Received June 18, 2019, accepted July 18, 2019, date of publication July 29, 2019, date of current version August 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2931824

A New Hybrid Algorithm for Solving Large Scale Global Optimization Problems

XIANGJUAN WU, YUPING WANG, (Senior Member, IEEE), JUNHUA LIU, AND NINGLEI FAN
School of Computer Science and Technology, Xidian University, Xi'an 710071, China

Corresponding author: Yuping Wang (ywang@xidian.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872281, and in part by the Natural Science Foundation of Shaanxi Province under Grant 2016JZ022.

ABSTRACT For large scale global optimization (LSGO) problems, many algorithms have been proposed in recent years. However, there are still some issues to be further handled since the search space grows exponentially and the problem solving becomes more and more difficult as the problem scale becomes larger and larger. In this paper, we propose a new hybrid algorithm for solving large-scale global optimization problems. First, we adopt an existing group algorithm to divide the large-scale problem into several small-scale problems. Second, a modified self-adaptive discrete scan method is designed to roughly scan the whole search space and then focus the search on the promising regions. Third, a hybrid search strategy is proposed, which adaptively chooses the one-dimensional search scheme or the covariance matrix adaptation evolutionary strategy to solve the subproblems of separable, partially (additively) separable problems or non-separable problems, respectively. To demonstrate the performance of the proposed algorithm, we conduct the experiments on 15 difficult LSGO problems in CEC'2013 benchmark suite and compare the performance of the proposed algorithm with that of the several state-of-the-art algorithms. The results show that the proposed algorithm is more effective than the compared algorithms in terms of solution accuracy.

INDEX TERMS Large scale global optimization, hybrid algorithm, one-dimensional search, covariance matrix adaptation evolutionary strategy.

I. INTRODUCTION

Many real world problems can be modeled as continuous large scale global function optimization problems, and these problems appear in a variety of fields such as biometrics, aerospace, network antennas, data mining and so on [1]–[4]. Usually, a global optimization problem with dimension higher than or equal to 1000 is called a large scale global optimization (LSGO) problem.

At present, there are two important challenges in solving LSGO problems. The first one is that the search space for LSGO problems grows exponentially as the problem dimension increases [5]. For example, for a 1000-dimension global optimization problem, when the search range in each dimension is [-50, 50], the search space contains as large as 100^{1000} points even if we only consider integers in each dimension. The extremely huge search space will make it impossible to search the whole space because the existing optimization algorithms cannot explore the entire space within valid time or limited computing resources. Liu and Wang [6] proposed a self-adaptive discrete scan method trying to shrink the search space by finding the promising region in the huge search space quickly and roughly from the current best solution. However, this scan method adopt a fixed shrink rate which cannot be suitable to different situations and different stages of evolution process, affecting the performance of scan method greatly. Therefore, designing a reasonable self-adaptive scan method to narrow the search space and to focus the search range on the most potential domain can make a great contribution to the whole optimization algorithm.

Another major challenge is that the LSGO problems cannot be solved effectively by using the traditional mathematical optimization algorithms such as Quasi-Newton method [7], conjugate gradient method [8] and so on. This is attributed to the fact that the LSGO problems are generally complex, non-convex and non-differentiable. Evolutionary Algorithms (EAs) such as genetic algorithms (GAs), estimation of distribution algorithms (EDAs), evolution strategy (ES), particle swarm optimization (PSO), ant colony optimization (ACO), etc. have been widely applied in LSGO

The associate editor coordinating the review of this manuscript and approving it for publication was Xiangtao Li.

problems because of their characteristics of self-organized, self-adaptive and derivative free [9]. These algorithms work well for solving medium and small size optimization problems. However, it has been demonstrated that the performance of EAs deteriorates rapidly as the problem dimension increases [10], which is called 'the curse of dimensionality'. Therefore, designing an effective and efficient algorithm for LSGO problems is a non-trivial task. It has attracted many researchers and a great deal of research works have been done in the past decades. An efficient approach for solving LSGO problems usually uses a divide-and-conquer strategy and tries to divide a large-scale problem into a set of smaller sub-problems. By optimizing these sub-problems separately, one can get the global optimal solution for the original problem. Cooperative Co-evolution (CC) has been proposed by Potter and De Jong [11] based on the above strategy. To improve the performance of CC, in recent years, many decomposition strategies have been developed, such as differential grouping (DG) and its variants [12]–[16], recursive differential grouping (RDG) and its variants [17], [18], formula-based grouping(FBG) [19] and so on. Recent studies show that these state-of-the-art decomposition methods can decompose most of the existing large-scale partially separable benchmarks with 100 percent accuracy [20]. The definitions of separable, partially (additively) separable and non-separable problems can be found in literatures [12], [21], [22]. The existing grouping-based optimization algorithms, such as DECC-RDG [17], CMAESCC-RDG [17], CMAESCC-RDG2 [18], and CBCC-DG2 [16] improve the performance of CC framework effectively in solving LSGO problems. However, the existing algorithms are still far unsatisfactory because the best solutions they obtained for the majority benchmarks are far away from the true global optimal solutions. Thus, developing more effective algorithms still remain an arduous challenge.

In this paper, we propose a new hybrid algorithm based on a modified self-adaptive discrete scan method and a sub-problem solving strategy called the combination optimization strategy. The modified self-adaptive discrete scan method guides search from the whole search space to a narrowed promising region with a self-adaptive shrink rate. By using the self-adaptive shrink rate, discrete scan method can adjust the shrink speed of search space adaptively to fit the different situations on different evolutionary stages. As a result, it can provide potential initial solutions for the algorithms to solve sub-problems. And then, a strategy combining the covariance matrix adaption evolution strategy (CMAES) with one-dimensional search method to optimize the sub-component is designed to improve the performance of the hybrid algorithm.

The remainder of this paper is organized as follows. Section 2 describes the related work on solving large scale optimization problems briefly. Section 3 presents the proposed method in detail. Numerical experiments are carried out and analyzed in Section 4. Section 5 concludes the paper and points out the future works.

II. RELATED WORK

Large-scale global optimization is a hot research field in the past decade. A large number of metaheuristic approaches have been developed. In this section, we give a brief description about the state-of-the-art ones for solving LSGO problems. These algorithms can be divided into two classes: decomposition based algorithms which decompose the problem into a set of small-scale sub-components (sub-problems) and non-decomposition based algorithms which do not explicitly decompose the LSGO problems [6].

A. DECOMPOSITION BASED ALGORITHMS

CC based algorithm is a typical class of decomposition based algorithms. The proposed algorithm belongs to this class and we shall briefly introduce CC based algorithms. A CC based algorithm consists of two phases: decomposition and optimization [17]. The whole problem is divided into several sub-problems and optimizing each sub-problems independently. Then the solutions obtained in all sub-problems are combined into one solution as the current solution of the whole problem in each cycle. One major issue affecting the performance of the CC based algorithms is the way of the decomposition [12], [13], [23]. In recent years, many decomposition methods have been proposed to decompose the LSGO problem into small sub components. These decomposition methods can be classified into two categories: manual decomposition and automatic decomposition [17]. Also, different optimization methods are designed or selected to solve each sub-problem in a round robin fashion.

In manual decomposition methods, the size of each sub-component and the number of sub-components are manually designed. For example, in literature [11], the univariable grouping method is proposed to decompose a n-dimension problem into n 1-dimension sub-problems, which does not consider the interaction between decision variables. Yang et al. [24] propose a random grouping method (RG), which randomly splits decision variables into m s-dimension sub-components and use differential evolution under CC framework (DECC-G) to optimize each sub-problem. The 'random' strategy ensures that each variable has the same chance to be grouped in any of the sub-problems to increase the chance of assigning two interacting decision variables to one group. And also a particle swarm optimization (PSO) [25] and an artificial bee colony (ABC) [26] algorithm are incorporated with RG to improve the optimization performance. However, this grouping method needs the user to predetermine the sub-component size. These algorithms have proven to be very effective for fully separable problems. But the performance deteriorates when the interaction exists among three or more variables. The reason is that these methods do not take the structure of the potential interaction between decision variables into account fully. For problems with non-separable sub-problems, Omidvar et al. [27] develop algorithm DECC-D incorporating differential evolutionary (DE) with the delta grouping. In delta grouping, it assumes that when assigning interacting variables into different subgroups, the improvement on the objective function value of the interacting variables is limited. The delta grouping classifies the variables with the smaller delta value into the same subgroup, where the delta value refers to the amount of change on the objective function by a certain variable in each iteration. The DECC-G based on delta grouping improves the performance of DE effectively. However when it is applied to the problems with more than one non-separable groups, its performance drops significantly.

To overcome the disadvantage of the manual decomposition and improve the decomposition accuracy, many automatic decomposition methods have been proposed in decade years, where the grouping methods are based on the interaction check (i.e., by identifying the interaction) between variables and assign the interacting variables into the same group without setting the size of sub-component or the number of sub-components manually. A representative automatic decomposition method, cooperative co-evolution with variable interaction learning (CCVIL), is proposed based on the non-monotonicity detection by Chen et al. [28]. In this method, if the monotonicity of a variable does not vary with the value of another variable, the two variables are considered to be independent. Otherwise they are considered to be interact with each other. The CCVIL algorithm has been proven to be more accurate than most manual decomposition methods. However, the change of monotonicity cannot be detected easily in the limited number of function evaluations, so the performance of the CCVIL is limited by the maximum number of function evaluations specified in the optimization algorithms. In literature [12], differential grouping (DG) is developed based on the non-linearity detection. If the differential of objective function value caused by perturbing one variable changes for different values of another variable, the two variables are defined to be interacted. The experiment results show that DG has a higher grouping accuracy than CCVIL on the CEC'2010 benchmark problems. It has also been demonstrated by experiments that the DECC-DG algorithm using this grouping method has achieved the better results than the predecessors on CEC'2010 benchmark problems, but it still cannot find the global optimal solutions for many benchmarks. The main reason is that the DG grouping method can only find directly related variables, so the variables with indirect interaction cannot be correctly grouped. To overcome this issue, Mei et al. [14] develop a global differential grouping (GDG) method with an interaction matrix between variables which indicates whether the variables are related to each other or not. GDG outperforms other five algorithms on CEC'2010 benchmark suite when it is embedded into a variant of covariance matrix adaption evolution strategy (CMAES) [29] under CC framework (CC-GDG-CMAES). Since both DG and GDG need calculate all the differential values between every pair variables, their computational complexity is high. To tackle this problem, a recursive differential grouping (RDG) is proposed based on non-linearity detection [17]. It firstly identifies the interaction between the chosen variable x_i and the remaining variables. If there is any interaction identified, the set of remaining variables is divided into two subsets $G1$ and $G2$ equally. And then the interaction between x_i and each subset is checked, respectively. This process is recursively executed until interaction between x_i and all of the variables are identified. If no variable interacted with x_i, x_i will be placed into a separable group. When no interaction can be identified, RDG outputs each variable as a separable group, i.e., all the variables are independent. Later, Sun et al. [17] proposed CMAESCC-RDG in which CMAES is adopted as the subcomponent optimizer. Compared with MOS and MA-SW-Chains, CMAESCC-RDG performs better than these two state-of-the-art algorithms on CEC'2010 benchmark and CEC'2013 benchmark suite. However, in RDG, the parameter to control the threshold value determining whether two variables interact with each other or not should be set manually. Sun et al. [18] proposed an adaptive threshold parameter estimation scheme with recursive differential grouping called RDG2. RDG2 achieves higher decomposition accuracy than RDG.

The aforementioned grouping methods are all proposed for the black box optimization problems where the information of function expression is supposed unknown or unavailable. However, in many real world problems, the function expressions are known or available. In this situation, it is very possible to design more effective variable group method. For example, in literature [19], Wang et al. develop a formula based grouping (FBG) for white box problems where the information of function expression is known or available. This method can accurately identify the interaction of variables and can group variables correctly by scanning function expressions. In this paper, we suppose that the problem to be tackled is a black box problem and we shall adopt RDG2 as the decomposition method.

An variant of CC methods is the contribution based cooperative co-evolution framework (CBCC) [30], which combines CC with a computation resource allocation scheme. CBCC tries to improve the efficiency of the CC methods by assigning different computational resources to different sub-components based on their contribution to the improvement of objective function value. CBCC has three major variants: CBCC1 [30], CBCC2 [30] and CBCC3 [20]. In CBCC1, sub-groups are optimized one by one in each cycle in CC framework, and then the sub-group with the largest cumulative contribution is optimized with one more chance, whereas CBCC2 chooses the sub-group with the largest cumulative contribution to optimize repeatedly until it enters a stagnant state. The experimental results on the CEC'2013 benchmark suite [31] show that the resource allocation policy of CBCC1 encourages exploration search and that of CBCC2 encourages the exploitation search, respectively. To balance the exploration and exploitation, CBCC3 uses the contribution in a single cycle instead of cumulative contributions as the measure of contribution and gets better results. In this paper, we adopt main idea of the resource allocation scheme in CBCC3 to assign more computation resources to the group with the largest contribution in each cycle.

B. NON-DECOMPOSITION BASED ALGORITHMS

Apart from the decomposition based algorithms, there are many approaches which do not explicitly decompose the problems. These algorithms often combine multiple algorithms or multiple techniques together to improve the ability of solving large-scale optimization problems. For example, the MA-SW-Chains algorithm proposed in literature [32] is a memetic algorithm which uses steady-state GA as a global search method and Solis and Wets algorithm as a local search method. At the same time, a local search chain is applied to store the local search state of the variable so that the same parameters can be used in the next time when the local search is performed on the variable. In literature [33], a multiple trajectory search (MTS) is investigated to solve the LSGO problems. In the MTS, the performance of three local search methods is tested and the best one is chosen to optimize the problems. LaTorre et al. [34] [35] propose a combination algorithm called multiple offspring sampling (MOS). It is a dynamic hybrid evolutionary algorithm seamlessly combining different types of algorithms, e.g. genetic algorithm, Solis and Wets, opposition based differential evolution [36], MTS-LS and so on. MOS selects one algorithm among these above algorithms dynamically based on their performance in every phase to optimize the LSGO problems and gets very good performance (it is the champion algorithm in CEC'2013 and CEC'2015 algorithm competitions). Later, the authors in literature [37] propose a modified whale optimization algorithm (MWOA), which effectively combines three techniques. A nonlinear dynamic strategy based on cosine function is proposed to control the optimization parameters firstly. Then, a quadratic interpolation method is used to improve the ability of local search, and finally the Lévy-flight strategy is adopted to make the algorithm jump out of the local optimal solution. Recently, Molina et al. [38] propose a SHADE algorithm that combines the DE algorithm with a set of local search algorithms in an iterative manner. The appropriate local optimization algorithm is selected according to their improvements in the previous optimization phase. It is also a dynamical hybrid algorithm. The experiment results show that the aforementioned non-decomposition based algorithms perform better than the state-of-the-art decomposition based algorithms. This indicates that the appropriate combination strategy in non-decomposition based algorithms is very important and can improve the performance of algorithms significantly for the LSGO problems. However, the performance of these non-decomposition based algorithms are far not satisfactory because they cannot find a good approximate global solution for many benchmark problems. Therefore, designing an effective combination strategy to improve the search ability of non-decomposition based algorithms is still a challenging and urgent issue.

Besides the aforementioned methods, there are some other non-decomposition based algorithms proposed to improve the ability of solving LSGO problems, e.g. auxiliary function methods [39]–[42] and parallel algorithms [43], [44], etc. In this paper, we shall make use the advantages of both decomposition based algorithms and non-decomposition based algorithms and focus the research on designing effective combination strategy and combining it with CBCC framework.

III. PROPOSED METHOD

In this section, we develop a new hybrid algorithm for solving LSGO problems. The algorithm consists of two phases, decomposition phase and optimization phase. Its pseudo code is shown in Algorithm 1. In the first phase (i.e. decomposition phase), the decomposition method RDG2 is used to divide the problem into m small scale sub-problems. In the second phase (i.e. optimization phase), the modified self-adaptive discrete scan method (mSaDS) proposed in Section III-A is used to scan the whole search space quickly and roughly, and narrow it to a promising region. Then the combination optimization algorithm(COA) designed in Section III-B combining different optimization methods based on the separability of sub-problems is applied to optimize the problem. The details of mSaDS and the combination strategy are described in the following sections.

Algorithm 1 The New Hybrid Optimization Algorithm (mSaDS-COA)

```
1: $(m, x_1, x_2, \ldots, x_m) = RDG2(f, lb, ub, n)$; //decompose the problem into $m$ subcomponents by RDG2
2: initialize the population;
3: calculate the fitness of each individual;
4: find the best member $bestmen$ and its function value $bestval$, respectively;
5: while $FEs \leq FExMax$ do // $FEs$ is the number of function evaluations, and $FEsMax$ is the maximum number of function evaluations.
6:     use mSaDS to roughly scan the whole search space;
7:     use COA to optimize the problem;
8: end while
```

A. MODIFIED SELF-ADAPTIVE DISCRETE SCAN METHOD

As discussion in Section I, one major difficulty of the LSGO problems is that the huge search space makes the problems too complex. The modified self-adaptive discrete scan method (mSaDS) is proposed to find the promising regions and reduce the whole search space. In mSaDS, the continuous search space was divided into discrete one by inserting a certain amount of points along each dimension uniformly. These discrete points are referred to the levels on each dimension. Only such points consisting of all these possible levels from all dimensions are searched in mSaDS. Thus the number of points searched will be decreased from infinite to finite and the search space is narrowed. And then the search will be focused on the promising region. The detail is as follows.

We suppose the current best solution is $x^* = (x_1{}^*, x_2{}^*, \ldots, x_n{}^*)^T$ and its function value is $f^* = f(x^*)$. Without loss of generality, we take *ith* dimension as an example. Assume that the current upper bound and lower bound of *ith* dimension is $ub(i)$ and $lb(i)$ respectively.

Firstly, we distribute uniformly *levelnum* points $l_1, l_2, \ldots, l_{levelnum}$ in the continuous region $[lb(i), ub(i)]$ while the variables in the other dimensions keep fixed values at x^*. Evaluate every level and find the best one $l_i{}^*$ along dimension i.

Secondly, it is very possible for the neighbor region around this level to be a more promising region in dimension i. The neighbor region can be defined as

$$U(l_i{}^*) = [l_i^* - r * d, l_i^* + r * d],$$

where

$$d = ub(i) - lb(i)$$

and r is the shrink rate between 0 and 1. We update the current upper and lower bounds of the *ith* dimension by (1) and (2) respectively to narrow the search region. If the updated $lb(i)$ or $ub(i)$ exceeds the boundary of the search region, we reset it to the value before updated.

$$ub(i) = l_i{}^* + r * d \tag{1}$$
$$lb(i) = l_i{}^* - r * d \tag{2}$$

Thirdly, if the length of $[lb(i), ub(i)]$ is less than a predetermined threshold *bgap_min*, repeat the above narrowing process along next dimension until the narrowing process on all dimensions is finished.

The performance of the proposed algorithm is sensitive to the parameter r. The larger the value of r is, the slower the space shrinks. If the value is set too small, the search space would shrink too fast in the early generations and the global optimal solution may be missed. However, if the value is set too large, the method cannot effectively reduce the search space and especially cannot quickly find a potential region in the later stage of iterations. Also, if the value of r is taken a fixed value (e.g., as in the scan method [6]), the corresponding scan method cannot fit the requirements of different situations and different stages. To overcome this shortcoming, we propose a self-adaptive scan method with adaptive shrink rate called a modified self-adaptive discrete scan method (mSaDS). The adaptive shrink rate is given by (3).

$$r = \frac{e^{1/\sqrt{iter}} - 1}{e^{1/\sqrt{iter}} + 1} \tag{3}$$

where *iter* is the current iteration and r adaptively decreases with the increase of *iter*. In the early stage of evolution, the value of r would be relatively large which means that the search space would reduce slowly to keep the diversity of solutions and as the process advances (i.e. as the *iter* increases), r would become smaller and smaller, narrowing the space fast on a promising region.

The pseudo code of the modified self-adaptive discrete scan method with adaptive shrink rate is present in Algorithm 2.

Algorithm 2 The Modified Self-Adaptive Discrete Scan Method (mSaDS)

```
Require: The population size: popsize; The lower and upper bounds lb and ub of the variables; The number of points distributed in ith dimension: levelnum; The current best solution: bestmem; The current best function value: bestval; The minimum boundary gap: bgap_min; The maximum number of iterations: iterNum.
Ensure: The updated lower and upper bounds lb and ub of the variables; The new best solution: bestmem; The new best function value: bestval;
1:  while $iter \leq iterNum$ do
2:      for $i = 1$ to $dim$ do
          (Note that dim is the total number of selected dimensions.)
3:          $levelNum = popsize$;
4:          $level\_gap = (ub(i) - lb(i))/(levelnum - 1)$;
          (Note that level_gap is the level gap of the search region.)
5:          if $(ub(i) - lb(i)) < bgap\_min$ then
6:              continue;
7:          else
8:              $levels = lb : level\_gap : ub$;
9:          end if
10:         evaluate the function values of all the levels;
11:         find the new best member and its function value denoted as bestmen_new and bestval_new respectively;
12:         if $bestval\_new < bestval$ then
13:             $bestval = bestval\_new$;
14:             $bestmem = bestmen\_new$;
15:         end if
16:         calculate shrink rate $r = \frac{e^{1/\sqrt{iter}}-1}{e^{1/\sqrt{iter}}+1}$;
17:         update $lb(i)$ and $ub(i)$ by (2) and (1) respectively;
18:         if $lb(i)$ or $ub(i)$ exceeds the boundary of the search region then
19:             reset it to the original value;
20:         end if
21:         $iter = iter + 1$;
22:     end for
23: end while
```

B. A COMBINATION OPTIMIZATION ALGORITHM

After scanning the entire search space by mSaDS, we can locate promising regions and find good solutions for LSGO problems. The solutions will be used as the initial points for further optimization. In the following optimization step, we propose a combination optimization algorithm (COA) based on the characteristics of one-dimensional search methods, CMAES and CBCC for solving LSGO problems.

CMAES is a kind of estimation of distribution algorithm which was proposed for solving difficult non-convex continuous optimization problems. It samples the population under a normal distribution $N(m_e, C)$, which can be expressed as follows.

$$N(m_e, C) \sim m_e + C^{\frac{1}{2}} N(0, \mathbf{I}) \tag{4}$$

where m_e is the mean value and C is the covariance matrix. The main idea of CMAES is iteratively updating the covariance matrix based on the offsprings to guide the search towards the areas with lower objective function values. For more details of CMAES, please refer to literatures [14] and [29]. The numerical experiment results of CMAESCC-RDG [17], CMAESCC-RDG2 [18] and CC-GDG-CMAES [14] demonstrate that the performance of CMAES outperforms many other algorithms for non-separable LSGO problems.

Based on the above analysis, we develop an combination strategy integrating CMAES and a one-dimensional search method according to whether the subcomponent is separable or not. For a LSGO problem, after it is decomposed by RDG2 into m subgroups, the variables with interaction are classified into one group, which is called a non-separable subcomponent, and the variables which have no interaction with other variables form one group. Such group is called a separable sub-component. The main idea of the combination strategy is that assigning CMAES to optimize the non-separable subcomponent and applying a one-dimensional search method to optimize the separable subcomponent dynamically. In this paper, mSaDS proposed in Section III-A is a one-dimensional search method which will be used for optimizing a separable component in the combination optimization algorithm. And also, in order to make use of the computing resources efficiently, we combine the two aforementioned optimization algorithms, i.e. CMAES and mSaDS, into CBCC framework to form a combination optimization algorithm. The pseudo code of the detailed combination optimization algorithm is shown in Algorithm 3. From Algorithm 3, we can see that the main framework of the combination optimization algorithm can be summarized in the following three steps:

Step 1. In each round of optimization, we optimize each subcomponent by any optimization algorithm under CC framework to provide initial improvements for all subcomponents (Lines 5-9). And then sort the subcomponents according to their contributions on the improvements of the whole function value.

Step 2. We choose the subcomponent with the biggest contribution and give it one more chance to be optimized (Lines 14-18). If the chosen subcomponent is a separable subcomponent, we use mSaDS as the optimizer to optimize the subcomponent, otherwise, we use CMAES as the optimizer. Then we rank all the subcomponents again according to their contributions on the improvements of the whole function value (Lines 19-21).

Step 3. If the biggest contribution of all subcomponents is smaller than a threshold. Then return to step 1 to carry out the next round of optimization until termination condition is met.

Algorithm 3 The Combination Optimization Algorithm (COA)

```
Require: The current best solution: bestmem; The current best function value: bestval;
Ensure: The updated best solution: bestmem; The updated best function value: bestval;
 1: set the current best individual $cv = bestmen$;
 2: $f^* = bestval$;
 3: $\Delta = zeros(1, m)$; // initialize the improvement vector for all subcomponents
 4: while $FEs \leq FExMax$ do // $FEs$ is the number of function evaluations, and $FEsMax$ is the maximum number of function evaluations.
 5:     for $i = 1$ to $m$ do
 6:         $f_p = f^*$;
 7:         $(cv, f^*) = optimizer(cv, x_i)$; //$x_i$ is a vector containing all the variables in $ith$ subcomponent
 8:         $\Delta_i = f_p - f^*$;
 9:     end for
10:     $(C, index) = sorting(\Delta, descending)$;
11:     $j = index(1)$;
12:     while $C_1 > epsilon$ and $FEs < FExMax$ do
13:         $f_p = f^*$
14:         if $jth$ subcomponent is a separable subcomponent then
15:             $(cv, f^*) = mSaDS(cv, x_j)$; // $x_j$ is a vector containing all the variables in $jth$ subcomponent; mSaDS is used as a one-dimensional search method;
16:         else
17:             $(cv, f^*) = CMAES(cv, x_j)$;
18:         end if
19:         $\Delta_j = f_p - f^*$;
20:         $(C, index) = sorting(\Delta, descending)$;
21:         $j = index(1)$;
22:     end while
23: end while
24: if $f^* < bestval$ then
25:     $bestval = f^*$;
26:     $bestmem = cv$;
27: end if
28: return($bestmen, bestval$)
```

IV. EXPERIMENTAL DESIGN

In order to evaluate the performance of the proposed mSaDS-COA, we conduct numerical experiments on the CEC'2013 large scale global optimization benchmark suite. The suite consists of 15 benchmark functions with 1000 dimensions which can be divided into three categories by their characteristics. $f1 - f3$ are fully separable functions. $f4 - f11$ are partially separable functions, in which

$f4-f7$ has a separable subcomponent with 700 variables, whereas $f8-f11$ have no separable subcomponent. $f12-f15$ are non-separable functions. The details of the benchmark can be found in literature [31]. All the numerical experiments are implemented in Matlab R2014b on a computer with Intel(R) Core(TM) i7-3770, CPU @3.40GHz and 8.00G RAM. Three state-of-the-art algorithms are chosen as the comparison algorithms. The maximum number of function evaluations FEsMax is set to 3.0×10^6 in all algorithms. And all algorithms are executed 25 independent runs for each test problem.

A. PARAMETER SETTING

The parameters of the proposed algorithm are as follows. For mSaDS, the minimum value of boundary gap *bgap_min* is set to $1.0e-3$. When the difference between the updated upper and lower bounds of *ith* dimension is smaller than this value, the algorithm will go to the next dimension. The population size is 30. In the optimization phase, the improvement threshold *epsilon* is set to *bestval* $\times$ $(1.0e-7)$, where *bestval* is the current best function value. For CMAES, all the parameters are set in the standard way as in CMAESCC-RDG and CMAESCC-RDG2.

B. RESULTS AND ANALYSIS

To demonstrate the performance of the proposed algorithm, we compare the results of mSaDS-COA with those of TPHA [6], CMAESCC-RDG2 [18], and MOS [34]. TPHA is a two phase hybrid algorithm and our proposed mSaDS is modified based on its first phase method. CMAESCC-RDG2 is the state-of-the-art algorithm developed in literature [18], whose decomposition method (i.e. RDG2) and optimization method (i.e. CMAES) are both used in our proposed algorithm. MOS performs very well on CEC'2013 benchmark functions and it is the champion algorithm in Algorithm Competition of CEC'2013 and CEC'2015. The comparison results are shown in Table 1, where 'Median' denotes the median of the function values of 25 independent runs, 'Mean' represents the mean of the function values of 25 independent runs and 'Std' is the standard deviation. The best results are highlighted using T-test with significance level $\rho = 0.05$. And the results of T-test of mSaDS-COA and the other compared algorithms are shown in Table 2. In Table 2, 'A' represents acceptance, which means the results on a function between the given algorithm and mSaDS-COA have no statistical difference. And 'R' denotes rejection, which indicates the results between the given algorithm and mSaDS-COA on a function are different significantly.

From Table 1, we can find that mSaDS-COA performs better than TPHA with 10 wins, 2 loses and 3 ties. mSaDS-COA outperforms TPHA on all of the partially separable problems $f4-f11$. Especially on functions $f4, f7, f8$ and $f11$, the mean of function values is improved by 2, 17, 9, 9 orders of magnitudes respectively, which strongly demonstrates that the modified self-adaptive discrete scan method and the combination strategy are effective. The modified self-adaptive discrete scan method used in mSaDS-COA can keep the diversity of the solutions and find the more potential search regions than the scan method proposed in TPHA. At the same time, the advantages of COA can be fully utilized in the optimization phase, since the separable subcomponents and non-separable subcomponents may exits simultaneously in partially separable problems. Furthermore, contribution-based combination optimization algorithm can assign an appropriate optimization algorithm (mSaDS or CMAES) to a specific subcomponent. In this way, the proposed algorithm can make full use of computing resources to improve the performance of the algorithm, which guarantees the effectiveness of mSaDS-COA.

To better understand the performance of mSaDS-COA, we give the comparison of the convergence curves of mSaDS-COA and TPHA on $f7$ and $f8$ in Figs. 1 and 2 for examples. In the figures, the horizontal axis represents the number of function evaluations in evolutionary process. The vertical axis denotes the mean of the best function values found. In the scan step (i.e. the step in which the number of function evolutions is less than about 3×10^5), we can see that the curves of mSaDS-COA converge slower than TPHA in the beginning, which can keep more potential good solutions and good search regions. As the process of the scanning advances, the curves decrease faster than TPHA which guarantees the search focus on the promising regions quickly in the later stage of the scan step. These illustrate that the discrete scan method with a adaptive shrink rate proposed in this paper is more effective than that with a fixed rate proposed in TPHA. Moreover, the curves of mSaDS-COA converge to relatively smaller values than TPHA on these two functions in the optimization phase, which demonstrates that the performance of COA proposed in Section III-B is significantly better than the optimization algorithm used in TPHA.

Comparing with CMAESCC-RDG2, mSaDS-COA gets 10 wins, 1 loses and 4 ties. On fully separable problems, mSaDS-COA finds the better optimal solutions on $f1$ and $f2$, which shows that mSaDS as a one-dimensional search algorithm is more effective than CMAES used in CMAESCC-RDG2 for fully separable problems. For partially separable problems $f4-f11$, mSaDS-COA achieves the best results on 6 out of 8 benchmark functions. Especially on functions $f4, f8$ and $f11$, the mean values are improved by 4, 6, 12 orders of magnitudes, respectively. In principle, the decomposition method and optimization method used in CMAESCC-RDG2 are also applied in mSaDS-COA, however, the scan method and the combination strategy are not used in CMAESCC-RDG2. That means the difference between the two algorithms is the usage of our proposed mSaDS and combination strategy, so the excellent optimization results obtained by mSaDS-COA demonstrate that our proposed algorithm is effective for partially separable problems. For non-separable problems, from Table 1 and Table 2,

TABLE 1. Comparison between mSaDS-COA and other state-of-the-art algorithms on CEC'2013 benchmark functions.

Funtions		mSaDS-COA	MOS	CMAESCC-RDG2	TPHA
f1	Mean	**0.00e+00**	**0.00e+00**	2.78e+05	**0.00e+00**
	Median	**0.00e+00**	**0.00e+00**	2.76e+05	**0.00e+00**
	Std	0.00e+00	0.00e+00	3.16e+04	0.00e+00
f2	Mean	**0.00e+00**	8.32e+02	4.70e+03	**0.00e+00**
	Median	**0.00e+00**	8.36e+02	4.70e+03	**0.00e+00**
	Std	0.00e+00	4.48e+01	2.05e+02	0.00e+00
f3	Mean	2.00e+01	**9.17e-13**	2.04e+01	2.00e+01
	Median	2.00e+01	**9.10e-13**	2.04e+01	2.00e+01
	Std	4.27e-10	5.23e-14	4.34e-02	1.09e-14
f4	Mean	**1.69e+02**	1.74e+08	5.83e+06	1.12e+04
	Median	**1.69e+02**	1.56e+08	5.81e+06	1.11e+04
	Std	9.80e-03	8.03e+07	6.32e+05	2.46e+03
f5	Mean	**1.63e+06**	6.94e+06	2.23e+06	4.48e+06
	Median	**1.65e+06**	6.79e+06	2.24e+06	4.29e+06
	Std	1.61e+05	9.03e+05	3.22e+05	8.67e+05
f6	Mean	9.96e+05	**1.48e+05**	9.95e+05	1.06e+06
	Median	9.96e+05	**1.39e+05**	9.95e+05	1.06e+06
	Std	3.60e-01	6.56e+04	6.54e+01	3.71e+03
f7	Mean	**8.06e-22**	1.62e+04	4.04e-16	5.43e-05
	Median	**7.23e-22**	1.62e+04	3.12e-19	8.75e-05
	Std	2.74e-22	9.29e+03	1.48e-15	4.36e-05
f8	Mean	**2.23e+00**	8.00e+12	8.70e+06	7.09e+08
	Median	**2.12e+00**	8.08e+12	8.15e+06	7.43e+08
	Std	7.71e-01	3.14e+12	3.61e+06	3.79e+08
f9	Mean	**1.13e+08**	3.83e+08	1.67e+08	3.18e+08
	Median	**1.53e+08**	3.87e+08	1.74e+08	2.81e+08
	Std	9.11e+06	6.42e+07	2.56e+07	7.78e+07
f10	Mean	9.05e+07	**9.02e+05**	9.10e+07	9.39e+07
	Median	9.05e+07	**1.18e+06**	9.05e+07	9.40e+07
	Std	1.58e+02	5.17e+05	1.30e+06	3.97e+05
f11	Mean	**5.55e-09**	5.22e+07	8.68e+03	3.08e+00
	Median	**1.94e-09**	4.48e+07	2.81e+03	2.20e+00
	Std	6.47e-09	2.10e+07	1.24e+04	1.94e+00
f12	Mean	9.59e+02	**2.47e+02**	9.81e+02	2.94e+02
	Median	9.99e+02	**2.46e+02**	1.01e+03	3.20e+02
	Std	9.93e+01	2.59e+02	7.30e+01	9.77e+01
f13	Mean	1.06e+06	3.40e+06	**9.31e+05**	1.07e+10
	Median	9.57e+05	3.30e+06	**9.04e+05**	1.17e+10
	Std	2.07e+05	1.08e+06	1.60e+05	2.40e+09
f14	Mean	2.65e+07	**2.56e+07**	2.68e+07	1.96e+11
	Median	2.82e+07	**2.42e+07**	2.65e+07	1.97e+11
	Std	7.51e+06	8.11e+06	1.88e+06	2.06e+10
f15	Mean	2.21e+06	2.35e+06	2.26e+06	**1.99e+06**
	Median	2.18e+06	2.38e+06	2.23e+06	**2.13e+06**
	Std	2.75e+05	1.98e+05	2.45e+05	2.26e+05

we can find that there is no significant statistical difference between the optimization results of CMAESCC-RDG2 and mSaDS-COA. The reason is that on non-separable problems, all variables are divided into only one subgroup and the proposed combination optimization strategy of mSaDS-COA is identical to CMAES used in CMAESCC-RDG2. In this case,

TABLE 2. Results of T-test of mSaDS-COA and others.

Funtions	MOS	CMAESCC-RDG2	TPHA
f1	1, A	1.85e-24, R	1, A
f2	3.38e-32, R	2.17e-34, R	1, A
f3	7.95e-98, R	6.15e-25, R	1, A
f4	1.00e-10, R	6.02e-25, R	1.32e-17, R
f5	3.55e-20, R	1.52e-08, R	2.12e-14, R
f6	1.95e-28, R	3.53e-30, R	1.97e-31, R
f7	6.66e-09, R	1.85e-01, A	1.95e-06, R
f8	3.59e-12, R	1.15e-11, R	1.78e-09, R
f9	7.19e-17, R	5.57e-10, R	2.04e-12, R
f10	1.83e-55, R	6.64e-02, A	3.51e-24, R
f11	6.02e-12, R	1.84e-03, R	3.62e-08, R
f12	3.07e-12, R	3.81e-01, A	3.13e-18, R
f13	1.44e-10, R	2.12e-02, R	1.51e-17, R
f14	6.88e-01, A	8.48e-01, A	2.90e-25, R
f15	4.98e-02, A	5.04e-01, A	5.00e-03, R

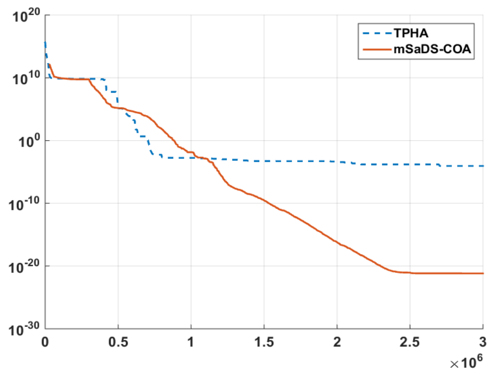


FIGURE 1. The convergence curves of mSaDS-COA and TPHA on f7 problem.

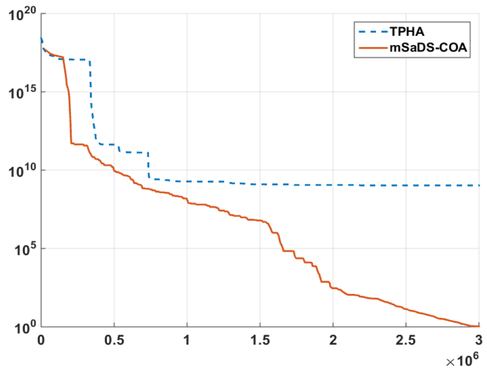


FIGURE 2. The convergence curves of mSaDS-COA and TPHA on f8 problem.

the proposed algorithm is basically the same as CMAESCC-RDG2 on non-separable problems. So the results obtained are nearly equal.

As for MOS, we can see that mSaDS-COA performs better than MOS on 9 functions: $f2, f4, f5, f7, f8, f9, f11, f13$ and $f15$, whereas it performs worse than MOS on 5 functions: $f3, f6, f10, f12$ and $f14$. On partially separable problems,

TABLE 3. The average ranking of each algorithm on 15 benchmark functions.

Algorithm	AR
mSaDS-COA	**1.7**
MOS	2.6
CMAESCC-RDG2	2.7
TPHA	2.6

mSaDS-COA outperforms MOS significantly. For example, we can find that mSaDS-COA improves the mean value of function $f4$ from $1.74e+08$ to $1.69e+02$, which improves the best objective function value by 6 orders of magnitudes. For function $f7$, the mean value $1.62e+04$ is improved to $8.06e-22$ by using our proposed algorithm. For function $f8$, mSaDS-COA reduces the mean value from $8.00e+12$ to $2.23e-00$, improving the optimization results by 13 orders of magnitudes. And also, for function $f11$, the mean value obtained by mSaDS-COA is $5.55e-09$, while the value by MOS is $5.22e+07$. However, on functions $f3, f6$ and $f10$, mSaDS-COA is significantly worse than MOS. The reason is that $f3, f6$ and $f10$ are Ackley functions and there exits many local optimal solutions. MOS integrates many efficient global search algorithms with local search algorithms, and it use a more complex combination strategy on many efficient algorithms. Thus it can often jump out of the local solutions by using a proper global search algorithm to get better results for fully non-separable problems with lots of locally optimal solutions. This means that for this kind of problems, we have to do further work to develop a more efficient and effective strategy for our proposed algorithm in the future.

In overall, we can find that mSaDS-COA achieves the best solutions on 8 test problems and MOS achieves the best solutions on 6 test problems, whereas TPHA obtains the best solutions on only 3 test problems and CMAESCC-RDG2 obtains only 1 best solutions. The number of optimal solutions obtained by the mSaDS-COA is far more than the champion algorithm MOS. Moreover, the optimal solutions obtained by mSaDS-COA are much better than those obtained by MOS. For example, on four problems, mSaDS-COA got solutions with errors less than $1.00e+00$, while MOS did so only on two problems.

In order to further compare the comprehensive performance of the compared algorithms, a performance measure called Average Ranking (AR) [17] , is defined and is used as follows.

$$AR = \frac{\sum_{i=1}^{N} rank_i}{N} \tag{5}$$

where N represents the total number of functions, and $rank_i$ denotes the rank of the algorithm for ith function. The smaller the AR is, the better performance the algorithm has.

Table 3 gives the average ranking of each compared algorithm on CEC'2013 benchmark functions. The best results are shown in bold.

As shown in Table 3, the average ranking of mSaDS-COA is 1.7, the smallest one among the four compared algorithms. This result means that the comprehensive performance of mSaDS-COA is better than MOS, CMAESCC-RDG2, and TPHA on solving large scale optimization problems, which is consistent with the aforementioned analysis.

V. CONCLUSION

In this paper, we propose a new hybrid algorithm for LSGO problems based on contribution-based CC framework. Firstly, we use a variant of recursive differential grouping to judge the separability and decompose the problem. And then a modified self-adaptive discrete scan method is applied to scan the whole search space roughly, where the shrink rate can be adaptively adjusted with the different evolution stage, which would make the search focus on the promising regions. Based on these, a combination optimization algorithm is developed to tackle the separable and non-separable problems by using different strategies. It combines the characteristics of mSaDS, CMAES and CBCC, and assigns different optimization algorithms to different problems and different groups dynamically. In this way, the proposed algorithm can effectively handle LSGO problems. However, the proposed algorithm performs not very well on non-separable problems, since all variables of a non-separable problem is assigned to one subgroups by RDG2, i.e., the problem is not decomposed at all and the proposed algorithm should face a large scale optimization problem, where the proposed algorithm only combines two strategies mSaDS and CMAES, and its search ability is not so strong as that of MOS which combines many efficient strategies. This means that for this kind of problems, we have to design some effective decomposition methods in the future.

REFERENCES

[1] R. Tang, L. Xin, and J. Lai, "A novel optimal energy-management strategy for a maritime hybrid energy system based on large-scale global optimization," *Appl. Energy*, vol. 228, pp. 254–264, Oct. 2018.

[2] N. R. Sabar, J. Abawajy, and J. L. Yearwood, "Heterogeneous cooperative co-evolution memetic differential evolution algorithm for big data optimization problems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 315–327, Aug. 2017.

[3] R. Paz, E. Pei, M. Monzón, F. Ortega, and L. Suárez, "Lightweight parametric design optimization for 4D printed parts," *Integr. Comput.-Aided Eng.*, vol. 24, no. 3, pp. 225–240, 2017.

[4] R. Mencia, M. R. Sierra, C. Mencia, and R. Varela, "Genetic algorithms for the scheduling problem with arbitrary precedence relations and skilled operators," *Integr. Comput.-Aided Eng.*, vol. 23, no. 3, pp. 269–285, 2016.

[5] M. N. Omidvar, X. Li, and K. Tang, "Designing benchmark problems for large-scale continuous optimization," *Inf. Sci.*, vol. 316, pp. 419–436, Sep. 2015.

[6] H. Liu, Y. Wang, L. Liu, and X. Li, "A two phase hybrid algorithm with a new decomposition method for large scale optimization," *Intergr. Comput.-Aided Eng.*, vol. 25, no. 4, pp. 349–367, 2018.

[7] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Trans. Math. Softw.*, vol. 23, no. 4, pp. 550–560, 1997.

[8] W. W. Hager and H. Zhang, "Algorithm 851: CG_DESCENT, a conjugate gradient method with guaranteed descent," *ACM Trans. Math. Softw.*, vol. 32, no. 1, pp. 113–137, 2006.

[9] L. M. Rios and N. V. Sahinidis, "Derivative-free optimization: A review of algorithms and comparison of software implementations," *J. Global Optim.*, vol. 56, no. 3, pp. 1247–1293, 2013.

[10] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *Proc. IEEE Congr. Evol. Comput.*, Seoul, South Korea, May 2001, pp. 1101–1108.

[11] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. 3rd Conf. Parallel Problem Solving Nature*, vol. 2, 1994, pp. 249–257.

[12] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 378–393, Jun. 2014.

[13] Y. Sun, M. Kirley, and S. K. Halgamuge, "Extended differential grouping for large scale global optimization with direct and indirect variable interactions," in *Proc. Annu. Conf. Genet. Evol. Comput.* Berlin, Germany: Springer, 2015, pp. 313–320.

[14] Y. Mei, X. Yao, X. Li, and M. N. Omidvar, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Trans. Math. Softw.*, vol. 42, no. 2, p. 13, 2016.

[15] X.-M. Hu, F.-L. He, W.-N. Chen, and J. Zhang, "Cooperation coevolution with fast interdependency identification for large scale optimization," *Inf. Sci.*, vol. 381, pp. 142–160, Mar. 2017.

[16] 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 Trans. Evol. Comput.*, vol. 21, no. 6, pp. 929–942, Dec. 2017.

[17] S. Yuan, M. Kirley, and S. K. Halgamuge, "A recursive decomposition method for large scale continuous optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 5, pp. 647–661, Oct. 2018.

[18] Y. Sun, M. N. Omidvar, M. Kirley, and X. Li, "Adaptive threshold parameter estimation with recursive differential grouping for problem decomposition," in *Proc. Annu. Conf. Genet. Evol. Comput.*, Kyoto, Japan, 2018, pp. 15–19.

[19] Y. Wang, H. Liu, F. Wei, T. Zong, and X. Li, "Cooperative coevolution with formula-based variable grouping for large-scale global optimization," *Evol. Comput.*, vol. 26, no. 4, pp. 569–596, 2018.

[20] M. N. Omidvar, B. Kazimipour, X. Li, and Y. Xin, "CBCC3—A contribution-based cooperative co-evolutionary algorithm with improved exploration/exploitation balance," in *Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, Jul. 2016, pp. 3541–3548.

[21] A. Anne, H. Nikolaus, M. Nikolas, R. Raymond, and S. Marc, "Bio-inspired continuous optimization: The coming of age," in *Proc. IEEE Congr. Evol. Comput.*, Singapore, Sep. 2007, pp. 3117–3124.

[22] X. Li, "Decomposition and cooperative coevolution techniques for large scale global optimization," in *Proc. Companion Publication Conf. Genet. Evol. Comput.*, Vancouver, BC, Canada, 2014, pp. 819–838.

[23] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Metaheuristics in large-scale global continues optimization: A survey," *Inf. Sci.*, vol. 295, pp. 407–428, Feb. 2015.

[24] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inf. Sci.*, vol. 178, no. 15, pp. 2985–2999, 2008.

[25] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 210–224, Apr. 2012.

[26] Y. Ren and Y. Wu, "An efficient algorithm for high-dimensional function optimization," *Soft Comput.*, vol. 17, no. 6, pp. 995–1004, 2013.

[27] M. N. Omidvar, X. Li, and Y. Xin, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, Jul. 2010, pp. 1–8.

[28] W. Chen, T. Weise, Z. Yang, and T. Ke, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *Proc. Int. Conf. Parallel Problem Solving Nature*. Berlin, Germany: Springer, 2010, pp. 300–309.

[29] N. Hansen, "The CMA evolution strategy: A tutorial," pp. 1–39, 2016, *arXiv:1604.00772*. [Online]. Available: https://arxiv.org/abs/1604.00772

[30] M. N. Omidvar, X. Li, and Y. Xin, "Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms," in *Proc. Annu. Conf. Genet. Evol. Comput.*, Dublin, Ireland, 2011, pp. 1115–1122.

[31] X. Li, K. Tang, M. N. Omidvar, Z. Yang, and K. Qin, "Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization," School Comput. Sci. Inf. Technol., RMIT Univ., Melbourne, VIC, Australia, Tech. Rep., 2013, pp. 1–23.

[32] D. Molina, M. Lozano, and F. Herrera, "MA-SW-Chains: Memetic algorithm based on local search chains for large scale continuous global optimization," in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, Jul. 2010, pp. 3153–3160.

[33] L.-Y. Tseng and C. Chen, "Multiple trajectory search for large scale global optimization," in *Proc. IEEE Congr. Evol. Comput.*, Hong Kong, Jun. 2008, pp. 3052–3059.

[34] A. Latorre, S. Muelas, and J.-M. Peña, "Evaluating the multiple offspring sampling framework on complex continuous optimization functions," *Memetic Comput.*, vol. 5, no. 4, pp. 295–309, 2013.

[35] A. LaTorre, S. Muelas, and J.-M. Peña, "Large scale global optimization: Experimental results with MOS-based hybrid algorithms," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, Mexico, Jun. 2013, pp. 2742–2749.

[36] H. Wang, Z. Wu, and S. Rahnamayan, "Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems," *Soft Comput.*, vol. 15, no. 11, pp. 2127–2140, Nov. 2011.

[37] Y. Sun, X. Wang, Y. Chen, and Z. Liu, "A modified whale optimization algorithm for large-scale global optimization problems," *Expert Syst. Appl.*, vol. 114, pp. 563–577, Dec. 2018.

[38] D. Molina, A. Latorre, and H. Francisco, "SHADE with iterative local search for large-scale global optimization," in *Proc. IEEE Congr. Evol. Comput.*, Rio de Janeiro, Brazil, Jul. 2018, pp. 1252–1259.

[39] H. Lin, Y. Wang, Y. Gao, and X. Wang, "A filled function method for global optimization with inequality constraints," *Comput. Appl. Math.*, vol. 37, no. 2, pp. 1524–1536, 2016.

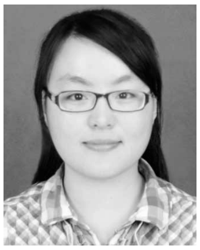
[40] W. Fei, Y. Wang, and H. Lin, "A new filled function method with two parameters for global optimization," *J. Optim. Theory Appl.*, vol. 163, no. 2, pp. 510–527, 2014.

[41] H. Lin, Y. Wang, S. Guan, and X. Liu, "A new filled function method for unconstrained global optimization," *Int. J. Comput. Math.*, vol. 94, no. 12, pp. 2283–2296, 2017.

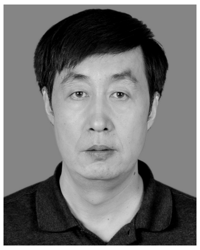
[42] H. Lin, Y. Gao, X. Wang, and S. Su, "A filled function which has the same local minimizer of the objective function," *Optim. Lett.*, vol. 13, no. 4, pp. 761–776, 2018. doi: 10.1007/s11590-018-1275-5.

[43] F. Caraffini, F. Neri, G. Iacca, and A. Mol, "Parallel memetic structures," *Inf. Sci.*, vol. 227, no. 4, pp. 60–82, 2013.

[44] E. Alba and J. M. Troya, "A survey of parallel distributed genetic algorithms," *Complexity*, vol. 4, no. 4, pp. 31–52, 1999.

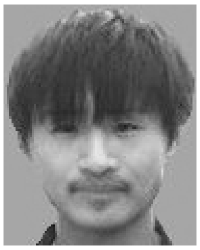


XIANGJUAN WU received the B.S. and M.S. degrees from the Nanjing University of Information Science and Technology, Nanjing, China, in 2009 and 2012, respectively. She is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Xidian University, Xi'an, China. Her research interests include large scale optimization and evolutionary computation.



YUPING WANG received the Ph.D. degree from the Department of Mathematics, Xi'an Jiaotong Unversity, Xi'an, China, in 1993. He is currently a Full Professor with the School of Computer Science and Technology, Xidian University, Xi'an. His research interests include optimization algorithms, optimization modeling for big data, and network scheduling.

JUNHUA LIU received the B.S. degree in mathematics and applied mathematics from Yuncheng University, Yuncheng, China, in 2012, and the M.S. degree in computational mathematics from North Minzu University, Yinchuan, China, in 2015. She is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Xidian University, Xi'an, China. Her research interests include multi/many-objective optimization and evolutionary computation.



NINGLEI FAN received the B.S. degree from Shanxi University, Taiyuan, China, in 2017. He is currently pursuing the M.S. degree with the School of Computer Science and Technology, Xidian University, Xi'an, China. His research interests include evolutionary computing and large scale optimization.