Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 47 (2026), No. 6, pp. 2165–2191
DOI : 10.47974/JIOS-2005

A large-scale optimization method based on gradient approximation and hybrid grouping

Haiyan Liu *
Yi Cheng †
Siyu Gao §
Hanzhe Yang ‡
Department of Software Engineering
School of Computer Science and Technology
Xi'an University of Posts and Telecommunications
Xi'an
Shaanxi 710121
China

Abstract

With advances in science and technology, optimization problems increasingly involve numerous decision variables, posing significant challenges for optimization algorithms. These problems are often mentioned as LSGO problems (large-scale global optimization) and have received considerable attention. LSGO problem is very challenging to solve due to the high dimensions, countless local optimal solutions, and huge computational burden. In this paper, a gradient approximation and hybrid grouping method (GAHG) is proposed to address such challenges. Firstly, we design an adaptive gradient approximation based search algorithm to make a quick search in a reduced search region and find preliminary good solutions. During this stage, important information about the decision variables will be stored for future usage. Further more, we introduce a hybrid grouping strategy designed to efficiently and reasonably decompose the large-scale fully non-separable problems. The hybrid grouping method consists of five decomposing strategies to assign potentially highly related variables into the same group. The performance of GAHG is evaluated on widely used large-scale benchmark suite CEC'2010 and CEC'2013 large-scale test problems.

* *E-mail:* hyliu83@126.com (Corresponding Author)
† *E-mail:* cyjy2000@163.com
§ *E-mail:* gsy0408150240@163.com
‡ *E-mail:* 490007952@qq.com

Experimental results and statistical analysis are made, which indicate the proposed GAHG is effective and efficient in solving large-scale problems.

Subject Classification: *90C06 (Large-scale problems in mathematical programming).*

Keywords: *Large-scale global optimization, Hybrid grouping, Co-evolutionary strategy, Gradient approximation, Adaptive method.*

Introduction

Global optimization is a discipline that selects the best strategies and decisions in production, life, and scientific research and has a broad range of applications in human production practice. As technology continues to advance, optimization problems involve more decision variables and grow in scale. When an optimization problem contains between several hundred and several thousand decision variables, we often refer to it as large-scale global optimization (LSGO). LSGO has found applications in diverse fields including seismic waveform inversion [1], traffic vehicle scheduling [2], and fleet routing allocation [3]. While these applications typically involve high-dimensional data, the principles of LSGO have also been proven crucial in scenarios involving small and medium sized datasets with highly correlated variables. For example, optimizing hyperparameters for ensemble models applied to medical datasets of small and medium scale [4] requires complex large-scale search strategies to balance the intricate trade-offs among hyper parameters, dataset size, and imbalance ratio.

LSGO problems can be described as Eq. (1) :

$$arg \min_{(x_1,x_2,\ldots,x_n)} f(x_1, x_2, .., x_n) \tag{1}$$

where $f: R^n \to R$ is an objective function, $x_1, x_2, \ldots x_n$ are the decision variables, and n is the dimension of the problem. In the CEC'2013 benchmark suite, each test function has a dimensionality of 1000.

LSGO is very challenging, many efficient traditional optimization methods cannot be applied, and evolutionary algorithms can quickly lose their effectiveness and advantages. The reasons may be due to the following three factors:

(i) Dimensionality challenge: as the number of variables grows, the size of the search space grows at an exponential rate, which greatly raises the difficulty of identifying the global optimum. Optimization algorithms need to explore a vast search space to find good solutions, which consumes substantial time and computational resources. Taking CEC'2013 test suite as example, the global optimums of many test problems cannot be located even by the most state-of-the-art algorithms;

(ii) Local optima: as the problem dimension increases, the number of local optima increases rapidly. Optimization algorithms often become trapped in local optima, failing to reach the global optimal solution;

(iii) Computational burden: LSGO problems require high function evaluations to obtain good results, which puts tremendous pressure on computational resources and time. Traditional optimization algorithms become inefficient in high-dimensional spaces.

To better solve LSGO problems, two approaches are commonly used: decomposition-based cooperative co-evolution methods and other methods that do not make explicit decomposition. At present, the cooperative co-evolution (CC) framework [5] has been widely adopted in LSGO. The CC framework aims to divide an N-dimensional problem into M sub-problems, and solves each sub-problem separately(e.g., employing Evolutionary Algorithms). In the Cooperative Co-evolution framework, while optimizing variables of one sub-problem, all other variables are kept fixed (often fixed at their current best values). Through this process, sub-problems evolve cooperatively, and their results are combined to generate the final solution for LSGO problem. The CC framework presents an efficient method to solve optimization challenges. But the result can not be satisfactory if the large-scale problem cannot be properly divided based on the dependencies among decision variables.

The interaction between variables can usually be classified into two types: separable and non-separable. When the change of a variable has an independent effect on the overall function value, this variable is separable. On the other hand, when the change of a variable has a dependent effect (influenced by other variables) on the overall function value, these variables are non-separable because they interact with each other.

Assuming a large-scale problem involving n variables, if these n variables are all independent, it indicates that the problem is fully separable. If m variables (m < n) are independent, then the problem is partially separable. Otherwise, if all n variables are interacted with each other, the problem is called fully non-separable. For a better understanding of separable and non-separable problems, We present three simple examples, each containing just four variables.

$$f_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 \tag{2}$$

$$f_2(x) = (x_1 + x_2 + x_3 + x_4)^2 \tag{3}$$

$$f_3(x) = (x_1 + x_2)^2 + (x_3 - x_4)^2 \tag{4}$$

Because the four variables of x_1, x_2, x_3, and x_4 are unrelated (mutually independent and do not affect each other) in Eq. (2), the function $f_1(x)$ is a fully separable problem. We can observe that the individual variables of the function $f_2(x)$ interact in Eq. (3) . This problem is a non-separable problem. Looking at Eq. (4), it can be observed that $f_3(x)$ is partially separable, as its variables can be separated into two independent groups $\{x_1, x_2\}$, $\{x_3, x_4\}$, which can be optimized separately. In the literature [6], the authors found that the interaction between variables can significantly affect the convergence rate of optimization algorithms. Therefore, the correct decomposition of large-scale problems is essential for LSGO problems. Over the past decades, researchers have been continually developing novel decomposition techniques that consider the interaction between

variables, thereby improving the effectiveness of optimization algorithms and tackling the difficulties inherent in LSGO problems. However, there are still two problems to be solved:

(i) How to efficiently optimize each dimension of a LSGO problem?
(ii) How to handle a fully non-separable LSGO problem?

In this paper, we introduce an adaptive one-dimensional search method based on gradient approximation to optimize each dimension of LSGO problems, and propose a hybrid grouping method to effectively decompose fully non-separable problems. By combining the proposed two methods, we design an efficient solution for solving LSGO problems.

II. Related work

Large-scale global optimization problem has received increasing attention. Many scholars have researched and proposed various solutions to this problem. Among these approaches, the CC framework has been widely utilized. Therefore, determining how to effectively decompose the problem is crucial in addressing the LSGO problem. Some early grouping strategies had been proposed by researchers include single-dimensional grouping [7], random grouping [8], Delta grouping [9], and non-fixed grouping strategy [10]. By employing these strategies, researchers can simplify and directly decompose the LSGO problem. However, with a growing number of interacting variables, the accuracy of identifying and grouping variables using these strategies may become compromised. In tackle this issue, Omidvar *et al.* introduced the Differential Grouping method (DG) [11] that improves grouping efficiency by dividing related variables into subgroups based on their interactions; nevertheless, DG is limited to detecting only direct relationships between variables, thereby overlooking potential indirect dependencies. Consequently, Sun, Kirley, and Halgamuge further optimized the DG grouping and introduced the Extended Differential Grouping method (XDG) [12] to address the limitations of the DG method. In literature [13], Omidvar *et al.* an enhanced variant of the differential grouping approach, referred to as the Effective Differential Grouping (DG2). DG2 improves the accuracy of the grouping by estimating errors via variable interactions. Sun, Kirley, and Halgamuge, introduced the Recursive Differential Group method (RDG) [14], which iteratively analyzes how a selected decision variable interacts with the remaining variables and groups all decision variables with mutual relationships into the same subgroup. Philip *et al.* introduced the Global Differential Grouping method (GDG) [15]. GDG decomposes LSGO problems by modeling the problems and employing differential grouping strategies, thereby enhancing the accuracy of the grouping process.

Due to the limitations of these methods in solving the LSGO problem, some researchers have proposed a series of further improvements: Yang *et al.* in [16] introduced an efficient recursive differential grouping method (ERDG) that reduces function evaluation by utilizing historical information about variable interactions. Previous research [17] has shown that the RDG method is further

enhanced with a three-level recursive differential grouping (TRDG). TRDG divides a set into three subsets during the grouping stage and detects relationships between each subset and the set being analyzed, thereby reducing the recursive depth of the algorithm. In literature [18], a three-level recursive differential grouping (NTRDG), which extends the TRDG approach, is introduced. NTRDG utilizes historical data during recursive searches to reduce redundant interactions among variable sets. Furthermore, it improves the test-set interaction technique by lowering the required fitness evaluations per test from three to two, thereby achieving a substantial reduction in the overall number of function evaluations. To boost the effectiveness of RDG while lowering resource demands during the process of decomposition, Zheng, Xu, and Chen proposed Three Stages Recursive Differential Grouping method (TSRDG) in a recent study [19]. This approach classifies functions as separable or non-separable by exploiting the independence among separable variable groups, thereby lowering the computational resources needed for decomposition. Meanwhile, When utilizing the TSRDG method, information from the first two stages will be leveraged to avoid generating redundant information and checking redundant mutual relationships, thereby reducing the computational cost of the decomposition stage. To further simplify the decomposition method, [20] presents a bidirectional-detection differential grouping (BDDG) method. The approach utilizes a bidirectional detection framework to determine variable relationships and introduces an adaptive intake strategy to enhance decomposition accuracy within this framework. BDDG can classify problems as separable, fully non-separable, or partially separable while requiring fewer function evaluations. The existing literature [21] indicates that a minimum hash-based decomposition strategy (MHD) can effectively decompose large-scale problems. This strategy applies the concept of minimum hashing in high-dimensional decision space to decompose the LSGO problem, thereby enhancing the efficiency of decision variable decomposition through the inherent properties of minimum hashing. These methods attempt to improve and optimize the grouping methods in different ways to increase the efficiency and performance of solving the LSGO problem.

To reduce the computational cost, Kumar, Das, and Mallipeddi introduce an efficient differential grouping (EDG) method [22]. EDG utilizes past relationships between already established variable groups to examine interactions in the remaining groups. This approach reduces computational resource usage while maintaining the accuracy of the grouping results. Li et al. [23] proposed a novel approach named dual differential grouping (DDG) aimed at improving decomposition and optimization for LSGO problems. By supporting both additively and multiplicatively separable functions, this approach markedly increases the potential applications of CC. Zhong et al. [24] proposed an algorithm named surrogate ensemble assisted differential evolution combined with efficient dual differential grouping (SEADECC-EDDG). The method extends the efficient recursive differential grouping (ERDG) framework and incorporates the multiplicative interaction detection feature of Dual DG (DDG), allowing both additive and multiplicative interactions to be identified concurrently without extra fitness evaluations.

Furthermore, In literature [25] a Fast Independency Identification (FII) is introduced, which involves comparing the current variable with all remaining variables, avoiding pairwise comparisons and saving a considerable amount of resources. Nevertheless, the identification accuracy of the FII method remains low. Sedigheh Mahdavi et al. proposed the K-means mean grouping in literature [2], which groups decision variables according to their impact on fitness improvement. Those subgroups making the largest contribution to the fitness value receive more iterations (FEs) during the optimization process.

The aforementioned grouping methods only consider the input and output of the problem, and all of them are black-box grouping methods. However, in real life, many problems have known expressions, and these problems are referred to as white-box problems, where utilizing these expressions makes sense. Formula Based Grouping (FBG), a white-box grouping method, was introduced by Yang *et al.* in [26]. FBG directly establishes mutual relationships between the variables through the known expressions to improve grouping accuracy. Nonetheless, FBG is only able to identify that the LSGO problem is fully non-separable, without the capability to carry out decomposition. Liu et al. [27] developed the Contribution-based Decomposition (CBD) method to address this problem. The method ranks and groups variables according to their contributions, enhancing overall accuracy. However, after decomposition, some sub-problems may still contain numerous decision variables in large-scale problems. Hence, In [28], Liu, Wang, and Fan introduced a hybrid deep grouping method (HDG), which integrates considerations of variable interactions and their importance. Nevertheless, this approach is not sufficiently intelligent at present to detect dependencies among variables in fully non-separable problems. A study by Liu *et al.* [29] demonstrated that the CBD method was improved by introducing a multi-grouping strategy that includes 5 different decomposition methods. This method provides 14 distinct grouping outcomes for fully non-separable problems. It ensures that interacting variables have a greater chance of being placed in the same subgroup to alleviate the negative impact of decomposing interactive variables that should not be decomposed.

Furthermore, methods have emerged that optimize the LSGO problem by optimizing the characteristics of subcomponents generated by the CC framework. Subcomponents generated by the CC framework may exhibit different features, which should be taken into consideration when setting up subpopulations for optimizing these subcomponents. CCFR [30] is a current CC framework that distributes computational resources among subpopulations according to their role in enhancing the best overall objective value. However, the method divides population into subpopulations of equal size, which ignores the features of the subproblem. Therefore, a study by Yang *et al.* [31] proposed an improved version of CCFR, called CCFR2. CCFR2 algorithm can adjust the size of the subpopulation according to the subproblem solution. Additionally, when a subpopulation is chosen for evolution across multiple consecutive co-evolutionary cycles, CCFR2 avoids re-evaluating the individuals at the start of each cycle, thereby reducing the computational cost required to achieve the

optimal overall solution. However, CCFR2 may be affected by the curse of dimensionality during the search process, which is less efficient when the problem has high dimensionality, and the grouping method may lack flexibility and accuracy. Therefore, a new co-evolutionary framework is proposed, CCFR3 [32] adaptively assesses the contribution of subpopulations during each co-evolutionary cycle. This algorithm enables more frequent allocation of computational resources among subpopulations, providing them with increased evolutionary opportunities. As a result, the method accelerates the convergence of CC and improves its ability to find the global optimum. Yang *et al.* proposed a CCFR2 with automatic enhancement of population diversity, called CCFR2-AEPD [33]. This adaptive approach is designed to improve population diversity and has been incorporated into a contribution-based CC framework. An effective parallel master-slave server model is introduced in a recent study [34], which utilizes multiple slave servers to simultaneously process a subproblem, thereby accelerating the computation speed. Kelkawi, El-Abd, and Ahmad in [35] proposed a CC framework for parallel implementation, which uses the parallel technology of GPU and the shared memory and global memory of CUDA platform to parallel the subcomponents of the problem and accelerate the optimization process, so as to solve the LSGO problem.

In addition to the decomposition-based methods mentioned above, there are also methods that solve the LSGO problem without explicit decomposition. Previous research [36] introduced the Multiple Trajectory Search (MTS) algorithm, which employs a set of three distinct local search strategies to address LSGO problems. In each iteration, the most effective strategy among them is selected to guide the subsequent optimization step. Molina, Lozano, and Herrera in [37] present the MA-SSW-Chain that combines a scalable LS algorithm with the classical Solis Wets algorithm. It links different local search information and allocates an appropriate algorithm for each individual according to its characteristics. Multiple Offspring Sampling-based hybrid algorithm (MOS) is introduced in the literature [38], which proposes a new framework for efficient cooperation among multiple algorithms to find the best combination.

In addition, there are other methods available to divide and overcome the LSGO problem. In [39], Cheng and Jin proposed a competitive swarm optimizer (CSO), in which a pair of individuals is randomly chosen to compete, and the losing individual adjusts its state according to the winner. Yang *et al.* [40] introduced the Segment-based Predominant Learning Swarm Optimizer (SPLSO), employing segmented and dominant learning strategies to speed up the search process and improve diversity. Wang, Wang, and Sun [41] proposed a particle swarm optimization algorithm based on reinforcement learning and level structures (RLLPSO), which employs a hierarchical framework and a reinforcement learning-based level control strategy to address LSGO problems, improving both search efficiency and convergence. Deng et al. [42] introduced a swarm optimizer for large-scale problems, named RBLSO, which incorporates ranking-based biased learning. RBLSO incorporates two learning mechanisms, ranking paired learning (RPL) and biased center learning (BCL), which are

designed to increase diversity and accelerate convergence. In the existing study [43], Wang and Jiang combined the Gibbs sampling algorithm and the adaptive particle swarm optimization algorithm to propose a new method for solving LSGO problem, called Gibbs-APS. The method fundamentally employs multivariate Gaussian distributions to model particle swarm distributions, aiming to achieve an improved LSGO solution. Disruption Particle Swarm Optimization (DPSO) is proposed in [44]. This approach integrates the physics-based disruption concept with Cauchy 's mutation and the PSO algorithm, aiming to maintain a balance between global search and local refinement while helping particles avoid local optima.

In [45], Wang *et al.* showed the capabilities of AGLDPSO, an adaptive granularity learning distributed PSO method, for handling large-scale optimization problems. AGLDPSO employs a master-slave multi-subpopulation distribution model for co-evolution, and enables full exchange of information between different populations, thereby enhancing the diversity of populations. Zhang and Lin [46] introduced a Particle Swarm Optimization algorithm named TLS-PSO, which incorporates three distinct learning strategies. The method combines the mid-point-example learning strategy and stochastic learning strategy in the updating equations to create a Medium-point-example and Random Learning Particle Swarm Optimization algorithm (MRL-PSO). Dynamic linear adjustment is utilized to enhance its search capability. In the study [47], introduced PSO-DBCD, a particle swarm optimization algorithm designed to dynamically balance convergence and diversity. The approach incorporates a competitive multi-cluster framework and a convergence-driven learning scheme. Additionally, it employs an entropy-based local partial set metric to regulate convergence intensity and leverages local diversity information to implement a diversity-guided learning mechanism, enhancing the capability of the algorithm to maintain population diversity. PSO-DBCD is capable of explicitly managing both convergence pressure and diversity simultaneously, which is beneficial for maintaining a balance between convergence and diversity. Liu *et al.* put forward A bi-directional learning particle swarm optimization (BLPSO) in [48]. On the one hand, in this approach, particles with insufficient diversity acquire knowledge from particles possessing better diversity, improving the collective diversity. On the other hand, BLPSO leverages better-adapted particles as learning examples to aid in faster convergence. A high-speed homogeneous learning-based particle swarm optimizer (HLPSO) is proposed in the literature [49]. The method balances exploration and exploitation by introducing the concept of uniform learning, while also incorporating a new framework for evaluating population diversity, allowing real-time adjustment of evolutionary strategies. To expedite the convergence of the high-dimensional problem, an accelerated evolutionary strategy incorporating an autophagy mechanism is proposed. This mechanism seeks to reduce unnecessary fitness evaluations and speed up convergence. A study by Huang *et al.* [50] presented a modified competitive swarm optimization algorithm that introduces a three-stage co-evolutionary strategy (TPCSO). TPCSO employs a three-stage co-evolutionary approach to regulate the

exploration and exploitation of two subpopulations across different evolutionary phases. It also introduces a new update strategy to explore more promising areas through information exchange between the two subpopulations. This effectively strengthens the populations capacity to increase the efficiency of the evolutionary process.

Although researchers have proposed numerous algorithms to tackle LSGO problem, the results have not been satisfactory. The fundamental reason is that the problem has a high dimensionality, leading to the "curse of dimensionality". In this paper, a new one-dimensional search method is proposed to solve the problem. This method aims to make quick scans and reduce the search area, also it can explore the contribution of the decision variables and store the important information for the subsequent optimization. Additionally, a new hybrid grouping method is introduced to effectively decompose the fully non-separable problem. This method offers five grouping strategies and can alleviate the adverse effects of decomposition variables by providing more grouping results and larger grouping sizes. In theory, fully non-separable problems cannot be divided because all the variables are interconnected. But the efficiency of solving the problem is very low if it is not decomposed. Therefore, in this paper, the hybrid grouping method is proposed to enhance the ease and efficiency of optimizing fully non-separable LSGO problems. Additionally, the method offers a range of decomposition results, increasing the probability of interactive variables being grouped together within the same subgroup.

3. A large-scale optimization method based on gradient approximation and hybrid grouping

The proposed method mainly contains two stages in optimizing large-scale problems. In the first stage, an adaptive one-dimensional search algorithm based on gradient approximation is designed to make a quick rough search. In the second stage, we develop a novel hybrid grouping method to decompose fully non-separable large-scale problems. The hybrid grouping method consists of four decomposition strategies that enable a more reasonable decomposition, leading to a significant reduction in the vast search space of the large-scale problem.

A. *A gradient approximation based search algorithm for large-scale problems*

The gradient approximation based adaptive one-dimensional search method is designed to roughly target the region where better solutions may be found. The main idea is to generate new points using the gradient approximation matrix and the adaptive step matrix, and then update the optimal solutions by comparing them with the current optimal solution. During this procedure, we also narrowed the search region continuously using the information we got. The detailed procedure is as follows.

Firstly, the search region is partitioned into $(NP-1)$ intervals for each dimension and the endpoints of corresponding interval is used to generate matrix sub_pop. Then the gradient matrix pk at each interval endpoint (node) is calculated using Eq. (5).

$$pk(i) = \frac{f(pop_{i+1}) - f(pop_i)}{(sub\_pop_{i+1} - sub\_pop_i) \cdot 10^{exp}}; \quad (5)$$

where

$$exp = floor(abs(log_{10}(f(pop_{i+1}) - f(pop_i)))); \quad (6)$$

We use the difference quotient instead of the derivative to approximate the gradient with the aim of reducing complexity and saving computational resources. The parameter exponent exp is introduced to prevent the difference of the function values from becoming too large, which may result in an excessively large gradient and make the subsequent operation difficult. So exp stores the difference in orders of magnitude. In the situation that exp is calculated as INF or -INF when the difference of function value is very small, we replace exp with a 0 value.

Then, within each sub-region, we design an adaptive step size parameter for the generation of new points. Eq. (7) shows the calculation of the step size $lamda$ where i represents the $i-th$ node and vector G is the cumulative gradient sum with an initial value of 0.

$$\begin{aligned} G(i) &= G(i) + abs(pk(i)) \\ lamda &= \frac{lamda(i)}{(G(i)+0.1)} \end{aligned} \quad (7)$$

Initially, all values of $lambda$ are set to 0.5. the advantage of which is that it allows x_a and x_b to be adjusted moderately relative to sub_{pop}, thus preventing the results from deviating excessively. In Eq. (7) the numerical literal 0.1 is used to prevent a divide-by-zero operation. For each node of sub_pop, the step size is adaptively determined and the $lambda$ vector is subsequently updated using the aforementioned method. For each sub-region, we use Eq. (8) to generate new points where $[lb, ub]$ is the boundary of the problem.

$$\begin{aligned} x_a &= sub\_pop + lamda \cdot pk; \\ x_b &= sub\_pop - lamda \cdot pk; \end{aligned} \quad (8)$$

The my_bound function is used to ensure that the generated new points do not exceed the boundary. If a new point exceeds the upper or lower boundary, it will be set to equal the boundary. So within each sub-region, we can obtain two new points except for the first and last region from which only one point can be generated.

Then, the new generated points will be evaluated under the cooperative co-evolutionary framework and comparisons will be made with current best solution. The current best solution will be updated if there is a better solution among the new generated points. The cumulative improvement $F(i)$ will be updated according to Eq. (9).

$$F(i) = (bestval - bestNew)/bestval; \quad (9)$$

Note that the improvement is normalized to eliminate the bias of extremely large improvement, which can flatten all the others. As a result, the improvement value for each dimension is a real number between 0 and 1.

Algorithm 1 presents the pseudo-code of the gradient approximation based one-dimensional search method.

Algorithm 1 An adaptive one-dimensional search method based on approximate gradient

Require: The search space $[lbound, ubound]$, The current best solution $bestmem$ and best function value $bestval$, The times that the line search method will be executed: $iter$

Ensure: The improvement matrix F_{mat}, The best points (best member) matrix bm_{hist}, The new best number $bestmem$ and best function value $bestval$

Randomly permute the order of all dimensions;
while $times \leq iter$ **do**
  for $i = 1{:}D$ **do**
    Divide the search area into sub-areas;
    if The length of the sub-area < $epsilon$ **then**
      Skip and move to the next dimension;
    else
      Calculate the gradient of each region node using Eq. (5);
      An adaptive step vector ($lamda$) is generated from the gradient vector (pk) (using Eq. (7));
      Generate new points according to Eq. (8);
      Evaluate all newly generated points and record the new optimal solution $memNew$ and the function value $bestnew$;
      if $bestnew < bestval$ **then**
        The amount of improvement is calculated according to Eq. (9);
        Update $bestnew$ and $bestval$;
        Reduce the search region centered on point $memNew$;
      else
        Reduce the search region centered on point $bestmem$;
      end if
    end if
  end for
  Add the best member in bm_{hist};
  Record improvement matrix F_{mat};
  $times = times + 1$;
end while

Algorithm 1 will execute for 10 times and some important information will be recorded such as the improvement matrix, best value matrix, etc. This information will be used to guide the decomposition process in the subsequent phase. By employing this valuable information, we can gain a deeper understanding of the problem's properties and effectively divide the large-scale problem into smaller and medium-sized subcomponents.

B. *A hybrid Grouping Method for Fully Non-Separable large-scale Problems*

To improve search performance and expedite convergence, many LSGO algorithms often decompose the problem into multiple subproblems. However, theoretically, a fully non-separable problem cannot be decomposed because all the variables are interconnected, and the efficiency of optimization will be significantly reduced since the scale of the problem is too large. In literature [27], Contribution-Based Decomposition (CBD) approach was introduced to decompose the fully non-separable problem. However, this grouping method is not efficient because all the variables are interconnected, and CBD only provides one fixed way to decompose.

Therefore, we propose a hybrid grouping method combining the concepts of contribution, clustering, and statistics to alleviate this problem.

1) *A decomposition Method Based on Density of Contribution:* Contribution refers to the extent of influence that a decision variable can make to the overall objective function. The contribution information of decision variables reflects valuable characteristics and should make good use of. It can not only be used to guide the division of large-scale problems, but also, we can adaptively adjust the allocation of computational resources accordingly.

Therefore for fully non-separable problems, we design decomposition method according to variable contributions. Variables with similar contributions may share some common characteristics and thus should be grouped together, otherwise, these variables should be decomposed. Instead of sorting and grouping in a fixed way as CBD does, in this paper, we use the clustering mechanism to make the decomposition. Since clustering provides a more natural and flexible way of grouping, so we can get better decomposition results. In this paper, a Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [51] is applied as clustering method. Using contribution information as density, DBSCAN can effectively cluster data points into different groups, leading to a more accurate division of subgroups for large-scale problems.

After the clustering, large-scale problems can be decomposed into smaller-scale subproblems, where variables contained within each subgroup have similar contribution which means they may be highly interacted. Also, through the decomposition, not only the search space is highly reduced, but the complexity of problem is also reduced greatly. Thus, the convergence speed and operational efficiency of the algorithm can be improved, and unnecessary computational overhead can be minimized. The main procedure of this decomposition method can be described in the following steps.

Step 1: Calculate the average contribution value of each variable according to the improvement matrix F_{mat}, then the average contribution values will be sorted in descending order and stored in a vector $sortedF$.

Step 2: Calculate the average of all numbers starting from the first non-zero number in the contribution difference between each pair of variables to generate the distance matrix $epsilon$ according to adaptive Eq. (10).

Step 3: Mark all points in the dataset as unvisited.

Step 4: Use algorithm 2 to find all data points that are density-reachable from point p, or noise points (an isolated point not belonging to any cluster) to form a cluster.

Step 5: If the selected data point p is a border point (neither a core point nor a noise point), select another unvisited data point.

Step 6: Repeat steps 3 and 4 until all points have been visited.

Step 7: If there are still data points remaining, assign them to a new cluster.

$$\begin{aligned} &D(i,j) = sortedF(i) - sortedF(j) \\ &subset\_D = D(i,j{:}n); \\ &epsilon(i) = mean(subset\_D(:))./(1 + log(i)); \end{aligned} \tag{10}$$

Eq. (10) is used to control the radius or distance parameter ε for clustering to search for neighboring points. In Eq. (10), n is the dimensionality of the problem, i denotes the ith data in vector $sortedF$. $subset\_D$ stores all numbers in matrix D starting from the first non-zero number, the numerical literal 1 is used to prevent a divide-by-zero operation. To get better decomposition results, the cluster size should be moderate because when the cluster size is too large or small, the group size can be either too big or too small. A small $Minpts$ will increase the number of clusters and separate sparse data points, and it may lead to excessively refined results. On the other hand, a larger $Minpts$ will decrease the number of clusters since more dense connections are required and may cause potential issues of over-concentration. Thus $Minpts = 15$ to make a balance, ensuring cluster stability and meeting the requirements of clustering results.

It should be noted that $epsilon$ is not a fixed input parameter, but a vector of size $1 * n$, where the data size is related to the contribution of each dimension during the one-dimensional search. This means that the value of $epsilon$ will be dynamically adjusted according to the characteristics of variables.

The DBSCAN algorithm is typically used to group data while removing noise points. In this paper, noise points are chosen to be included directly in the current group instead of being removed. Algorithm 2 shows the detailed process of the density-reachable method.

2) *A Decomposition Method Based on Hierarchy of Contribution:* Since all variables of fully non-separable problems interact with each other, more decomposition diversity can provide a better combination of related variables and thus obtain better optimization results. Thus, in this section, a new decomposition method is proposed in which contributions of variables are combined with hierarchical clustering to make a decomposition. Hierarchical clustering is a method of establishing new clusters by merging clusters with high similarity[52]. The core concept of the hierarchical of contribution based grouping method is to

treat each variable as an independent cluster and then establish new clusters by merging clusters with high similarity (small contribution difference) until the predetermined number of clusters k is reached.

Algorithm 2 pseudo-code of the density-reachable method

Require: Distance vector $epsilon$, Matrix D, The minimum number to include in a neighborhood $MinPts$
Ensure: New cluster class
1: Select the first data point p and mark it as visited.
2: Query the number of points around the point that are less than or equal to $epsilon\{i\}$ $(i = 1)$ and denote it as t;
3: **if** t is less than $MinPts$ **then**
4: Add the index of the noise point directly to the vector $indices$;
5: **else**
6: Create a new cluster, and add the index of the point to $indices$;
7: Set variable k
8: **while** Continuously accessing **do**
9: Find the first density reachable point q within distance $epsilon\{i\}$;
10: **if** The point has not been visited, mark it as visited **then**
11: Query the number of points within distance $epsilon\{j\}$ in the vicinity of the point q (where j is the index of point q);
12: Add the index of this point to $indices$;
13: **if** The number of points in the vicinity is greater than or equal to $Minpts$, indicating that the point is density reachable **then**
14: Include this point in the same cluster class;
15: **end if**
16: **end if**
17: Increment k and continue iterating to the next directly density reachable point;
18: **if** All directly density reachable points have been traversed **then**
19: Perform grouping and save the vector $indices$ in $group\{C\}$;
20: Exit the loop;
21: **end if**
22: **end while**
23: **end if**

First, the improvement matrix will be sorted in descending order row by row to obtain the vector $sortedF$ and ind, where $sortedF$ stores contributions and ind is the corresponding indices. Next, we subtract the contributions of variable i and the variables following it to obtain matrix D. To facilitate the subsequent operations, the matrix D is transformed into a symmetric matrix with diagonal elements set to 0. The equations of calculating matrix D is defined as Eq. (11).

$$\begin{aligned} &D(i,j) = sortedF(i) - sortedF(j); \\ &D = D + D.'; \\ &D = D - diag(diag(D)); \end{aligned} \tag{11}$$

After obtaining the distance matrix D, we construct the initial cluster tree and specify the number of clusters $k = 5$. Then, the cluster tree is divided and labeled using the $cluster()$ function of MATLAB. Data with the same cluster label will be put into the same group.

Algorithm 3 implements the hierarchy of contribution based grouping method in pseudo-code. The pseudo-code of the hierarchy of contribution based grouping method is summarized in Algorithm 3.

Algorithm 3: A Grouping Method Based On Hierarchy of Contribution

Require: The improvement matrix F_{mat}
Ensure: The decomposition results
1: The number of rows to get the improvement matrix F_{mat} is saved in the variable $rows$;
2: **for** Go through each row **do**
3: The ith row of the improvement matrix F_{mat} will be arranged in descending order, and the results will be stored in the vector $sortedF$ and ind;
4: Generate the distance matrix D according to Eq. (11);
5: Construct the initial clustering tree.
6: Merging the closest clusters: Identify the two closest clusters and merge them into a new cluster;
7: Updating the distance matrix: Update the distance matrix by re-calculating the distance between the merged cluster and the other clusters;
8: Repeat the above two steps until all data points are merged into one cluster or a predefined stopping condition is met (such as the number of clusters);
9: Grouping each cluster puts the data with the same final cluster label into a group.
10: **end for**

Since the improvement matrix F_{mat} has 10 rows, Algorithm 3 can obtain 10 different decomposition results.

3) *A decomposition method based on features of data:* To enhance the decomposition diversity of fully non-separable large-scale problems, we designed another two methods based on data features. The statistical concepts of data can be seen as data features such as the value of mean, variance, standard deviation, mode, median, etc. The first feature based decomposition method utilizes mode and median values to classify decision variables. Thus, decision variables with similar mode and median values of contribution will be grouped together. The core concept of this decomposition method is shown in the following steps.

Step 1: Analyze the contribution information of all decision variables stored in the improvement matrix F_{mat}.

Step 2: Get the mode value of each decision variable. If there is no mode value, use the median value instead. Save the results in vector $result$.

Step 3: Sort the vector $result$ in descending order and decompose it into 10 different groups with a group size not exceeding 100.

The second feature based decomposition method uses the exponent index of variable contributions to classify decision variables. The exponent index represents the order of magnitude of contribution that a decision variable has on the overall objective function and thus can reflect the significance of decision variables. So decomposition could be made on this method allows us to mitigate the impact of excessively large or small contribution data on the grouping result, thus making a more rational decomposition. The main steps of the method are as follows:

Step 1: Calculate the exponent indices of all data in the improvement matrix F_{mat} and store them in matrix H.

Step 2: Find the mode value of each column in H, and record the corresponding row and column numbers.

Step 3: Retrieve the corresponding values in the improvement matrix F_{mat} and calculate their average. Save the results in the vector $result$.

Step 4: Sort the vector $result$ in descending order and group the data into sets of 100, with the final set possibly containing fewer than 100 elements.

To enhance the diversity of decomposition, random grouping is also utilized as an additional method. Therefore, the proposed hybrid grouping method encompasses 5 decomposition strategies that provide up to 14 decomposition outcomes for fully non-separable LSGO problem.

C. *The overall framework of the proposed method*

The overall framework of the method is structured as two stages. In the first stage, adaptive search method based on gradient approximation is proposed to make a rough and quick search while obtaining useful contribution information of the decision variables. In this stage, we use differential quotient to approximate gradient since the objective function is not continuous or differentiable. To make efficient calculations, we perform matrix operations to compute all difference quotients at one time. To better adapt different function variations, we design a new way to adjust the step size to generate new points. After the adaptive search algorithm, the contribution information of the decision variables will be calculated and stored in a matrix.

In the second stage, a hybrid grouping method is designed to reasonably decompose the fully non-separable LSGO problem. This method consists of five grouping approaches as described in the previous section: a contribution density based decomposition method, a contribution hierarchy based decomposition method, a mode-median feature based decomposition method, a contribution exponent index feature based decomposition method, and a random grouping strategy. a mode and median feature-based decomposition method a contribution exponent index feature-based decomposition method. The overall method follows these basic ideas and steps:

(i) Disrupt the order of all decision variables to make the algorithm more adaptable.

(ii) Use the gradient approximation based search method (Algorithm 1) to optimize each dimension of the problem.

(iii) Use FBG [26] to detect and decompose separable problems.

(iv) Use the hybrid decomposition method as detailed in Section III-B, the fully non-separable problems can be decomposed into 14 different grouping results.

(v) Based on the magnitude of improvement, optimize each sub-problem using SaNSDE [53] and the Matlab fmincon algorithm.

The algorithm terminates when the maximum number of function evaluations reaches the specified limit of 3e+6.

IV. Numerical Experiments

The proposed optimization method based on gradient approximation and hybrid grouping (GAHG) is implemented and tested on the CEC'2010 [54] and CEC'2013 [55] large-scale benchmark suites. The CEC'2010 benchmark suite consists of 20 test functions, each of which is 1000-dimensional, with f1-f3 being fully separable functions, f4-f18 being partially separable, and f19 and f20 being fully non-separable. The CEC'2013 benchmark suite consists of 15 test functions, each of which is 1000-dimensional, with f1-f3 being fully separable, f4-f11 being partially separable, and f12-f15 being fully non-separable. The global minimum function values of these functions are all 0. The following parameters are used in the experiments: the population size (NP) is 30, the number of function evaluations (FEs) is 3.0e+6. The proposed algorithm is executed 30 independent runs for each test function, and the average function value (Mean) and standard deviation (Std) are recorded.

The proposed algorithm is compared with some well-known and advanced large-scale optimization algorithms such as AGLDPSO [45], CBCC-DG2 [13], DECC-DG [11], RLLPSO [41] and TPCSO [50].

Tables I and II present the comparative results for the CEC'2010 and CEC'2013 benchmark suites, respectively. Significant mean results are shown in bold to enhance clarity. At the same time, Z-test at $\alpha = 0.05$ is used to evaluate whether GAHG performs significantly better than (+),similar to ($\approx$) or significantly worse than (-) the corresponding algorithm respectively.

Table I

The obtained experiment results on the 1000-D CEC'2010 functions

		GAHG	DECC-DG	RLLPSO	AGLDPSO	TPCSO
f1	Mean	3.83e-03	5.04e+04(+)	4.48e-22(-)	1.22e-21(-)	1.57e-21(-)
	Std	3.55e-04	2.26e+05	1.70e-22	8.14e-22	1.83e-22
f2	Mean	9.91e-11	4.38e+03(+)	9.72e+02(+)	2.35e+03(+)	4.51e+02(+)
	Std	2.19e-11	1.55e+02	5.71e+01	9.05e+02	1.05e+01
f3	Mean	9.86e-06	1.68e+01(+)	8.67e-14(-)	1.32e-13(-)	3.41e-14(-)
	Std	4.90e-07	2.20e-01	1.08e-14	1.27e-15	1.95e-15
f4	Mean	9.60e+03	3.60e+12(+)	6.50e+11(+)	1.10e+11(+)	8.60e+11(+)
	Std	1.43e+03	9.78e+11	5.02e+10	9.13e+10	1.42e+11
f5	Mean	1.39e+08	9.07e+07(-)	1.26e+07(-)	1.12e+07(-)	3.18e+06(-)
	Std	3.89e+07	1.83e+07	2.64e+06	3.37e+06	1.66e+06
f6	Mean	3.14e+06	3.85e+04(-)	3.35e-01(-)	1.99e+01(-)	3.58e-09(-)
	Std	1.06e+06	2.07e+05	5.17e-01	1.93e-02	2.77e-11
f7	Mean	1.02e-01	2.53e+04(+)	2.37e-01(+)	8.43e+04(+)	2.93e+02(+)
	Std	4.67e-03	1.30e+04	1.35e-01	6.32e+04	1.24e+02
f8	Mean	9.84e-02	4.63e+07(+)	1.43e+07(+)	2.96e+04(+)	2.79e+07(+)
	Std	7.88e-03	2.64e+07	1.13e+07	9.75e+03	5.08e+06
f9	Mean	1.01e+08	5.43e+07(-)	3.00e+07(-)	2.20e+07(-)	3.97e+07(-)
	Std	3.12e+07	3.52e+06	1.13e+06	2.36e+06	6.04e+06
f10	Mean	2.18e+03	4.23e+03(+)	9.00e+02(-)	2.38e+03(+)	4.22e+02(-)
	Std	2.55e+02	1.33e+02	5.79e+01	1.91e+02	3.73e+01
f11	Mean	8.06e+01	1.29e+01(-)	2.41e+00(-)	2.00e+01(-)	1.87e-13(-)
	Std	6.62e+00	5.78e-01	3.11e+00	2.78e+00	9.70e-14
f12	Mean	3.00e-06	1.58e+04(+)	1.12e+04(+)	1.33e+03(+)	3.24e+04(+)
	Std	5.48e-06	2.23e+03	6.09e+02	7.28e+02	1.14e+04
f13	Mean	5.65e-02	8.60e+03(+)	9.92e+02(+)	5.35e+02(+)	5.13e+02(+)
	Std	6.05e-03	3.87e+03	1.44e+02	1.42e+02	1.69e+02
f14	Mean	7.52e+05	3.07e+08(+)	1.11e+08(+)	6.30e+07(+)	1.47e+08(+)
	Std	3.40e+05	1.39e+07	2.32e+06	4.37e+06	8.83e+06
f15	Mean	4.20e+03	6.07e+03(+)	8.41e+02(-)	2.34e+03(-)	7.96e+02(-)
	Std	2.96e+02	1.50e+02	4.71e+01	1.95e+02	1.55e+02
f16	Mean	1.49e+02	1.49e-01(-)	1.88e+00(-)	6.21e+00(-)	3.23e-13(-)
	Std	6.71e+00	3.82e-01	3.02e+00	2.68e+00	7.71e-14
f17	Mean	1.59e-08	1.26e+05(+)	5.31e+04(+)	2.42e+04(+)	1.78e+05(+)
	Std	3.63e-09	4.43e+03	1.63e+03	2.54e+03	4.98e+04
f18	Mean	6.64e-01	3.21e+09(+)	2.10e+03(+)	1.40e+03(+)	1.20e+03(+)
	Std	1.84e+00	8.81e+08	3.90e+02	2.85e+02	3.66e+02
f19	Mean	4.49e+05	2.17e+06(+)	8.71e+05(+)	8.43e+05(+)	2.95e+06(+)
	Std	3.62e+04	1.22e+05	2.31e+04	4.61e+04	2.98e+05
f20	Mean	2.63e-03	9.18e+10(+)	1.84e+03(+)	1.44e+03(+)	9.96e+02(+)
	Std	1.27e-02	8.99e+09	1.68e+02	1.56e+02	5.24e+01

Table II

The obtained experiment results on the 1000-D CEC'2013 functions

		GAHG	CBCC-DG2	RLLPSO	AGLDPSO	TPCSO
f1	Mean	4.12e-03	8.65e+05(+)	9.01e-22(-)	1.46e-21(-)	2.38e-21(-)
	Std	2.97e-04	1.16e+06	3.50e-22	1.19e-22	2.30e-22
f2	Mean	1.78e-07	1.41e+04(+)	1.04e+03(+)	2.56e+03(+)	5.55e+02(+)
	Std	1.96e-07	1.54e+03	7.40e+01	1.29e+02	1.35e+01
f3	Mean	2.00e+01	2.06e+01(+)	2.16e+01(+)	2.16e+01(+)	2.14e+01(+)
	Std	4.30e-06	8.62e-03	6.21e-03	4.96e-03	1.36e-02
f4	Mean	8.25e+03	3.39e+07(+)	4.44e+09(+)	1.13e+09(+)	5.88e+09(+)
	Std	4.49e+03	1.77e+07	4.99e+08	7.72e+08	7.24e+08
f5	Mean	4.27e+06	2.14e+06(-)	7.25e+05(-)	8.57e+05(-)	5.00e+05(-)
	Std	6.82e+05	4.24e+05	1.01e+05	2.30e+05	1.35e+04
f6	Mean	1.00e+06	1.05e+06(+)	1.06e+06(+)	1.05e+06(+)	1.02e+06(+)
	Std	1.59e+04	3.37e+03	9.95e+02	1.58e+03	3.70e+03
f7	Mean	3.37e-08	2.95e+07(+)	9.84e+05(+)	3.71e+06(+)	1.04e+06(+)
	Std	3.30e-08	2.78e+07	2.47e+05	2.38e+06	3.49e+05
f8	Mean	1.06e+08	4.28e+10(+)	8.56e+13(+)	1.36e+13(+)	1.17e+14(+)
	Std	3.76e+07	8.86e+10	1.07e+13	4.73e+12	1.52e+13
f9	Mean	3.51e+08	1.70e+08(-)	3.85e+07(-)	1.59e+08(-)	2.89e+07(-)
	Std	7.63e+07	3.16e+07	8.14e+06	2.51e+07	2.86e+06
f10	Mean	9.15e+07	9.28e+07(+)	9.40e+07(+)	9.39e+07(+)	9.11e+07(≈)
	Std	1.06e+06	7.16e+05	2.15e+05	3.50e+05	3.54e+05
f11	Mean	5.06e-01	6.59e+08(+)	9.25e+11(+)	9.35e+11(+)	7.50e+07(+)
	Std	8.89e-01	2.80e+08	8.70e+09	4.59e+10	9.42e+06
f12	Mean	3.14e-04	5.81e+07(+)	1.87e+03(+)	1.53e+03(+)	1.60e+03(+)
	Std	1.39e-03	7.83e+07	1.72e+02	2.79e+02	2.60e+01
f13	Mean	2.49e+07	6.03e+08(+)	2.17e+08(+)	2.12e+10(+)	1.68e+08(+)
	Std	1.48e+07	1.37e+08	5.46e+07	3.49e+09	3.98e+07
f14	Mean	5.31e+07	1.11e+09(+)	5.06e+07(≈)	9.46e+07(+)	9.15e+07(+)
	Std	1.50e+07	1.07e+09	2.16e+07	5.68e+06	2.72e+07
f15	Mean	1.72e+07	7.11e+06(-)	1.63e+06(-)	2.73e+06(-)	8.11e+06(-)
	Std	3.95e+06	1.38e+06	5.88e+04	1.19e+05	4.02e+05

A. *Comparison results on CEC'2010 benchmark suite*

Firstly, the detailed comparison results on the CEC'2010 test suite are presented in Table I. According to Table I, GAHG demonstrates better performance relative to the other algorithms. GAHG gets 11 best results over 20 test functions, while DECC-DG gets none best results, RLLPSO gets 4 best results, AGLDPSO gets 2 best results, and TPCSO gets 7 best results.

For fully separable test problems f1-f3, it seems that GAHG performs worse than RLLPSO, AGLDPSO and TPCSO with only one best results. However, we can see that GAHG successfully achieved the global optimal solutions for all

these three problems with an accuracy error as small as $e-3, e-11$ and $e-6$ respectively while the other three algorithms failed to locate the global optimal solution for $f2$ (with an accuracy error as large as $e+2, e+3$ and $e+2$ respectively). From this perspective, we can say that GAHG is more stable and performs better than the other compared algorithms. Since for fully separable problems, only the proposed gradient approximation based algorithm described in section III-A is used, the good performance indicates that the proposed algorithm is effective.

For partially separable functions f4-f18, GAHG achieves the best performance on 8 out of all 15 test problems (f4, f7, f8, f12, f13, f14, f17, f18). On the contrary, all the other compared algorithms can only achieve the best performance on no more than five problems.

For fully non-separable problems f19 and f20, the performance of GAHG is much better than the compared algorithms. Especially for problem $f20$, GAHG successfully located the global optimal solution with an accuracy error as small as $e-3$ while the other compared algorithms all failed (with an accuracy error as large as $e+2, e+3$ or even $e+10$). The good performance indicates the proposed hybrid grouping strategy is effective. By decomposing the problem, the optimization method can obtain better efficiency. By increasing the diversity of the decomposition results using the proposed hybrid grouping strategy, we can obtain better optimization results on fully non-separable large-scale problems.

Overall, the proposed GAHG algorithm achieved competitive good results: it outperforms DECC-DG for 14 wins 5 losses and 1 tie, outperforms RLLPSO for 11 wins and 9 losses. outperforms AGLDPSO algorithm for 12 wins and 8 losses, and outperforms TPCSO for 11 wins and 9 losses. GAHG shows good performance on the CEC'2010 benchmark suite. The gradient approximation based method and the hybrid grouping method are effective.

B. *Comparison on CEC'2013 benchmark suite*

CEC'2013 large-scale benchmark suite contains 15 test problems. It is the most challenging benchmark by now and there is no algorithm ever successfully located all 15 global optimal solutions. Thus we use this benchmark suite to test and compare the performance of the proposed method. Table II displays the optimization results of the proposed method and the other four algorithms.

Firstly, for fully separable problems f1-f3, GAHG outperformed all the other four algorithms with two wins on f2 and f3. And for test problem $f1$, GAHG is only slightly inferior than the other three algorithms with an accuracy error $e-3$ which is very close to the global optimal value. Since for fully separable problems, only the gradient approximation based search algorithm is used, it indicates the effectiveness and efficiency of the proposed algorithm.

Secondly, we can see that for the eight partially separable problems f4-f11, GAHG achieved the best performance. GAHG performs best on five problems (f4, f6, f7, f8, f11) while other compared algorithms can achieve the best performance on no more than three test problems. For partially separable problems, the proposed gradient approximation based search algorithm is applied

and contribution information of each decision variable is collected. Together with the contribution information, the self-adaptive deferential algorithm SaNSDE and the Matlab fmincon method perform well. This shows the effectiveness and robustness of GAHG in solving partially separable functions.

Lastly, for the four fully non-separable problems f12-f15, we can see that GAHG also performs best with three wins. GAHG achieves three best results (f12-f14) and the second best algorithm RLLPSO achieves two best results. So we can get the conclusion that GAHG has significant advantages in dealing with fully non-separable hard LSGO problems. Since for fully non-separable problems, theoretically, there is no way to decompose the problem, and thus the algorithm may be ineffective. We use the proposed hybrid grouping method to reasonably decompose the problem and then the optimization can be more effective and efficient. However, as is demonstrated in the comparative table, even the best performance results are far from global optimal value, e.g., the best results for f12, f13, and f14 are as far as $e+7$ and for f15 the best result is as far as $e+6$ from the global optimal value. This indicates that large-scale optimization is very challenging and much more effort should be continued to make on it.

Overall, GAHG performs better than other compared algorithms on 11 test problems: GAHG is better than RLLPSO (9 wins, 5 loses, 1 tie), AGLDPSO (10 wins, 5 losses), TPCSO (10 wins, 4 losses, 1 tie), and CBCC-DG2 (12 wins, 3 losses).

In summary, based on the results of the comparison on the CEC'2013 large-scale benchmark test suite, we can conclude that the proposed GAHG method has good performance on different function types and has unique advantages when dealing with fully separable, and fully non-separable LSGO problems.

V. Conclusions

Solving the LSGO problem is extremely difficult because of their high-dimensionality, countless local optimal solutions and significant computational load. In order to effectively solve these problems, a new method named GAHG is proposed in which a gradient approximation based search algorithm and a hybrid grouping method is designed. In the first stage of GAHG, we design an adaptive gradient approximation based one-dimensional search algorithm. This search algorithm can make quick scans and reduce the search area, also it can explore the contribution of the decision variables and store the important information for the subsequent optimization. In the second stage, a hybrid grouping method is introduced for fully non-separable large-scale problems. It aims to decompose fully non-separable LSGO problems and enhance the ease and effectiveness of optimization. At the same time, by providing various decomposition results, it can also ensure that interactive variables have a greater chance of being put into the same subgroup. In this way, we can make better use of the interaction between variables to further improve the performance of global optimization. The proposed gradient approximation based search algorithm and hybrid grouping method is a useful and effective new approach for solving

complex large-scale global optimization problems. Experimental results show that the current algorithms cannot find the best solutions in all cases. This supports the view that there is no free lunch theorem, that is, no algorithm can be applied to all cases and find the best solutions to LSGO problems. Even the most advanced algorithms can't solve these problems completely. Therefore, large-scale optimization remains a challenging task, and developing improved search and decomposition strategies will be the main focus of our future research.

VI. Acknowledgements

This work was supported by National Natural Science Foundation of China (No. 62002289).

References

[1] C. Wang and J. Gao, "High-dimensional waveform inversion with cooperative coevolutionary differential evolution algorithm," *IEEE Geoscience and Remote Sensing Letters*, vol. 9, no. 2, pp. 297-301 (2012).

[2] S. Mahdavi, S. Rahnamayan, and M. E. Shiri, "Multilevel framework for large-scale global optimization," *Soft Computing,* vol. 21, pp. 4111-4140 (2017).

[3] M. Sam, A. DAriano, F. Corman, and D. Pacciarelli, "Metaheuristics for efficient aircraft scheduling and re-routing at busy terminal control areas," *Transportation Research Part C: Emerging Technologies,* vol. 80, pp. 485-511 (2017).

[4] V. J. Kadam and S. M. J. and, "Performance analysis of hyperparameter optimization methods for ensemble learning with small and medium sized medical datasets," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 23, no. 1, pp. 115-123 (2020).

[5] M. A. Potter and K. A. D. Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1-29 (2000).

[6] R. Salomon, "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms," *Biosystems*, vol. 39, no. 3, pp. 263-278 (1996).

[7] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in Parallel Problem Solving from Nature—PPSN III (Y. Davidor, H.-P. Schwefel, and R. Männer, eds.), (Berlin, Heidelberg), pp. 249-257, Springer Berlin Heidelberg (1994).

[8] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences,* vol. 178, no. 15, pp. 2985-2999, Nature Inspired Problem-Solving (2008).

[9] M. Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," Proceedings of the IEEE Congress on Evolutionary Computation (CEC2010) ; Conference date: 23-07-2010 (July 2010).

[10] Z. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp. 1663-1670 (2008).

[11] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, pp. 378-393 (JUN 2014).

[12] Y. Sun, M. Kirley, and S. K. Halgamuge, "Extended differential grouping for large scale global optimization with direct and indirect variable interactions," Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (2015).

[13] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, "Dg2: A faster and more accurate differential grouping for large-scale black-box optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 6, pp. 929-942 (2017).

[14] Y. Sun, M. Kirley, and S. K. Halgamuge, "A recursive decomposition method for large scale continuous optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 5, pp. 647-661 (2018).

[15] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Trans. Math. Softw.*, vol. 42 (June 2016).

[16] M. Yang, A. Zhou, C. Li, and X. Yao, "An efficient recursive differential grouping for large-scale continuous problems," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 1, pp. 159-171 (2021).

[17] H.-b. Xu, F. Li, and H. Shen, "A three-level recursive differential grouping method for large-scale continuous optimization," *IEEE Access,* vol. 141946–141957, pp. 8 (2020).

[18] X. Liu, F. Li, Y. Xu, and H. Shen, "A new three-level recursive differential grouping method for large-scale optimization," in 2022

34th Chinese Control and Decision Conference (CCDC), pp. 236-241 (2022).

[19] L. Zheng, G. Xu, and W. Chen, "Three stages recursive differential grouping for large-scale global optimization," *IEEE ACCESS*, vol. 11, pp. 109734-109746 (2023).

[20] Y. Sun and H. Yue, "An improved decomposition method for large-scale global optimization: bidirectional-detection differential grouping," *Applied Intelligence*, vol. 52, pp. 11569-11591 (AUG 2022).

[21] J. Ma, F. Chang, and X. Yu, "Large-scale evolutionary optimization approach based on decision space decomposition," *Frontiers in Energy Research*, vol. 10, JUN 9 (2022).

[22] A. Kumar, S. Das, and R. Mallipeddi, "An efficient differential grouping algorithm for large-scale global optimization," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 1, pp. 32-46 (2024).

[23] J.-Y. Li, Z.-H. Zhan, K. C. Tan, and J. Zhang, "Dual differential grouping: A more general decomposition method for large-scale optimization," *IEEE Transactions on Cybernetics,* vol. 53, no. 6, pp. 3624-3638 (2023).

[24] R. Zhong, E. Zhang, and M. Munetomo, "Cooperative coevolutionary surrogate ensemble-assisted differential evolution with efficient dual differential grouping for large-scale expensive optimization problems," *Complex & Intelligent Systems*, vol. 10, pp. 2129-2149, 4 (2024).

[25] X.-M. Hu, F.-L. He, W.-N. Chen, and J. Zhang, "Cooperation coevolution with fast interdependency identification for large scale optimization," *Information Sciences*, vol. 381, pp. 142-160 (2017).

[26] Y. Wang, H. Liu, F. Wei, T. Zong, and X. Li, "Cooperative Coevolution with Formula-Based Variable Grouping for Large-Scale Global Optimization," *Evolutionary Computation,* vol. 26, pp. 569-596, 12 (2018).

[27] H. Liu, Y. Wang, L. Liu, and X. Li, "A two phase hybrid algorithm with a new decomposition method for large scale optimization," *Integrated Computer-Aided Engineering*, vol. 25, pp. 1-19, 04 (2018).

[28] H. Liu, Y. Wang, and N. Fan, "A hybrid deep grouping algorithm for large scale global optimization," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 6, pp. 1112-1124 (2020).

[29] H. Liu, Y. Cheng, S. Xue, and S. Tuo, "A space-reduction based three-phase approach for large-scale optimization," *APPLIED SOFT COMPUTING*, vol. 144 (SEP 2023).

[30] M. Yang, M. N. Omidvar, C. Li, X. Li, Z. Cai, B. Kazimipour, and X. Yao, "Efficient resource allocation in cooperative co-evolution for large-scale global optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 493-505 (2017).

[31] M. Yang, A. Zhou, C. Li, J. Guan, and X. Yan, "Ccfr2: A more efficient cooperative co-evolutionary framework for large-scale global optimization," *Information Sciences*, vol. 512, pp. 64-79 (2020).

[32] M. Yang, A. Zhou, X. Lu, Z. Cai, C. Li, and J. Guan, "Ccfr3: A cooperative co-evolution with efficient resource allocation for large-scale global optimization," *Expert Systems with Applications*, vol. 203, p. 117397 (2022).

[33] M. Yang, J. Gao, A. Zhou, C. Li, and X. Yao, "Contribution-based cooperative co-evolution with adaptive population diversity for large-scale global optimization [research frontier]," *IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE*, vol. 18, pp. 56-68 (AUG 2023).

[34] R. Jarray, M. A. Dhaifallah, H. Rezk, and S. Bouallègue, "Parallel cooperative coevolutionary grey wolf optimizer for path planning problem of unmanned aerial vehicles," Sensors (Basel, Switzerland), vol. 22 (2022).

[35] A. Kelkawi, M. El-Abd, and I. Ahmad, "Gpu-based cooperative co-evolution for large-scale global optimization," *Neural Computing and Applications*, vol. 35, pp. 4621-4642 (Feb. 2023). Publisher Copyright: 2022, The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature.

[36] L.-Y. Tseng and C. Chen, "Multiple trajectory search for large scale global optimization," in 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp. 3052-3059 (2008).

[37] D. Molina, M. Lozano, and F. Herrera, "Ma-sw-chains: Memetic algorithm based on local search chains for large scale continuous global optimization," in *IEEE congress on evolutionary computation*, pp. 1-8, IEEE (2010).

[38] A. LaTorre, S. Muelas, and J.-M. Peña, "Large scale global optimization: Experimental results with mos-based hybrid algorithms," in

2013 IEEE congress on evolutionary computation, pp. 2742-2749, IEEE (2013).

[39] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 191-204 (2015).

[40] Q. Yang, W.-N. Chen, T. Gu, H. Zhang, J. D. Deng, Y. Li, and J. Zhang, "Segment-based predominant learning swarm optimizer for large-scale optimization," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2896-2910 (2017).

[41] F. Wang, X. Wang, and S. Sun, "A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization," *Inf. Sci.*, vol. 602, p. 298-312 (jul 2022).

[42] H. Deng, L. Peng, H. Zhang, B. Yang, and Z. Chen, "Ranking-based biased learning swarm optimizer for large-scale optimization," *Inf. Sci.*, vol. 493, p. 120-137 (Aug 2019).

[43] M. Wang and M. Jiang, "Combing gibbs-sampling with adaptive particle swarm for large scale global optimization," in 2018 Tenth International Conference on Advanced Computational Intelligence (ICACI), pp. 856-860 (2018).

[44] Y. Liu and Guiyan Hu, "A modified particle swarm optimization for large-scale numerical optimizations and engineering design problems," *Journal of Intelligent Manufacturing*, vol. 30, no. 6 (2019).

[45] Z.-J. Wang, Z.-H. Zhan, S. Kwong, H. Jin, and J. Zhang, "Adaptive granularity learning distributed particle swarm optimization for large-scale optimization," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1175-1188 (2021).

[46] X. Zhang and Q. Lin, "Three-learning strategy particle swarm algorithm for global optimization problems," *Information Sciences*, vol. 593, pp. 289-313 (2022).

[47] D. Li, L. Wang, W. Guo, M. Zhang, B. Hu, and Q. Wu, "A particle swarm optimizer with dynamic balance of convergence and diversity for large-scale optimization," *Applied Soft Computing*, vol. 132, p. 109852 (2023).

[48] S. Liu, Z.-J. Wang, Y.-G. Wang, S. Kwong, and J. Zhang, "Bi-directional learning particle swarm optimization for large-scale optimization," *Applied Soft Computing*, vol. 149, p. 110990 (2023).

[49] W.-Y. Fu, "Accelerated high-dimensional global optimization: A particle swarm optimizer incorporating homogeneous learning and autophagy mechanisms," *Information Sciences*, vol. 648, p. 119573 (2023).

[50] C. Huang, X. Zhou, X. Ran, Y. Liu, W. Deng, and W. Deng, "Co-evolutionary competitive swarm optimizer with three-phase for large-scale complex optimization problem," *Information Sciences*, vol. 619, pp. 2-18 (2023).

[51] M. Ester, H. P. Kriegel, J. Sander, and X. Xiaowei, "A density-based algorithm for discovering clusters in large spatial databases with noise," 12 1996.

[52] J. H. Ward Jr, "Hierarchical grouping to optimize an objective function," *Journal of the American statistical association*, vol. 58, no. 301, pp. 236-244 (1963).

[53] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp. 1110-1116 (2008).

[54] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark functions for the cec'2010 special session and competition on large-scale," (2009).

[55] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. Ping Chen, C. M. Chen, and Z. Yang, "Benchmark functions for the cec'2013 special session and competition on large-scale global optimization," (2008).

Received November, 2024.