International Journal of Machine Learning and Cybernetics (2020) 11:729–745
https://doi.org/10.1007/s13042-019-01030-4

ORIGINAL ARTICLE

Large-scale evolutionary optimization: a survey and experimental comparative study

Jun-Rong Jian[1,2] · **Zhi-Hui Zhan**[1,2] · **Jun Zhang**[1,2]

Received: 27 August 2019 / Accepted: 29 October 2019 / Published online: 14 November 2019

Abstract
In the last decades, global optimization problems are very common in many research fields of science and engineering and lots of evolutionary computation algorithms have been used to deal with such problems, such as differential evolution (DE) and particle swarm optimization (PSO). However, the algorithms performance rapidly decreases as the increasement of the problem dimension. In order to solve large-scale global optimization problems more efficiently, a lot of improved evolutionary computation algorithms, especially the improved DE or improved PSO algorithms have been proposed. In this paper, we want to analyze the differences and characteristics of various large-scale evolutionary optimization (LSEO) algorithms on some benchmark functions. We adopt the CEC2010 and the CEC2013 large-scale optimization benchmark functions to compare the performance of seven well-known LSEO algorithms. Then, we try to figure out which algorithms perform better on different types of benchmark functions based on simulation results. Finally, we give some potential future research directions of LSEO algorithms and make a conclusion.

Keywords Differential evolution · Particle swarm optimization · Large-scale global optimization · Large-scale evolutionary optimization algorithms

1 Introduction

In recent years, global optimization is a very important field in science and engineering because it often appears in many real-world optimization problems [1, 2]. To deal with such problems, lots of evolutionary computation (EC) algorithms includes evolutionary algorithms (EAs) such as genetic algorithm (GA) [3], differential evolution (DE) [4–7], and estimation of distribution algorithm (EDA) [8, 9], and swarm intelligence (SI) such as particle swarm optimization (PSO) [10, 11], ant colony optimization (ACO) [12], and artificial bee colony (ABC) [13] have been proposed. These algorithms have been shown great advantages on many numerical and combination optimization problems. However, most EC algorithms still suffer from the "curse of dimensionality" [14, 15], meaning that as the dimension of these problems increase, the performance of the EC algorithms will deteriorate rapidly. In real life, many optimization problems become more and more complex due to the increase of dimension. So, how to design and improve EC algorithms for large-scale optimization problems has attracted great attention from the scholars all over the world. This has aroused a popular and rapid developed research topic in EC community called large-scale evolutionary optimization (LSEO).

Generally speaking, there are two approaches that can improve the ability of EC to develop LSEO algorithms for solving large-scale optimization problems, namely, decomposition and non-decomposition. According to such a classification, the typical LSEO algorithms can be categorized as decomposition algorithms based on cooperative co-evolution (CC) method and non-decomposition algorithms that all the variables are considered as a whole, as shown in Fig. 1.

In the decomposition approach, the intuitive idea is to decompose an entire large-scale optimization problem into a number of smaller subproblems which are easier to be solved, and then optimize all the subproblems to achieve the purpose of optimizing the large-scale optimization problem. This idea is also known as "divide-and-conquer" which was first appeared in René Descartes' book *A Discourse on*

✉ Zhi-Hui Zhan
zhanapollo@163.com

[1] School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China

[2] State Key Laboratory of Subtropical Building Science, South China University of Technology, Guangzhou 510006, China

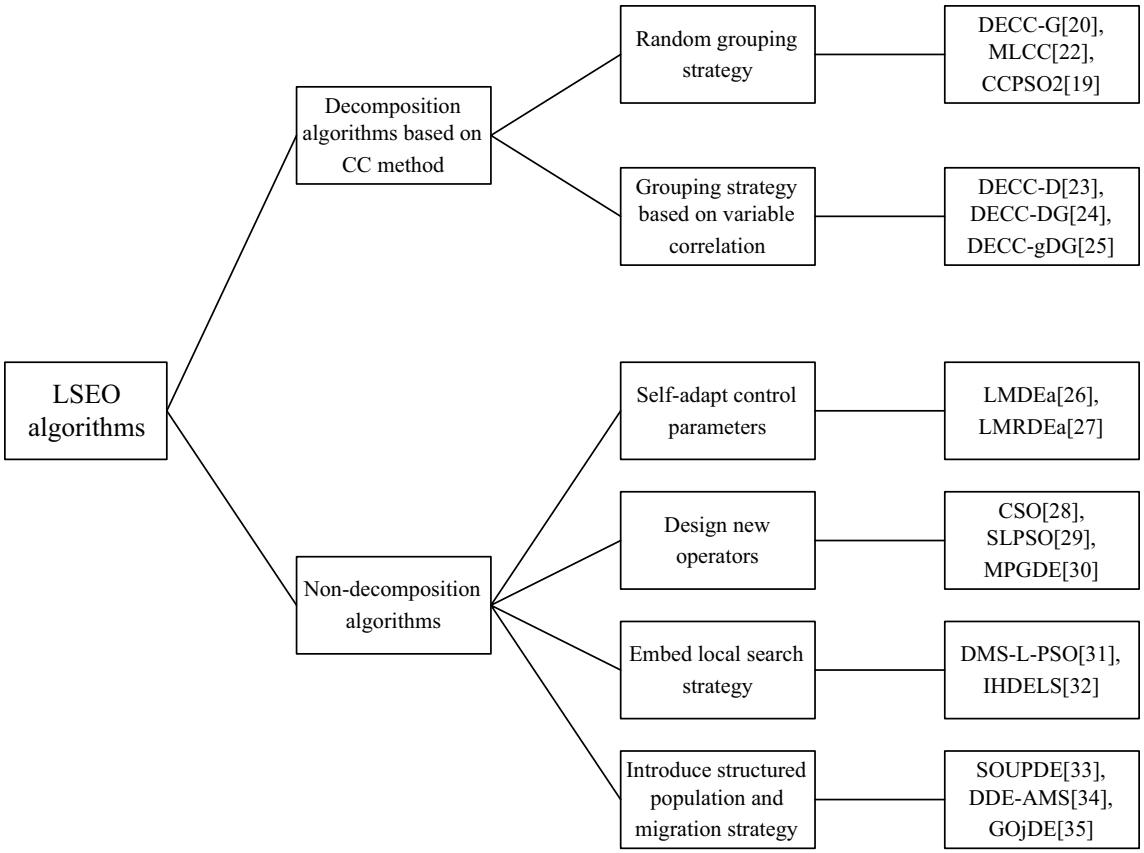

Fig. 1 Major categories of LSEO algorithms

Method [16]. The CC method which has been proposed by Potter and De Jong [17] is a famous and common method to decompose large-scale optimization problems. They first used this method in GA, termed as cooperative co-evolution GA (CCGA) [17], and decomposed an n-dimensional problem into n 1-dimensional problems. Liu et al. [15] proposed the fast evolutionary programming (FEP) algorithm with CC method, termed as FEPCC. However, FEPCC performs poorly in nonseparable functions because it often traps in a local optimal. Van den Bergh and Engelbrecht [18] were the first to apply the CC method to PSO and developed CCPSO. Unlike CCGA that separates all dimensions separately, the CCPSO decomposes an n-dimensional problem into k s-dimensional problems ($s \ll n$). In order to improve the diversity of the population and avoid local optimal, Li and Yao [19] proposed an improved CCPSO variant called CCPSO2 based on a random grouping (RG) strategy which proposed by Yang et al. [20]. In addition to the RG strategy, CCPSO2 also employs the Gaussian and Cauchy-based update rules with a local neighborhood topology to enhance the search ability. The CC method was also applied in DE to designed the DECC algorithm by Shi et al. [21]. They proposed a new decomposition strategy, called splitting-in-half strategy, which decomposed the decision variables into two subcomponents with equally size and each was evolved by a separate subpopulation [21]. To further reduce the scale of the problem, Yang et al. [20] proposed a new algorithm called DECC-G, where G means grouping. Similar to CPSO, DECC-G also uses the RG strategy to decompose a large-scale problem into several small-scale problems. Besides, DECC-G uses adaptive weighting for co-evolution among subproblems to improve the performance of the algorithm. However, DECC-G has a parameter which is difficult to be determined, that is, the size of the subproblems. In other word, the parameter is also called group size. To deal with this problem, multilevel cooperative co-evolution (MLCC) method was proposed by Yang et al. [22]. In MLCC, a set of possible group sizes are provided based on the RG strategy instead of using a fixed value. In addition to the RG strategy, various decomposition strategies combined with DE have been proposed, such as delta grouping strategy (DECC-D) [23], differential grouping (DG) strategy (DECC-DG) [24], and graph-based DG strategy (DECC-gDG) [25]. Moreover, Omidvar et al. [23] proposed a DECC-D variant called DECC-DML that self-adapted the group sizes. The development roadmap of the decomposition algorithms based on CC method is illustrated as Fig. 2.

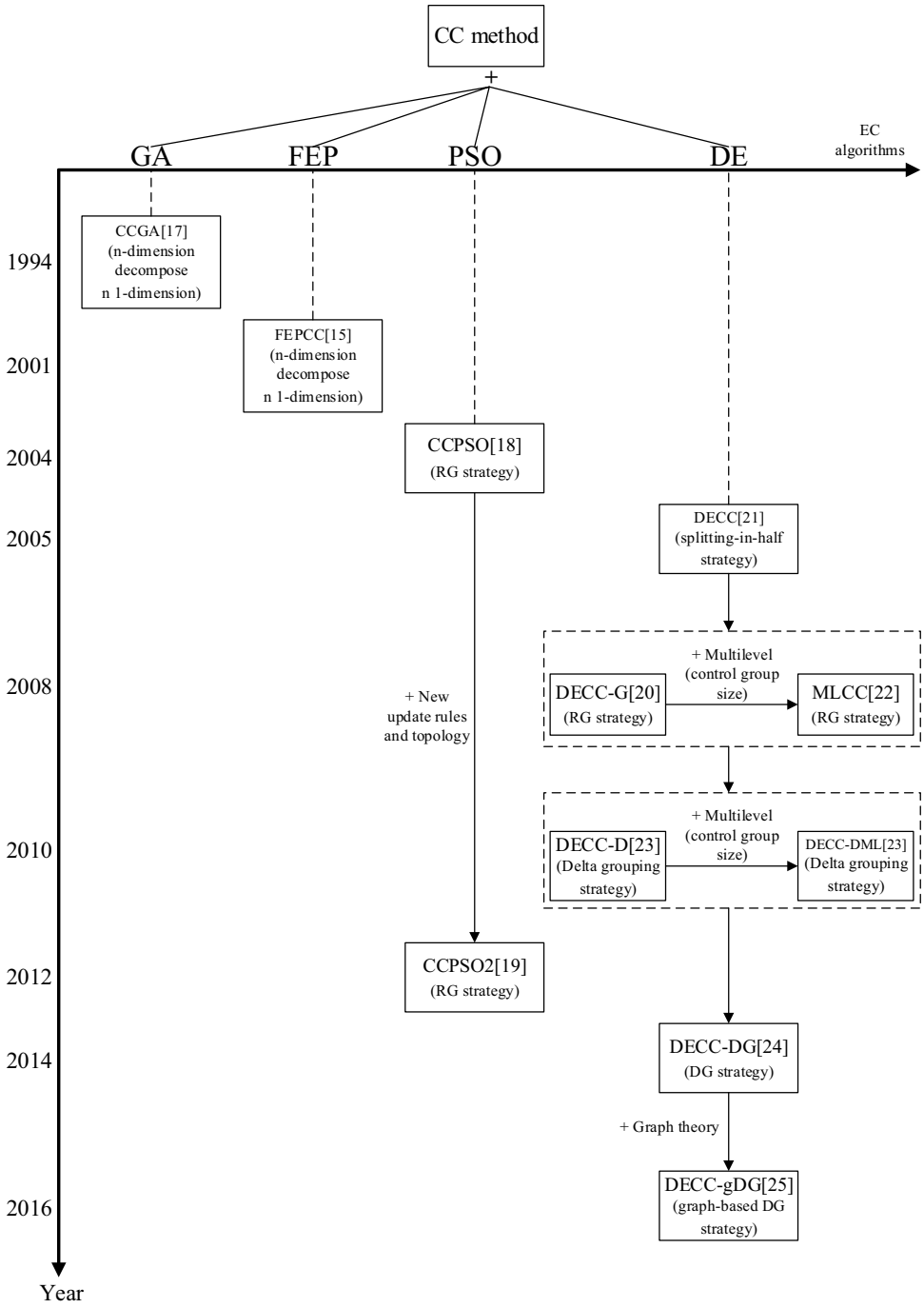

Fig. 2 The development roadmap of the decomposition algorithms based on CC method

There is no doubt that CC method is the most intuitive way to solve large-scale optimization problems and has achieved great success. However, CC method is highly sensitive to the decomposition strategies and has poor performance for nonseparable functions.

So, many researchers also considered the non-decomposition approach to tackle large-scale optimization problems. Unlike the decomposition approach, this approach considers all the decision variables of large-scale optimization problems as a whole instead of decomposing them. In the non-decomposition approach, researchers often solve large-scale optimization problems by (1) self-adapting control parameters [26, 27], (2) designing new operators [28–30], (3) embedding local search strategy [31, 32], and (4) introducing structured population and migration strategy [33–36]. Among the numerous EC algorithms, DE and PSO have more advantages than other algorithms because of their simplicity and efficiency. Therefore, many non-decomposition strategies based on these two algorithms have been proposed. Herein, according to the algorithms listed in

Fig. 1, we survey the LSEO algorithms based on the above four categories.

1. Firstly, on self-adapting control parameters, DE with landscape modality detection and a diversity archive (LMDEa) was proposed by Takahama and Sakai [26]. LMDEa can self-adapt the control parameters dynamically by modality detection and shows competitive result for the large-scale optimization problems. In order to enhance a more well-balanced exploration and exploitation ability, Kushida et al. [27] proposed an LMDEa variant which introduced an idea based on ranking, called LMRDEa. In LMRDEa, we can control the parameters as well as mutation strategy by detecting the landscape modality.
2. Secondly, on designing new operators, Cheng and Jin proposed a competitive swarm optimizer (CSO) [28] and a social learning PSO (SLPSO) [29] for large-scale optimization by designing new operators. In CSO, a pairwise competition mechanism is introduced to increase the population diversity and address premature convergence. In SLPSO, a social learning mechanism is introduced and each particle learns from any better particles instead of personal best and global best in the whole swarm. In addition, SLPSO also adopts a dimension-dependent parameter control method to ease the burden of parameter settings. Yang et al. [30] proposed a multiple parents guided DE (MPGDE) algorithm for large-scale optimization problems without using the decomposition approach.
3. Thirdly, on embedding local search strategy, the local search strategy is introduced to improve the performance of the EC algorithms. Zhao et al. [31] proposed a dynamic multi-swarm PSO with local search (DMS-L-PSO). Not only do the particles exchange information by regrouping subswarms frequently to avoid trapping into the local optimal, but also the Quasi-Newton method is adopted to speed up the convergence in DMS-L-PSO. Similarly, Molina and Herrera [32] proposed an iterative hybridization of DE algorithm with local search to solve large-scale optimization problems, called IHDELS.
4. Fourthly, on introducing structured population and migration strategy, many researchers adopt multi-population strategies which can achieve parallelization and distribution to deal with large-scale optimization problems and some appropriate population migration strategies are introduced to further improve the algorithms. Ge et al. [33] proposed a distributed DE based on adaptive mergence and split (DDE-AMS) for large-scale optimization problems. DDE-AMS designs a novel mergence and split operators to make full use of population resource which can improve the performance of large-scale optimization problems. Weber et al. [34] proposed a shuffle or update parallel DE (SOUPDE) which characterized by multi-population to avoid premature convergence. SOUPDE adopts a shuffling operation by randomly rearranging the individuals over the subpopulations and updating the parameters of the subpopulations. Similarly, Wang et al. [35] proposed a parallel DE variant for solving large-scale optimization problems, which called GOjDE. Moreover, Liang and Suganthan [36] proposed a dynamic multi-swarm PSO (DMS-PSO), which enhanced the diversity of population by constantly changing the neighborhood structure.

Overall, the above LSEO algorithms have their own advantages and disadvantages. It is no doubt that no algorithm can perform best on every kind of problems because of the "no free lunch" theorem [37]. In order to further understand the performance of different algorithms for different kinds of large-scale optimization problems, we choose seven representative LSEO algorithms (i.e., DECC-G, MLCC, DECC-DG, CCPSO2, CSO, SLPSO, and DMS-L-PSO) for comparison on some large-scale optimization functions, including CEC2010 test suits [38] and CEC2013 test suits [39]. Among the seven algorithms, there are four decomposition algorithms based on CC framework, including random grouping strategy (DECC-G, MLCC, and CCPSO2) and grouping strategy based on variable relationship (DECC-DG), and three non-decomposition algorithms which all the variables are considered as a whole. Among three non-decomposition algorithms, two algorithms adopt new operators (CSO and SLPSO), while the other adopts the operator of standard PSO (DMS-L-PSO). However, DMS-L-PSO adopts multi-population strategy and introduces local search strategy. We try to investigate and find out whether these seven algorithms have any preferences and difficulties on some kinds of test functions. Therefore, in this paper, we first list the experimental results of each algorithm on test suits through experiments. Then we analyze and investigate the performance of seven algorithms on different kinds of the problems based on experimental results. Finally, the conclusion and the discussion are presented.

The remainder of this paper is organized as follows. Section 2 briefly reviews the seven LSEO algorithms which are used to compare in this paper. Sections 3 and 4 present the experimental results of the seven algorithms on CEC2010 test suits and CEC2013 test suits respectively. The future research directions of LSEO algorithms are discussed in Sect. 5. In the end, Sect. 6 summarizes and concludes this paper.

2 Preliminaries

In this section, we are going to briefly introduce the LSEO algorithms which are used for comparison. Due to the simplicity and efficiency of DE and PSO, the LSEO

algorithms we use for comparison are all variants of DE and PSO. Therefore, we first present the standard DE and PSO algorithms, and then briefly review the seven LSEO algorithms for experimental comparative study.

2.1 DE

DE was first proposed by Storn and Price [4] as a population-based algorithm to search for the potential solutions. In initialization, NP individuals in a population $\boldsymbol{P} = \{\boldsymbol{x}_i, i = 1, 2, \ldots, NP\}$ are randomly generated within a search space, where i is the individual index. Each individual i refer as $\boldsymbol{x}_i = \{x_{i1}, x_{i2}, \ldots, x_{iD}\}$, where D is the problem dimension.

After initializing the population, the evolutionary process is carried out. Generally speaking, there are three operators during the evolutionary process, namely mutation, crossover, and selection. In mutation operator, each individual i create a mutation vector $\boldsymbol{v}_i = \{v_{i1}, v_{i2}, \ldots, v_{iD}\}$. So far, many mutation strategies have been proposed in the literature. Among them, two widely used mutation strategies in DE are listed as follows:

DE/rand/1:

$$\boldsymbol{v}_i = \boldsymbol{x}_{r1} + F \cdot (\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}) \tag{1}$$

DE/best/1:

$$\boldsymbol{v}_i = \boldsymbol{x}_{best} + F \cdot (\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3}) \tag{2}$$

where parameter r_1, r_2, and r_3 are distinct integer and also different from the index i which are randomly selected within $[1, NP]$; $\boldsymbol{x}_{best}$ is the global best individual found so far and parameter F is the amplification factor which controls the differential information between two random individuals.

After mutation, DE performs the crossover operation, which create a trial vector $\boldsymbol{u}_i = \{u_{i1}, u_{i2}, \ldots, u_{iD}\}$ for each individual from vector $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$. Crossover operation can be formulated as:

$$u_{ij} = \begin{cases} v_{ij}, & \text{if rand}(0,1) \le CR \text{ or } j = j_{rand} \\ x_{ij}, & \text{otherwise} \end{cases} \tag{3}$$

where $rand(0, 1)$ is a uniform value within $[0, 1]$; j denotes the dimension; CR is the crossover rate and j_{rand} is an integer randomly selected from $[1, D]$ which is used to make sure that at least one dimension of the $\boldsymbol{u}_i$ is different form $\boldsymbol{x}_i$.

Finally, selection operation is performed. DE choose the better individual into the next generation by comparing the fitness of $\boldsymbol{x}_i$ and $\boldsymbol{u}_i$. For minimization problems, the selection operation can be formulated as:

$$\boldsymbol{x}_i = \begin{cases} \boldsymbol{u}_i, & \text{if } f(\boldsymbol{u}_i) \le f(\boldsymbol{x}_i) \\ \boldsymbol{x}_i, & \text{otherwise} \end{cases} \tag{4}$$

where $f()$ is the fitness function of the problems.

2.2 PSO

PSO was introduced by Kennedy and Eberhart [10, 11] in 1995, which used a swarm of particles to simulate the swarm intelligence behaviors of birds flocking to find the optimal solution. In PSO, each particle i has three vectors, position vector $\boldsymbol{x}_i = \{x_{i1}, x_{i2}, \ldots, x_{iD}\}$, velocity vector $\boldsymbol{v}_i = \{v_{i1}, v_{i2}, \ldots, v_{iD}\}$ and personal historical best position vector $\boldsymbol{pbest}_i = \{p_{i1}, p_{i2}, \ldots, p_{iD}\}$, where D is the dimensions of the problem. In the initialization, the position and velocity of each particle are randomly set within the corresponding ranges. The personal historical best position vector $\boldsymbol{p}_i$ is set to $\boldsymbol{x}_i$. The best particle's position vector of the whole swarm by calculating the fitness values of all the particles is denoted as the global best position vector $\boldsymbol{gbest} = \{g_{i1}, g_{i2}, \ldots, g_{iD}\}$.

During the evolutionary process, the velocity and the position of the particle i on dimension j are updated as follows:

$$v_{ij} = \omega \cdot v_{ij} + c_1 \cdot rand_{1j} \cdot (pbest_{ij} - x_{ij}) + c_2 \cdot rand_{2j} \cdot (gbest_j - x_{ij}), \tag{5}$$

$$x_{ij} = x_{ij} + v_{ij} \tag{6}$$

where ω is the inertia weight [40], c_1 and c_2 are the acceleration coefficients [11], and $rand_1$ and $rand_2$ are the two uniformly distributed random numbers which are generated within [0, 1] independently for the dimension j; $pbest_{ij}$ is the jth dimension of the best position found by particle i so far. The $gbest_j$ is the jth dimension of the best position found by all particles so far.

Generally, PSO algorithms are divided into two version, global-version PSO (GPSO) and local-version PSO (LPSO) [11]. They use global best particle ***gbest*** to guide all particles to update their velocity and position in GPSO while use local best particle ***lbest*** in LPSO. Therefore, in LPSO, neighborhood is constructed with a small group of particles and ***lbest*** is the best position found by all particles in the neighborhood so far.

2.3 DECC-G

The DECC-G algorithm was introduced by Yang et al. [20]. which is one of the earliest algorithms to apply DE with CC method. This algorithm attempts to decompose a large-scale optimization problem into several subproblems which are much smaller than the whole problem by random grouping. In other words, an n-dimensional

problem vector is decomposed into m ($m \ll n$) s-dimensional subcomponents, and random grouping means that each variable has an equal chance to be assigned to any of the subcomponents. After that, we use a variant of DE, the Self-adaptive Neighborhood Search DE (SANSDE) [41] to optimize all the subcomponents independently. Obviously, it is much more efficient and effective to optimize subcomponents than to optimize an entire large-scale problem. When all the subcomponents are optimized, they are combined again to construct the solution vector. It should be noted that in DECC-G, the grouping structure will be changed dynamically, that is, the random grouping operation should be carried out before each generation starts. The motivation behind this is to increase the chance of optimizing interacting variables together.

Furthermore, an adaptive weighting strategy has been proposed in DECC-G. Some interdependent subcomponents can coadaptation by adaptive weighting strategy. A weight is applied to each of the subcomponents after the evolution process in each generation and all the weights will construct a weight vector. Then, the weight vector will be evolved with standard DE algorithm.

2.4 MLCC

In order to overcome the shortcomings of DECC-G, Yang et al. [22] proposed a new framework called multilevel cooperative coevolution (MLCC) for large-scale optimization problems. In DECC-G, one of major disadvantages is that it has a parameter which is difficult to determine, called the group size and this parameter has a great impact on the performance of different types of problems. The small group size is good for separable problems while large group size is proper for nonseparable problems. So, MLCC was proposed to improve DECC-G from the perspective of framework by assigning a set of group sizes.

In MLCC, several problem decomposers with different group sizes are designed to construct a decomposer pool. Each decomposer in the pool represents different interaction levels between variables. At the beginning of each generation, MLCC need to select a problem decomposer from the decomposer pool based on their probability. And then, similar to DECC-G, random grouping strategy with the selected decomposer and evolving each subcomponent with SANSDE will be carried out. At the end of each generation, the performance growth rate of the selected decomposer will be calculated and updated. Note that, each decomposer in the pool will record the performance growth rate. If the performance growth rate is high, the probability of this decomposer being selected is high; on the contrary, if the performance growth rate is low, this decomposer has less chance to be selected. With such strategy, MLCC is able to self-adapt to select decomposer with proper group size according to the problem features and the evolution state. The details of the MLCC framework can be referred to [22].

2.5 DECC-DG

In order to further explore the interaction between the decision variables, Omidvar et al. [24] proposed an automatic decomposition strategy called differential grouping. Similarly, this decomposition strategy is also applied in CC method and combined with DE algorithm, which is called DECC-DG. Unlike DECC-G and MLCC which use random grouping without the prior knowledge of the problems, DECC-DG decompose the large-scale optimization problems by differential grouping which can detect the interaction between decision variables, so that the interacting variables are assigned to the same subcomponents and the interdependence between subcomponents is kept to a minimum.

Differential grouping is an effective way to identify interaction between two decision variables and group the interacting variables together. We can judge whether the two decision variables x_i and x_j are interaction by the following formula:

$$\begin{aligned} &\Delta_{\delta,x_i} f(\boldsymbol{x})|_{x_i=a,x_j=b_1} \neq \Delta_{\delta,x_i} f(\boldsymbol{x})|_{x_i=a,x_j=b_2}, \\ &\Delta_{\delta,x_i} f(\boldsymbol{x}) = f(\ldots,x_i+\delta,\ldots) - f(\ldots,x_i,\ldots) \end{aligned} \tag{7}$$

where a, b_1, and b_2 are three arbitrary values; b_1 is not equal to b_2 and $\delta \neq 0$; $f()$ is the fitness function of the problems. If (7) is satisfied, x_i and x_j can be considered to be interaction, and these two decision variables can be grouped in the same subcomponent. In other words, if x_i and x_j are interacting variables, the value of function is change by add a perturbation to x_i for different values of x_j.

In fact, DECC-DG is similar to some DECC algorithm variants, except that before the evolutionary process starts, the differential grouping strategy is carried out to decompose large-scale problems into several subcomponents. Later, each subcomponents will be optimized with SANSDE algorithm.

2.6 CCPSO2

The CCPSO2 algorithm was proposed on the basis of CCPSO algorithm by Li and Yao [19]. CCPSO algorithm is similar to DECC-G algorithm which incorporate random grouping strategy and adaptive weighting strategy. The only difference is in the selection of evolutionary algorithm. CCPSO uses standard PSO algorithm while DECC-G uses SANSDE algorithm. In order to further improve

the performance and reliability of CCPSO, CCPSO2 was proposed which adopted several new strategy.

In CCPSO2, Li and Yao improve the standard PSO algorithm by introducing Gaussian and Cauchy distributions, and remove the adaptive weighing strategy which is adopted in CCPSO. Moreover, an ***lbest*** ring topology structure is adopted in CCPSO2 to increase the diversity of the population and avoid convergence prematurely. For the novel PSO model, the velocity vector does not need to be used, but instead, it generate the new particle positions by using Gaussian and Cauchy distributions which can sampling around the personal best and the neighborhood best. Each particle position $\boldsymbol{x}_i$ is updated by:

$$x_{ij} = \begin{cases} y_{ij} + C(1)|pbest_{ij} - lbest_{ij}|, & \text{if } rand \le p \\ y'_{ij} + \mathrm{N}(0,1)|pbest_{ij} - lbest_{ij}|, & \text{otherwise} \end{cases} \tag{8}$$

where $C(1)$ and $N(0, 1)$ is the number that is generated based on Cauchy distribution and Gaussian distribution, and their standard deviation are both set $|y_{ij} - y'_{ij}|$; j denotes the dimension; *rand* is a random number generated uniformly from [0, 1]; p is a specified probability; $pbest_{ij}$ denotes the jth dimension of the personal best of the ith particle; $lbest_{ij}$ denotes the jth dimension of the local neighborhood best of the ith particle. The local neighborhood best is chosen among three particles (the current ith particle and its immediate left and right neighbors) based on ring topology structure.

Besides, similar to MLCC, a different group size in a set can be randomly chosen at each generation instead of using a fixed group size in CCPSO2. However, CCPSO2 uses a simpler approach. If the global best fitness value does not improve after a generation, a new group size from the set is randomly chosen again, otherwise, the group size remains unchanged.

2.7 CSO

In order to achieve a good balance between exploration and exploitation and address premature convergence, Cheng and Jin [28] proposed a novel competitive swarm optimizer (CSO) based on PSO for large-scale optimization. In consideration of the fact that most PSO variant update particles based on the global best position ***gbest*** and the personal best position ***pbest*** which lead to convergence prematurely, the update of particles in CSO is driven by a pairwise competition mechanism between two particles instead of using ***gbest*** and ***pbest***. This mechanism can address premature convergence and maintain the diversity of the population because each particle has a chance to learn from any particles.

In CSO, all the particles are randomly allocated in pairs in each generation. Assume that the swarm size is m which is an even number, so there are $m/2$ pairs of particles. In each pairs, a pairwise competition is carried out between two particles. The winner (the particle which has better fitness value) goes to the next generation directly while the loser (the particle which has worse fitness value) will update its velocity and position by learning from the winner as follows:

$$v_{lj} = r_1 v_{lj} + r_2(x_{wj} - x_{lj}) + \varphi r_3(\bar{x}_j - x_{lj}) \tag{9}$$

$$x_{lj} = x_{lj} + v_{lj} \tag{10}$$

where r_1, r_2, and r_3 are three random numbers generated uniformly from [0, 1]; x_{wj}, x_{lj} and v_{wj}, v_{lj} denote the jth dimension of the position and velocity of the winner and loser respectively; $\bar{x}_j$ denotes the mean position value of the jth dimension of all particles in current swarm; φ is a parameter that controls the influence of $\bar{x}_j$ .Therefore, only half of the particles position in swarm are updated in each generation in CSO which can save some fitness evolution.

2.8 SLPSO

Cheng and Jin [29] proposed a another new PSO variant called Social Learning PSO (SLPSO), where neither the personal best position ***pbest*** nor global best position ***gbest*** will be used to update the particle position like CSO. In SLPSO, each particle learns from any better particles in the current swarm instead of learning from historical best particle position. The social learning mechanism make the particles learn from each other dynamically and interactively which can maintain the diversity of the swarm and avoid premature convergence.

At the beginning of each generation, sort all the particles in the swarm according to their fitness value. Then, each particle, except for the best one, will learn from the particles that better than itself. Note that, if the number of the better particles is more than one, select one of them randomly. Assume that the current updated particle is i, and the selected particle for learning is k. The learning process is shown as follows:

$$v_{ij} = r_1 v_{ij} + r_2(x_{kj} - x_{ij}) + \varepsilon r_3(\bar{x}_j - x_{ij}) \tag{11}$$

$$x_{ij} = \begin{cases} x_{ij} + v_{ij}, & \text{if } p_i(t) \le P_i \\ x_{ij}, & \text{otherwise} \end{cases} \tag{12}$$

where j is the dimension of the vector; r_1, r_2, and r_3 are three random numbers generated uniformly from [0, 1]; x_{ij}, x_{kj} and v_{ij}, v_{kj} denote the jth dimension of the position vector and velocity vector of particle i and k respectively; $\bar{x}_j$ denotes the mean position value of the jth dimension of all particles in

current swarm; ε is a parameter that controls the influence of $\bar{x}_j$. In (12), P_i is called the learning probability which control whether the particles are updated or not. Each particle have their own learning probability. Generally speaking, the better the fitness value of a particle is, the lower the learning probability will be. $p_i(t)$ is a randomly generated number from [0, 1].

In addition, SLPSO adopts a dimension-dependent parameter control method in order to ease the burden of parameter settings.

2.9 DMS-L-PSO

DMS-L-PSO is a dynamic multi-swarm particle swarm optimizer with local search which is proposed by Zhao et al. [31]. In order to increase the diversity of population, the population is divide into small sized swarms and each subswarm evolved independently in DMS-L-PSO. In addition, every R (called regrouping period) generation, the population is regrouped randomly which can dynamically change the subswarms structure and exchange the information among the particles to enhance the diverstiy. At the same time, a new PSO variant is introduced in DMS-L-PSO. In the new PSO variant, when updating the positions of the particles, half of the dimension are the same as its personal best position ***pbest*** and the other half of the dimensions are updated as follows:

$$v_{ij} = \omega \cdot v_{ij} + c_1 \cdot r_1 \cdot (pbest_{ij} - x_{ij}) + c_2 \cdot r_2 \cdot (lbest_{ij} - x_{ij}) \tag{13}$$

$$x_{ij} = x_{ij} + v_{ij} \tag{14}$$

where ω is the inertia weight fixed to be 0.729; c_1 and c_2 is the accelerate coefficient fixed to be 1.49445; r_1 and r_2 are two random value from [0, 1]; ***lbest*** is the best particle position in the current subswarm; x_{ij} and v_{ij} represent the jth dimension of position vector and velocity vector of particle i respectively.

In order to speed up the convergence of the population and give a better search in the better local areas, local search is introduced in DMS-L-PSO. Every L (called local search period) generation, the personal best position ***pbest*** of five particles will be randomly chosen to do the local search by Quasi-Newton method. Then, the ***pbest*** which is nearest (according to Euclidean distance) to the refined solution will be replaced with the refined solutions if the refined solution is better.

Besides, when 90% of maximum fitness evaluations have been used, all the particles in subswarms are reconstituted into one population. Then, the global PSO algorithm is adopted to continue optimizing the population until the fitness evaluations is exhausted.

3 Comparison studies on CEC2010

In this section, the comparisons experiments of the seven LSEO algorithms which mentioned in Sect. 1 on CEC2010 test suits are carried out. The CEC2010 test suits contain 20 test functions. These functions can be classified into the following five groups.

1. Separable functions (f_1−f_3).
2. Single-group m-nonseparable functions (f_4−f_8).
3. (n/2 m) group m-nonseparable functions (f_9−f_{13}).
4. (n/m) group m-nonseparable functions (f_{14}−f_{18}).
5. Nonseparable functions (f_{19}−f_{20}).

where n represents the dimension of the problem and m represents the number of variables in each nonseparable subcomponent. The specific form and characteristics of these functions can be referred to [38].

To make a fair comparison, in our experiments, the dimensions of the problem n are set as 1000 and the maximum number of function evaluations (*MaxFEs*) is set as 3e6 for all 7 LSEO algorithms. For the parameter setting of all the algorithms, parameters are set according to their original papers. In addition, to make the results more convincing, all the algorithms need to run 25 times independently for statistics and calculate the mean results and standard deviation.

The experimental results of 7 LSEO algorithms on CEC2010 test suites are shown in Table 1. For clarity, the best results for each function are highlighted in boldface. In addition, we count the number of functions that perform the best for each algorithm and show them on the last line of the results. From this statistical data, it can be seen that among the 20 test functions, CSO has the best performance on seven functions, DECC-DG and SLPSO both have the best performance on five functions, while other algorithms have poor performance. MLCC, DECC-G, CCPSO2 and DMS-L-PSO have the best performance on only 2, 1, 0 and 0 functions respectively. However, as can be seen from the specific experimental results in Table 1:

For the first group of three separable functions (f_1−f_3), MLCC performs better than other algorithms except f_3, especially on f_1. MLCC can converge to the optimal value on f_1. This may benefit from the multi-level grouping strategy in MLCC algorithm and this strategy has great advantages in resolving the separable functions, but on the more complex separable function f_3, CSO performs better than other algorithms.

For the second group of 15 partially-separable functions (f_4−f_{18}), DECC-DG, CSO and SLPSO perform better than other algorithms. On the partially-separable functions with fewer groups (f_4−f_{13}), the performance of CSO and SLPSO are better than that of DECC-DG except f_7 and f_{12} (unimodal

Table 1 Experimental results of 7 LSEO algorithms on cec2010 test suits

Fun	DECC-G	MLCC	DECC-DG	CCPSO2
	Mean ± std	Mean ± std	Mean ± std	Mean ± std
f_1	1.36E−14 ± 2.22E−15	**0 ± 0**	1.61E+01 ± 1.89E+01	1.64E+00 ± 2.28E+00
f_2	4.88E+01 ± 1.31E+01	**3.18E−01 ± 5.43E−01**	4.47E+03 ± 2.09E+02	7.49E+00 ± 1.74E+00
f_3	1.71E+00 ± 3.33E−01	8.17E−02 ± 3.11E−01	1.67E+01 ± 3.44E−01	8.89E−03 ± 2.10E−03
f_4	1.25E+13 ± 2.85E+12	1.54E+13 ± 6.57E+12	3.82E + 12 ± 6.75E + 11	1.55E + 12 ± 7.41E + 11
f_5	2.60E + 08 ± 8.16E + 07	3.13E + 08 ± 1.09E + 08	1.54E + 08 ± 1.90E + 07	4.53E + 08 ± 1.18E + 08
f_6	4.76E + 06 ± 6.97E + 05	1.61E + 07 ± 4.61E + 06	1.64E + 01 ± 1.92E−01	1.92E + 07 ± 1.07E + 06
f_7	8.39E + 06 ± 7.56E + 06	1.81E + 06 ± 2.85E + 06	**5.81E + 03 ± 2.66E + 03**	1.70E + 08 ± 3.23E + 08
f_8	4.72E + 07 ± 3.08E + 07	3.76E + 07 ± 3.27E + 07	3.94E + 07 ± 2.98E + 07	3.31E + 07 ± 2.97E + 07
f_9	2.54E + 08 ± 1.01E + 07	1.19E + 08 ± 1.44E + 07	5.95E + 07 ± 9.21E + 06	1.14E + 08 ± 3.60E + 07
f_{10}	9.22E + 03 ± 4.37E + 02	2.98E + 03 ± 3.70E + 02	4.55E + 03 ± 1.21E + 02	5.71E + 03 ± 1.03E + 03
f_{11}	2.52E + 01 ± 1.18E + 00	1.96E + 02 ± 3.30E + 00	1.13E + 01 ± 5.09E−01	1.98E + 02 ± 2.74E−01
f_{12}	3.91E + 04 ± 5.81E + 03	3.60E + 04 ± 6.49E + 03	**2.53E + 03 ± 3.14E + 02**	2.78E + 04 ± 7.58E + 03
f_{13}	3.13E + 03 ± 1.15E + 03	2.37E + 03 ± 1.64E + 03	4.86E + 03 ± 2.73E + 03	1.28E + 03 ± 1.82E + 02
f_{14}	5.77E + 08 ± 2.18E + 07	3.24E + 08 ± 1.97E + 07	3.40E + 08 ± 1.85E + 07	3.22E + 08 ± 1.46E + 08
f_{15}	9.79E + 03 ± 2.74E + 03	7.17E + 03 ± 1.14E + 03	**5.84E + 03 ± 6.02E + 01**	1.02E + 04 ± 8.90E + 02
f_{16}	8.50E + 01 ± 1.18E + 01	3.81E + 02 ± 4.93E + 01	**7.23E−13 ± 5.50E−14**	3.97E + 02 ± 4.60E−01
f_{17}	1.63E + 05 ± 9.60E + 03	1.56E + 05 ± 1.03E + 04	**4.18E + 04 ± 1.14E + 03**	1.41E + 05 ± 5.81E + 04
f_{18}	9.00E + 03 ± 1.09E + 03	6.83E + 03 ± 5.99E + 03	1.51E + 10 ± 1.86E + 09	2.87E + 03 ± 3.73E + 02
f_{19}	**7.33E + 05 ± 4.61E + 04**	1.31E + 06 ± 1.05E + 05	1.71E + 06 ± 1.04E + 05	1.41E + 06 ± 8.90E + 04
f_{20}	3.47E + 03 ± 2.48E + 02	2.03E + 03 ± 1.88E + 02	6.17E + 10 ± 6.59E + 09	1.97E + 03 ± 2.40E + 02
The number of best result	1	2	5	0

Fun	CSO	SLPSO	DMS-L-PSO
	Mean ± Std	Mean ± Std	Mean ± Std
f_1	4.50E−12 ± 5.94E−13	8.73E−18 ± 3.30E−18	4.88E + 09 ± 1.34E + 08
f_2	7.42E + 03 ± 2.86E + 02	1.93E + 03 ± 1.12E + 02	6.17E + 03 ± 1.87E + 02
f_3	**2.60E−09 ± 2.62E−10**	1.88E + 00 ± 3.30E−01	1.74E + 01 ± 5.26E−02
f_4	7.25E + 11 ± 1.23E + 11	**2.99E + 11 ± 7.16E + 10**	3.90E + 13 ± 4.40E + 12
f_5	**1.15E + 07 ± 1.62E + 06**	3.17E + 07 ± 6.21E + 06	1.04E + 08 ± 7.83E + 06
f_6	**8.21E−07 ± 2.68E−08**	2.08E + 01 ± 2.63E + 00	1.66E + 06 ± 1.46E + 05
f_7	2.01E + 04 ± 3.86E + 03	6.49E + 04 ± 5.60E + 04	2.42E + 10 ± 1.63E + 09
f_8	3.87E + 07 ± 6.81E + 04	**7.81E + 06 ± 1.56E + 06**	1.43E + 08 ± 3.42E + 07
f_9	7.03E + 07 ± 5.73E + 06	**3.30E + 07 ± 4.46E + 06**	5.79E + 09 ± 2.02E + 08
f_{10}	9.60E + 03 ± 7.67E + 01	**2.56E + 03 ± 2.17E + 02**	5.88E + 03 ± 2.48E + 02
f_{11}	**4.02E−08 ± 5.12E−09**	2.32E + 01 ± 2.10E + 00	1.82E + 02 ± 1.38E + 00
f_{12}	4.37E + 05 ± 6.22E + 04	1.75E + 04 ± 9.07E + 03	2.84E + 06 ± 1.10E + 05
f_{13}	**6.29E + 02 ± 2.32E + 02**	9.59E + 02 ± 3.74E + 02	9.68E + 07 ± 2.62E + 07
f_{14}	2.49E + 08 ± 1.53E + 07	**8.41E + 07 ± 6.31E + 06**	5.02E + 09 ± 3.43E + 08
f_{15}	1.01E + 04 ± 5.23E + 01	1.12E + 04 ± 8.65E + 01	6.21E + 03 ± 2.76E + 02
f_{16}	5.89E−08 ± 5.61E−09	2.51E + 01 ± 1.16E + 01	3.39E + 02 ± 9.52E−01
f_{17}	2.20E + 06 ± 1.55E + 05	9.00E + 04 ± 1.58E + 04	2.67E + 06 ± 1.54E + 05
f_{18}	**1.73E + 03 ± 5.22E + 02**	2.77E + 03 ± 8.33E + 02	2.82E + 09 ± 5.30E + 08
f_{19}	1.01E + 07 ± 5.64E + 05	5.10E + 06 ± 7.05E + 05	1.63E + 07 ± 6.70E + 05
f_{20}	**1.05E + 03 ± 1.49E + 02**	1.85E + 03 ± 2.59E + 02	4.10E + 09 ± 7.56E + 08
The number of best result	7	5	0

function). This may be because DECC-DG consumes a lot of unnecessary function evaluations (FEs) for differential grouping in solving simple partially-separable functions, resulting in less FEs for evolution. While on the partially-separable functions with more groups ($f_{14}-f_{18}$), such as f_{15}, f_{16}, f_{17}, DECC-DG performs better than other algorithms thanks to the differential grouping strategy. In this kind of partially-separable functions which is more difficult to solve, appropriate grouping strategy like differential grouping can get better performance.

For the third group of two nonseparable functions ($f_{19}-f_{20}$), DECC-G performs the best on f_{19} (unimodal function) while CSO performs the best on f_{20} (multimodal function). DECC-DG performs poorly on nonseparable functions.

In addition, we rank the seven algorithms on each function according to their experimental results, as shown in Table 2. From the ranking results, SLPSO performs the best among the seven algorithms and CSO is ranked the second. For the decomposition algorithms based on CC framework, DECC-DG performs the best. DMS-L-PSO is ranked the last and this may be caused by the evolution operator of this algorithm which is the same as the standard PSO.

In order to further study the evolutionary process of different algorithms on CEC2010 test suits, we select six representative functions f_1, f_4, f_8, f_{11}, f_{16} and f_{20} and draw the convergence curves of the seven algorithms on these six functions, as shown in Fig. 3.

From Fig. 3a, we can see that only MLCC and SLPSO have a fast convergence speed and converge to better solutions. However, SLPSO will stagnate at the later stage of evolution and only MLCC can converge to the optimal value on f_1. On partially-separable functions f_4 and f_8, as shown in Fig. 3b, c, SLPSO is better than other algorithms in terms of convergence speed and final results. On partially-separable function f_{11}, as shown in Fig. 3d, only CSO can converge continuously and rapidly, while other algorithms fall into local optimal prematurely. Besides, on partially-separable function f_{16}, only CSO and DECC-DG can converge to better solutions quickly and the final result of DECC-DG is better than that of CSO. Finally, from Fig. 3f, we can see the convergence curves of seven algorithms on nonseparable function f_{20}. Except DECC-DG and DMS-L-PSO, other algorithms have the similar performance. They can find the better solutions and have faster convergence speed than DECC-DG and DMS-L-PSO.

To sum up, the performance of CSO and SLPSO are better than other decomposition algorithms based on CC framework (DECC-G, MLCC, DECC-DG, and CCPSO2). The evolution operators of these two algorithms are different from that of the standard PSO. The evolution operator of CSO is based on competitive relationship, while SLPSO

Table 2 Ranking results of 7 LSEO algorithms on cec2010 test suits

Fun	DECC-G Rank	MLCC Rank	DECC-DG Rank	CCPSO2 Rank	CSO Rank	SLPSO Rank	DMS-L-PSO Rank
f_1	3	**1**	6	5	4	2	7
f_2	3	**1**	5	2	7	4	6
f_3	4	3	6	2	**1**	5	7
f_4	5	6	4	3	2	**1**	7
f_5	5	6	4	7	**1**	2	3
f_6	5	6	2	7	**1**	3	4
f_7	5	4	**1**	6	2	3	7
f_8	6	3	5	2	4	**1**	7
f_9	6	5	2	4	3	**1**	7
f_{10}	6	2	3	4	7	**1**	5
f_{11}	4	6	2	7	**1**	3	5
f_{12}	5	4	**1**	3	6	2	7
f_{13}	5	4	6	3	**1**	2	7
f_{14}	6	4	5	3	2	**1**	7
f_{15}	4	3	**1**	6	5	7	2
f_{16}	4	6	**1**	7	2	3	5
f_{17}	5	4	**1**	3	6	2	7
f_{18}	5	4	7	3	**1**	2	6
f_{19}	**1**	2	4	3	6	5	7
f_{20}	5	4	7	3	**1**	2	6
Ave. rank	4.6	3.9	3.65	4.15	3.15	**2.6**	5.95
Final rank	6	4	3	5	2	**1**	7

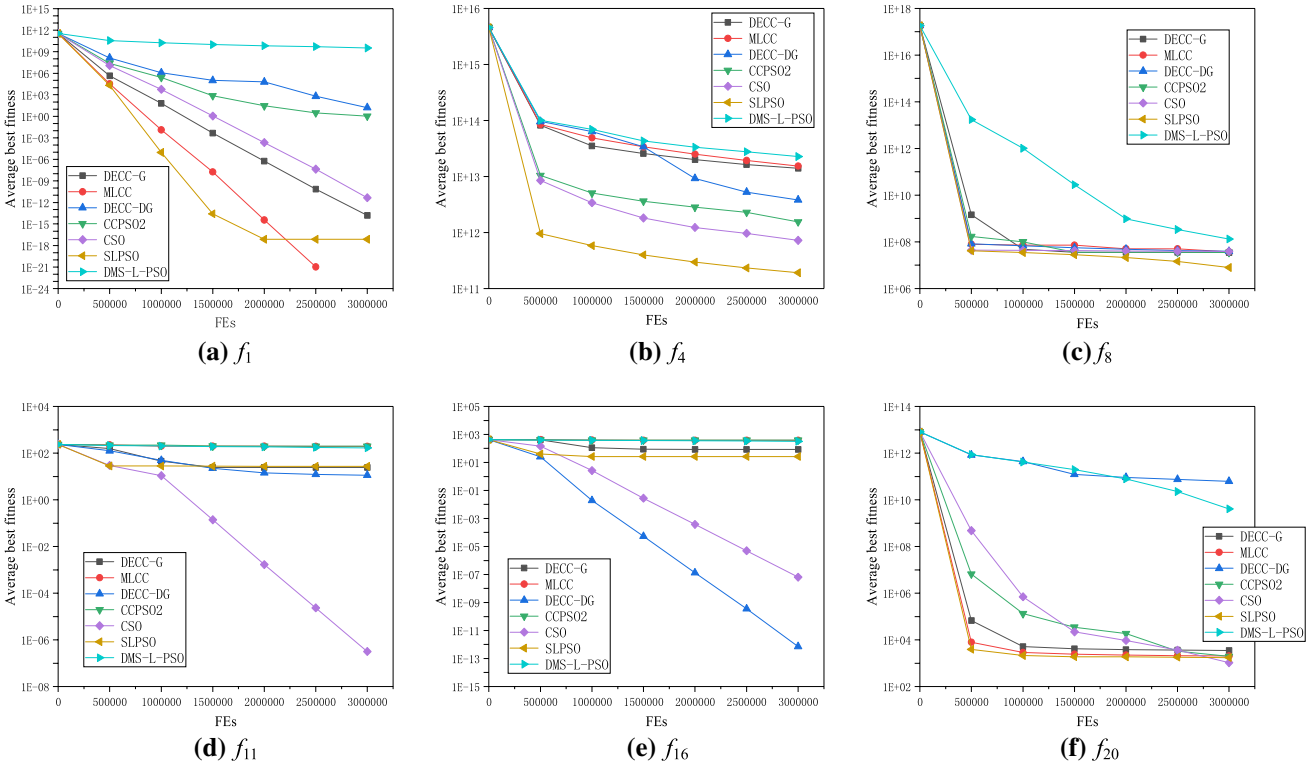

Fig. 3 Convergence curves of 7 algorithms on 6 representative functions from cec2010

is based on social learning. Therefore, the good evolution operator will greatly improve the performance of LSEO algorithms. The poor performance of DMS-L-PSO may be because it still adopts the evolution operator of the standard PSO. For the decomposition algorithms based on CC framework, DECC-DG has the best overall performance, especially in solving partially-separable functions. However, DECC-DG performs worse on separable and nonseparable functions than MLCC and CCPSO2 which are based on random grouping strategy, which may be caused by excessive consumption of FEs by differential grouping strategy. Moreover, MLCC performs better than other algorithms in solving simple separable functions.

4 Comparison Studies on CEC2013

To further compare the ability of seven algorithms to solve the large-scale optimization problems, we adopted another large-scale benchmark function suites—CEC2013 test suites to test the performance of the algorithms. The CEC2013 test suites are more complex and more difficult to solve than CEC2010, so they can better reflect the performance of the algorithms to solve the large-scale optimization problems. The CEC2013 test suits contain 15 test functions. These test functions can be classified into the following four groups.

1. Separable functions (f_1-f_3).
2. Partially-separable functions (f_4-f_{11}).
3. Overlapping functions ($f_{12}-f_{14}$).
4. Nonseparable functions (f_{15}).

The specific form and characteristics of these functions can be referred to [38].

Similar to the experiments on CEC2010 test suits, in this experiments, the dimensions of the problem n are also set as 1000 (Note: the dimensions of f_{13} and f_{14} are set as 905) and the maximum number of function evaluations (*MaxFES*) is also set as 3e6 for all 7 LSEO algorithms.

The experimental results of seven LSEO algorithms on CEC2013 test suites are shown in Table 3. For clarity, the best results for each function are highlighted in boldface. Similarly, we count the number of functions that perform the best for each algorithm and show them on the last line of the results. From this statistical data, it can be seen that SLPSO has the best performance on 6 out of 15 test functions, more than other algorithms. While MLCC, CCPSO2, CSO, DECC-DG, DMS-L-PSO, and DECC-G have the best performance of only 4, 2, 2, 1, 1 and 0 functions respectively.

Table 3 Experimental results of 7 algorithms on cec2013 test suits

Fun	DECC-G	MLCC	DECC-DG	CCPSO2
	Mean ± std	Mean ± std	Mean ± std	Mean ± std
f_1	2.16E−12 ± 1.03E−12	**8.10E−26 ± 1.62E−25**	1.25E + 03 ± 1.78E + 03	5.26E + 00 ± 1.62E + 00
f_2	4.90E + 01 ± 2.25E + 01	**8.29E + 00 ± 5.58E + 00**	1.28E + 04 ± 4.62E + 02	1.24E + 01 ± 1.00E + 00
f_3	2.01E + 01 ± 2.27E−03	**2.00E + 01 ± 1.65E−03**	2.14E + 01 ± 1.52E−02	**2.00E + 01 ± 4.00E−05**
f_4	1.42E + 11 ± 6.49E + 10	8.76E + 10 ± 2.85E + 10	5.24E + 10 ± 3.36E + 10	1.61E + 10 ± 7.64E + 09
f_5	7.53E + 06 ± 1.61E + 06	1.02E + 07 ± 2.09E + 06	5.82E + 06 ± 4.39E + 05	1.70E + 07 ± 5.15E + 06
f_6	1.06E + 06 ± 5.82E + 02	1.05E + 06 ± 3.67E + 03	1.06E + 06 ± 9.04E + 02	1.05E + 06 ± 9.50E + 03
f_7	3.98E + 08 ± 3.03E + 08	4.30E + 08 ± 2.18E + 08	8.35E + 08 ± 7.66E + 08	1.16E + 08 ± 8.71E + 07
f_8	2.94E + 15 ± 1.29E + 15	4.59E + 15 ± 3.71E + 15	4.59E + 15 ± 5.25E + 14	6.20E + 14 ± 5.84E + 14
f_9	5.97E + 08 ± 1.21E + 08	8.98E + 08 ± 2.26E + 08	5.00E + 08 ± 2.44E + 07	3.18E + 09 ± 6.06E + 08
f_{10}	9.30E + 07 ± 5.53E + 05	**9.22E + 07 ± 3.84E + 05**	9.46E + 07 ± 3.47E + 04	9.37E + 07 ± 4.40E + 05
f_{11}	5.90E + 10 ± 4.91E + 10	1.20E + 11 ± 4.46E + 10	**2.35E + 10 ± 1.45E + 10**	9.30E + 11 ± 8.16E + 09
f_{12}	3.36E + 03 ± 2.69E + 02	2.10E + 03 ± 1.99E + 02	1.63E + 11 ± 1.61E + 10	2.02E + 03 ± 8.87E + 01
f_{13}	4.54E + 09 ± 6.53E + 08	8.19E + 09 ± 3.58E + 09	1.98E + 10 ± 8.21E + 09	2.04E + 09 ± 6.06E + 08
f_{14}	7.53E + 10 ± 3.44E + 10	1.18E + 11 ± 6.86E + 10	1.86E + 10 ± 9.40E + 09	1.42E + 11 ± 9.87E + 10
f_{15}	4.76E + 06 ± 4.16E + 05	6.70E + 06 ± 1.04E + 06	9.51E + 06 ± 9.59E + 05	**3.67E + 06 ± 2.54E + 06**
The number of best result	0	4	1	2

Fun	CSO	SLPSO	DMS-L-PSO
	Mean ± std	Mean ± std	Mean ± std
f_1	3.67E−12 ± 1.04E−12	1.09E−17 ± 2.50E−18	5.70E + 09 ± 1.44E + 08
f_2	7.04E + 03 ± 3.56E + 02	2.13E + 03 ± 1.36E + 02	1.24E + 04 ± 2.69E + 02
f_3	2.16E + 01 ± 5.44E−03	2.16E + 01 ± 1.45E−02	2.14E + 01 ± 1.92E−02
f_4	1.26E + 10 ± 1.91E + 09	**4.35E + 09 ± 9.48E + 08**	9.00E + 11 ± 3.78E + 10
f_5	8.62E + 05 ± 3.20E + 04	**8.41E + 05 ± 1.75E + 05**	5.49E + 06 ± 4.19E + 05
f_6	1.06E + 06 ± 1.05E + 03	1.06E + 06 ± 1.48E + 03	**1.03E + 06 ± 3.99E + 03**
f_7	7.62E + 06 ± 1.35E + 06	**1.63E + 06 ± 7.05E + 05**	3.55E + 09 ± 2.61E + 08
f_8	3.50E + 14 ± 3.59E + 13	**1.03E + 14 ± 3.62E + 13**	6.79E + 15 ± 1.38E + 15
f_9	**3.94E + 07 ± 7.45E + 06**	8.25E + 07 ± 2.03E + 07	5.05E + 08 ± 2.66E + 07
f_{10}	9.41E + 07 ± 1.49E + 05	9.25E + 07 ± 1.67E + 06	9.31E + 07 ± 3.11E + 05
f_{11}	3.58E + 11 ± 4.62E + 09	9.33E + 11 ± 1.46E + 10	4.96E + 11 ± 4.01E + 10
f_{12}	**1.28E + 03 ± 8.23E + 01**	1.78E + 03 ± 1.74E + 02	4.42E + 09 ± 8.34E + 08
f_{13}	8.06E + 08 ± 1.22E + 08	**4.65E + 08 ± 2.35E + 08**	1.16E + 11 ± 1.05E + 10
f_{14}	5.17E + 09 ± 2.86E + 09	**3.28E + 08 ± 5.17E + 08**	1.25E + 12 ± 1.59E + 11
f_{15}	1.74E + 07 ± 6.44E + 05	7.87E + 07 ± 6.50E + 06	1.60E + 09 ± 6.18E + 08
The number of best result	2	6	1

However, as can be seen from the specific experimental results in Table 3:

For the first group of three separable functions (f_1-f_3), MLCC still shows great advantage and performs better than other algorithms.

For the second group of eight partially-separable functions (f_4-f_{11}), CSO and SLPSO perform better than other algorithms. SLPSO performs the best on 4 functions (f_4, f_5, f_7, f_8). Although CSO performs not as well as SLPSO on these functions, it is not much different from SLPSO in terms of results. CSO performs the best on f_9. Similar to the experimental results on CEC2010, DECC-DG performs the best on the complex partially-separable function f_{11}.

For the third group of three overlapping functions ($f_{12}-f_{14}$), it is also the case that CSO and SLPSO perform better than other algorithms. While the performance of DECC-DG and DMS-L-PSO are very poor.

For the last nonseparable function (f_{15}), the decomposition algorithms based on CC framework (DECC-G, MLCC, DECC-DG, and CCPSO2) performs better than the non-decomposition algorithms (CSO, SLPSO, and DMS-L-PSO).

Table 4 Ranking results of 7 algorithms on cec2013 test suits

Fun	DECC-G Rank	MLCC Rank	DECC-DG Rank	CCPSO2 Rank	CSO Rank	SLPSO Rank	DMS-L-PSO Rank
f_1	3	**1**	6	5	4	2	7
f_2	3	**1**	7	2	5	4	6
f_3	3	**1**	4	**1**	6	6	4
f_4	6	5	4	3	2	**1**	7
f_5	5	6	4	7	2	**1**	3
f_6	4	2	4	2	4	4	**1**
f_7	4	5	6	3	2	**1**	7
f_8	4	5	5	3	2	**1**	7
f_9	5	6	3	7	**1**	2	4
f_{10}	3	**1**	7	5	6	2	4
f_{11}	2	3	**1**	6	4	7	5
f_{12}	5	4	7	3	**1**	2	6
f_{13}	4	5	6	3	2	**1**	7
f_{14}	4	5	3	6	2	**1**	7
f_{15}	2	3	4	**1**	5	6	7
Ave. rank	3.8	3.53	4.73	3.8	3.2	**2.73**	5.47
Final rank	4	3	6	4	2	**1**	7

Similarly, according to their experimental results, we also give the specific ranking of the seven algorithms on each function in Table 4. From Table 4, we can see that SLPSO and CSO are still ranked first and second, which further illustrates the advantages of these two algorithms. For the decomposition algorithms based on CC framework, the ranking of MLCC is the best. DECC-DG loses its competitiveness on CEC2013 test suits, which indicates that when faces with more complex test functions, good grouping strategy does not greatly improve the performance of the algorithms. DMS-L-PSO is still ranked the last.

Similar to the experiment on CEC2010 test suits, we also select six representative functions $f_1, f_4, f_9, f_{11}, f_{13}$ and f_{15} and draw the convergence curves of the seven algorithms on these six functions, as shown in Fig. 4.

On separable function f_1, as shown in Fig. 4(a), the convergence curves of the algorithms is similar to that of f_1 on CEC2010 test suits. MLCC and SLPSO converge faster than other algorithms. However, SLPSO will fall into the local optimal at the later stage of evolution and MLCC can find the better result. On partially-separable functions f_4, f_9 and overlapping function f_{13}, as shown in Fig. 4b, c, e, CSO and SLPSO are better than other algorithms in terms of convergence speed and final results. Besides, on partially-separable function f_{11}, as shown in Fig. 4d, although DECC-DG can find the best solutions and have fastest convergence speed, in fact, other algorithms are very similar to its converge curve. Similar phenomenon also appears on nonseparable function f_{15}, as shown in Fig. 4f. Except DMS-L-PSO, the convergence curves of other algorithms are very similar, but the final result of CCPSO2 is better than other algorithms.

In conclusion, CEC2013 test functions are more complex than CEC2010 test functions, so LSEO algorithms are less effective in solving CEC2013 test suits. However, in terms of the performance of each algorithm, CSO and SLPSO still perform better than other algorithms. While DECC-DG is not capable of solving more complicated problems and its performance is not as good as that of other decomposition algorithms based on CC framework. Moreover, MLCC still highlights its advantages on separable functions.

5 Future directions

At present, with the development of big data, artificial intelligence, and high performance computing, the optimization problems we encounter in the real world are becoming more and more complex and large-scale. In order to solve the large-scale optimization problems, LSEO algorithms have attracted more and more researchers' attention in recent decades. Researchers have also proposed many strategies and methods to improve the LSEO algorithms. However, due to the complexity of large-scale optimization problems, LSEO algorithms still face many challenges. Therefore, how to improve the performance of LSEO algorithms will continue to be a hot topic. In this section, we will present some future research directions of LSEO algorithms.

1. Perfect decomposition of large-scale optimization problem by CC method. The ultimate goal of the CC method should group all variables into optimal or near-optimal subcomponents, where the variables in the same sub-

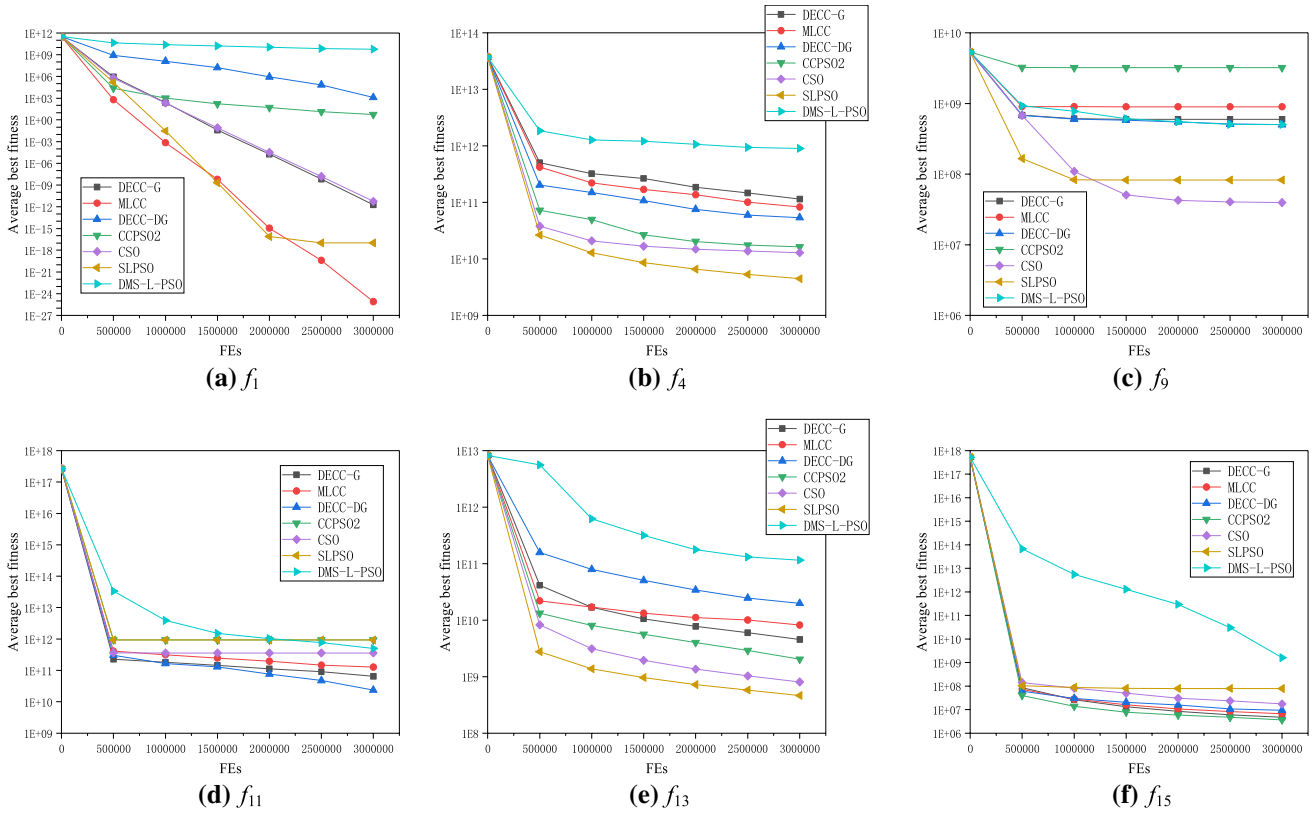

Fig. 4 Convergence curves of 7 algorithms on 6 representative functions from cec2013

component are with linkage while the variants in different subcomponents are without or with weak linkage. Although there are many decomposition strategies at present, such as delta grouping strategy, DG strategy, graph-based DG strategy, and so on, most of the decomposition strategies only focus on the construction of non-separable subcomponents. That is, only consider the related variables that are divided into a group and do not consider the construction of separable subcomponents. Therefore, the future decomposition strategies based on CC method need to consider the construction of both non-separable and separable subcomponents, so as to realize the optimal decomposition and improve the performance of the LSEO algorithms. Moreover, the problem characteristics can be used to help decompose the variables perfectly. For example, Zhang et al. [42] recently proposed an efficient CC based bare-bones PSO (CCBBPSO) with function independent decomposition (FID) according to the different independent functional characteristics of different variables.

2. Light computational burden for decomposition. The decomposition strategies based on CC method not only need to improve the decomposition accuracy, but also should be better to consume as little FEs as possible, so that more FEs can be used to the evolutionary process and to make the algorithms perform better. For example, DECC-DG performs poorly on completely separable functions and non-separable functions, which may be because it consumes too many FEs when adopting DG strategy to group variables. In addition, for non-separable functions, even if a good decomposition strategy is used to group the variables, all variables will eventually be divided into one group, which is equivalent to solving the whole large-scale optimization function without achieving the goal of "divide-and-conquer". Therefore, a fast method which has low consumption of FEs is needed to determine whether the problem is a completely separable problem or a non-separable problem. For example, some variables can be randomly selected to judge whether there is a relationship between them. If there is, the problem is judged to be a non-separable problem. Other decomposition strategies that can reduce FEs consumption need further study.
3. Efficient allocation of computational resources to different subcomponents. Most optimization problems in the real world have the imbalance property. In other words, when we decompose a large-scale optimization problem into several subproblems, the optimization of some subproblems have a great impact on the final results of the whole optimization problem, while the rest of the

subproblems have insignificant impact on the whole optimization problem. According to the imbalance of the optimization problem, it is obvious to allocate more computational resources to subprobelms which contribute more to the whole optimization problem. Therefore, considering the imbalance of the optimization problems will become another research direction of LSEO algorithms. We can introduce weights to different subproblems to express their contribution to the whole optimization problem. Omidvar et al. [43] proposed a contribution-based CC method to allocate the available computational resources to the subcomponents based on their contributions, called CBCC (CBCC1, CBCC2). Later, in order to find a better balance between exploration and exploitation, Omidvar et al. [44] proposed an improved CBCC variant called CBCC3. At present, the research on the imbalance of optimization problem is still in its infancy, which provides us with another way to design new LSEO algorithms.

4. Multiple populations and distributed algorithm to enhance global search ability. Beside the decomposition method, there is another significant method to deal with large-scale optimization problems by considering all the variables as a whole. However, due to the high dimension of large-scale optimization problem, the search space of the problem is very large and there are many local optimal solutions. Therefore, many distributed and parallel algorithms have been proposed to deal with large-scale optimization problems. Generally speaking, LSEO algorithms can realize distribution by means of multi-population strategy, which a whole population is decomposed into multiple subpopulations, and each subpopulation evolves independently. At the same time, an appropriate migration strategy should be designed to further improve the diversity of the population and the performance of the algorithms. Wang et al. [45] proposed a variant of distributed PSO algorithm which adopted dynamic group learning strategy, called DGLDPSO. Wu et al. [46] also proposed a distributed DE algorithm which based on multi-population strategy, called MPEDE. MPEDE not only adopts distributed strategy, but also integrates multiple mutation strategies to improve the performance of the algorithms. Moreover, some LSEO algorithms use multiple processors to achieve parallelization [47, 48]. Each subpopulation has a processor for processing, and all subpopulations evolve in the processor in parallel, which can reduce the running time and accelerate the efficiency of LSEO algorithm. In conclusion, we can consider introducing distributed and parallel methods to deal with large-scale optimization problems in the future.
5. Scalability of LSEO algorithms. At present, most of the researchers focus on large-scale optimization problems with 500 or 1000 dimensions. In the face of more complex problems with larger dimensions (such as 2000, 3000 or even 5000 dimensions), the performance of some LSEO algorithms drop sharply, which indicate that the scalability of most algorithms are poor. Therefore, scalability is very important for LSEO algorithms, and it is also one of the future research directions.
6. At last, application of real-world large-scale optimization problems. In recent years, a number of LSEO algorithms with good performance have been proposed. However, most of the algorithms are tested by using large-scale benchmark function sets. Therefore, an obvious question was raised that the good performance of the algorithm on the benchmark function sets does not mean that the algorithm can also perform well in solving real-world problems. So, we need to apply LSEO algorithm to real-world problems or design corresponding algorithms according to the characteristics of real-world problems. Nowadays, there are many complex problems such as dynamic optimization problems [49, 50], multimodal optimization problems [51–53], and multi/many-objective optimization problems [54, 55], and also the real-world problems such as cloud workflow scheduling problems [56–58], and community detection and inference [59, 60] that have drawn more and more attention to the researchers. These problems would be much more challenge in their large-scale version. Therefore, the LSEO algorithms we design in the future needs to be connected with real-world problems.

6 Conclusion

In this paper, we first introduce the research status of large-scale optimization problems and list some state-of-the-art LSEO algorithms. Then, we select seven representative algorithms and make a brief introduction, including the decomposition algorithms based on CC framework and non-decomposition algorithms which all the variables are considered as a whole. In addition, we adopt two commonly used large-scale benchmark function sets—CEC2010 test suits and CEC2013 test suits to compare the performance of these seven LSEO algorithms. Meanwhile, in order to further compare these seven LSEO algorithms, their convergence curves on some representative functions are plotted to analyze their characteristics and advantages. From the experimental results, SLPSO and CSO perform better than other algorithms. While MLCC is suitable for solving simple separable functions and DECC-DG can find better results on some relatively complex partially-separable functions. However, when faced with some extremely complex test functions, the performance of DECC-DG is not as

good as some decomposition algorithms which based on random grouping strategy. Besides, DMS-L-PSO performs worse than other comparative algorithms. Moreover, we also discuss the future research directions of LSEO algorithms. Especially, we think that the function independent decomposition strategy is worthy studying in CC based LSEO, like the CCBBPSO algorithm [42], and the distributed multi-population is worthy studying in non-CC based LSEO, like the DGLDPSO algorithm [45].

Acknowledgements This work was supported in part by the Outstanding Youth Science Foundation under Grant 61822602, in part by the National Natural Science Foundations of China (NSFC) under Grant 61772207 and Grant 61873097, in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003, and in part by the Guangdong-Hong Kong Joint Innovation Platform under Grant 2018B050502006.

References

1. Shi GY, Dong JL (2002) Optimization methods. Higher Education Press, Beijing
2. Fletcher R (1987) Practical methods of optimization. Wiley-Interscience, New York
3. Holland JH (1992) Genetic algorithms. Sci Am 267(1):66–72
4. Storn R, Price K (1997) Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. J Global Opt 11(4):341–359
5. Storn R (1996) On the usage of differential evolution for function optimization. In: 1996 biennial conference of the North American fuzzy information processing, pp 519–523
6. Cui L, Li G, Lin Q, Chen J, Lu N (2016) Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations. Comput Oper Res 67:155–173
7. Li G, Lin Q, Cui L, Du Z, Liang Z, Chen J, Lu N, Ming Z (2016) A novel hybrid differential evolution algorithm with modified CoDE and JADE. Appl Soft Comput 47:577–599
8. Muhlenbein H (1996) From recombination of genes to the estimation of distributions I. binary parameters. In: International Conference on Parallel Problem Solving from Nature. Springer, Berlin, Heidelberg, pp 178–187
9. Zhang QF, Sun JY, Tsang E, Ford J (2004) Hybrid estimation of distribution algorithm for global optimization. Eng Comput 21(1):91–107
10. Kennedy J, Eberhart RC (1995) Particle swarm optimization. IEEE Int. Conf. Neural Netw, Perth, pp 1942–1948
11. Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: the 6th Int. Symp. Micromachine Human Sci. Nagoya, pp 39–43
12. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern B Cybern 26(1):29–41
13. Cui L, Li G, Luo Y, Chen F, Ming Z, Lu N, Lu J (2018) An enhanced artificial bee colony algorithm with dual-population framework. Swarm Evol Comput 43:184–206
14. Yang ZY, Tang K, Yao X (2008) Large scale evolutionary optimization using cooperative coevolution. Inf Sci 178(15):2985–2999
15. Liu Y, Yao X, Zhao Q, Higuchi T (2001) Scaling up fast evolutionary programming with cooperative coevolution. In: IEEE Congr. Evol. Comput., pp 1101–1108
16. Descartes R (1956) Discourse on method, 1st edn. Perentice Hall, Upper Saddle River
17. Potter MA, Jong KAD (1994) A cooperative coevolutionary approach to function optimization. In: International Conference on Parallel Problem Solving from Nature, pp 249–257
18. Bergh FV, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput 8(3):225–239
19. Li X, Yao X (2012) Cooperatively coevolving particle swarms for large scale optimization. IEEE Trans Evol Comput 16(2):210–224
20. Yang Z, Tang K, Yao X (2008) Large scale evolutionary optimization using cooperative coevolution. Inf Sci 178(15):2985–2999
21. Shi Y, Teng H, Li Z (2005) Cooperative co-evolutionary differential evolution for function optimization. In: International Conference on Natural Computation, pp 1080–1088
22. Yang Z, Tang K, Yao X (2008) Multilevel cooperative coevolution for large scale optimization. In: IEEE Congr. Evol. Comput., pp 1663–1670
23. Omidvar MN, Li X, Yao X (2010) Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In: IEEE Congr. Evol. Comput., pp 1762–1769
24. Omidvar M, Li X, Mei Y, Yao X (2014) Cooperative co-evolution with differential grouping for large scale optimization. IEEE Trans Evol Comput 18(3):378–393
25. Ling YB, Li HJ, Cao B (2016) Cooperative co-evolution with graph-based differential grouping for large scale global optimization. In: IEEE International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, pp 95–102
26. Takahama T, Sakai S (2012) Large scale optimization by differential evolution with landscape modality detection and a diversity archive. In: IEEE Congr. Evol. Comput., pp 2842–2849
27. Kushida J, Hara A, Takahama T (2015) Rank-based differential evolution with multiple mutation strategies for large scale global optimization. In: IEEE Congr. Evol. Comput., pp 353–360
28. Ran C, Jin YC (2015) A competitive swarm optimizer for large scale optimization. IEEE Trans Cybern 45(2):191–204
29. Ran C, Jin YC (2015) A social learning particle swarm optimization algorithm for scalable optimization. Inf Sci 291:43–60
30. Yang Q, Xie HY, Chen WN, Zhang J (2016) Multiple parents guided differential evolution for large scale optimization. In: IEEE Congr. Evol. Comput., pp 3549–3556
31. Zhao SZ, Liang JJ, Suganthan PN, Tasgetiren MF (2008) Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In: IEEE Congr. Evol. Comput., pp 3845–3852
32. Molina D, Herrera F (2015) Iterative hybridization of DE with local search for the cec2015 special session on large scale global optimization. In: IEEE Congr. Evol. Comput., pp 1974–1978
33. Ge YF, Yu WJ, Lin Y, Gong YJ, Zhan ZH, Chen WN, Zhang J (2018) Distributed differential evolution based on adaptive mergence and split for large-scale optimization. IEEE Trans Cybern 48(7):2166–2180
34. Weber M, Neri F, Tirronen V (2011) Shuffle or update parallel differential evolution for large-scale optimization. Appl Soft Comput 15(11):2089–2107
35. Wang H, Rahnamayan S, Wu ZJ (2013) Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems. J Parallel Distrib Comput 73(1):62–73
36. Liang JJ, Suganthan PN (2005) Dynamic multi-swarm particle swarm optimizer. In: IEEE Int. Swarm Intelligence Symposium, pp 124–129
37. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
38. Tang K, Li X, Suganthan P, Yang Z, Weise T (2009) Benchmark functions for the cec 2010 special session and competition on large scale global optimization. In: Technical Report, Nature

Inspired Computation and Applications Laboratory, USTC, China
39. Li X, Tang K, Omidvar MN, Yang Z, Qin K (2013) Benchmark functions for the cec 2013 special session and competition on large scale global optimization. In: Evol. Comput. Mach. Learn. Subpopulation, Tech. Rep. RMIT University, Melbourne
40. Shi Y, Eberhart RC (1998) A modified particle swarm optimizer. In: IEEE World Congr. Comput. Intell., pp 69–73
41. Yang Z, Tang K, Yao X (2007) Differential evolution for high-dimensional function optimization In: IEEE Congr. Evol. Comput., pp 3523–3530
42. Zhang X, Du KJ, Zhan ZH, Kwong S, Gu TL, Zhang J (2019) Cooperative co-evolutionary bare-bones particle swarm optimization with function independent decomposition for large-scale supply chain network design with uncertainties. IEEE Trans Cybern. https://doi.org/10.1109/TCYB.2019.2933499
43. Omidvar MN, Li X, Yao X (2011) Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms. In: Conference on Genetic and Evolutionary Computation, pp 1115–1122
44. Omidvar MN, Kazimipour B, Li X, Yao X (2016) CBCC3—a contribution-based cooperative co-evolutionary algorithm with improved exploration/exploitation balance. In: IEEE Congr. Evol. Comput., pp 3541–3548
45. Wang ZJ, Zhan ZH, Yu WJ, Lin Y, Zhang J, Gu TL, Zhang J (2019) Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling. IEEE Trans Cybern. https://doi.org/10.1109/TCYB.2019.2933499
46. Wu G, Mallipeddi R, Suganthan PN, Wang R, Chen H (2016) Differential evolution with multi-population based ensemble of mutation strategies. Inf Sci 329:329–345
47. Glotic A, Glotic A, Kitak P, Pihler J, Ticar I (2014) Parallel self-adaptive differential evolution algorithm for solving short-term hydro scheduling problem. IEEE Trans Power Syst 29(5):2347–2358
48. Zhan ZH, Liu X, Zhang H, Yu Z, Weng J, Li Y, Gu T, Zhang J (2017) Cloudde: a heterogeneous differential evolution algorithm and its distributed cloud version. IEEE Trans Parallel Distrib Syst 28(3):704–716
49. Liu XF, Zhan ZH, Gu TL, Kwong S, Lu Z, Duh HBL, Zhang J (2019) Neural network-based information transfer for dynamic optimization. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2019.2920887
50. Liu XF, Zhan ZH, Zhang J (2018) Neural network for change direction prediction in dynamic optimization. IEEE Access 6:72649–72662
51. Zhao H, Zhan ZH, Lin Y, Chen X, Luo XN, Zhang J, Kwong S, Zhang J (2019) Local binary pattern-based adaptive differential evolution for multimodal optimization problems. IEEE Trans Cybern. https://doi.org/10.1109/TCYB.2019.2927780
52. Wang ZJ, Zhan ZH, Lin Y, Yu WJ, Wang H, Kwong S, Zhang J (2019) Automatic niching differential evolution with contour prediction approach for multimodal optimization problems. IEEE Trans Evol Comput. https://doi.org/10.1109/tevc.2019.2910721
53. Wang ZJ, Zhan ZH, Lin Y, Yu WJ, Yuan HQ, Gu TL, Kwong S, Zhang J (2018) Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems. IEEE Trans Evol Comput 22(6):894–908
54. Zhan ZH, Li J, Cao J, Zhang J, Chung H, Shi YH (2013) Multiple populations for multiple objectives: a coevolutionary technique for solving multiobjective optimization problems. IEEE Trans Cybern 43(2):445–463
55. Liu XF, Zhan ZH, Gao Y, Zhang J, Kwong S, Zhang J (2019) Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization. IEEE Trans Evol Comput 23(4):587–602
56. Chen ZG, Zhan ZH, Lin Y, Gong YJ, Yuan HQ, Gu TL, Kwong S, Zhang J (2019) Multiobjective cloud workflow scheduling: a multiple populations ant colony system approach. IEEE Trans Cybern 49(8):2912–2926
57. Zhan ZH, Liu XF, Gong YJ, Zhang J, Chung HSH, Li Y (2015) Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Comput Surv 47(4):1–33
58. Liu XF, Zhan ZH, Deng D, Li Y, Gu TL, Zhang J (2018) An energy efficient ant colony system for virtual machine placement in cloud computing. IEEE Trans Evol Comput 22(1):113–128
59. Ma L, Gong M, Liu J, Cai Q, Jiao L (2014) Multi-level learning based memetic algorithm for community detection. Appl Soft Comput. 19:121–133
60. Ma L, Li J, Lin Q, Gong M, Coello CAC, Ming Z (2019) Reliable link inference for network data with community structure. IEEE Trans Cybern 49(9):3347–3361