2013, pp. 377-383.

[37] I. Loshchilov, "CMA-ES with restarts for solving CEC 2013 benchmark problems," in Proc. IEEE Congr. Evol. Comput., Cancun, Mexico, Jun. 2013, pp. 369-376.

[38] T. Liao and T. Stutzle, "Benchmark results for a simple hybrid algorithm on the CEC 2013 benchmark set for real-parameter optimization," in Proc. IEEE Congr. Evol. Comput., Cancun, Mexico, Jun. 2013, pp. 1938-1944.

[39] J. L. Rueda and I. Erlich, "Hybrid mean-variance mapping optimization for solving the IEEE-CEC 2013 competition problems," in Proc. IEEE Congr. Evol. Comput., Cancun, Mexico, Jun. 2013, pp. 1664-1671.

[40] B. Lacroix, D. Molina, and F. Herrera, "Dynamically updated region based memetic algorithm for the 2013 CEC special session and competition on real parameter single objective optimization," in Proc. IEEE Congr. Evol. Comput., Cancun, Mexico, Jun. 2013, pp. 1945-1951.
[65] F. Caraffini, G. Iacca, F. Neri, L. Piciocchi, and E. Mininni, “A CMA-ES super-fit scheme for the re-sampled inheritance search,” in Proc. IEEE Congr. Evol. Comput., Cancun, Mexico, Jun. 2013, pp. 1123–1130.

[66] M. El-Abd, “Testing a particle swarm optimization and artificial bee colony hybrid algorithm on the CEC13 benchmarks,” in Proc. IEEE Congr. Evol. Comput., Jun. 2013, pp. 2215–2220.

[67] F. V. Nepomuceno and A. P. Engelbrecht, “A self-adaptive heterogeneous pso for real-parameter optimization,” in Proc. IEEE Congr. Evol. Comput., Cancun, Mexico, Jun. 2013, pp. 361–368.

[68] S. M. Elsayed, R. A. Sarker, and D. L. Essam, “A genetic algorithm for solving the CEC'2013 competition problems on real-parameter optimization,” in Proc. IEEE Congr. Evol. Comput., Jun. 2013, pp. 356–360.

[69] G. Papa and J. Silc, “The continuous differential ant-stigmergy algorithm applied to real-world optimization problems,” in Proc. IEEE Congr. Evol. Comput., Jun. 2013, pp. 1131–1137.

[70] P. Korosec and J. Silc, “The continuous differential ant-stigmergy algorithm applied on real-parameter single objective optimization problems,” in Proc. IEEE Congr. Evol. Comput., Jun. 2013, pp. 1658–1665.

[71] Q. Fan and X. Yan, “Self-adaptive differential evolution algorithm with discrete mutation control parameters,” Expert Syst. Appl., vol. 42, no. 3, pp. 1551–1572, Feb. 2015.

[72] Y. Wang, H.-X. Li, T. Huang, and L. Li, “Differential evolution based on covariance matrix learning and bimodal distribution parameter setting,” Appl. Soft Comput., vol. 18, pp. 232–247, May 2014.

[73] A. W. Mohamed, A. A. Hadi, and K. M. Jambi, “Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization,” Swarm Evol. Comput., vol. 50, Nov. 2019, Art. no. 100455.

[74] M. Asafuddoula, T. Ray, and R. Sarker, “An adaptive differential evolution algorithm and its performance on real world optimization problems,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2011, pp. 1057–1062.

[75] G. Reynoso-Meza, J. Sanchez, X. Blasco, and J. M. Herrero, “Hybrid DE algorithm with adaptive crossover operator for solving real-world numerical optimization problems,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2011, pp. 1551–1556.

[76] A. LaTorre, S. Muelas, and J.-M. Pena, “Benchmarking a hybrid DE-RHC algorithm on real world problems,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2011, pp. 1027–1033.

[77] H. K. Singh and T. Ray, “Performance of a hybrid EA-DE-memetic algorithm on CEC2011 real world optimization problems,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2011, pp. 1322–1326.

[78] Y. Wang, B. Li, and K. Zhang, “Estimation of distribution and differential evolution cooperation for real-world numerical optimization problems,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2011, pp. 1315–1321.

[79] P. Suganthan, “Testing evolutionary algorithms on real-world numerical optimization problems,” School Electr. Electron. Eng., Nanyang Technol. Univ., Singapore, 2011. [Online]. Available: http://www3.ntu.edu.sg/home/epnsugan/index_files/CEC11-RWP/CEC2011_ranking.pdf

[80] R. Mallipeddi and P. N. Suganthan, “Ensemble differential evolution algorithm for CEC2011 problems,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2011, pp. 1557–1564.

[81] S. M. Elsayed, R. A. Sarker, and D. L. Essam, “Differential evolution with multiple strategies for solving CEC2011 real-world numerical optimization problems,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2011, pp. 1041–1048.

[82] A. Mandal, A. K. Das, P. Mukherjee, S. Das, and P. N. Suganthan, “Modified differential evolution with local search algorithm for real world optimization,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2011, pp. 1565–1572.

[83] U. Halder, S. Das, D. Maity, A. Abraham, and P. Dasgupta, “Self-adaptive cluster based and weed inspired differential evolution algorithm for real world optimization,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2011, pp. 750–756.

[84] P. Korosec and J. Silc, “The continuous differential ant-stigmergy algorithm applied to real-world optimization problems,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2011, pp. 1327–1334.

[85] S. M. Elsayed, R. A. Sarker, and D. L. Essam, “GA with a new multi-parent crossover for solving IEEE-CEC2011 competition problems,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2011, pp. 1034–1040.

[86] S. Bandaru, R. Tulshyan, and K. Deb, “Modified SBX and adaptive mutation for real world single objective optimization,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2011, pp. 1335–1342.

[87] A. Saha and T. Ray, “How does the good old genetic algorithm fare at real world optimization?” in Proc. IEEE Congr. Evol. Comput. (CEC), Jun. 2011, pp. 1049–1056.