Applied Intelligence (2024) 54:4585–4601
https://doi.org/10.1007/s10489-024-05390-5

Evolutionary dynamic grouping based cooperative co-evolution algorithm for large-scale optimization

Wanting Yang[1,2] · Jianchang Liu[1,2] · Shubin Tan[1,2] · Wei Zhang[1,2] · Yuanchao Liu[1,2]

Accepted: 9 March 2024 / Published online: 1 April 2024

Abstract
To effectively address large-scale optimization problems, this paper proposes an evolutionary dynamic grouping (EDG) based cooperative co-evolution (CC) algorithm. In the proposed algorithm, a novel decomposition method is designed to generate the sub-components of decision variables dynamically. Additionally, an evolutionary search method based on the fireworks search strategy is proposed to enhance the searchability of the algorithm. The performance of the proposed algorithm is assessed using two benchmark suites, IEEE CEC'2010 and IEEE CEC'2013, as well as a real-world optimization problem, the 0/1 Knapsack Problem (KP). Experimental results demonstrate that the proposed algorithm achieves competitive results when compared with other state-of-the-art algorithms.

Keywords Large-scale optimization · Cooperative co-evolution (CC) · Dynamic grouping · Fireworks search strategy

1 Introduction

Over the past decade, the dimensional space of optimization problems has grown significantly in real-world applications [1, 2]. Consequently, large-scale global optimization problems (LSGOPs), involving at least thousands of decision variables [3, 4], have emerged as a vibrant area of research. The dimensionality of the decision variables is a major factor in the complexity of optimization problems. The exponential growth in the size of the search space with respect to the number of decision variables affects the number of candidate solutions in the search space. Generally, the dimensionality of the decision variables also has a direct impact on the computational cost of the optimization and the computational feasibility of detecting correlation between pairs of variables [5]. Therefore, LSGOPs are valuable but challenging to solve. Recently, some attempts have been made for LSGOPs [6].

Evolutionary algorithms (EAs) have demonstrated significant success in solving complex optimization problems [7–13]. However, due to the "curse of dimensionality" [14], it is still difficult for traditional EAs to address LSGOPs [15, 16]. Consequently, numerous researchers have undertaken efforts to develop EAs tailored for LSGOPs [15, 17]. These algorithms are commonly known as large-scale evolutionary algorithms (LSEAs). The existing LSEAs can be roughly categorized into metaheuristics and divide-and-conquer methods [18].

The metaheuristics LSEAs mainly focus on enhancing the searchability of algorithms. In other words, various schemes are designed to maintain population diversity in the large-scale space. For instance, Hansen and Ostermeier [19] developed a covariance matrix adaptation evolution strategy (CMA-ES) algorithm utilizing two evolution paths to preserve diversity. Ros and Hansen [20] proposed a variant of CMA-EA, called sep-CMA-ES, to alleviate the time cost associated with covariance matrix calculation in CMA-ES. Additionally, Molina et al. [21] introduced a memetic algorithm, named MA-SW-Chains, which assigns each search intensity to each individual through chaining different local search operators. LaTorre et al. [22] presented a multi-offspring generation framework (MOS) that combines a Genetic Algorithm (GA) with two local searches. Moreover, several algorithms based on particle swarm optimization (PSO) have been proposed. Specifically, Cheng and Jin [23] proposed a competitive particle swarm algorithm (CSO) employing a paired competition mechanism to generate the

✉ Jianchang Liu
liujianchang@ise.neu.edu.cn

1 State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang, China

2 College of Information Science and Engineering, Northeastern University, Shenyang, China

next population. They also presented a social learning particle swarm optimization algorithm (SL-PSO) [24], which updates the positions of particles by learning from those with better fitness values. Jian et al. [25] proposed a local search strategy, ARS, based on a novel region-based encoding scheme to enhance SL-PSO, forming the SLPSO-ARS. Wang et al. [26, 27] proposed two variants of distributed PSO (DPSO), named dynamic group learning DPSO (DGLDPSO) and adaptive granularity learning DPSO (AGLDPSO). These algorithms increase population diversity by dynamically changing the structure of the subpopulations. Furthermore, Ge et al. [28] developed a novel distributed differential evolution (DE) algorithm with an adaptive population model, named DDE-AMS. Hadi et al. [29] proposed a hybrid algorithm, MLSHADE-SPA, based on three DE strategies and a modified Multiple Trajectory Search (MTS). Later, Koçer and Uymaz [30] proposed an improved MLSHADE-SPA (IMLSHADE-SPA) with a novel local search method. Zhang and Lan [31] introduced a memetic algorithm, MPCE & SSALS, utilizing a multiparent crossover evolution strategy and a step-size adaptive local search method for local exploitation. Recently, Li et al. [32] proposed a dynamic sine cosine algorithm, named DSCA, which includes a nonlinear random convergence parameter to update the equation dynamically, balancing the exploration and exploitation of SCA. Sanjoy et al proposed an enhanced whale optimization algorithm (eWOA) [33]. In eWOA, a selection parameter is introduced, and the co-efficient vectors in the whale optimization algorithm (WOA) are modified.

In contrast to the first category, the divide-and-conquer LSEAs, mainly cooperative co-evolutionary EAs (CCEAs), decompose original LSGOPs into several sub-components and optimize them separately. CCEAs can effectively reduce the search space, which makes it more conducive to solving complex LSGOPs. The primary challenge in employing CC for solving LSGOPs lies in the choice of the problem decomposition method. Consequently, numerous attempts have been made to design effective decomposition methods. For instance, Van den Bergh and Engelbrecht [34] proposed a decision grouping method where a D-dimensional problem is evenly divided into k small-dimensional sub-problems. To consider the variable interdependencies, Yang et al. [35] proposed a random grouping (RG), which randomly decomposes variables into a fixed number of sub-components at each generation. Subsequently, they developed a method to determine the sub-component sizes from a provided set based on the probability of historical performance measures [36]. Omidvar et al. [37] proposed another competitive decomposition method, called delta grouping, which identifies the interacting variables by measuring the averaged difference in a certain variable across the entire population. It was also extended to an improved version that can adaptively determine the size of sub-components, as in [36]. However, these decomposition methods do not detect the underlying structure of variable interactions. To address this, Omidvar et al. [14] proposed a differential grouping (DG) decomposition method. In DG, the interactions between each variable are identified by detecting the fitness changes when perturbing the variables. Sun et al. [38] developed an extended differential grouping (XDG) method to detect the indirect interactions between variables. Additionally, Mei et al. [39] proposed a global differential grouping (GDG) method. Later, Omidvar et al. [40] presented an improved version of DG, named DG2, which exhibits better efficiency and grouping accuracy than DG. Recently, Sun et al. [41] developed a recursive differential grouping (RDG) method based on the bisection method, which significantly improved the efficiency of problem decomposition in terms of time complexity. Chen et al. [42] proposed an efficient adaptive differential grouping algorithm (EDGA) for large-scale black-box optimization problems. In addition, some algorithms [43–45] are proposed to allocate different computing resources based on the contribution of the sub-problems, and others [46, 47] based on modified CC technique are also developed.

The CCEAs have proven successful in addressing LSGOPs, which can effectively reduce problem difficulty through a divide-and-conquer approach. The full potential of CCEAs is realized by meticulously designing efficient decomposition and evolutionary search methods. As mentioned above, the decomposition methods can be classified into two different approaches, namely, the manual decomposition method [34–37] and the automatic decomposition method [14, 38–42]. In manual decomposition methods, the number of sub-components is manually determined. These methods work well on fully separable problems while exhibiting shortcomings in solving non-separable problems due to they do not identify the underlying structure of variable interactions. Conversely, automatic decomposition methods automatically assign decision variables to different sub-components based on their underlying structure, in which the interacting variables can be placed into the same sub-component. However, a drawback is that the decomposed sub-components will remain unchanged during the evolution, with only the best-so-far solutions exchanged between different of them. This makes the automatic decomposition methods insufficient to compensate for information. In particular, there are often challenges in accurately decomposing real-world problems. Moreover, these methods are also unsuitable for both fully separable and fully non-separable problems, as they will place only one variable in each sub-component or group all variables into one big sub-component. Therefore, an effective decomposition method that strikes a balance between the two approaches is needed. Compared with the decomposition method, the evolutionary search method is equally

crucial for enhancing CCEA performance, but limited attention has been devoted to improving these search methods.

To address the above issues, this paper proposes an evolutionary dynamic grouping based cooperative co-evolution algorithm, named EDGCC. The proposed EDGCC mainly aims to enhance the informative collaborators among subcomponents and improve the searchability of the algorithm in solving LSGOPs. The major contributions of this paper are summarized as follows:

(1) An evolutionary dynamic grouping (EDG) method is proposed. In EDG, the sub-components are dynamically generated based on the designed selection, crossover, mutation operations, and an adaptive frequency scheme. It considers both the underlying structure of variable interactions and the transfer of information between different sub-components.
(2) An evolutionary search method based on the fireworks search strategy is proposed. In this strategy, the operations of generating offspring populations in the fireworks algorithm are first time introduced to the CCEAs for LSGOPs. It is performed after each cycle of sub-component optimization, which works with the sub-component optimizer to enhance the diversity of the population.
(3) An evolutionary dynamic grouping-based cooperative co-evolution (EDGCC) algorithm is proposed for LSGOPs. To demonstrate the effectiveness of EDGCC, a comprehensive empirical study has been conducted. The experimental results indicate that the proposed algorithm achieves promising performance compared with four state-of-the-art methods.

The rest of this paper is organized as follows. In Section 2, a review of related work on the proposed algorithm is provided. In Section 3, the details of the proposed algorithm are described. Experiments are conducted in Section 4. Finally, the conclusions and future work are given in Section 5.

2 Related work

In this section, first, a brief introduction to the LSGOP is given. Then, the idea of the CC technique is elaborated. Finally, the firework algorithm is described.

2.1 Large-scale global optimization problems

Without loss of generality, an LSGOP can be formulated as follows:

$$minf\left(\mathrm{x}\right), \mathrm{x} = [x_1, x_2, \cdots, x_D] \in \mathcal{X}. \tag{1}$$

where $f\left(\mathrm{x}\right)$ is the objective function, $\mathrm{x} = [x_1, x_2, \cdots, x_D]$ denotes a D-dimensional decision vector, $\mathcal{X} \in \mathbb{R}^D$ indicates the feasible solution set. In the LSGOP, $D \geq 1000$ [48].

There are three categories for LSGOPs according to the interaction relationship among variables, namely, fully separable LSGOP, partially separable LSGOP, and fully non-separable LSGOP. The separable LSGOP is defined as follows:

$$\begin{aligned} &\underset{\mathrm{S}}{\mathrm{arg}min} f\left(\mathrm{x}\right) = \\ &\left(\underset{\mathrm{S_1}}{\mathrm{arg}min} f\left(x_1, \cdots\right), \cdots, \underset{\mathrm{S_m}}{\mathrm{arg}min} f\left(\cdots, x_m\right)\right). \end{aligned} \tag{2}$$

where $\{\mathrm{S}_1, \mathrm{S}_2 \cdots, \mathrm{S_m}\}$ represents the m disjoint sub-components of x. If $m = D$, the LSGOP defined by (2) is a fully separable LSGOP, and if $m = 1$, the LSGOP defined by (2) is a fully non-separable LSGOP. However, LSGOP defined by (2) is a partially separable LSGOP when $m \neq D$ and $m \neq 1$.

2.2 Cooperative co-evolution technique

The cooperative co-evolution(CC) technique is a popular approach for tackling large-scale optimization problems [49–51]. It is attributed to the divide-and-conquer strategy, which decomposes a large-scale problem into numbers of smaller sub-problems and optimizes them separately. The performance of CCEAs for solving LSGOPs is sensitive to the choice of decomposition method. As reviewed in Section 1, a number of decomposition methods have been proposed. In this subsection, we detail the recently developed recursive differential grouping (RDG) method [41], which can efficiently and accurately decompose the problems based on the interaction of decision variables. The detailed description of RDG is as follows:

Notation: Let X be the set of decision variables $\{x_1, \cdots, x_D\}$ and U_X be the set of unit vectors in the decision space $\mathbb{R}^D$. Let X_1 be a subset of variables X and U_{X_1} be a subset of unit vectors U_X. For any unit vector $\mathbf{u} = (u_1, \cdots, u_D) \in U_{X_1}$, we have

$$u_i = 0, \text{if} x_i \notin X_1. \tag{3}$$

Theorem: Let $f : \mathbb{R}^D \to \mathbb{R}$ be an objective function; $X_1 \subset X$ and $X_2 \subset X$ be two mutually exclusive subsets of decision variables: $X_1 \cap X_2 = \phi$. There is some interaction between decision variables in X_1 and X_2, if there exist two unit vectors $\mathbf{u}_1 \in U_{X_1}$ and $\mathbf{u}_2 \in U_{X_2}$, two real numbers $l_1, l_2 > 0$, and a candidate solution $\mathbf{x}^*$ in the decision space, that satisfied the following:

$$\begin{aligned} &f\left(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_2\right) - f\left(\mathbf{x}^* + l_2\mathbf{u}_2\right) \neq \\ &f\left(\mathbf{x}^* + l_1\mathbf{u}_1\right) - f\left(\mathbf{x}^*\right). \end{aligned} \tag{4}$$

If there is an interaction between X_1 and X_2, X_2 will be further divided into two equal-sized and mutually exclusive subsets. Then, the interactions between X_1 and these subsets are examined. The above process is repeated until RDG finds all the variables that interact with X_1. However, if the formula is not held, X_1 and X_2 will be determined as mutually separable sets.

In RDG, the interrelationship is examined between a pair of sets of variables but not a pair of variables. During this binary search procedure, if there is not an interaction between X_1 and a subset, RDG does not further examine the interrelationship between X_1 and the variables in this subset. Therefore, RDG greatly improved the efficiency of the time complexity compared with DG [14] and DG2 [40]. According to [41], the computational complexity of RDG is $O\left(D\log_2 D\right)$ for decomposing a D-dimensional problem, while DG is $D^2 + D$, and DG2 is $\left[\left(D^2 + D + 2\right)/2\right]$.

2.3 Fireworks algorithm

The fireworks algorithm (FWA) [52] is a swarm intelligence algorithm inspired by the phenomenon of fireworks explosion in the night sky. Liu et al. [53] have presented theoretical analysis and proved that FWA is an absorbing Markov stochastic process. In recent years, there are many pieces of research for FWA in solving practical problems [54, 55].

FWA mainly follows the general framework of EAs. However, it proposes a new search manner that imitates the process of fireworks explosion to search the potential space by a stochastic explosion process within the local space. In FWA, there are three main operations, namely, explosion, Gaussian mutation, and selection. At first, several fireworks are initialized randomly. Then, explosion sparks of these fireworks are generated by the explosion operation. To maintain the diversity of the population and balance the global and local search, the explosion amplitude and the population of the generated sparks are different between fireworks. Specifically, a firework with lower fitness has a higher explosion amplitude, which generates a smaller population within a more extensive range. On the contrary, a firework with better fitness will have a lower explosion amplitude and a larger population. In other words, exploration is achieved by those fireworks with a large explosion amplitude to escape from local minima, while exploitation is achieved by those fireworks with a small explosion amplitude to reinforce the local search ability in promising areas. Moreover, the mutation operation mutates the locations of the fireworks to generate mutation sparks, which are used to further improve the diversity of the swarm. Finally, the selection operation selects among the set of the generated two types of sparks and the original fireworks for the next generation.

3 The proposed algorithm

In this section, the main framework of the proposed algorithm EDGCC is first elaborated. Then, the evolutionary dynamic grouping (EDG) method and the fireworks search strategy as two critical components of EDGCC are introduced in detail.

3.1 Main framework of EDGCC

The main framework of the proposed EDGCC is given in Algorithm 1. It starts by randomly initializing a population, denoted as P, with size N, in which each solution consists of D decision variables. Then, the RDG [41] method is employed to detect the interaction of all decision variables. The RDG method obtains several non-separable variable groups and a separable variable group, which are denoted as $nonseps$ and $seps$, respectively.

In the main loop, to begin with, the sub-components are generated by the EDG method. Subsequently, all sub-components are optimized by the sub-component optimizer one by one. Meanwhile, the function value improvement ΔF of each sub-component is calculated by subtracting the function values of the best solutions in the original population and the optimized new population. This is utilized for the selection operation of the EDG method. Finally, the fireworks search strategy is executed to further enhance the exploration and exploitation of the sub-component optimizer. It is essential to note that the meaning of parameters in the fireworks search strategy is shown in Table 1. The above optimization process is iterated until the termination condition is satisfied. The subsequent sections elaborate on the EDG method and the fireworks search strategy.

3.2 Evolutionary dynamic grouping method

In this subsection, the designed evolutionary dynamic grouping (EDG) method is elaborated. Its main idea is to generate sub-components dynamically, just like the implementation process of the genetic algorithm. Algorithm 2 presents the details of the designed EDG method, which includes two parts, i.e., sub-component initialization and sub-component evolution.

At the beginning of EDG, the decision variables are decomposed into several sub-components, which are used for the initial optimization. During this process, the sub-components are decomposed based on the variables preprocessing by the RDG method. Depending on the detection results of the RDG method, the decision variables are divided into three situations corresponding to different sub-component decomposition methods: 1) for fully separable or fully non-separable problems, all variables are randomly and uniformly divided into several sub-components, where

Table 1 The parameters of fireworks search strategy

character	meaning
$snum$	the number of solutions selected for "explosion sparks"
$gnum$	the number of solutions selected for "Gaussian sparks"
M_e	the control parameter for the number of "explosion sparks"
a	the control parameter for the maximum number of the "explosion sparks"
b	the control parameter for the minimum number of the "explosion sparks"
$\hat{A}$	the maximum explosion amplitude
A_{init}	the initial minimum explosion amplitude
A_{final}	the final minimum explosion amplitude

Algorithm 1 Main Framework of the Proposed EDGCC.

Require: N (population size), f (fitness function with decision variables X_D), ub(upper bounds), lb (lower bounds), ϵ (the threshold), s (the preset sub-components dimensions), m (the number of groups divided in each sub-component), $snum$, $gnum$, M_e, a, b, $\hat{A}$, A_{init}, A_{final} (the parameters of fireworks search strategy)
Ensure: P (final population)
$cyc = 0$;
$S = \phi$;
$\Delta F = \phi$;
$P \leftarrow$ Initialize population with N solutions;
$[nonseps, seps] \leftarrow RDG(f, \text{ub}, \text{lb}, \epsilon)$;
while $termination\ condition\ is\ not\ reached$ **do**
  $subcomponent_{1:n} \leftarrow EDG(cyc, \Delta F, nonseps, seps, s, m)$//Algorithm 2;
  for $i = 1 : n$ **do**
    $P \leftarrow Optimizer(P, f, subcomponent_i)$;
    $\Delta F_i = best - bestnew$;
  end for
  $cyc = cyc + 1$;
  $P \leftarrow FireworksSearchStrategy(P, f, snum, gnum, M_e, \hat{A}, a, b, A_{init}, A_{final})$//Algorithm 3;
end while
return P

the dimension of each sub-component is s; 2) for a partially separable problem and the number of separable variables $|seps|$ is half or more of D, each non-separable group is regarded as a sub-component, and the separable variables are randomly and uniformly divided into $|nonseps|$ sub-components; 3) otherwise, each non-separable group is regarded as a sub-component, and all separable variables are placed in one sub-component. It is worth noteworthy that the first two situations in the above variable decomposition differ from the RDG method, while the third situation is the same as the RDG method. This distinction is intentional. Specifically, the sub-components in the first situation are advantageous for the crossover and selection operators, whereas the sub-components decomposed by the RDG method make crossover and selection operations impractical. The second situation aims to prevent an imbalance in the number of variables between the separable variable sub-component and other sub-components, which easily appears in the RDG method. Thus, the sub-component of separable variables is further divided in the second situation. Following sub-component decomposition, the variables within each sub-component are further randomly divided into m groups, which serve as the genes for the crossover operation.

After the sub-component initialization, sub-component evolution is performed to dynamically generate the sub-components that are conducive to improving the searchability, as the optimization process proceeds. To visually present the sub-component evolution, Fig. 1 visually illustrates the sub-component evolution, which contains the selection, crossover, and mutation operations. Specifically, two sub-components (i and j) are first selected based on having the smallest function value improvement ΔF. The motivation for such selection is the contribution of these two sub-components to the optimization is poor, and thus the variables of these two sub-components should be decomposed again. Subsequently, the crossover operation is performed to generate new sub-components by exchanging variables in a randomly chosen group k from the selected sub-components i and j. Finally, a mutation operation is proposed to enhance the variable diversity in the groups. In this operation, a group g of the sub-component u is randomly selected, and a random variable x_d is added to this group.

In general, the early phase of an algorithm should emphasize the exploration ability, while the later stages should emphasize the exploitation ability. In EDG, the sub-components are dynamically generated to enhance the transfer of information between different sub-components as optimization progresses. A higher frequency of information transfer implies that more information is utilized to guide the generation of offspring populations, resulting in improved exploration ability. Conversely, a lower frequency of information transfer is more conducive to enhancing the exploitation ability of the algorithm. Therefore, an adaptive frequency scheme of the sub-component evolution is designed to adjust the search focus. In this scheme, the frequency of sub-component evolution is higher in the early stages of the algorithm and decreases as the algorithm pro-

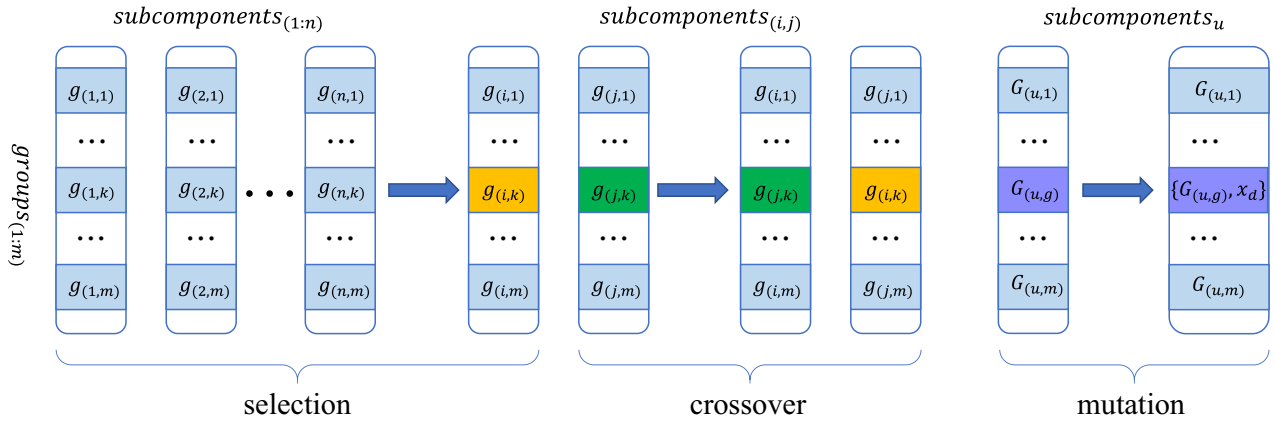


Fig. 1 The diagram of sub-components evolution, which contains the selection, crossover and mutation operations

gresses. Specifically, if the cycle number of sub-component optimization belongs to an assigned set $\hat{S}$, the sub-component evolution is performed. The cycle number set $\hat{S}$ is assigned in the following manner:

$$\hat{S} = \left\{\hat{S};\ (cyc^2 + cyc + 2)/2\right\},\ Wang2022 \tag{5}$$

where cyc is the number of sub-component optimization cycles. In other words, $cyc = \{1, 2, \cdots, cyc_{max}\}$, where cyc_{max} represents the maximum value of the cycle number when the termination condition is reached. To have a clear understanding of it, Fig. 2 gives an illustration. Figure 2(a) shows a part of the function $f(x) = (x^2 + x + 2)/2$ and Fig. 2(b) gives the values of $f(x)$ when x is set to $x = \{0, 1, 2, 3, 4, 5\}$. It is clear that the interval between the function values of two adjacent variables gradually increases. Therefore, the probability of sub-component evolution is greater at the early stage of the optimization process, while the possibility gets smaller as the optimization proceeds.

3.3 Fireworks search strategy

After the EDG method, the sub-components are optimized by the sub-components optimizer. To further enhance the performance of the algorithm, an evolutionary search method based on the fireworks search strategy is performed. Algorithm 3 provides the details of the fireworks search strategy. At first, $snum$ solutions in the current population P are randomly selected as the initial fireworks. Their quality (i.e., fitness) is evaluated to determine the number of explosion sparks and the explosion amplitudes. Then, the fireworks explode to generate several "explosion sparks" within their local space. Later, the "Gaussian sparks" are generated by conducting the search in a local Gaussian space around $gnum$ randomly selected fireworks. It should be noted that if the generated spark exceeds the search range, it will be mapped to another location in the search space by a mapping strategy. Finally, to retain the information and pass it to the next iteration, the

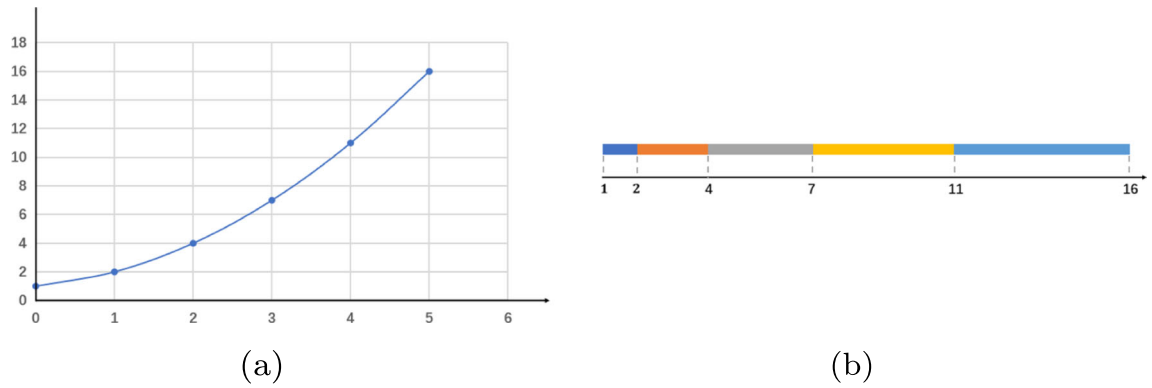


Fig. 2 The illustration of the function used in the adaptive frequency scheme. (a) The function image. (b) The function values

Algorithm 2 The Evolutionary Dynamic Grouping Method.

Require: cyc (the cycle number of sub-component optimization), $nonseps$ (the non-separable groups divided by RDG), $seps$ (the separable variables divided by RDG), s (the preset sub-components dimensions), m (the number of groups divided in each sub-component), ΔF (the function value improvement of each sub-component)

Ensure: $subcomponent$ (the regenerated sub-components)

if $cyc = 0$ **then**
  if $|seps| = D$ or $|nonseps| = 1 \cap |seps| = 0$ **then**
    $n = ceil(D/s)$;
    $subcomponent \leftarrow Randomgroup(X_D, n)$;
  else
    if $|seps| >= D/2$ **then**
      $subseps \leftarrow Randomgroup(seps, |nonseps|)$;
      $n = |nonseps| + |nonseps|$;
      $subcomponent \leftarrow \{nonseps; subseps\}$;
    else
      $n = |nonseps| + 1$;
      $subcomponent_{1:n} \leftarrow \{nonseps; seps\}$;
    end if
  end if
  for $i = 1 : n$ **do**
    $groups \leftarrow Randomgroup(subcomponent_i, m)$;
  end for
end if
$\hat{S} \leftarrow \left\{\hat{S}; (cyc^2 + cyc + 2)/2\right\}$;
if $cyc \in S$ **then**
  Sort the ΔF in ascending order;
  $\Delta F(1) = \Delta F_i$;
  $\Delta F(2) = \Delta F_j$;
  Randomly select $k \in [1, m]$;
  $subcomponent_i^{'}(group_k) = subcomponent_i(group_k)$;
  $subcomponent_i(group_k) = subcomponent_j(group_k)$;
  $subcomponent_j(group_k) = subcomponent_i^{'}(group_k)$;
  Randomly select $u \in [1, n]$, $g \in [1, m]$, $d \in [1, D]$;
  $subcomponent_u(group_g) = \left\{subcomponent_u(group_g); d\right\}$;
end if
return $subcomponent$

new solutions are selected from the set of the initial fireworks and generated sparks to replace the original solutions of P.

In this paper, the operations of an enhanced version of FWA (EFWA) [56] is introduced to the fireworks search strategy. Compared to the traditional FWA, EFWA incorporates five modifications. Specifically, for each dimension k, the explosion amplitude A_i^k is bound as follows:

$$A_i^k = \begin{cases} A_{min}^k \ if A_i^k < A_{min}^k, \\ A_i^k \quad otherwise, \end{cases} \tag{6}$$

where,

$$A_{min}^k(t) = A_{init} - \frac{A_{init} - A_{final}}{evals_{max}} \sqrt{(2 * evals_{max} - t)\, t}, \tag{7}$$

and

$$A_i = \hat{A} \cdot \frac{f(X_i) - y_{min} + \varepsilon}{\sum_{i=1}^{N} (f(X_i) - y_{min}) + \varepsilon}, \tag{8}$$

t represents the number of function evaluations, $evals_{max}$ denotes the maximum number of evaluations, A_{init} and A_{final} are the initial and final minimum explosion amplitude, $\hat{A}$ represents the constant to control the explosion amplitude, $y_{max} = max(f(X_i))$, $y_{min} = min(f(X_i))$, ε is the machine epsilon. The lower bound A_{min} of the explosion amplitude is proposed to prevent the explosion amplitude from being too small that the locations of the explosion sparks are almost the same as the firework themselves. This paper uses the non-linear decreasing function as A_{min}. In the early stages of the search, A_{min} is set to a high value to facilitate exploration. As the number of evaluations increases, it is lowered to allow for better exploitation capabilities around good locations. Moreover, the number of explosion sparks s for each firework X_i is calculated as follows:

$$s_i = M_e \cdot \frac{y_{max} - f(X_i) + \varepsilon}{\sum_{i=1}^{N} (y_{max} - f(X_i)) + \varepsilon}, \tag{9}$$

where M_e represents the constant to control the number of explosion sparks. To avoid the decisive influence of the fireworks in good positions, the number of sparks is determined by the following:

$$s_i = \begin{cases} round(aM_e) \ \text{if} s_i < aM_e, \\ round(bM_e) \ \text{if} s_i > bM_e, \\ round(s_i) \quad otherwise. \end{cases} \tag{10}$$

where a and b are constant parameters that confine the range of the "explosion sparks" size.

Furthermore, the operation that generates the "explosion sparks" is to add different offset displacements ΔX^k in each dimension. And the Gaussian mutation operation uses the $\tilde{X}_i^k = \tilde{X}_i^k + (\tilde{X}_B^i - \tilde{X}_i^k) * e$ to generate "Gaussian sparks", where X_B is the location of the currently best firework or explosion spark found so far, and $e = \mathcal{N}(0, 1)$. This mutation operation will expand along the direction between the current firework position and the optimal firework position, which ensures diversity in the search as well as the global movement to find the best location found so far. If the generated sparks are out of bounds, the uniform random mapping strategy $\bar{X_i^k} = X_{min}^k + rand * \left(X_{max}^k - X_{min}^k\right)$ is used. This mapping strategy can avoid focusing the locations of the mapping on the origin.

In addition, the $Elitism - Randomselection$ (ERP) [57] operation is introduced to select the solution. In this operation, the best candidate solution is selected first. Then, the other individuals are selected randomly.

Algorithm 3 The Fireworks Search Strategy.

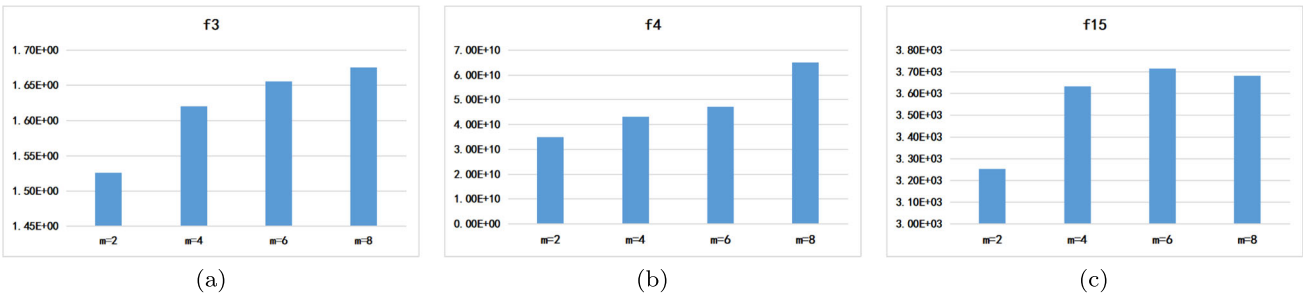
Require: P (current population), $snum$, $gnum$, M_e, a, b, $\hat{A}$, A_{init}, A_{final} (the parameters of fireworks search strategy)
Ensure: P (final population)
Randomly select $snum$ solutions X from P as the initial "explosion sparks" $\hat{X}$;
Set $z^k = round(rand(0,1)), k = 1, 2, \cdots, D$;
for $i = 1 : sum$ **do**
  for each dimension of $\hat{X}_i^k$, where $z^k == 1$ **do**
    $\Delta X^k = A_i \times rand(-1, 1)$;
    $\hat{X}_i^k = \hat{X}_i^k + \Delta X^k$;
    if $\hat{X}_i^k$ out of bounds **then**
      map $\hat{X}_i^k$ to potential space;
    end if
  end for
end for
for $j = 1 : gnum$ **do**
  Randomly select one solutions X as the initial "Gaussian sparks" $\tilde{X}$;
  $e = Gaussian(0, 1)$;
  for each dimension of $\tilde{X}_i^k$, where $z^k == 1$ **do**
    $\tilde{X}_i^k = \tilde{X}_i^k + (X_B^k - \tilde{X}_i^k) * e$, where X_B is the position of the best firework found so far;
    if $\tilde{X}_i^k$ out of bounds **then**
      map $\tilde{X}_i^k$ to potential space;
    end if
  end for
end for
$P \leftarrow$ Select the solutions from the X as well as all explosion and Gaussian sparks to replace the original solutions X of original P;
return P

4 Experimental study

In this section, a comprehensive experimental study is conducted to examine the performance of the proposed EDGCC. First, the experimental settings and the parameter sensitivity analysis are presented. Subsequently, a set of experiments are conducted to evaluate the performance of the proposed EDGCC. Finally, a discussion is given.

4.1 Experimental settings

1) Compared Algorithms: Four recently proposed state-of-the-art LSEAs, namely, DSCA [32], eWOA [33], DECC-RDG [41] and EDGA [42] are used for comparison. The DSCA and eWOA are the metaheuristics LSEAs. The EDGA and DECC-RDG are the divide-and-conquer LSEAs.
2) Benchmark Suites: Two widely used large-scale global optimization benchmark suites are used. The first one is IEEE CEC'2010 [58], which contains 20 test functions. These functions can be classified into fully separable LSGOPs ($f_1 - f_3$), partially separable LSGOPs ($f_4 - f_{18}$), and fully non-separable LSGOPs ($f_{19} - f_{20}$). The other one is IEEE CEC'2013 [59], which contains 15 test functions. These functions can be classified into fully separable LSGOPs ($f_1 - f_3$), partially separable LSGOPs ($f_4 - f_{11}$), and fully non-separable LSGOPs ($f_{12} - f_{15}$). The number of decision variables of each test functions is 1000 in both benchmark suites except for the functions f_{13} and f_{14} in CEC'2013, have 905 decision variables.
3) Parameters: In the EDG method, as recommended by [35], s is set to 100. Moreover, the group number of each sub-component m is set to 2. In the fireworks search strategy, all parameters are set the same as recommended in [56], where $snum = 5$, $gnum = 5$, $M_e = 50$, $\hat{A} = 40$, $a = 0.04$, $b = 0.8$, $A_{init} = \left(X_{max}^k - X_{min}^k\right) \times 0.02$, and $A_{final} = \left(X_{max}^k - X_{min}^k\right) \times 0.001$.
4) Stopping Condition and Population Size: The population size N of the algorithms is set to 50. In each run, the maximum number of function evolutions for each algorithm is set to 3×10^6.
5) Performance Metrics: The mean and standard deviation of the function value obtained from 25 independent runs on each problem. To test the difference for statistical sig-



(a) (b) (c)

Fig. 3 The mean function values of different parameter m on f_3, f_4, f_{15} of CEC'2010. Statistical results are obtained by 25 independent runs. (a) The function values on f_3. (b) The function values on f_4. (c) The function values on f_{15}

Table 2 Comparisons among EDGCC and compared algorithms on the CEC'2010 benchmark functions

Functions	Quality	eWOA	DSCA	EADG	DECC-RDG	EDGCC
f1	Mean	1.86E+11 −	4.39E+11 −	2.44E+05 −	2.07E+00 −	**2.79E–19**
	Std	8.76E+09	7.24E+10	5.41E+05	6.75E+00	**6.69E–19**
f2	Mean	1.65E+04 −	2.67E+04 −	4.30E+03 −	4.38E+03 −	**3.80E+02**
	Std	1.03E+02	3.24E+03	4.09E+02	1.72E+02	**5.52E+01**
f3	Mean	2.10E+01 −	2.16E+01 −	1.11E+01 −	1.65E+01 −	**1.53E+00**
	Std	1.67E−02	4.36E−02	1.10E+00	3.35E−01	**2.01E–01**
f4	Mean	6.52E+13 −	2.00E+14 −	3.26E+10 −	6.68E+11 −	**3.50E+10**
	Std	1.52E+13	9.15E+13	2.55E+10	3.33E+11	**1.97E+10**
f5	Mean	2.45E+08 −	3.51E+08 −	**6.59E+07** +	1.28E+08 −	7.77E+07
	Std	3.07E+07	1.70E+07	**1.24E+07**	1.92E+07	1.88E+07
f6	Mean	7.92E+06 −	9.55E+06 −	3.52E+04 −	**1.61E+01** ≈	1.66E+01
	Std	2.40E+05	8.69E+05	1.76E+05	**3.64E−01**	2.28E+00
f7	Mean	1.09E+12 −	5.30E+12 −	1.15E+04 −	2.16E+01 −	**1.65E–02**
	Std	4.51E+01	3.10E+12	5.90E+03	7.56E+01	**5.37E–03**
f8	Mean	4.74E+16 −	2.31E+17 −	3.28E+05 −	1.15E+04 −	**9.99E–03**
	Std	4.96E+15	1.12E+17	1.10E+06	5.90E+03	**4.25E–03**
f9	Mean	9.07E+10 −	2.27E+11 −	3.72E+07 −	3.72E+07 −	**1.11E+07**
	Std	4.01E+09	4.40E+10	2.79E+07	2.79E+06	**1.17E+06**
f10	Mean	1.22E+04 −	1.55E+04 −	3.20E+03 −	4.33E+03 −	**1.93E+03**
	Std	9.84E+01	2.00E+03	1.36E+02	1.39E+02	**4.18E+02**
f11	Mean	1.11E+02 −	1.38E+02 −	2.58E+01 ≈	**1.03E+01** +	2.65E+01
	Std	2.51E−01	7.41E+00	2.83E+00	**8.50E-01**	2.09E+00
f12	Mean	2.03E+07 −	4.85E+07 −	2.66E+04 −	1.53E+03 −	**1.04E+01**
	Std	3.73E+06	1.31E+07	1.08E+04	4.66E+02	**3.92E+00**
f13	Mean	6.78E+11 −	4.14E+12 −	1.33E+04 −	7.12E+02 −	**6.27E+02**
	Std	4.91E+09	2.23E+12	5.24E+03	2.52E+02	**1.41E+02**
f14	Mean	1.06E+11 −	2.18E+11 −	**2.15E+07** +	3.47E+08 −	2.36E+07
	Std	5.53E+09	1.03E+10	**1.64E+06**	2.31E+07	2.73E+06
f15	Mean	3.66E+03 −	4.18E+03 −	**2.72E+03** +	5.84E+03 −	3.25E+03
	Std	1.55E+02	6.97E+01	**2.48E+02**	1.01E+02	6.08E+02
f16	Mean	8.76E+01 −	1.13E+02 −	1.84E+01 ≈	**2.67E-13** +	1.87E+01
	Std	1.59E−01	6.70E+00	3.08E+00	**9.81E-15**	4.34E+00
f17	Mean	1.87E+07 −	4.55E+07 −	8.30E+00 +	4.07E+04 −	**1.14E+01**
	Std	6.04E+06	3.76E+06	3.00E+00	2.55E+03	**2.28E+00**
f18	Mean	7.70E+11 −	4.88E+12 −	1.16E+03 ≈	1.20E+03 −	**1.09E+03**
	Std	6.67E+09	6.08E+11	1.62E+02	1.07E+02	**1.41E+02**
f19	Mean	1.76E+08 −	4.31E+08 −	**9.04E+05** +	1.71E+06 +	4.47E+06
	Std	4.70E+07	1.37E+08	**4.99E+04**	8.91E+04	3.82E+06
f20	Mean	1.61E+12 −	1.12E+13 −	7.43E+07 −	6.96E+03 −	**2.06E+03**
	Std	1.25E+10	1.51E+12	2.29E+08	1.27E+04	**1.40E+02**
No.best		0	0	4	3	13
+ / −/ ≈		0/20/0	0/20/0	5/12/3	3/16/1	

Statistical results are obtained by 25 independent runs
The bold entries are the best results that obtained by the tested algorithm

nificance, the Wilcoxon rank-sum test with a significance level of 0.05 is employed to assess whether one algorithm is better than another in terms of the function value. The symbols " + ", " − ", " ≈ " are used to indicate that the compared algorithm is significantly better than, worse than, and similar to EDGCC, respectively.

The sub-component optimizer in this paper is implemented by SaNSDE [60], which is the same as the compared DECC-RDG. It should be noted that only m is the parameter introduced by the proposed algorithm. The sensitivity analysis of m is given in the following subsection.

4.2 Parameter sensitivity analysis

In the proposed EDGCC, there is a major parameter, i.e., the group number of each sub-component m, which is used to determine the number of exchange variables in the crossover operation of the EDG method. In this study, some comparisons are designed to study the influence of the above parameter on the performance of EDGCC. Specifically, four values for m are chosen to test the performance of these different parameter settings.

Figure 3 plots the mean function values of each parameter, where m is set to 2,4,6,8 on f_3, f_4, and f_{15} of CEC'2010 benchmark suit, respectively. From Fig. 3, it can be seen

Table 3 Comparisons among EDGCC and compared algorithms on the CEC'2013 benchmark functions

Functions	Quality	eWOA	DSCA	EADG	DECC-RDG	EDGCC
f1	Mean	1.92E+11 −	3.59E+11 −	1.02E+06 −	3.73+01 −	**8.06E–22**
	Std	6.84E+09	1.17E+11	2.25E+06	1.24+02	**8.30E–22**
f2	Mean	4.31E+04 −	1.20E+05 −	1.11E+04 −	1.27+04 −	**1.01E+02**
	Std	2.80E+02	6.05E+04	1.65E+03	6.40+02	**1.90E+01**
f3	Mean	2.10E+01 −	2.15E+01 −	2.06E+01 ≈	2.13+01 −	**2.01E+01**
	Std	2.10E−02	9.29E−02	9.07E−03	1.64−02	**1.58E–03**
f4	Mean	1.84E+13 −	5.01E+13 −	**2.97E+08** +	4.44+10 −	1.05E+09
	Std	8.40E+12	3.71E+07	**1.51E+08**	1.77+10	1.33E+09
f5	Mean	4.02E+07 −	8.92E+07 −	**2.27E+06** +	5.09+06 −	2.53E+06
	Std	2.23E+06	3.71E+07	**3.06E+05**	4.81+05	4.26E+05
f6	Mean	**1.03E+06** ≈	1.05E+06 ≈	1.06E+06 ≈	1.06+06 ≈	1.06E+06
	Std	**2.05E+03**	8.44E+03	2.15E+03	1.21+03	2.25E+03
f7	Mean	1.80E+14 −	1.27E+16 −	**6.10E+05** +	6.42+07 −	3.18E+07
	Std	7.79E+13	1.11E+16	**2.70E+06**	2.97+07	2.12E+07
f8	Mean	6.76E+17 −	3.16E+18 −	**9.20E+13** +	5.04+15 −	3.55E+14
	Std	4.76E+17	1.17E+18	**5.35E+13**	1.86+15	2.96E+14
f9	Mean	3.24E+09 −	5.98E+09 −	2.67E+08 −	4.82+08 −	**2.24E+08**
	Std	4.10E+08	1.16E+09	7.54E+07	3.06+07	**8.25E+07**
f10	Mean	**9.27E+07** +	9.58E+07 −	9.43E+07 ≈	9.44+07 ≈	9.43E+07
	Std	**8.19E+05**	2.94E+05	2.71E+05	2.06+05	4.35E+05
f11	Mean	6.82E+15 −	4.56E+18 −	1.58E+10 −	**5.38+08** +	1.04E+09
	Std	3.61E+15	6.17E+18	3.60E+10	**1.34+08**	1.92E+08
f12	Mean	1.68E+12 −	1.03E+13 −	4.44E+07 −	4.85+03 −	**1.63E+03**
	Std	1.15E+10	5.05E+12	1.83E+08	3.06+03	**1.13E+02**
f13	Mean	7.98E+15 −	2.35E+18 −	5.29E+08 −	3.06+09 −	**4.71E+08**
	Std	2.46E+15	4.36E+18	2.18E+08	6.68+08	**2.12E+08**
f14	Mean	1.71E+16 −	4.21E+17 −	**6.32E+08** +	2.87+09 +	4.00E+11
	Std	3.98E+15	4.23E+17	**7.70E+08**	1.73+09	2.36E+11
f15	Mean	2.13E+11 −	1.18E+18 −	**5.27E+06** +	9.75+06 +	5.99E+08
	Std	8.79E+10	1.28E+18	**1.70E+06**	1.91+06	3.37E+08
No.best		2	0	6	1	6
+ / −/ ≈		1/13/1	0/14/1	6/6/3	3/10/2	

Statistical results are obtained by 25 independent runs
The bold entries are the best results that obtained by the tested algorithm

that the different parameters can result in various performances on test benchmark problems. Generally, the function value increases as the increase of m. This is attributed to the difference between sub-components generated by the crossover operator and the original sub-components gradually decreases as the increase of m. Consequently, it can be recommended that $m = 2$ can be used as a general parameter setting for the proposed EDGCC.

4.3 Results on benchmark suits

In this subsection, the comparison experiments on benchmark suits are conducted. Tables 2 and 3 present the mean and standard deviation of the function value obtained from 25 independent runs on the CEC'2010 and CEC'2013 benchmark suite, respectively. The best result on each function is shown in the bold typeface.

As shown in Tables 2 and 3, EDGCC demonstrates superior performance, which achieves the best results on 19 out of 35 functions across two benchmark suites. Followed by EADG, which achieves the best on 10 functions. Moreover, DECC-RDG achieves the best on 4 functions, eWOA achieves the best on 2 functions, and DSCA on none. Specifically, for CEC'2010, EDGCC outperforms others in 13 functions, while EADG and DECC-RDG lead in 4 and 3 functions, respectively. For CEC'2013, both EDGCC and EADG excel in 6 functions, while eWOA and DECC-RDG demonstrate superiority in 2 and 1 functions, respectively. In general, the effectiveness of EDGCC and the other two CCEAs surpasses that of the compared metaheuristic algorithms. This can be attributed to the fact that the CCEAs use variable decomposition methods to reduce the search space.

Detailed view of each algorithm, the proposed EDGCC outperforms them in most functions. Specifically, in comparison to other metaheuristic algorithms, EDGCC is significantly better than eWOA only except for f_6 and f_{10} in CEC'2013. Similarly, EDGCC outperforms DSCA across all functions except for f_6 in CEC'2013. In comparison to the other CCEAs, EDGCC is better than EADG on 18 out of 35 functions and is statistically similar to EADG on 6 functions. Moreover, EDGCC outperforms DECC-RDG on 26 out of 35 functions and is defeated on only 6 functions.

The performance of EDGCC in solving fully separable LSGOPs (i.e., $f_1 - f_3$ in CEC'2010 and CEC'2013) and fully non-separable LSGOPs (i.e., $f_{19} - f_{20}$ in CEC'2010 and $f_{12} - f_{15}$ in CEC'2013) is better than all the compared algorithms. Additionally, for partially separable LSGOPs (i.e., $f_4 - f_{18}$ in CEC'2010 and $f_4 - f_{11}$ CEC'2013), the proposed EDGCC also performs well on the majority of them.

The above statistical comparisons confirm that the proposed EDGCC is more effective in solving the LSGOPs than the compared state-of-the-art LSEAs. This can be attributed to the fact that adequate information exchange between dif-

Table 4 Comparisons among EDGCC and two variants on the CEC'2010 benchmark functions

Functions		FCC	CC-EDG	EDGCC
f1	Mean	3.30E+00 −	5.20E−19 −	**2.79E–19**
	Std	6.50E+00	5.81E−19	**6.69E–19**
f2	Mean	3.37E+03 −	**7.24E+00** +	3.80E+02
	Std	2.43E+02	**4.51E+00**	5.52E+01
f3	Mean	7.38E+00 −	2.62E+00 −	**1.53E+00**
	Std	1.04E+00	1.65E−01	**2.01E–01**
f4	Mean	4.47E+10 −	3.60E+10 ≈	**3.50E+10**
	Std	1.97E+10	1.46E+10	**1.97E+10**
f5	Mean	7.96E+07 −	8.60E+07 −	**7.77E+07**
	Std	1.32E+07	1.78E+07	**1.88E+07**
f6	Mean	**1.64E+01** ≈	1.57E+05 −	1.66E+01
	Std	**1.37E+00**	4.28E+05	2.28E+00
f7	Mean	1.97E−02 −	1.90E+04 −	**1.65E–02**
	Std	1.31E−02	6.80E+03	**5.37E–03**
f8	Mean	1.22E−02 −	8.06E+05 −	**9.99E–03**
	Std	7.85E−03	1.65E+06	**4.25E–03**
f9	Mean	1.41E+07 −	**1.03E+07** ≈	1.11E+07
	Std	4.18E+06	**1.38E+06**	1.17E+06
f10	Mean	2.99E+03 −	1.99E+03 ≈	**1.93E+03**
	Std	2.80E+02	8.62E+01	**4.18E+02**
f11	Mean	**2.51E+01** −	2.86E+01 −	2.65E+01
	Std	**3.13E+00**	2.98E+00	2.09E+00
f12	Mean	3.65E+02 −	1.28E+01 −	**1.04E+01**
	Std	1.26E+02	1.70E+01	**3.92E+00**
f13	Mean	**3.37E+02** +	6.13E+02 ≈	6.27E+02
	Std	**7.85E+01**	8.62E+01	1.41E+02
f14	Mean	2.21E+07 −	**2.09E+07** +	2.36E+07
	Std	1.28E+06	**2.15E+06**	2.73E+06
f15	Mean	**2.99E+03** +	3.06E+03 −	3.25E+03
	Std	**5.94E+02**	2.53E+02	6.08E+02
f16	Mean	**1.81E+01** ≈	2.09E+01 −	1.87E+01
	Std	**4.53E+00**	3.88E+00	4.34E+00
f17	Mean	8.93E+00 +	**8.63E+00** +	1.14E+01
	Std	1.36E+00	**2.36E+00**	2.28E+00
f18	Mean	1.13E+03 ≈	1.14E+03 ≈	**1.09E+03**
	Std	1.10E+02	1.38E+02	**1.41E+02**
f19	Mean	**4.96E+05** +	3.39E+06 −	4.47E+06
	Std	**1.38E+04**	2.04E+06	3.82E+06
f20	Mean	3.18E+03 −	2.17E+03 −	**2.06E+03**
	Std	2.30E+02	1.59E+02	**1.40E+02**
No.best		6	4	10
EDGCC vs.		12 vs 8	13 vs 7	

Statistical results are obtained by 25 independent runs
The bold entries are the best results that obtained by the tested algorithm

ferent sub-components and an effective search strategy is necessary.

4.4 The effectiveness analysis of two important components

The proposed EDGCC has higher competitiveness in solving LSGOPs, which can contribute to the cooperation of the main components, i.e., EGD method and fireworks search strategy. To verify their respective effectiveness, two variants of EDGCC are designed and compared with the original EDGCC on the CEC'2010 benchmark suite, where two variants are respectively called CC-EGD, and FCC. In CC-EDG, the EGD method is preserved, while the fireworks search strategy is removed. Its purpose is to investigate the effectiveness of the fireworks search strategy. In FCC, the EGD method is abandoned. Instead, the fireworks search strategy is preserved, which aims to investigate the effectiveness of the EGD method. Note that all parameters in these three algorithms are consistent for a fair comparison.

Table 4 presents the comparison results among the original EDGCC and its two variants on the CEC'2010 benchmark suit. According to the table, EDGCC achieves the best results on 10 out of 20 test problems, while CC-EGD and FCC outperform in 6 and 4 test problems, respectively. This indicates that the two main components in EDGCC are important to the performance of EDGCC, which cooperate to make the EDGCC perform efficiently. Additionally, convergence curves obtained from EDGCC and its two variants on functions f_3 and f_{10} are depicted in Fig. 4, which can visually illustrate the distinctions. From the figure, it is evident that EDGCC outperforms its variants throughout the evolutionary process. In summary, the synergy of the main components makes the proposed algorithm perform noticeably.

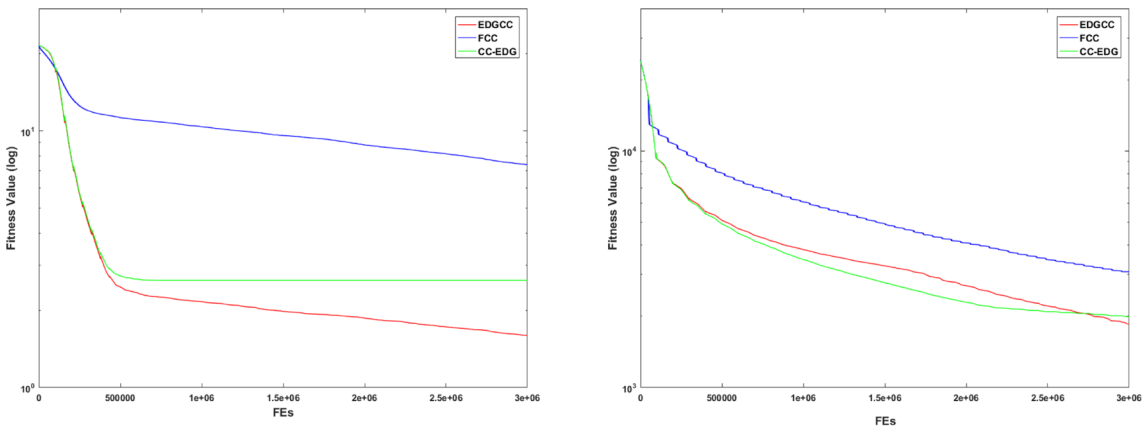
4.5 Computational complexity and time of the EDG method

The EDG method consists of two main components, i.e., sub-component initialization and sub-component evolution. The initial sub-components of decision variables are generated based on the RDG method, which is a recently proposed variable grouping method with low computational complexity. As shown in [41], the computational complexity of the initialization for a D-dimensional problem is O(Dlog(D)).

Following the sub-component initialization, the initial sub-components are dynamically evolving as the optimization process proceeds. This is an additional part compared with static decomposition methods. Table 5 gives the computational time of the sub-component evolution on CEC'2013 benchmark functions. It can be seen that the computational time of this part is very small. In essence, compared with static decomposition methods, the EDG method does not incur significant additional computational time.

4.6 Application in real-world problem

To further validate the effectiveness of the proposed EDGCC, the experimental study of the real-world optimization problem, i.e., the knapsack problem (KP), is conducted. In KP, there is a set of goods, which is associated with weight and profit values, and a backpack, which is associated with maximum capacity. The objective of the KP is to maximize the total value of a subset of goods within the constraints of the capacity of the knapsack. Therefore, it is a constrained optimization problem. The specific settings and Matlab code for the KP are available on PlatEMO [61]. The number of decision variables of KP D is set to 100, 200, 300, 500, and



(a) f_3 (b) f_{10}

Fig. 4 The convergence curves of EDGCC, and two variants FCC, CC-EDG for functions f_3 and f_{10}. (a) The convergence curves on f_3. (b) The convergence curves on f_{10}

Table 5 Computational time of the sub-component dynamic evolution on CEC'2013 benchmark functions

Func	f1	f2	f3	f4	f5	f6	f7
Time	8.89E−03	7.82E−03	2.71E−03	5.85E−02	4.61E−03	4.60E−03	1.13E−03
f8	f9	f10	f11	f12	f13	f14	f15
1.50E−03	2.93E−03	1.19E−01	1.88E−02	9.20E−03	2.56E−02	1.04E−02	9.67E−04

1000. The maximum number of function evaluations for each problem is set to $3000D$.

The result obtained by each algorithm from 25 independent runs on KP is presented in Table 6. According to the results, it can be seen that the proposed EDGCC achieves the best performance of all KPs with different variables. The EADG and DECC-RDG obtain similar results. However, eWOA cannot converge to the optimal solution at all, and the DSCA cannot find a solution that satisfies the constraint conditions in any runs. Generally, the effect of the CCEAs is better than the two metaheuristics algorithms. This observation is the same as the results that obtained on the benchmark suits.

In summary, the proposed EDGCC demonstrates a clear advantage compared with the peer competitors on the real-world KP.

4.7 Discussion

The above statistical comparisons confirm the effectiveness of the proposed EDGCC for solving LSGOPs. This can be attributed to the following reasons. First, with practical information compensation by the EDG method, information is transferred between different sub-components. Sub-components that are more favorable to the current optimization phase are generated. Additionally, the fireworks search strategy further improves the diversity of the population.

Although the proposed EDGCC has shown encouraging results in most test instances, the current study still has some limitations as well. First, many of the parameters in this paper are set as other relevant papers, while not analyzing them in greater depth. For example, for fully separable and fully non-separable problems, the decision variables are randomly divided into some initial sub-components. The size of each sub-component, which has a significant impact on the performance of the algorithm, is set as recommended by [35]. Therefore, the proposed EDGCC only achieves the best on 3 out of 6 fully non-separable problems, which fails to achieve the desired result. Additionally, it should be noted that the fireworks search strategy will consume the function evolution and potentially hamper the convergence speed of the algorithm. Although the limitation of function evolution frequency was not explicitly considered when tackling large-scale optimization problems, where the maximum number of function evaluations is set to be very large, it is necessary to design strategies to enhance population convergence.

5 Conclusion

The performance of the CC technique for solving LSGOPs will be influenced by the decomposition methods. However, most existing decomposition methods divide decision

Table 6 Comparisons among EDGCC and compared algorithms on the KP with 100, 200, 300, 500, and 1000 decision variables

Problem	D	Quality	eWOA	DSCA	EADG	DECCRDG	EDGCC
KP	100	Mean	5.72E+03 −	4.00E+03 −	2.72E+03 −	2.72E+03 −	**2.62E+03**
		Std	0.00E+00	2.71E+02	3.56E+01	6.33E+01	**8.05E+01**
	200	Mean	1.10E+04 −	8.10E+03 −	5.21E+03 −	5.22E+03 −	**4.94E+03**
		Std	0.00E+00	2.24E+02	5.76E+01	8.41E+01	**1.38E+02**
	300	Mean	1.71E+04 −	1.27E+04 −	8.28E+03 −	8.27E+03 −	**8.11E+03**
		Std	0.00E+00	6.45E+02	1.35E+02	6.27E+01	**8.20E+01**
	500	Mean	2.76E+04 −	2.04E+04 −	1.35E+04 −	1.34E+04 −	**1.31E+04**
		Std	0.00E+00	6.12E+02	1.31E+02	7.95E+01	**1.79E+02**
	1000	Mean	5.59E+4 −	4.24E+04 −	2.75E+04 −	2.73E+04 −	**2.70E+04**
		Std	0.00E+00	0.00E+00	2.67E+02	2.08E+02	**3.68E+02**
No.best			0	0	0	0	5
+ / − / ≈			0/5/0	0/5/1	0/5/0	0/5/0	

Statistical results are obtained by 25 independent runs
The bold entries are the best results that obtained by the tested algorithm

variables into fixed sub-components, which may not be conducive to the information transfer between the variables in different sub-components. Therefore, this paper proposes an evolutionary dynamic grouping based cooperative coevolution algorithm, named EDGCC.

In EDGCC, a novel variables decomposition method EDG is proposed to generate new sub-components dynamically in the optimization process. The idea of the EDG method is motivated by the evolution process of the genetic algorithm, while the selection, crossover, and mutation operations are redefined. To adjust the search focus in different stages of the algorithm, an adaptive frequency scheme of EDG is also proposed. Additionally, the fireworks search strategy is introduced to work with the sub-components optimizer, which can further improve the diversity of the population.

To examine the performance of the proposed EDGCC, a comprehensive experimental study is conducted. First, the sensitivity analysis of the unique parameter, i.e., the group number of each sub-component m. From the result, it can be seen that the performance of EDGCC will deteriorate as m increases. Therefore, $m = 2$ is used as a general parameter setting in this paper. Then, the comparisons among EDGCC and four recently developed LSEAs on two widely used CEC'2010 and CEC'2010 benchmarks are tested. Experimental results show the effectiveness of EDGCC, in which EDGCC achieves the best on 26 out of 35 functions. In order to verify the effectiveness of the EDG method and fireworks search strategy, EDGCC has been compared with two variants on CEC'2010. From the result, it can be seen that EDGCC obtains better convergence than the compared two variants. Moreover, the computational time of the sub-component dynamic evolution is tested. Finally, EDGCC is applied to solving the real-world KP with 100, 200, 300, 500, and 1000 decision variables. The experimental results show that the EDGCC algorithm has excellent performance in solving KP.

In the future work, we would like to focus on investigating if the EDG method can be extended for large-scale multi-objective optimization problems and developing the fireworks algorithm for large-scale problems.

Acknowledgements The authors are supported by the National Nature Science Foundation of China under Grant No.62273080.

Declarations

Conflict of interest All authors declare that they have no conflict of interest.

Ethical approval All authors declare that they have no conflict of interest.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

1. Goh SK, Tan KC, Al-Mamun A, Abbass HA (2015) Evolutionary big optimization (bigopt) of signals. In: 2015 IEEE congress on evolutionary computation (CEC). IEEE, pp 3332–3339. https://doi.org/10.1109/CEC.2015.7257307
2. Wang Y, Zhang Q, Wang GG (2023) Improving evolutionary algorithms with information feedback model for large-scale many-objective optimization. Appl Intell 53(10):11439–11473. https://doi.org/10.1007/s10489-022-03964-9
3. Omidvar MN, Li X, Yao X (2021) A review of population-based metaheuristics for large-scale black-box global optimization-part i. IEEE Trans Evol Comput 26(5):802–822. https://doi.org/10.1109/TEVC.2021.3130838
4. Omidvar MN, Li X, Yao X (2021) A review of population-based metaheuristics for large-scale black-box global optimization-part ii. IEEE Trans Evol Comput 26(5):823–843. https://doi.org/10.1109/TEVC.2021.3130835
5. Caraffini F, Neri F, Iacca G (2017) Large scale problems in practice: the effect of dimensionality on the interaction among variables. In: Applications of evolutionary computation: 20th European Conference, EvoApplications 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings, Part I 20. Springer, pp 636–652
6. Caraffini F, Neri F, Iacca G, Mol A (2013) Parallel memetic structures. Inf Sci 227:60–82. https://doi.org/10.1016/j.ins.2012.11.017
7. Tayarani-N MH, Yao X, Xu H (2014) Meta-heuristic algorithms in car engine design: A literature survey. IEEE Trans Evol Comput 19(5):609–629. https://doi.org/10.1109/TEVC.2014.2355174
8. Xue B, Zhang M, Browne WN, Yao X (2015) A survey on evolutionary computation approaches to feature selection. IEEE Trans Evol Comput 20(4):606–626. https://doi.org/10.1109/TEVC.2015.2504420
9. Sun Y, Xiao K, Wang S, Lv Q (2022) An evolutionary many-objective algorithm based on decomposition and hierarchical clustering selection. Appl Intell 1–46. https://doi.org/10.1007/s10489-021-02669-9
10. Zhang L, Wang L, Pan X, Qiu Q (2023) A reference vector adaptive strategy for balancing diversity and convergence in many-objective evolutionary algorithms. Appl Intell 53(7):7423–7438. https://doi.org/10.1007/s10489-022-03545-w
11. Abbaszadeh Shahri A, Khorsand Zak M, Abbaszadeh Shahri H (2022) A modified firefly algorithm applying on multi-objective radial-based function for blasting. Neural Comput Appl 1–17. https://doi.org/10.1007/s00521-021-06544-z
12. Balande U, Shrimankar D (2022) A modified teaching learning metaheuristic algorithm with opposite-based learning for permutation flow-shop scheduling problem. Evol Intell 15(1):57–79. https://doi.org/10.1007/s12065-020-00487-5
13. Abbaszadeh Shahri A, Kheiri A, Hamzeh A (2021) Subsurface topographic modeling using geospatial and data driven algorithm. ISPRS Int J Geo-Inf 10(5):341. https://doi.org/10.3390/ijgi10050341
14. Omidvar MN, Li X, Mei Y, Yao X (2013) Cooperative co-evolution with differential grouping for large scale optimization. IEEE Trans Evol Comput 18(3):378–393. https://doi.org/10.1109/TEVC.2013.2281543
15. LaTorre A, Muelas S, Peña JM (2015) A comprehensive comparison of large scale global optimizers. Inf Sci 316:517–549. https://doi.org/10.1016/j.ins.2014.09.031
16. Clark M, Ombuki-Berman B, Aksamit N, Engelbrecht A (2022) Cooperative particle swarm optimization decomposition methods for large-scale optimization. In: 2022 IEEE symposium series on computational intelligence (SSCI). IEEE, pp 1582–1591. https://doi.org/10.1109/SSCI51031.2022.10022095

17. Mahdavi S, Shiri ME, Rahnamayan S (2015) Metaheuristics in large-scale global continues optimization: A survey. Inf Sci 295:407–428. https://doi.org/10.1016/j.ins.2014.10.042
18. Peng X, Jin Y, Wang H (2018) Multimodal optimization enhanced cooperative coevolution for large-scale optimization. IEEE Trans Cybern 49(9):3507–3520. https://doi.org/10.1109/TCYB.2018.2846179
19. Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evol Comput 9(2):159–195. https://doi.org/10.1162/106365601750190398
20. Ros R, Hansen N (2008) A simple modification in cma-es achieving linear time and space complexity. In: International conference on parallel problem solving from nature. Springer, pp 296–305. https://doi.org/10.1007/978-3-540-87700-4_30
21. Molina D, Lozano M, Herrera F (2010) Ma-sw-chains: Memetic algorithm based on local search chains for large scale continuous global optimization. In: IEEE congress on evolutionary computation. IEEE, pp 1–8. https://doi.org/10.1109/CEC.2010.5586034
22. LaTorre A, Muelas S, Peña JM (2011) A mos-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test. Soft Comput 15:2187–2199. https://doi.org/10.1007/s00500-010-0646-3
23. Cheng R, Jin Y (2014) A competitive swarm optimizer for large scale optimization. IEEE Trans Cybern 45(2):191–204. https://doi.org/10.1109/TCYB.2014.2322602
24. Cheng R, Jin Y (2015) A social learning particle swarm optimization algorithm for scalable optimization. Inf Sci 291:43–60. https://doi.org/10.1016/j.ins.2014.08.039
25. Jian JR, Chen ZG, Zhan ZH, Zhang J (2021) Region encoding helps evolutionary computation evolve faster: A new solution encoding scheme in particle swarm for large-scale optimization. IEEE Trans Evol Comput 25(4):779–793. https://doi.org/10.1109/TEVC.2021.3065659
26. 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 50(6):2715–2729. https://doi.org/10.1109/TCYB.2019.2933499
27. Wang ZJ, Zhan ZH, Kwong S, Jin H, Zhang J (2020) Adaptive granularity learning distributed particle swarm optimization for large-scale optimization. IEEE Trans Cybern 51(3):1175–1188. https://doi.org/10.1109/TCYB.2020.2977956
28. Ge YF, Yu WJ, Lin Y, Gong YJ, Zhan ZH, Chen WN, Zhang J (2017) Distributed differential evolution based on adaptive mergence and split for large-scale optimization. IEEE Trans Cybern 48(7):2166–2180. https://doi.org/10.1109/TCYB.2017.2728725
29. Hadi AA, Mohamed AW, Jambi KM (2019) Lshade-spa memetic framework for solving large-scale optimization problems. Complex Intell Syst 5:25–40. https://doi.org/10.1007/s40747-018-0086-8
30. Koçer HG, Uymaz SA (2021) A novel local search method for lsgo with golden ratio and dynamic search step. Soft Comput 25(3):2115–2130. https://doi.org/10.1007/s00500-020-05284-x
31. Zhang W, Lan Y et al (2022) A novel memetic algorithm based on multiparent evolution and adaptive local search for large-scale global optimization. Comput Intell Neurosci 2022. https://doi.org/10.1155/2022/3558385
32. Li Y, Zhao Y, Liu J (2021) Dynamic sine cosine algorithm for large-scale global optimization problems. Expert Syst Appl 177:114950. https://doi.org/10.1016/j.eswa.2021.114950
33. Chakraborty S, Saha AK, Chakraborty R, Saha M (2021) An enhanced whale optimization algorithm for large scale optimization problems. Knowl-Based Syst 233:107543. https://doi.org/10.1016/j.knosys.2021.107543
34. Van den Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput 8(3):225–239. https://doi.org/10.1109/TEVC.2004.826069
35. Yang Z, Tang K, Yao X (2008) Large scale evolutionary optimization using cooperative coevolution. Inf Sci 178(15):2985–2999. https://doi.org/10.1016/j.ins.2008.02.017
36. Yang Z, Tang K, Yao X (2008b) Multilevel cooperative coevolution for large scale optimization. In: 2008 IEEE congress on evolutionary computation (IEEE World Congress on Computational Intelligence). IEEE, pp 1663–1670. https://doi.org/10.1109/CEC.2008.4631014
37. Omidvar MN, Li X, Yao X (2010) Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In: IEEE congress on evolutionary computation. IEEE, pp 1–8. https://doi.org/10.1109/CEC.2010.5585979
38. Sun Y, Kirley M, Halgamuge SK (2015) Extended differential grouping for large scale global optimization with direct and indirect variable interactions. In: Proceedings of the 2015 annual conference on genetic and evolutionary computation, pp 313–320. https://doi.org/10.1145/2739480.2754666
39. Mei Y, Omidvar MN, Li X, Yao X (2016) A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. ACM Trans Math Softw (TOMS) 42(2):1–24. https://doi.org/10.1145/2791291
40. Omidvar MN, Yang M, Mei Y, Li X, Yao X (2017) Dg2: A faster and more accurate differential grouping for large-scale black-box optimization. IEEE Trans Evol Comput 21(6):929–942. https://doi.org/10.1109/TEVC.2017.2694221
41. Sun Y, Kirley M, Halgamuge SK (2017) A recursive decomposition method for large scale continuous optimization. IEEE Trans Evol Comput 22(5):647–661. https://doi.org/10.1109/TEVC.2017.2778089
42. Chen A, Ren Z, Guo W, Liang Y, Feng Z (2022) An efficient adaptive differential grouping algorithm for large-scale black-box optimization. IEEE Trans Evol Comput. https://doi.org/10.1109/TEVC.2022.3170793
43. Omidvar MN, Kazimipour B, Li X, Yao X (2016) Cbcc3–a contribution-based cooperative co-evolutionary algorithm with improved exploration/exploitation balance. In: 2016 IEEE congress on evolutionary computation (CEC). IEEE, pp 3541–3548. https://doi.org/10.1109/CEC.2016.7744238
44. Kazimipour B, Omidvar MN, Qin AK, Li X, Yao X (2019) Bandit-based cooperative coevolution for tackling contribution imbalance in large-scale optimization problems. Appl Soft Comput 76:265–281. https://doi.org/10.1016/j.asoc.2018.12.007
45. Yang M, Omidvar MN, Li C, Li X, Cai Z, Kazimipour B, Yao X (2016) Efficient resource allocation in cooperative co-evolution for large-scale global optimization. IEEE Trans Evol Comput 21(4):493–505. https://doi.org/10.1109/TEVC.2016.2627581
46. Ren Z, Chen A, Wang M, Yang Y, Liang Y, Shang K (2020) Bi-hierarchical cooperative coevolution for large scale global optimization. IEEE Access 8:41913–41928. https://doi.org/10.1109/ACCESS.2020.2976488
47. Yang M, Zhou A, Li C, Guan J, Yan X (2020) Ccfr2: A more efficient cooperative co-evolutionary framework for large-scale global optimization. Inf Sci 512:64–79. https://doi.org/10.1016/j.ins.2019.09.065
48. Xu HB, Li F, Shen H (2020) A three-level recursive differential grouping method for large-scale continuous optimization. IEEE Access 8:141946–141957. https://doi.org/10.1109/ACCESS.2020.3013661
49. Mei Y, Li X, Yao X (2013) Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems. IEEE Trans Evol Comput 18(3):435–449. https://doi.org/10.1109/TEVC.2013.2281503

50. Sayed E, Essam D, Sarker R, Elsayed S (2015) Decomposition-based evolutionary algorithm for large scale constrained problems. Inf Sci 316:457–486. https://doi.org/10.1016/j.ins.2014.10.035
51. Goh CK, Tan KC (2008) A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. IEEE Trans Evol Comput 13(1):103–127. https://doi.org/10.1109/TEVC.2008.920671
52. Tan Y, Zhu Y (2010) Fireworks algorithm for optimization. In: Advances in swarm intelligence: first international conference, ICSI 2010, Beijing, China, June 12-15, 2010, Proceedings, Part I 1. Springer, pp 355–364. https://doi.org/10.1007/978-3-642-13495-1_44
53. Liu J, Zheng S, Tan Y (2014) Analysis on global convergence and time complexity of fireworks algorithm. In: 2014 IEEE Congress on evolutionary computation (CEC). IEEE, pp 3207–3213. https://doi.org/10.1109/CEC.2014.6900652
54. Zheng S, Tan Y (2013) A unified distance measure scheme for orientation coding in identification. In: 2013 IEEE Third international conference on information science and technology (ICIST). IEEE, pp 979–985. https://doi.org/10.1109/ICIST.2013.6747701
55. Imran AM, Kowsalya M (2014) A new power system reconfiguration scheme for power loss minimization and voltage profile enhancement using fireworks algorithm. Int J Electr Power Energy Syst 62:312–322. https://doi.org/10.1016/j.ijepes.2014.04.034
56. Zheng S, Janecek A, Tan Y (2013) Enhanced fireworks algorithm. In: 2013 IEEE congress on evolutionary computation. IEEE, pp 2069–2077. https://doi.org/10.1109/CEC.2013.6557813
57. Engelbrecht AP (2006) Fundamentals of computational swarm intelligence. John Wiley & Sons Inc
58. Tang K, Yáo X, Suganthan PN, MacNish C, Chen YP, Chen CM, Yang Z (2007) Benchmark functions for the cec'2010 special session and competition on large-scale global optimization. Nature inspired computation and applications laboratory, USTC, China, vol 24, pp 1–18
59. Li X, Tang K, Omidvar MN, Yang Z, Qin K, China H (2013) Benchmark functions for the cec 2013 special session and competition on large-scale global optimization. Gene 7(33):8
60. Yang Z, Tang K, Yao X (2008) Self-adaptive differential evolution with neighborhood search. In: 2008 IEEE congress on evolutionary computation (IEEE World Congress on Computational Intelligence). IEEE, pp 1110–1116. https://doi.org/10.1109/CEC.2008.4630935
61. Tian Y, Cheng R, Zhang X, Jin Y (2017) Platemo: A matlab platform for evolutionary multi-objective optimization [educational forum]. IEEE Comput Intell Mag 12(4):73–87. https://doi.org/10.1109/MCI.2017.2742868

Wanting Yang received the B.S. degree in automation from Zhengzhou University, Zhengzhou, China in 2015. She is currently pursuing the Ph.D. degree in control science and engineering at Northeastern University, Shenyang, China.
Her current research interests include large-scale optimization and multiobjective optimization.

Jianchang Liu was born in Heishan City, Liaoning Province, P.R. China, in 1960. He received the B.S., M.S., and Ph.D. degrees from Northeastern University, Shenyang, China, in 1980, 1989, and 1998, respectively. He is currently a Professor at the Collage of Information Science and Engineering, Northeastern University, Shenyang, China.
His research interests include multi-objective optimization, modeling, control and optimization for complex process, intelligent control theory and application, fault diagnosis.

Shubin Tan received the B.Eng. in safety engineering, M. B.Eng. and Ph.D. degree in control theory and control engineering from Northeastern University, Shenyang, China, in 1996, 2003 and 2006, respectively.
He is currently an associate professor in the College of Information Science and Engineering, Northeastern University, Shenyang, China. His research interest covers modeling, control and optimization of industrial process, fault diagnosis.

Wei Zhang received the B.S. degree in automation from Shenyang Institute of Engineering at Shenyang, China, in 2019. He received the master degree in control engineering from Northeastern University, Shenyang, China, in 2022.
His current research interests include multi-objective optimization and its applications.

Yuanchao Liu received the B.S. degree in automation from Northeastern University at Qinhuangdao, Qinhuangdao, China in 2017. He received the master degree in control theory and control engineering and Ph.D. degree in control science and engineering from Northeastern University, Shenyang, China, in 2019 and 2023, respectively. He is currently a lecturer with the College of Information Science and Engineering, Northeastern University, Shenyang, China.
His current research interests include multiobjective optimization, robust optimization, dynamic optimization and data-driven optimization.