A Competitive Divide-and-Conquer Algorithm for Unconstrained Large-Scale Black-Box Optimization

Mei, Yi; Omidvar, Mohammad Nabi; Li, Xiaodong; Yao, Xin

DOI:
10.1145/2791291

Citation for published version (Harvard):
Mei, Y, Omidvar, MN, Li, X & Yao, X 2016, 'A Competitive Divide-and-Conquer Algorithm for Unconstrained Large-Scale Black-Box Optimization', ACM Transactions on Mathematical Software, vol. 42, no. 2, 13. https://doi.org/10.1145/2791291

A Competitive Divide-and-Conquer Algorithm for Unconstrained Large-Scale Black-Box Optimization

YI MEI, Victoria University of Wellington and RMIT University
MOHAMMAD NABI OMIDVAR, RMIT University
XIAODONG LI, RMIT University
XIN YAO, University of Birmingham

This paper proposes a competitive divide-and-conquer algorithm for solving large-scale black-box optimization problems, where there are thousands of decision variables, and the algebraic models of the problems are unavailable. We focus on problems that are partially additively separable, since this type of problem can be further decomposed into a number of smaller independent sub-problems. The proposed algorithm addresses two important issues in solving large-scale black-box optimization: (1) the identification of the independent sub-problems without explicitly knowing the formula of the objective function and (2) the optimization of the identified black-box sub-problems. First, a Global Differential Grouping (GDG) method is proposed to identify the independent sub-problems. Then, a variant of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is adopted to solve the sub-problems resulting from its rotation invariance property. GDG and CMA-ES work together under the cooperative co-evolution framework. The resultant algorithm named CC-GDG-CMAES is then evaluated on the CEC'2010 large-scale global optimization (LSGO) benchmark functions, which have a thousand decision variables and black-box objective functions. The experimental results show that on most test functions evaluated in this study, GDG manages to obtain an ideal partition of the index set of the decision variables, and CC-GDG-CMAES outperforms the state-of-the-art results. Moreover, the competitive performance of the well-known CMA-ES is extended from low-dimensional to high-dimensional black-box problems.

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Large-scale black-box optimization, decomposition, cooperative co-evolution, differential grouping, covariance matrix adaptation evolutionary strategy (CMA-ES)

ACM Reference Format:
Yi Mei, Xiaodong Li, Mohammad Nabi Omidvar and Xin Yao, 2014. A Competitive Divide-and-Conquer Algorithm for Unconstrained Large-Scale Black-Box Optimization *ACM Trans. Math. Softw.* V, N, Article A (January YYYY), 25 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

In science and engineering, a *black box* is a function or a system that transfers its input to its output with an unknown or imprecise internal transferring mechanism. Black boxes are often encountered in research and industrial fields. For example, in cogni-

This work was supported by an ARC Discovery grant (No. DP120102205), an NSFC grant (No. 61329302), and an EPSRC grant (No. EP/K001523/1). Xin Yao was supported by a Royal Society Wolfson Research Merit Award.
Author's addresses: Y. Mei, School of Engineering and Computer Science, Victoria University of Wellington, Kelburn 6012, New Zealand; M.N. Omidvar and X. Li, School of Computer Science and Information Technology, RMIT University, Melbourne, VIC 3000, Australia; X. Yao, CERCIA, School of Computer Science, University of Birmingham, B15 2TT Birmingham, UK.

DOI:http://dx.doi.org/10.1145/0000000.0000000

tive science, the human mind can be seen as a black box ([Friedenberg and Silverman 2006]). In physics, a physical system whose internal structure is unknown or too complicated to be fully understood may be referred to as a black box. In the computing area, a program may be considered to be a black box if it is a closed source program and the users have no access to its inner workings.

Black-box optimization aims to do optimization based on black-box systems. To be specific, a black-box optimization problem consists of a set of objective and constraint functions, at least one of which is a black box. Due to the wide distribution of black-box scenarios in practice, it is important to investigate how to effectively solve black-box optimization problems.

When dealing with black-box optimization problems, traditional mathematical programming methods such as the simplex algorithm [Dantzig and Thapa 1997] [Weglarz et al. 1977] [Azulay and Pique 2001], and gradient-based methods such as the Quasi-Newton method [Zhu et al. 1997] and the conjugate gradient method [Hager and Zhang 2006] are no longer applicable, since the internal information about the problem, such as the coefficients and derivative, is unavailable, or only partially available. In such a case, derivative-free algorithms are promising methods for solving black-box optimization problems as they take only the input and output of a function into account. There have been plenty of derivative-free algorithms developed in the literature, including various local search algorithms [Hooke and Jeeves 1961] [Nelder and Mead 1965] [Custódio and Vicente 2007] and global search algorithms [Holland 1975] [Eberhart and Kennedy 1995] [Yao et al. 1999]. A comprehensive survey of derivative-free algorithms can be found in [Rios and Sahinidis 2012].

Derivative-free algorithms are mainly based on sophisticated sampling mechanisms to search effectively within the solution space. Hence, their performances largely depend on the problem size, i.e., the number of decision variables of the problem. When the problem size increases, the size of the solution space increases exponentially. For example, in a binary optimization problem, when the problem dimension increases from 10 to 20, the size of the solution space increases from $2^{10} = 1024$ to $2^{20} = 1048576$. The rapid growth in the size of the search space makes it exceedingly difficult to find the global optimum by sampling/searching the entire space. To address this scalability issue, the divide-and-conquer strategy is a commonly-used approach for solving large-scale black-box optimization problems. One direction is the additive model approximation ([Andrews and Whang 1990] [Friedman and Silverman 1989] [Stone 1985] [Li et al. 2001]), which approximates the black-box function to a mathematical model that is represented as the sum of several component functions of the subsets of the variables, and then solves the optimization problems of each component function separately with mathematical programming methods. The other direction is to directly decompose the original large-scale problem into a number of smaller-sized sub-problems, and solve them individually. The coordinate descent methods [Bezdek et al. 1987] and block coordinate descent methods [Tseng 2001] [Richtárik and Takáč 2012] [Blondel et al. 2013] are representative divide-and-conquer methods that search for the local optimum by doing line search along one coordinate direction or block search in the space of a coordinate subset at the current point in each iteration. When optimizing the current coordinate or block, all the other coordinates or blocks are fixed to the best-so-far values (called the *collaborative values*). Cooperative co-evolution [Potter and De Jong 1994] [Liu et al. 2001] [Shi et al. 2005] [Van den Bergh and Engelbrecht 2004] [Li and Yao 2012] [Yang et al. 2008] [Chen et al. 2010] [Omidvar et al. 2010] [Omidvar et al. 2010] [Omidvar et al. 2014] is a generalized framework of the coordinate descent and block coordinate descent methods. Instead of being fixed to the best-so-far value, a set of candidate values is retained for each collaborating coordinate in cooperative co-evolution to reduce the greediness of the search. With divide-and-conquer strategies, the entire index set of the decision variables is divided into smaller subsets, and the corresponding decision variables, called the *subcomponents*, are optimized separately. This way, the solution space is much reduced and can be explored more effectively.

In order to ensure the efficacy of the divide-and-conquer strategies, we assume that the problem is unconstrained and has a partially additively separable objective function, i.e., $f(\mathbf{x}) = \sum_{i=1}^{g} f_i(\mathbf{x}_{\mathcal{I}_i})$, where $\mathbf{x} = (x_1, \dots, x_n)$ is the decision vector, $(\mathcal{I}_1, \dots, \mathcal{I}_g)$ is a partition of the index set $\{1, \dots, n\}$ and $\mathbf{x}_{\mathcal{I}_i}$ is the subcomponent corresponding to the index subset $\mathcal{I}_i$. For example, if $\mathcal{I}_i = \{1, 3, 5\}$, then $\mathbf{x}_{\mathcal{I}_i} = (x_1, x_3, x_5)$. Obviously, under such an assumption, the additive model approximation is the original problem, and is equivalent to the direct decomposition.

Although the objective function is partially additively separable, the independent sub-problems and the corresponding subcomponents are not straightforward for the black-box optimization problems, where the formula of the objective function cannot be known exactly. Therefore, when solving unconstrained large-scale black-box optimization problems with partially additively separable objective functions, two major issues should be addressed: (1) the identification of the independent sub-problems and the corresponding subcomponents without knowing the exact formula of the objective function and (2) the selection of competent optimizers for the nonseparable sub-problems. The former ensures that solving the sub-problems separately is equivalent to solving the overall problem. The latter (selecting a competent optimizer) enhances the capability of the algorithm to obtain optimal solution for each sub-problem.

In this paper, the above two issues are addressed by a Global Differential Grouping (GDG) and a variant of CMA-ES [Hansen 2011], respectively. Differential grouping [Omidvar et al. 2014] has been demonstrated to have a high accuracy of grouping interacting decision variables together without knowing any information about the formula of the objective function. GDG adopts the mechanism of differential grouping, and further improves its accuracy by maintaining the global information (i.e., the interaction matrix between variables) and taking the computational error into account. CMA-ES is one of the state-of-the-art derivative-free algorithms for nonseparable black-box optimization problems [Rios and Sahinidis 2012]. It conducts a rotation-invariant search which is independent of the coordinate system. Concretely, it estimates a covariance matrix that indicates the relationship between variables for improving the objective value. For quadratic objective functions, the covariance matrix gradually converges to the inverse Hessian matrix [Loshchilov et al. 2011]. In this case, CMA-ES transforms the coordinate system into the one defined by the inverse Hessian matrix, and thus transforms the nonseparable problem into a separable one. The competitiveness of CMA-ES has been verified on low-dimensional (e.g., no more than 40) GECCO'2009 black-box optimization benchmark functions [Hansen et al. 2010]. The subcomponents are expected to have low dimensions after the decomposition. Thus, one can expect that CMA-ES will perform well in optimizing the subcomponents. On the other hand, the covariance matrix estimation stage becomes computationally expensive when dealing with large-scale problems. From previous studies [Rios and Sahinidis 2012], the complexity of CMA-ES is dominated by the covariance matrix estimation, which has a complexity of $O(n^3)$, where n is the problem size. When decomposing the n-dimensional decision vector into n/s s-dimensional subcomponents, the complexity is reduced to $n/s \cdot O(s^3) = O(ns^2)$, where $s \ll n$. For example, when $n = 1000$ and $s = 20$, the complexity is reduced to $1000 \cdot 20^2/1000^3 = 1/2500$ of the original one.

The rest of the paper is organized as follows: The large-scale black-box optimization with partially additively separable objective function and divide-and-conquer strategies are introduced in Section 2. Then, the proposed algorithm, named Cooperative Co-evolution with GDG and CMA-ES (CC-GDG-CMAES), is described in Section 3.

The experimental studies are presented in Section 4. Finally, conclusions and future work are described in Section 5.

2. BACKGROUND

2.1. Large-Scale Black-Box Optimization

Without loss of generality, a black-box optimization problem can be stated as follows:

$$\min \; f(\mathbf{x}), \tag{1}$$

$$s.t.: \; \mathbf{h}(\mathbf{x}) = 0, \tag{2}$$

where $\mathbf{x} = (x_1, \ldots, x_n)$ is an n-dimensional *decision vector*, and each x_i $(i = 1, \ldots, n)$ is called a *decision variable*. $f(\mathbf{x})$ is the *objective function* to be minimized, and $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_m(\mathbf{x}))$ is a set of equality *constraints*. It is clear that all the inequality constraints can be transformed to equality ones by adding slack variables. In the above problem, at least one of the functions among the objective function $f(\mathbf{x})$ and the set of constraints $\mathbf{h}(\mathbf{x})$ is a black box.

When tackling the above constrained black-box optimization problem Eqs. (1)–(2), the traditional method of Lagrangian multipliers is no longer applicable, since it requires the calculation of the gradient of the Lagrangian. Concretely, the Lagrangian of the above problem is

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i h_i(\mathbf{x}), \tag{3}$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ is the $n \times 1$ vector of Lagrangian multipliers. Since at least one function in the set of $\{f(\mathbf{x}), h_1(\mathbf{x}), \ldots, h_m(\mathbf{x})\}$ is a black box, it is obvious that the Lagrangian $L(\mathbf{x}, \boldsymbol{\lambda})$ is also a black box, and thus the gradient $\nabla_{\mathbf{x},\boldsymbol{\lambda}} L$ is unavailable.

In this situation, the augmented Lagrangian method is an alternative to solve a constrained black-box optimization problem by replacing it with a series of unconstrained black-box optimization problems. Specifically, Eqs. (1)–(2) are replaced by the following unconstrained problems:

$$\min \Phi_k(\mathbf{x}) = f(\mathbf{x}) + \frac{\mu_k}{2} \sum_{i=1}^{m} h_i(\mathbf{x})^2 - \sum_{i=1}^{m} \lambda_i h_i(\mathbf{x}). \tag{4}$$

At each step, the optimal solution $\mathbf{x}_k^* = \arg\min \Phi_k(\mathbf{x})$ is obtained by some derivative-free method. Then, μ_k is increased and $\lambda_i \leftarrow \lambda_i - \mu_k h_i(\mathbf{x}_k^*)$ is updated. When μ_k becomes sufficiently large, the optimal solution $\mathbf{x}_k^*$ to the unconstrained problem is also the global optimum of the original constrained problem.

It can be seen that the most important issue in solving the constrained black-box optimization problems with the augmented Lagrangian method is to solve the corresponding unconstrained black-box optimization problems $\min \Phi_k(\mathbf{x})$ to obtain $\mathbf{x}_k^*$. In other words, once the unconstrained problems are solved, the original constrained problem can be solved directly. Thus, we focus on solving the unconstrained black-box optimization problems, which are stated as follows:

$$\min f(\mathbf{x}), \tag{5}$$

where $f(\mathbf{x})$ is a black box.

Then, a partially additively separable function is defined as follows:

Definition 2.1 (*Partially additively separable function*). Given an n-dimensional function $f(\mathbf{x})$, if there exists a partition $(\mathcal{I}_1, \ldots, \mathcal{I}_g)$ of the index set $\{1, \ldots, n\}$ along with a set of functions $f_1(\mathbf{x}_{\mathcal{I}_1}), \ldots, f_g(\mathbf{x}_{\mathcal{I}_g})$, so that $f(\mathbf{x}) = \sum_{i=1}^{g} f_i(\mathbf{x}_{\mathcal{I}_i})$, then $f(\mathbf{x})$ is called to be *partially additively separable*. In particular, if $|\mathcal{I}_i| = 1$, $\forall i = 1, \ldots, n$, then $f(\mathbf{x})$ is *completely additively separable*.

In Definition 2.1, the partition $(\mathcal{I}_1, \ldots, \mathcal{I}_g)$ is called an *ideal partition* of the index set. More formally, the definition of an ideal partition is given as follows:

Definition 2.2 (*Ideal partition*). Given an n-dimensional function $f(\mathbf{x})$ and a partition $(\mathcal{I}_1, \ldots, \mathcal{I}_g)$ of the index set $\{1, \ldots, n\}$, if there exists a set of functions $f_1(\mathbf{x}_{\mathcal{I}_1}), \ldots, f_g(\mathbf{x}_{\mathcal{I}_g})$, so that $f(\mathbf{x}) = \sum_{i=1}^{g} f_i(\mathbf{x}_{\mathcal{I}_i})$, then $(\mathcal{I}_1, \ldots, \mathcal{I}_g)$ is called an *ideal partition* of the index set. The corresponding functions $f_i(\mathbf{x}_{\mathcal{I}_i})$ $(i = 1, \ldots, g)$ are called a set of *ideal component functions*.

In our work, we focus on the unconstrained large-scale black-box optimization problems with partially additively separable objective functions, in which the number of decision variables n is large (i.e., $n \geq 1000$).

2.2. Divide-and-Conquer Strategies

A divide-and-conquer strategy in optimization typically includes the following two tasks: (1) the decomposition of the problem by dividing the set of decision variables into a number of subcomponents; and (2) the coordination of the optimization of the subcomponents. There are two types of problem decomposition depending on the different relationships between the subcomponents [Shan and Wang 2010]. One is called the *ideal decomposition*, which aims to obtain an ideal partition of the index set and the corresponding non-overlapping subcomponents. The other one is called the *coordination-based decomposition*, which leads to the subcomponents sharing common variables due to the interaction between them. In this paper, we focus on the ideal decomposition, since ideal partitions of the index set must exist for the partially additively separable objective functions investigated in our studies.

As indicated in Definition 2.1, the identification of the ideal decomposition consists of two tasks. One is obtaining an ideal partition $(\mathcal{I}_1, \ldots, \mathcal{I}_g)$ of the index set, and the other is the identification of the corresponding ideal component functions $f_1(\mathbf{x}_{\mathcal{I}_1}), \ldots, f_g(\mathbf{x}_{\mathcal{I}_g})$. Since $f(\mathbf{x})$ is a black box, it is impossible to obtain the exact formula of the component functions. Cooperative co-evolution is a suitable framework to obtain black-box component functions.

A typical cooperative co-evolution framework divides the entire optimization process into several *cycles*. In each cycle, the subcomponents are evolved in a round-robin fashion. Given the g subcomponents $\mathbf{x}_{\mathcal{I}_1}, \ldots, \mathbf{x}_{\mathcal{I}_g}$, a sub-population $\{\mathbf{s}_{i1}, \ldots, \mathbf{s}_{iN}\}$ of candidate solutions is maintained for $\mathbf{x}_{\mathcal{I}_i}$ ($\forall i = 1, \ldots, g$, N is the size of the sub-population), where each $\mathbf{s}_{ij}$ is called an *individual*. The individuals can be generated by either applying crossover and mutation operators to the previous individuals (genetic algorithms [Holland 1975]), or random sampling according to the distribution learnt from the past (estimation of distribution algorithms [Larrañaga and Lozano 2002] and CMA-ES [Hansen 2011]). Besides, a *context vector* [Van den Bergh and Engelbrecht 2004] [Li and Yao 2012] $\mathbf{cv} = (\mathbf{cv}_1, \ldots, \mathbf{cv}_g)$ is maintained for evaluation of individuals. When evaluating a given individual $\mathbf{s}_{ij}$, a temporary vector $\mathbf{v}_{ij} = (\mathbf{cv}_1, \ldots \mathbf{cv}_{i-1}, \mathbf{s}_{ij}, \mathbf{cv}_{i+1}, \ldots, \mathbf{cv}_g)$ is first obtained by replacing $\mathbf{cv}_i \in \mathbf{cv}$ with $\mathbf{s}_{ij}$. Then, the fitness value of $\mathbf{s}_{ij}$ is assigned to $f(\mathbf{v}_{ij})$. The context vector is randomly initialized and then updated after each cycle. A common strategy is to combine the best-fit individuals together. In this case, the cooperative co-evolution framework reduces to the coordinate descent (when each subcomponent consists of a single variable) or block coordinate descent methods. Fig. 1 gives an illustration of the above process of the cooperative co-evolution framework.

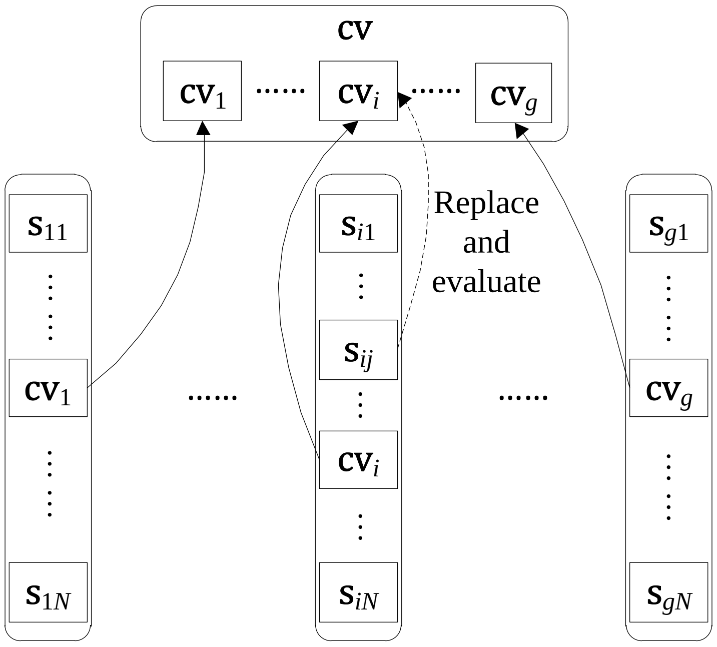

Fig. 1. Sub-populations and the context vector $\mathbf{cv}$ of the cooperative co-evolution framework, where $\mathbf{cv}_i$ is the best-fit individual in the sub-population $\{\mathbf{s}_{1i}, \ldots, \mathbf{s}_{iN}\}$. When evaluating $\mathbf{s}_{ij}$, it first replaces $\mathbf{cv}_i$ in $\mathbf{cv}$. Then, its fitness value is set to $f(\mathbf{cv}_1, \ldots, \mathbf{cv}_{i-1}, \mathbf{s}_{ij}, \mathbf{cv}_{i+1}, \ldots, \mathbf{cv}_g)$

To find the ideal decomposition, various fixed grouping schemes have been proposed (e.g., the one-dimensional [Liu et al. 2001], split-in-half [Shi et al. 2005] strategies and the more general one dividing into k s-dimensional subcomponents where $k \times s = n$ [Van den Bergh and Engelbrecht 2004]). All these approaches divide the index set according to its natural order before the optimization and no longer change the decomposition throughout the optimization process. In this case, it is hard to identify ideal partitions of the index set for the functions that are not fully separable. For example, x_1 and x_n (n is the number of variables) are never placed in the same group by the fixed grouping methods. Thus, the ideal decomposition can never be found for a function in which x_1 interacts with x_n.

To increase the probability of obtaining the ideal decomposition, Yang *et al.* [Yang et al. 2008] proposed a *random grouping* method, which randomly shuffles the indices of the variables to obtain a different partition at the beginning of each cycle. Suppose that in each cycle, there is a probability p $(0 < p < 1)$ of obtaining ideal partitions, then the probability of obtaining ideal partitions in at least one cycle out of the total T cycles is $1 - (1-p)^T \gg p$. The efficacy of random grouping has been verified on the CEC'2010 LSGO benchmark functions with 1000 to 2000 decision variables [Yang et al. 2008] [Li and Yao 2012]. To further increase such probability, Omidvar *et al.* [Omidvar et al. 2010] suggested more frequent random grouping (i.e., shorter cycles). Subsequent to random grouping, various dynamic grouping methods have been proposed to learn the structure of the ideal decomposition based on the interactions between variables (e.g., LEGO [Smith and Fogarty 1995], LLGA [Harik 1997]), adaptive coevolutionary optimization [Weicker and Weicker 1999], Cooperative Co-evolution with Variable Interaction Learning (CCVIL) [Chen et al. 2010], delta grouping [Omidvar et al. 2010] and route distance grouping [Mei et al. 2014]).

Although the proposed dynamic grouping methods significantly increase the chance of obtaining ideal partitions, such probability is still low, especially for the problems whose ideal partitions have complicated structures (e.g., consisting of many imbalanced subcomponents). In addition, the computational resource assigned to the optimization under the ideal decomposition is not adequate to reach the global optimum, or at least a good local optimum. Recently, Omidvar *et al.* [Omidvar et al. 2014] proposed the differential grouping method, which has a rigorous theoretical background that guarantees the correctness of the detected interactions between decision variables. It learns the interactions between variables, and groups them before the optimization, fixing the grouping throughout the optimization. Differential grouping has been empirically verified to be the state-of-the-art decomposition method based on the results obtained from the CEC'2010 large-scale global optimization (LSGO) benchmark functions [Omidvar et al. 2014].

3. COOPERATIVE CO-EVOLUTION WITH GLOBAL DIFFERENTIAL GROUPING AND CMA-ES

The partially additively separable objective function considered in this paper takes the following form:

$$f(\mathbf{x}) = \sum_{i=1}^{g} f_i(\mathbf{x}_{\mathcal{I}_i}). \tag{6}$$

Therefore, provided that an ideal partition $(\mathcal{I}_1, \ldots, \mathcal{I}_g)$ has been successfully obtained, any function with the following form is an ideal component function:

$$\mathit{fit}_i(\mathbf{x}_{\mathcal{I}_i}) = f(\mathbf{a}_1, \ldots, \mathbf{a}_{i-1}, \mathbf{x}_{\mathcal{I}_i}, \mathbf{a}_{i+1}, \ldots, \mathbf{a}_g), \forall\ \mathbf{a}_j \in \Omega(\mathbf{x}_{\mathcal{I}_j}), j \neq i, \tag{7}$$

where $\Omega(\mathbf{x}_{\mathcal{I}_j})$ indicates the solution space of $\mathbf{x}_{\mathcal{I}_j}$.

In the cooperative co-evolution framework, we set $\mathbf{a}_j = \mathbf{cv}_j^{(t)}$, $\forall j \neq i$, $t \geq 0$, where $\mathbf{cv}_j^{(t)}$ is the j^{th} context sub-vector at the t^{th} iteration. In other words, the cooperative co-evolution framework optimizes the subcomponents in terms of $\mathit{fit}_i^{(t)}(\mathbf{x}_{\mathcal{I}_i}) = f\big(\mathbf{cv}_1^{(t-1)}, \ldots, \mathbf{cv}_{i-1}^{(t-1)}, \mathbf{x}_{\mathcal{I}_i}, \mathbf{cv}_{i+1}^{(t-1)}, \ldots, \mathbf{cv}_n^{(t-1)}\big)$. If all the subcomponents converge to their own global optima $\mathbf{x}_{\mathcal{I}_i}^{*(t)} = \arg\min\{\mathit{fit}_i^{(t)}(\mathbf{x}_{\mathcal{I}_i})\}$ $(i = 1, \ldots, g)$, the entire solution must converge to the global optimum $\mathbf{x}^* = \arg\min\{f(\mathbf{x})\}$. Therefore, in the cooperative co-evolution framework, one only needs to identify an ideal partition of the index set. In our work, a Global Differential Grouping (GDG) is proposed to identify ideal partitions, and a variant of CMA-ES is adopted to optimize each subcomponent.

The proposed algorithm, named CC-GDG-CMAES, is described in Algorithm 1, where f is the objective function to be minimized and $\mathbf{x} = (x_1, \ldots, x_n)$ is the decision vector. The parameter n is the problem size. The vectors $\mathbf{ub} = (ub_1, \ldots, ub_n)$ and $\mathbf{lb} = (lb_1, \ldots, lb_n)$ indicate the upper and lower bounds of the variables $(x_1, \ldots, x_n)$. The parameter ϵ is the threshold to detect interactions by GDG, and Γ_{max} is the maximal number of fitness evaluations. The entire algorithm can be divided into three phases: the decomposition phase, the initialization phase, and the optimization phase. During the decomposition phase (line 2), the partition $(\mathcal{I}_1, \ldots, \mathcal{I}_g)$ of the index set is obtained by `GDG()` along with the number of subcomponents g and the number of fitness evaluations Γ that have been consumed. The partition is then fixed throughout the subsequent phases. During the initialization phase (lines 4–8), the parameters $\mathcal{A}$ of CMA-ES are initialized in the standard way [Hansen 2011]. Then, the context vector $\mathbf{cv} = (\mathbf{cv}_1, \ldots, \mathbf{cv}_g)$ is randomly initialized by `randinit()`. The optimization phase (lines 10–14) is divided into a number of cycles. Within each cycle, an iteration of CMA-ES is applied to optimize the subcomponents sequentially by `CMAESIter()`, and

ALGORITHM 1: $(\mathbf{x}^*, f^*) = \texttt{CC-GDG-CMAES}(f, \mathbf{x}, n, \mathbf{ub}, \mathbf{lb}, \epsilon, \Gamma_{max})$

```
/* (Phase 1): Decomposition */;
$(g, \mathcal{I}_1, \ldots, \mathcal{I}_g, \Gamma) = \texttt{GDG}(f, \mathbf{x}, n, \mathbf{ub}, \mathbf{lb}, \epsilon)$;
/* (Phase 2): Initialization */;
Initialize the CMA-ES parameters $\mathcal{A}$;
for $i = 1 \to g$ do
    $\Gamma_i = 0$;
end
$\mathbf{cv} = \texttt{randinit}(\mathbf{ub}, \mathbf{lb}, n)$;
/* (Phase 3): Optimization */;
while $\sum_{i=1}^{g} \Gamma_i < \Gamma_{max} - \Gamma$ do
    for $i = 1 \to g$ do
        $(\mathbf{cv}, \mathcal{A}, \Gamma_i) = \texttt{CMAESIter}(\mathbf{cv}, \mathcal{I}_i, \mathcal{A}, \Gamma_i)$;
    end
end
$\mathbf{x}^* = \mathbf{cv}$, $f^* = f(\mathbf{x}^*)$;
return $(\mathbf{x}^*, f^*)$;
```

the corresponding coordinates of **cv** are replaced by the latest best-fit individual (line 12). The CMA-ES parameters $\mathcal{A}$ and number of fitness evaluations Γ_i are updated as well. The algorithm stops when the total number of fitness evaluations exceeds the maximal number of fitness evaluations Γ_{max}. Finally, the best-so-far solution $\mathbf{x}^*$ is set to **cv** and its objective value f^* is obtained.

Next, the details of GDG will be given in Section 3.1, followed by a description of the CMA-ES subcomponent in Section 3.2.

3.1. Global Differential Grouping

The Global Differential Grouping (GDG) method is extended from the differential grouping method [Omidvar et al. 2014], which groups the variables based on the interaction between them. To facilitate the description of GDG, two complementary relationships between variables are first defined in Definitions 3.1 and 3.2.

Definition 3.1 (*Interaction*). Given a function $f(x_1, \ldots, x_n)$, for any pair of variables x_p and x_q, if $\exists (a, b)$, $\frac{\partial^2 f}{\partial x_p \partial x_q}|_{x_p=a, x_q=b} \neq 0$, then x_p and x_q are said to *interact* with each other.

Definition 3.2 (*Independence*). Given a function $f(x_1, \ldots, x_n)$, for any pair of variables x_p and x_q, if $\forall (a, b)$, $\frac{\partial^2 f}{\partial x_p \partial x_q}|_{x_p=a, x_q=b} = 0$, then x_p and x_q are said to be *independent* from each other.

It is obvious that two variables either interact with or are independent from each other. Based on the above definitions, a variable is called *separable* if it is independent from all the other variables, and *nonseparable* if it interacts with at least one of the other variables.

The differential grouping method was derived from Theorem 3.3 [Omidvar et al. 2014], which is described below:

THEOREM 3.3. $\forall a, b_1 \neq b_2, \delta \in \mathbb{R}, \delta \neq 0$, *if* $f(\mathbf{x})$ *is a partially separable function, and the following condition holds*

$$\Delta_{\delta, x_p}[f](\mathbf{x})|_{x_p=a, x_q=b_1} \neq \Delta_{\delta, x_p}[f](\mathbf{x})|_{x_p=a, x_q=b_2}, \quad (8)$$

ALGORITHM 2: $(g, \mathcal{I}_1, \ldots, \mathcal{I}_g, \mathcal{I}_{sep}, \Gamma) = \texttt{DG}(f, n, \mathbf{ub}, \mathbf{lb}, \epsilon)$

```
$\mathcal{D} = \{1, ..., n\}$, $\mathcal{I}_{sep} = \{\}$, $g = 0$, $\Gamma = 0$;
for $i \in \mathcal{D}$ do
    $\mathcal{I}_{tmp} = \{i\}$;
    for $j \in \mathcal{D} \setminus \{i\}$ do
        $\mathbf{p}_1 = \mathbf{lb}$, $\mathbf{p}_2 = \mathbf{p}_1$, $\mathbf{p}_2(i) = ub_i$;
        $\Delta_1 = f(\mathbf{p}_1) - f(\mathbf{p}_2)$;
        $\mathbf{p}_1(j) = 0$, $\mathbf{p}_2(j) = 0$;
        $\Delta_2 = f(\mathbf{p}_1) - f(\mathbf{p}_2)$;
        $\Gamma \leftarrow \Gamma + 4$;
        if $|\Delta_1 - \Delta_2| > \epsilon$ then
            $\mathcal{I}_{tmp} \leftarrow \mathcal{I}_{tmp} \cup \{j\}$, $\mathcal{D} \leftarrow \mathcal{D} \setminus \{j\}$;
        end
    end
    if $|\mathcal{I}_{tmp}| = 1$ then
        $\mathcal{I}_{sep} \leftarrow \mathcal{I}_{sep} \cup \mathcal{I}_{tmp}$;
    else
        $g \leftarrow g + 1$;
        $\mathcal{I}_g = \mathcal{I}_{tmp}$;
    end
end
return $(g, \mathcal{I}_1, \ldots, \mathcal{I}_g, \mathcal{I}_{sep}, \Gamma)$;
```

where

$$\Delta_{\delta,x_p}[f](\mathbf{x}) = f(\ldots, x_p + \delta, \ldots) - f(\ldots, x_p, \ldots) \tag{9}$$

refers to the forward difference of f with respect to variable x_p with interval δ, then x_p and x_q interact with each other,

Based on Theorem 3.3, one can detect the interaction between any two different variables x_i and x_j by checking whether the condition Eq. (8) holds. The differential grouping method is described in Algorithm 2, which implements Theorem 3.3 by setting $a = lb_i$, $b_1 = lb_j$, $b_2 = 0$, $\delta = ub_i - lb_i$ and $x_k = lb_k$ for all $k \in \{1, \ldots, n\} \setminus \{i, j\}$. Then, $\Delta_1 = \Delta_{\delta,x_p}[f](\mathbf{x})|_{x_p=a,x_q=b_1}$ and $\Delta_2 = \Delta_{\delta,x_p}[f](\mathbf{x})|_{x_p=a,x_q=b_2}$. For any pair of variables (x_i, x_j), if the difference between Δ_1 and Δ_2 is greater than the predefined threshold ϵ, then the two variables are considered to interact with each other and their indices are placed in the same subset. In addition, the indices of all separable variables are placed in a single separable index subset $\mathcal{I}_{sep}$.

Although Algorithm 2 has been demonstrated to be effective in grouping the interacting variables together, and can potentially save a considerable number of fitness evolutions when the subcomponents are mutually exclusive, it has three major drawbacks. First, the comparison between the variables is not complete and many interactions between variables may be missed. For example, suppose that there are three variables (x_1, x_2, x_3), where x_2 interacts with x_1 and x_3, and x_1 is independent from x_3 (e.g., $f(x_1, x_2, x_3) = x_1x_2 + x_2x_3$). In Algorithm 2, the index of x_2 is removed from the set $\mathcal{D}$ immediately after its interaction with x_1 has been detected. Then, the interaction between x_2 and x_3 cannot be detected, and the final index partition will be $\mathcal{I}_1 = \{1, 2\}$ and $\mathcal{I}_2 = \{3\}$, which is different from the unique ideal partition $\mathcal{I}_1 = \{1, 2, 3\}$. This phenomenon happens on the Rosenbrock function [Rosenbrock 1960], where each variable interacts with at most two of the other variables.

The second drawback is that, the grouping performance is sensitive to the threshold ϵ. Theoretically, the value of ϵ can be set to 0, since any positive difference between Δ_1 and Δ_2 implies interaction between the corresponding variables. However, in practice, the computer operations incur computational errors and cause non-zero values of $|\Delta_1 - \Delta_2|$ for independent variables. To tolerate such computational errors, a positive small threshold ϵ is proposed and employed in line 10 [Omidvar et al. 2014], and three values of 10^{-1}, 10^{-3} and 10^{-6} are empirically compared. The results show that the best ϵ value varies from function to function, which indicates that ϵ needs to be set adaptively as per the test function rather than fixed.

The third problem is that, it cannot deal with fully separable functions or the functions with a large portion of separable variables. It only keeps a single index subset $\mathcal{I}_{sep}$ for all the separable variables. As a result, it cannot optimize the corresponding subcomponent $\mathbf{x}_{\mathcal{I}_{sep}}$ effectively. For example, it optimizes all the variables of the fully separable functions without any decomposition. It has been shown that the decision of the optimal subcomponent size for fully separable functions is still an open and challenging task, and any intermediary decomposition between the two extreme decompositions is better [Omidvar et al. 2014].

To address the first issue, the entire matrix $\Lambda_{n\times n}$ of all the $|\Delta_1 - \Delta_2|$'s is calculated and maintained, where Λ_{ij} is the $|\Delta_1 - \Delta_2|$ value between the variables x_i and x_j. Then, a matrix $\Theta_{n\times n}$ of the interaction between the variables is obtained from $\Lambda_{n\times n}$ and ϵ. The entry Θ_{ij} takes 1 if $\Lambda_{ij} > \epsilon$, and 0 otherwise. The Θ matrix is also known as the *design structure matrix* in the area of system decomposition and integration [Browning 2001], which indicates whether the objects of the system relate to each other. Finally, the decomposition of the variables can be modelled as the computation of the connected components of the graph with the node adjacency matrix of Θ, which can be easily solved in linear time in terms of n using breadth-first search or depth-first search [Hopcroft and Tarjan 1973]. The grouping method based on the Θ matrix is called the Global Differential Grouping (GDG), as it maintains the global information of the interactions between variables. An example of GDG is given below. Given a function $f(\mathbf{x}) = x_1x_2 + x_1x_4 + x_2x_4 + x_3x_5x_6 + x_5x_6x_7$ with a unique lower bound of -1 and an upper bound of 1 for all the variables, the Λ matrix obtained by GDG is shown in Eq. (10). With a sufficiently small ϵ, the Θ matrix is then shown in Eq. (11). Considering Eq. (11) as the adjacency matrix of a graph, as shown in Fig. 2, the connected components are $\{x_1, x_2, x_4\}$ and $\{x_3, x_5, x_6, x_7\}$.

$$\Lambda = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \\ \begin{pmatrix} 0 & 2 & 0 & 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 2 & 0 \\ 2 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 4 & 2 \\ 0 & 0 & 2 & 0 & 4 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 0 \end{pmatrix} \end{array}, \tag{10}$$

$$\Theta = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}. \tag{11}$$

Then, to address the sensitivity of the grouping performance to the threshold ϵ and to determine the best ϵ value specifically for each function, an intuitive method is used

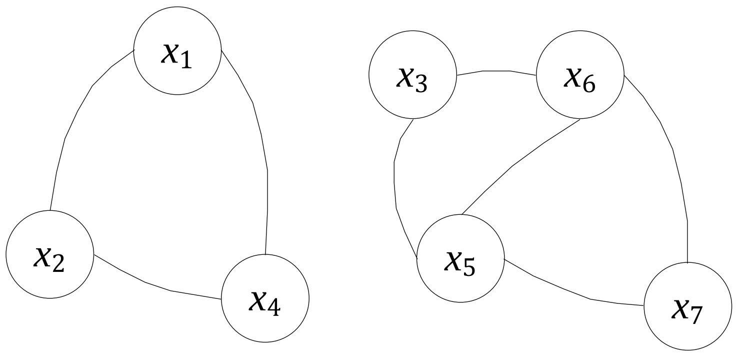

Fig. 2. The graph corresponding to the adjacency matrix Eq. (11).

to estimate the magnitude of the computational error based on the magnitude of the objective space. Specifically, K points $\mathbf{x}_1, \ldots, \mathbf{x}_K$ are randomly sampled in the decision space, and their objective values $f(\mathbf{x}_1), \ldots, f(\mathbf{x}_K)$ are evaluated. Then, we choose the ϵ value as follows:

$$\epsilon = \alpha \cdot \min\{|f(\mathbf{x}_1)|, \ldots, |f(\mathbf{x}_K)|\}, \tag{12}$$

where α is the controlling coefficient which will be set to 10^{-10} in the experimental studies.

Finally, a proper ideal partition is still to be decided for the fully separable functions or the functions with a large portion of separable variables. To this end, a simple scheme is adopted. It is known that if the original partition of the index set is ideal, then any further partition for the separable index subset $\mathcal{I}_{sep}$ will lead to another ideal partition [Omidvar et al. 2014]. Then, the separable index subset is further divided into smaller subsets whose sizes are set according to a rule of thumb, which is to choose a value that is small enough to be within the capacity of the subcomponent optimizer, but it should not be made any smaller [Omidvar et al. 2014]. Based on the previous studies [Hansen et al. 2010], CMA-ES performs well for the functions with up to 40 dimensions. Its competitiveness may not be maintained on large-scale problems, mainly due to the rapid increase of the covariance matrix [Omidvar and Li 2010] [Ros and Hansen 2008]. Here, the subcomponent size is set to 20, which is a more conservative value. Therefore, if $\mathcal{I}_{sep}$ is still large, it is further divided into 20-dimensional or smaller subsets. For example, if $|\mathcal{I}_{sep}| = 950$, then it is further divided into 47 20-dimensional subsets plus a 10-dimensional subset.

The pseudocode of GDG is described in Algorithm 3. All the fitness values used to obtain the Λ and Θ matrices are stored in the scalar F_1, the vectors $\mathbf{F}_2$ and $\mathbf{F}_3$, and the matrix $\mathbf{F}_4$. Then, the partition $(\mathcal{I}'_1, \ldots, \mathcal{I}'_k)$ is obtained by the function `ConnComp()`, which computes the connected components based on the node adjacency matrix Θ and considers each separable variable as a single component, since it is an isolated node in the graph. This can be done simply by depth-first search or breadth-first search in linear time, whose details can be found in [Hopcroft and Tarjan 1973]. Then, the indices of all the separable variables are grouped into $\mathcal{I}_{sep}$ (lines 24 – 31) and further divided into 20-dimensional or smaller subsets (lines 32 – 40). It can be seen that GDG takes a fixed number of fitness evaluations of

$$\Gamma = 1 + 2n + \frac{n(n-1)}{2} = \frac{n^2 + 3n + 2}{2}. \tag{13}$$

ALGORITHM 3: $(g, \mathcal{I}_1, \dots, \mathcal{I}_g, \Gamma) = \texttt{GDG}(f, n, \mathbf{ub}, \mathbf{lb}, \epsilon)$

```
$\mathbf{p}_1 = \mathbf{lb}$, $F_1 = f(\mathbf{p}_1)$, $\Gamma = 1$;
$\mathbf{F}_2 = \mathbf{0}_{1\times n}$, $\mathbf{F}_3 = \mathbf{0}_{1\times n}$, $\mathbf{F}_4 = \mathbf{0}_{n\times n}$;
for $i = 1 \to n$ do
    $\mathbf{p}_2 = \mathbf{p}_1$, $\mathbf{p}_2(i) = ub_i$, $\mathbf{F}_2(i) = f(\mathbf{p}_2)$, $\Gamma \leftarrow \Gamma + 1$;
    $\mathbf{p}_3 = \mathbf{p}_1$, $\mathbf{p}_3(i) = 0$, $\mathbf{F}_3(i) = f(\mathbf{p}_3)$, $\Gamma \leftarrow \Gamma + 1$;
end
for $i = 1 \to n - 1$ do
    for $j = i + 1 \to n$ do
        $\mathbf{p}_4 = \mathbf{p}_1$, $\mathbf{p}_4(i) = ub_i$, $\mathbf{p}_4(j) = 0$;
        $\mathbf{F}_4(i, j) = \mathbf{F}_4(j, i) = f(\mathbf{p}_4)$, $\Gamma \leftarrow \Gamma + 1$;
    end
end
$\Lambda = \mathbf{0}_{n\times n}$, $\Theta = \mathbf{0}_{n\times n}$;
for $i = 1 \to n - 1$ do
    for $j = i + 1 \to n$ do
        $\Delta_1 = F_1 - \mathbf{F}_2(i)$, $\Delta_2 = \mathbf{F}_3(j) - \mathbf{F}_4(i, j)$;
        $\Lambda_{ij} = |\Delta_1 - \Delta_2|$;
        if $\Lambda_{ij} > \epsilon$ then
            $\Theta_{ij} = 1$;
        end
    end
end
$(k, \mathcal{I}'_1, \dots, \mathcal{I}'_k) = \texttt{ConnComp}(\Theta)$ ;                    // Computing the connected components
$\mathcal{I}_{sep} = \{\}$, $g = 0$;
for $i = 1 \to k$ do
    if $|\mathcal{I}'_i| = 1$ then
        $\mathcal{I}_{sep} \leftarrow \mathcal{I}_{sep} \cup \mathcal{I}'_i$;
    else
        $g \leftarrow g + 1$, $\mathcal{I}_g = \mathcal{I}'_i$;
    end
end
while $|\mathcal{I}_{sep}| > 0$ do
    $g \leftarrow g + 1$, $\mathcal{I}_g = \{\}$;
    for $i \in \mathcal{I}_{sep}$ do
        $\mathcal{I}_g \leftarrow \mathcal{I} \cup \{i\}$, $\mathcal{I}_{sep} \leftarrow \mathcal{I}_{sep} \setminus \{i\}$;
        if $|\mathcal{I}_{sep}| = 0 \vee |\mathcal{I}_g| = 20$ then
            break;
        end
    end
end
return $(g, \mathcal{I}_1, \dots, \mathcal{I}_g, \Gamma)$;
```

3.2. CMA-ES subcomponent optimizer

CMA-ES [Hansen and Ostermeier 1996] was proposed for solving difficult non-linear non-convex continuous optimization problems. As the name implies, the basic idea of CMA-ES is the covariance matrix adaptation. It samples the population based on the current covariance matrix between the variables, and updates the covariance matrix by the distribution of the best subset (e.g., half) of the individuals, weighted by their fitness values. Details of CMA-ES can be found in [Hansen 2011].

ALGORITHM 4: $\mathbf{cv} = \texttt{randinit}(\mathbf{ub}, \mathbf{lb}, n)$

```
1 $\mathbf{x}^{(0)} = \mathbf{lb} + (\mathbf{lb} + \mathbf{ub})/2$, $\boldsymbol{\sigma}^{(0)} = 0.3 \cdot (\mathbf{ub} - \mathbf{lb})$;
2 $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_{n\times n})$;
3 $\mathbf{cv} = \mathbf{x}^{(0)} + \boldsymbol{\sigma}^{(0)} \circ \mathbf{z}$ ;          // $\circ$ is the entrywise product of two vectors/matrices
4 return $\mathbf{cv}$;
```

The CMA-ES subcomponent optimizer is directly derived from the minimalistic implementation of CMA-ES (`purecmaes.m`), which is given on the official website [1]. Here, we only list the differences between the variant adopted in our approach (`CMAESIter()` in Algorithm 1) and the original CMA-ES (`purecmaes.m`), without giving the full details of the algorithm.

(1) `CMAESIter()` runs a single iteration of `purecmaes.m` instead of a number of iterations.
(2) In `CMAESIter()`, the fitness function of each individual $\mathbf{p}_{ij}$ for the index subset $\mathcal{I}_i$ is defined as $f(\mathbf{cv}_1, \ldots, \mathbf{cv}_{i-1}, \mathbf{p}_{ij}, \mathbf{cv}_{i+1}, \ldots, \mathbf{cv}_g)$, where $\mathbf{cv}_k$ is the best-so-far individual for $\mathbf{x}_{\mathcal{I}_k}$, $\forall k \in \{1, \ldots, g\} \setminus \{i\}$, since the explicit fitness function $f_i(\mathbf{p}_{ij})$ is not available. Due to the fact that CMA-ES uses truncation selection at each iteration, $f(\mathbf{cv}_1, \ldots, \mathbf{cv}_{i-1}, \mathbf{p}_{ij}, \mathbf{cv}_{i+1}, \ldots, \mathbf{cv}_g)$ leads to the same CMA-ES process as $f_i(\mathbf{p}_{ij})$.
(3) The initialization of `CMAESIter()` is different from that of `purecmaes.m`. First, the initial mean is randomly sampled from a uniform distribution between 0 and 1 in `purecmaes.m`, while it is fixed to the center of the solution space $\mathbf{lb} + (\mathbf{lb} + \mathbf{ub})/2$ in `CMAESIter()`. Second, the initial standard deviation σ is a scalar and is fixed to 0.5 in `purecmaes.m` regardless of the range of the decision variables. In `CMAESIter()`, on the other hand, σ_i is initialized to $0.3 \cdot (ub_i - lb_i)$ for each decision variable respectively to take the range of the decision variables into account. Finally, in our algorithm, the newly introduced context vector **cv** is initialized by `randinit()`, which is described in Algorithm 4.

Note that in CMA-ES, the population size λ^k for the subcomponent $\mathbf{x}_{\mathcal{I}_k}$ monotonously increases with the problem size $|\mathcal{I}_k|$ ($\lambda^k = 4 + \lfloor 3\log(|\mathcal{I}_k|)\rfloor$). Such an adaptive setting of population size addresses the issue of contributions of different subcomponents mentioned in [Omidvar et al. 2011] [Omidvar et al. 2014] to some extent. This is consistent with the intuition that a larger subcomponent contributes more to the objective function and thus needs to be allocated to more computational resources by a larger population size.

4. EXPERIMENTAL STUDIES

In the experimental studies, the proposed CC-GDG-CMAES is evaluated on the CEC'2010 large-scale global optimization (LSGO) benchmark functions [Tang et al. 2009], which consist of 20 1000-dimensional benchmark functions (f_1 to f_{20}) that can be divided into five categories. The details of the categorization of the functions are given in Table I.

First, the accuracy of GDG is evaluated by comparing the interaction matrix obtained by GDG with the true interaction matrix. Then, the final results obtained by CC-GDG-CMAES are compared with that of DECC-I [Omidvar et al. 2014], MA-SW-Chains [Molina et al. 2010], CMA-ES [Hansen 2011] (the minimalistic implementation `purecmaes.m`) and the Cooperative Co-evolution with Ideal Grouping and CMA-ES (CC-

[1]https://www.lri.fr/~hansen/purecmaes.m

Table I. The categorization of the CEC'2010 LSGO benchmark functions.

Category	Functions	Description
Fully separable	f_1 to f_3	1000 separable variables
Single-group 50-nonseparable	f_4 to f_8	One 50-dimensional nonseparable group plus 950 separable variables
10-group 50-nonseparable	f_9 to f_{13}	10 50-dimensional nonseparable groups plus 500 separable variables
20-group 50-nonseparable	f_{14} to f_{18}	20 50-dimensional nonseparable groups
Nonseparable	f_{19} to f_{20}	Single 1000-dimensional nonseparable group

Table II. Description of the compared algorithms in the experimental studies.

Algorithm	Description
DECC-I [Omidvar et al. 2014]	Differential Evolution with Cooperative Co-evolution and Ideal Decomposition
MA-SW-Chains [Molina et al. 2010]	Memetic Algorithm with adaptive local search intensity and *Solis Wets*' local search algorithm
CMA-ES [Hansen 2011]	Evolution Strategy with Covariance Matrix Adaptation
CC-IG-CMAES	Cooperative Co-evolution with Ideal Decomposition and CMA-ES
CC-GDG-CMAES	Cooperative Co-evolution with Global Differential Grouping and CMA-ES

IG-CMAES). DECC-I and MA-SW-Chains are the representatives of the state-of-the-art algorithms for solving large-scale black-box optimization problems. DECC-I adopts the differential evolution with cooperative co-evolution [Yang et al. 2008] and an ideal partition of the index set obtained manually using the knowledge of the benchmark functions. In this ideal partition, all the separable variables are placed in a single group called the *separable group*. MA-SW-Chains is a memetic algorithm that assigns local search intensity to individuals by chaining different local search applications to explore more effectively in the search space. CMA-ES is one of the state-of-the-art algorithms for small-sized black-box optimization problems [Hansen et al. 2010], and can be seen as a special case of CC-GDG-CMAES where there is no decomposition. Therefore, the comparison with CMA-ES will show the efficacy of GDG. CC-IG-CMAES replaces the GDG part of CC-GDG-CMAES by the pre-selected ideal partition of the index set. Therefore, the comparison with CC-IG-CMAES shows the effect on the final solution when GDG cannot find any ideal partition. The compared algorithms in the experimental studies are summarized in Table II.

4.1. Experimental Settings

The parameter settings of CC-GDG-CMAES are given in Table III. All the parameters of CMA-ES are set in the standard way suggested on the official website [2]. Thus, besides the stopping criterion Γ_{max}, there are only two parameters related to GDG (Eq. (12)) – K and α. Here, $\Gamma_{max} = 3 \times 10^6$ is a commonly-used setting which has been adopted by most of the previous studies including the compared DECC-I and MA-SW-Chains. All the compared algorithms were run 25 times independently.

4.2. Results and Discussions

4.2.1. Grouping Accuracy. First, the accuracy of GDG is evaluated on f_1 to f_{20}. To this end, three measures are defined to indicate the accuracies of identifying the (1) interaction; (2) independence and (3) both types of relationships. They are defined as

[2]https://www.lri.fr/~hansen/purecmaes.m

Table III. The parameter settings of the compared algorithms.

Parameter	**Description**	**Value**
K	Samples to estimate the magnitude of objective space	10
α	Coefficient to obtain ϵ	10^{-10}
Γ_{max}	Maximal fitness evaluations	3×10^6

Table IV. Accuracies of identifying interaction, independence and both types of relationships of GDG with $\alpha = 10^{-10}$ on the CEC'2010 LSGO benchmark functions.

Measure	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
ρ_1	-	-	-	100%	100%	100%	100%	100%	100%	100%
ρ_2	100%	100%	2.8%	100%	100%	100%	100%	100%	100%	100%
ρ_3	100%	100%	2.8%	100%	100%	100%	100%	100%	100%	100%
Measure	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
ρ_1	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
ρ_2	75.5%	100%	100%	100%	100%	100%	100%	100%	-	100%
ρ_3	76.1%	100%	100%	100%	100%	100%	100%	100%	100%	100%

follows:

$$\rho_1 = \frac{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\Theta \circ \Theta_{ideal})_{i,j}}{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\Theta_{ideal})_{i,j}} \times 100\%, \tag{14}$$

$$\rho_2 = \frac{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}((\mathbf{1}_{n\times n} - \Theta) \circ (\mathbf{1}_{n\times n} - \Theta_{ideal}))_{i,j}}{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\mathbf{1}_{n\times n} - \Theta_{ideal})_{i,j}} \times 100\%, \tag{15}$$

$$\rho_3 = \frac{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\mathbf{1}_{n\times n} - |\Theta - \Theta_{ideal}|)_{i,j}}{n(n-1)/2} \times 100\%, \tag{16}$$

where Θ_{ideal} is the ideal interaction matrix. $(\Theta_{ideal})_{i,j}$ equals 1 if variable i and j interact with each other, and 0 otherwise. $|\Theta - \Theta_{ideal}|_{i,j}$ takes 0 if $\Theta_{i,j} = (\Theta_{ideal})_{i,j}$, and 1 otherwise. The operator "$\circ$" is the entrywise product of two matrices.

Table IV shows the three accuracies of GDG on the CEC'2010 LSGO benchmark functions. Note that there is no ρ_1 for f_1 to f_3, because all the variables are independent from each other and thus there is no interaction. There is no ρ_2 for f_{19} since all the variables interact with each other. It should be noted that the nonseparable Rosenbrock function f_{20} has both interaction and independence in the Θ matrix due to the chain-like interaction between the variables. That is, x_i only interacts with x_{i-1} and x_{i+1}, and is independent from all the other variables.

It can be seen that GDG has perfect accuracy on most of the benchmark functions. The low ρ_2 values on f_3 and f_{11} imply that most of the computational errors between independent variables are larger than ϵ on these two functions. When looking further into the details of f_3 and f_{11}, the parts of the objective functions including the separable variables are both found to be the Ackley function, which is stated as follows:

$$f_{Ackley}(\mathbf{x}) = 20 - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + e. \tag{17}$$

One can see that the Ackley function includes the exponential function. This is consistent with our intuition that exponential functions can induce larger computational

Table V. Accuracies of identifying interaction, independence and both types of relationships of GDG with $\alpha = 10^{-9}$ on the CEC'2010 LSGO benchmark functions.

Measure	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
ρ_1	-	-	-	100%	100%	100%	100%	100%	100%	100%
ρ_2	100%	100%	15.0%	100%	100%	100%	100%	100%	100%	100%
ρ_3	100%	100%	15.0%	100%	100%	100%	100%	100%	100%	100%
Measure	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
ρ_1	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
ρ_2	81.2%	100%	100%	100%	100%	100%	100%	100%	-	100%
ρ_3	81.6%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Table VI. Accuracies of identifying interaction, independence and both types of relationships of GDG with $\alpha = 10^{-8}$ on the CEC'2010 LSGO benchmark functions.

Measure	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
ρ_1	-	-	-	100%	100%	100%	100%	100%	100%	100%
ρ_2	100%	100%	60.1%	100%	100%	100%	100%	100%	100%	100%
ρ_3	100%	100%	60.1%	100%	100%	100%	100%	100%	100%	100%
Measure	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
ρ_1	100%	100%	100%	100%	100%	100%	100%	100%	100%	99.8%
ρ_2	96.5%	100%	100%	100%	100%	100%	100%	100%	-	100%
ρ_3	96.6%	100%	100%	100%	100%	100%	100%	100%	100%	100.0%

errors due to their higher complexity than the polynomial functions. The observation suggests a possibility of a more adaptive setting of ϵ based not only on the magnitude of objective space, but also on the complexity level of the objective function.

As a result, GDG obtained ideal partitions of the variables for all the test functions except f_3 and f_{11}. For f_3, all the 1000 separable variables were placed in a single nonseparable group. For f_{11}, the 500 nonseparable variables were perfectly divided into 10 50-dimensional nonseparable groups, and the 500 separable variables were placed in a single nonseparable group. Overall, GDG is demonstrated to be effective in identifying ideal partitions of the variables.

In order to investigate the sensitivity of the parameter α defined in Eq. (12) to the accuracy of GDG, comparative analysis is carried out using different α values. Tables V – VI show the accuracies achieved by $\alpha = 10^{-9}$ and 10^{-8}. One can see that for f_3 and f_{11}, the accuracy improves with the increase of α. However, when $\alpha = 10^{-8}$, some of the interactions in f_{20} can no longer be identified. When looking into the grouping results, the 1000 separable variables in f_3 were still placed in a single nonseparable group for both $\alpha = 10^{-9}$ and 10^{-8}. For f_{11}, the 500 separable variables were placed in a single nonseparable group when $\alpha = 10^{-9}$, and divided into a 165-dimensional separable group and a 335-nonseparable group when $\alpha = 10^{-8}$. For f_{20} and $\alpha = 10^{-8}$, the 1000 nonseparable variables are divided into three nonseparable groups. The first one is 679-dimensional, the second is 298-dimensional and the last is 23-dimensional. In summary, $\alpha = 10^{-9}$ obtained the highest accuracy among the three compared values. As for the grouping results which are practically used by the subsequent search process, $\alpha = 10^{-10}$ and $\alpha = 10^{-9}$ performed the same. Therefore, we simply keep $\alpha = 10^{-10}$ for the test functions without loss of grouping performance.

4.2.2. Computational Effort of Grouping. Eq. (13) gives the number of fitness evaluations required for carrying out variable grouping by GDG on n-dimensional problems. In the experimental studies, $n = 1000$, and thus the computational effort of grouping is

$$\Gamma = \frac{1000^2 + 3 \times 1000 + 2}{2} = 501501. \tag{18}$$

This is about $1/6$ of the total fitness evaluations $\Gamma_{max} = 3 \times 10^6$. There are nearly 2.5×10^6 fitness evaluations for the subsequent optimization, which is a considerable number.

4.2.3. Optimization Performance. Tables VII and VIII show the median, mean and standard deviation of the fitness values obtained by the 25 independent runs of the compared algorithms on the CEC'2010 LSGO benchmark functions. For testing the statistical significance of the results, first the Kruskal-Wallis one-way ANOVA [Sheskin 2003] with a significance level of 0.05 is used to find out if there is any significant difference between the algorithms. If a significant difference is detected, then a series of pair-wise Wilcoxon rank sum tests [Wilcoxon 1945] with a significance level of 0.05 is conducted with Bonferroni correction [Sheskin 2003] in order to find the best performing algorithm. Bonferroni correction is a simple technique for controlling the family-wise error rate. The family-wise error rate is the accumulation of type I errors when more than one pair-wise comparison is used to rank a set of results. The median of the best performing algorithm is marked in bold. Note that for each of the instances except f_3 and f_{11}, CC-GDG-CMAES manages to obtain an ideal partition, and is thus essentially the same as CC-IG-CMAES. If both algorithms perform significantly better than all the other algorithms, their median values are marked in bold. Overall, it is obvious that CC-GDG-CMAES and CC-IG-CMAES perform much better than the other algorithms. For f_5 and f_6, none of the algorithms perform significantly better than any of the other algorithms due to the large standard deviation of some algorithms (MA-SW-Chains for f_5, and CC-GDG-CMAES and CC-IG-CMAES for f_6). It is noteworthy that for f_6, CC-GDG-CMAES and CC-IG-CMAES obtain large mean values. However, their median values are the global optimum. When looking into the details of the results, it is found that the algorithms obtain the global optimum for 19 out of the 25 runs, but are stuck in very large local optima (with scale of 10^6) for the other 6 runs. The probability of obtaining the global optimum is still high ($19/25 = 76\%$).

Then, CC-GDG-CMAES is compared with CMA-ES and CC-IG-CMAES to verify the efficacy of GDG. From Tables VII and VIII, it is seen that CC-GDG-CMAES outperforms CMA-ES with respect to mean on 13 functions and median on 14 functions, while CC-GDG-CMAES is outperformed by CMA-ES on only one function. It obtains the same results as CMA-ES on f_3, f_{19} and f_{20} since it has no decomposition on these functions, and is thus equivalent to CMA-ES. It is outperformed by CC-IG-CMAES on only two functions (f_3 and f_{11}), since GDG failed to obtain any ideal partition.

Next, CC-GDG-CMAES is compared with DECC-I to show the advantage of CMA-ES as a subcomponent optimizer. It can be seen that CC-GDG-CMAES obtains better mean values on 14 functions and better median values on 15 functions. Note that DECC-I keeps all the separable variables in a single subcomponent, while CC-GDG-CMAES further divides them into at most 20-dimensional subcomponents. Therefore, CC-GDG-CMAES employs different decompositions for f_1, f_2, f_4 to f_{10}, f_{12} and f_{13}, where there are 500 to 1000 separable variables. However, for f_{14} to f_{20}, there is no separable variable and the ideal partitions are unique for both algorithms. CC-GDG-CMAES outperforms DECC-I on 5 out of these 7 functions, which indicates that CMA-ES is an effective subcomponent optimizer for large-scale black-box optimization. For f_3 and f_{11}, CC-GDG-CMAES does not obtain any better ideal partition, and places all the separable variables in a single group. When replacing the grouping of CC-GDG-CMAES with the ideal grouping, the resultant CC-IG-CMAES improves the mean and median on both f_3 and f_{11}, and reliably reaches the global optimum on f_3.

Table VII. Median, mean and standard deviation of the fitness values obtained by the 25 independent runs of the compared algorithms on the CEC'2010 LSGO benchmark functions from f_1 to f_8. Under the pairwise Wilcoxon rank sum test with Bonferroni correction and significance level of 0.05, the median of the best performing algorithms are marked in bold.

Functions		DECC-I	MA-SW-Chains	CMA-ES	CC-IG-CMAES	CC-GDG-CMAES
f_1	Median	1.51e-01	1.50e-14	8.49e+06	**0.00e+00**	**0.00e+00**
	Mean	4.21e+00	2.10e-14	8.60e+06	0.00e+00	0.00e+00
	Std	1.03e+01	1.99e-14	8.01e+05	0.00e+00	0.00e+00
f_2	Median	4.42e+03	**7.90e+02**	5.20e+03	1.61e+03	1.61e+03
	Mean	4.39e+03	8.10e+02	5.21e+03	1.60e+03	1.60e+03
	Std	1.97e+02	5.88e+01	2.20e+02	5.29e+01	5.29e+01
f_3	Median	1.67e+01	6.11e-13	2.17e+01	**0.00e+00**	2.17e+01
	Mean	1.67e+01	7.28e-13	2.17e+01	0.00e+00	2.17e+01
	Std	3.34e-01	3.40e-13	1.06e-02	0.00e+00	1.06e-02
f_4	Median	5.82e+11	3.54e+11	5.22e+13	**1.17e+10**	**1.17e+10**
	Mean	6.13e+11	3.53e+11	5.90e+13	1.60e+10	1.60e+10
	Std	2.08e+11	3.12e+10	1.98e+13	1.25e+10	1.25e+10
f_5	Median	1.37e+08	2.31e+08	6.44e+07	1.02e+08	1.02e+08
	Mean	1.34e+08	1.68e+08	6.37e+07	1.02e+08	1.02e+08
	Std	2.31e+07	1.04e+08	1.31e+07	1.32e+07	1.32e+07
f_6	Median	1.63e+01	1.60e+00	2.17e+01	0.00e+00	0.00e+00
	Mean	1.64e+01	8.14e+04	2.58e+06	6.03e+06	6.03e+06
	Std	2.66e-01	2.84e+05	7.13e+06	9.88e+06	9.88e+06
f_7	Median	4.34e+00	9.04e+01	1.41e+09	**0.00e+00**	**0.00e+00**
	Mean	2.97e+01	1.03e+02	1.35e+09	0.00e+00	0.00e+00
	Std	8.58e+01	8.70e+01	3.29e+08	0.00e+00	0.00e+00
f_8	Median	6.73e+00	**3.43e+06**	6.47e+08	2.20e+07	2.20e+07
	Mean	3.19e+05	1.41e+07	1.08e+09	2.90e+07	2.90e+07
	Std	1.10e+06	3.68e+07	1.35e+09	2.59e+07	2.59e+07
No. Best		0	2	0	4	3

Table IX shows the best fitness values over the 25 independent runs of the compared algorithms. For each function, the best result among the compared algorithms is marked in bold. MA-SW-Chains was the winner in 2010, and DECC-I is an algorithm that was developed later and showed comparative performance with MA-SW-Chains. It can be seen that CC-GDG-CMAES is much better than DECC-I and MA-SW-Chains in terms of best performance, and obtains the global optima for 6 test functions.

To compare CC-GDG-CMAES with CMA-ES and CC-IG-CMAES in more detail, their convergence curves are plotted for five selected functions, one from each category (shown in Table I), as shown in Figs. 3 to 7. Concretely, f_1 is selected from the fully separable functions, where CC-GDG-CMAES performs best among all the compared algorithms. f_5 is selected from the functions still with a large portion of separable variables, but CC-GDG-CMAES does not perform as well as on f_1. f_{11} is selected from the third category, where CC-GDG-CMAES is outperformed by DECC-I. f_{18} is selected from the partially separable functions with no separable variables, on which CC-GDG-CMAES performs significantly better. f_{19} is arbitrarily selected from the two nonseparable functions. In the figures, the x-axis stands for the fitness evaluations and the y-axis represents the average fitness value of the individuals in the current population over the 25 independent runs. The y-axis is in log scale. For f_1, f_5 and f_{18},

Table VIII. Median, mean and standard deviation of the fitness values obtained by the 25 independent runs of the compared algorithms on the CEC'2010 LSGO benchmark functions from f_9 to f_{20}. Under the pairwise Wilcoxon rank sum test with Bonferroni correction and significance level of 0.05, the median of the best performing algorithms are marked in bold.

Functions		DECC-I	MA-SW-Chains	CMA-ES	CC-IG-CMAES	CC-GDG-CMAES
f_9	Median	4.84e+07	1.40e+07	9.55e+06	**1.57e+03**	**1.57e+03**
	Mean	4.84e+07	1.41e+07	9.59e+06	1.74e+03	1.74e+03
	Std	6.56e+06	1.15e+06	1.07e+06	6.95e+02	6.95e+02
f_{10}	Median	4.31e+03	2.07e+03	5.17e+03	**1.80e+03**	**1.80e+03**
	Mean	4.34e+03	2.06e+03	5.20e+03	1.81e+03	1.81e+03
	Std	1.46e+02	1.40e+02	2.83e+02	8.89e+01	8.89e+01
f_{11}	Median	**1.03e+01**	3.70e+01	2.38e+02	4.39e+01	6.44e+01
	Mean	1.02e+01	3.77e+01	2.38e+02	5.07e+01	6.53e+01
	Std	1.14e+00	6.85e+00	1.97e-01	2.04e+01	2.82e+01
f_{12}	Median	1.35e+03	3.50e-06	**0.00e+00**	**0.00e+00**	**0.00e+00**
	Mean	1.48e+03	3.62e-06	0.00e+00	0.00e+00	0.00e+00
	Std	4.28e+02	5.92e-07	0.00e+00	0.00e+00	0.00e+00
f_{13}	Median	6.23e+02	1.07e+03	7.85e+04	**1.59e+02**	**1.59e+02**
	Mean	7.51e+02	1.25e+03	9.15e+04	2.72e+02	2.72e+02
	Std	3.70e+02	5.72e+02	6.77e+04	1.77e+02	1.77e+02
f_{14}	Median	3.34e+08	3.08e+07	1.07e+07	**0.00e+00**	**0.00e+00**
	Mean	3.38e+08	3.10e+07	1.06e+07	0.00e+00	0.00e+00
	Std	2.40e+07	2.19e+06	1.11e+06	0.00e+00	0.00e+00
f_{15}	Median	5.87e+03	2.71e+03	5.16e+03	**2.01e+03**	**2.01e+03**
	Mean	5.88e+03	2.72e+03	5.12e+03	2.00e+03	2.00e+03
	Std	9.89e+01	1.22e+02	1.98e+02	6.74e+01	6.74e+01
f_{16}	Median	**2.49e-13**	9.39e+01	4.33e+02	8.46e+01	8.46e+01
	Mean	2.47e-13	1.01e+02	4.33e+02	9.67e+01	9.67e+01
	Std	9.17e-15	1.45e+01	2.83e-01	3.78e+01	3.78e+01
f_{17}	Median	3.93e+04	1.26e+00	**0.00e+00**	**0.00e+00**	**0.00e+00**
	Mean	3.92e+04	1.24e+00	0.00e+00	0.00e+00	0.00e+00
	Std	2.75e+03	1.25e-01	0.00e+00	0.00e+00	0.00e+00
f_{18}	Median	1.19e+03	1.19e+03	2.51e+03	**5.79e+01**	**5.79e+01**
	Mean	1.17e+03	1.30e+03	3.36e+03	8.63e+01	8.63e+01
	Std	9.66e+01	4.36e+02	1.81e+03	8.76e+01	8.76e+01
f_{19}	Median	1.75e+06	**2.85e+05**	2.81e+06	2.81e+06	2.81e+06
	Mean	1.74e+06	2.85e+05	2.87e+06	2.87e+06	2.87e+06
	Std	9.54e+04	1.78e+04	6.61e+05	6.61e+05	6.61e+05
f_{20}	Median	3.87e+03	1.06e+03	**8.29e+02**	**8.29e+02**	**8.29e+02**
	Mean	4.14e+03	1.07e+03	8.54e+02	8.54e+02	8.54e+02
	Std	8.14e+02	7.29e+01	6.71e+01	6.71e+01	6.71e+01
No. Best		2	1	3	9	9

CC-IG-CMAES and CC-GDG-CMAES have the same convergence curves. For f_{19}, all three algorithms have the same convergence curve.

It can be seen that CC-IG-CMAES and CC-GDG-CMAES converge quickly on f_1, f_5 and f_{11}. This is because there are at least 500 separable variables in these functions, which are divided into low-dimensional subcomponents (at most 20). CMA-ES has been shown to be effective to search in a low-dimensional solution space and converges

Table IX. The best fitness values obtained by the 25 independent runs of the compared algorithms on the CEC'2010 LSGO benchmark functions. For each function, the best algorithm is marked in bold.

Functions	DECC-I	MA-SW-Chains	CMA-ES	CC-IG-CMAES	CC-GDG-CMAES
f_1	3.01e-03	3.18e-15	7.32e+06	**0.00e+00**	**0.00e+00**
f_2	4.00e+03	**7.04e+02**	4.82e+03	1.50e+03	1.50e+03
f_3	1.61e+01	3.34e-13	2.16e+01	**0.00e+00**	2.16e+01
f_4	2.51e+11	3.04e+11	3.61e+13	**3.48e+09**	**3.48e+09**
f_5	8.84e+07	**2.89e+07**	3.55e+07	8.06e+07	8.06e+07
f_6	1.60e+01	8.13e-07	2.17e+01	**0.00e+00**	**0.00e+00**
f_7	7.18e-01	3.35e-03	6.50e+08	**0.00e+00**	**0.00e+00**
f_8	**1.17e+00**	1.54e+06	4.86e+08	1.77e+07	1.77e+07
f_9	3.63e+07	1.19e+07	7.60e+06	**7.90e+02**	**7.90e+02**
f_{10}	4.14e+03	1.81e+03	4.75e+03	**1.64e+03**	**1.64e+03**
f_{11}	**8.50e+00**	2.74e+01	2.38e+02	2.00e+01	2.17e+01
f_{12}	1.02e+03	2.65e-06	**0.00e+00**	**0.00e+00**	**0.00e+00**
f_{13}	4.13e+02	3.86e+02	6.06e+03	**1.27e+02**	**1.27e+02**
f_{14}	3.04e+08	2.72e+07	8.83e+06	**0.00e+00**	**0.00e+00**
f_{15}	5.70e+03	2.48e+03	4.72e+03	**1.85e+03**	**1.85e+03**
f_{16}	**2.27e-13**	8.51e+01	4.32e+02	4.26e+01	4.26e+01
f_{17}	3.47e+04	1.04e+00	**0.00e+00**	**0.00e+00**	**0.00e+00**
f_{18}	9.74e+02	7.83e+02	6.49e+02	**2.38e+01**	**2.38e+01**
f_{19}	1.53e+06	**2.49e+05**	1.62e+06	1.62e+06	1.62e+06
f_{20}	3.20e+03	9.25e+02	**7.72e+02**	**7.72e+02**	**7.72e+02**
No. Best	3	3	3	14	13

quickly. The early convergence allows more fitness evaluations to identify the ideal decomposition if needed in the future. Although CC-GDG-CMAES is outperformed by DECC-I on f_{11}, it is still better than CMA-ES due to the advantage of decomposition. For f_{18}, the 1000 variables are decomposed into 20 50-dimensional nonseparable groups. In this case, the solution space is larger and may make CMA-ES spend more fitness evaluations to converge. For f_{19}, the algorithms do not converge since there is no decomposition and all the 1000 variables are placed in a single nonseparable group. This also implies the importance of decomposition to increase convergence of algorithm, even for the nonseparable functions.

5. CONCLUSION

This paper addresses the two important issues in solving large-scale black-box optimization with decomposition. First, the Global Differential Grouping (GDG) is derived from the differential grouping [Omidvar et al. 2014] to identify an ideal partition of variables. Then, CMA-ES is employed and modified to adapt to the Cooperative Coevolution (CC) framework as a subcomponent solver. The resultant algorithm, called CC-GDG-CMAES, has been demonstrated to outperform the state-of-the-art results on the CEC'2010 LSGO benchmark functions. Additionally, for most of the test functions, ideal partitions have been identified by GDG.

In the future, the assumption of partial additive separability of the objective function, on which many of the theorems in this paper are based, is to be relaxed. Therefore, the functions with other types of separability, or some nonseparable functions with weak interaction matrices such as the Rosenbrock function, cannot be decom-

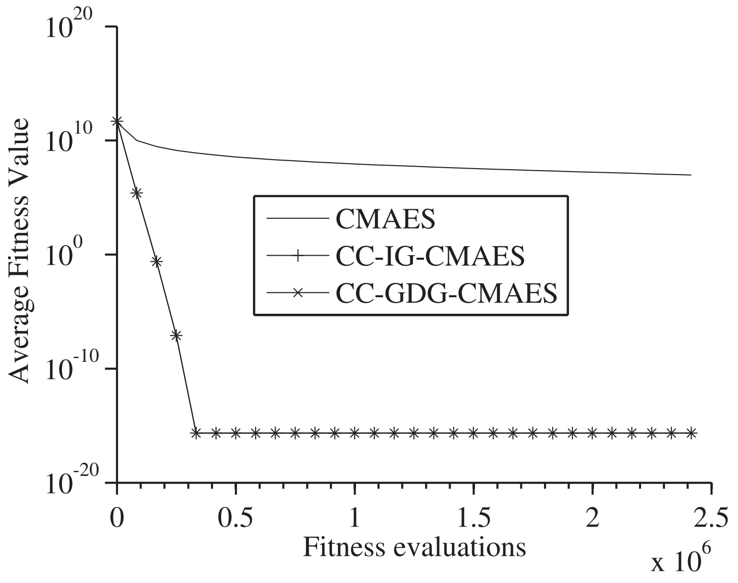

Fig. 3. Convergence curves of the compared algorithms on f_1.

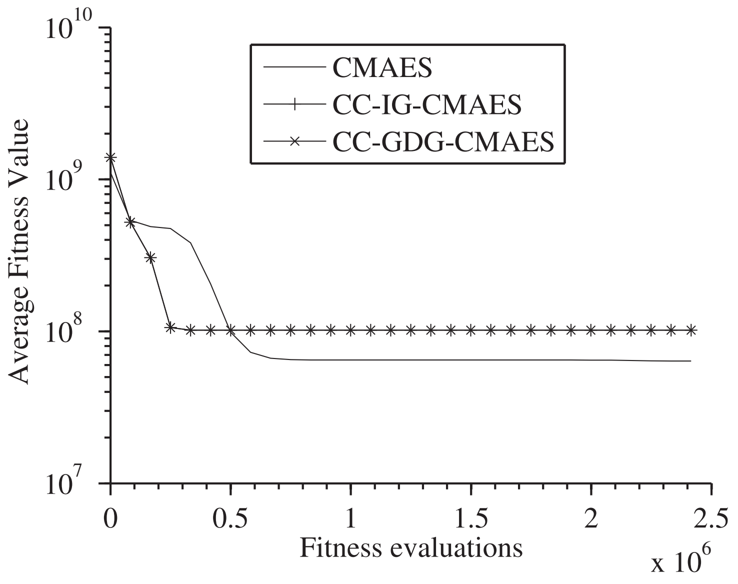

Fig. 4. Convergence curves of the compared algorithms on f_5.

posed, and thus cannot be solved effectively. Effective decomposition methods are to be designed for such nonseparable functions to explore the solution space more effectively. In addition, the strength of the interaction between variables is also to be taken into account to further improve the efficiency of the search, e.g., the variables with stronger interactions are optimized together more often than the ones with weaker interactions.

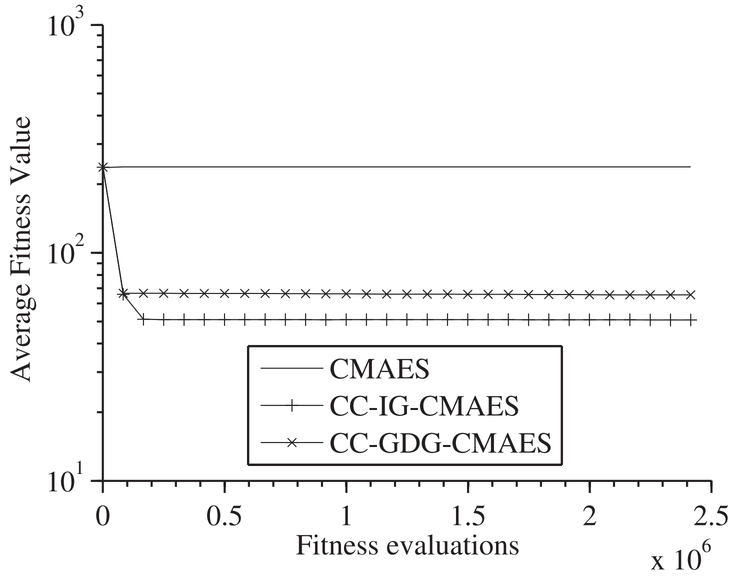

Fig. 5. Convergence curves of the compared algorithms on f_{11}.

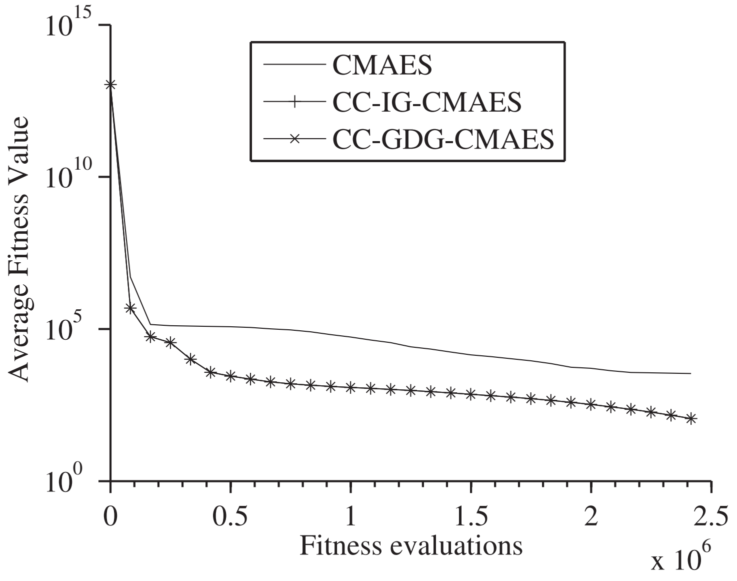

Fig. 6. Convergence curves of the compared algorithms on f_{18}.

APPENDIX

The MATLAB code of this algorithm has been uploaded to MATLAB Central File Exchange for free and easy download and use. The download link is: http://www.mathworks.com/matlabcentral/fileexchange/45783-the-cc-gdg-cmaes-algorithm.

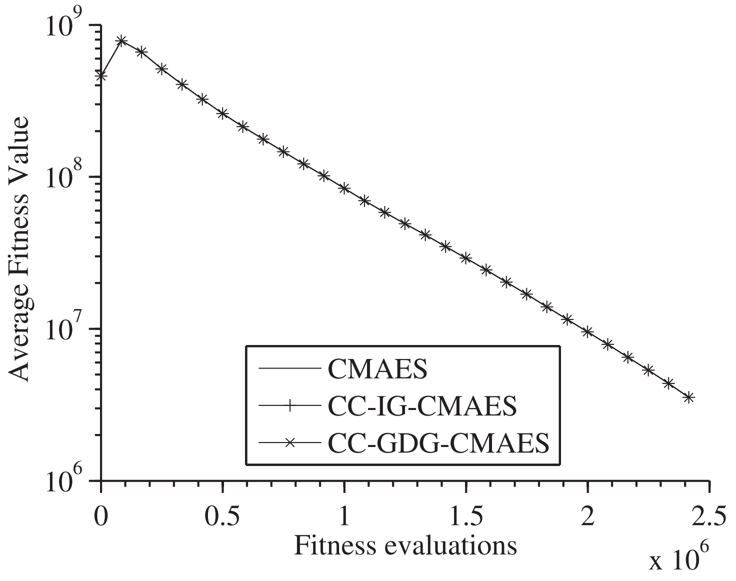

Fig. 7. Convergence curves of the compared algorithms on f_{19}.

ACKNOWLEDGMENTS

This work was supported by an ARC Discovery grant (No. DP120102205) and an EPSRC grant (No. EP/I010297/1). Xin Yao is supported by a Royal Society Wolfson Research Merit Award.

REFERENCES

D. Andrews and Y. Whang. 1990. Additive interactive regression models: circumvention of the curse of dimensionality. *Econometric Theory* 6, 4 (1990), 466–479.

D. Azulay and J. Pique. 2001. A revised simplex method with integer Q-matrices. *ACM Transactions on Mathematical Software (TOMS)* 27, 3 (2001), 350–360.

J.C. Bezdek, R.J. Hathaway, R.E. Howard, C.A. Wilson, and M.P. Windham. 1987. Local convergence analysis of a grouped variable version of coordinate descent. *Journal of Optimization Theory and Applications* 54, 3 (1987), 471–477.

M. Blondel, K. Seki, and K. Uehara. 2013. Block coordinate descent algorithms for large-scale sparse multiclass classification. *Machine learning* 93, 1 (2013), 31–52.

T.R. Browning. 2001. Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *IEEE Transactions on Engineering Management* 48, 3 (2001), 292–306.

W. Chen, T. Weise, Z. Yang, and K. Tang. 2010. Large-Scale Global Optimization Using Cooperative Coevolution with Variable Interaction Learning. *Parallel Problem Solving from Nature, PPSN XI* (2010), 300–309.

A.L. Custódio and L.N. Vicente. 2007. Using sampling and simplex derivatives in pattern search methods. *SIAM Journal on Optimization* 18, 2 (2007), 537–555.

G.B. Dantzig and M.N. Thapa. 1997. *Linear programming: 1: Introduction*. Vol. 1. Springer.

R. Eberhart and J. Kennedy. 1995. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*. IEEE, 39–43.

J. Friedenberg and G. Silverman. 2006. Mind as a Black Box: The Behaviorist Approach. In *Cognitive science: An introduction to the study of mind*. Sage, 85–88.

J.H. Friedman and B.W. Silverman. 1989. Flexible parsimonious smoothing and additive modeling. *Technometrics* 31, 1 (1989), 3–21.

W.W. Hager and H. Zhang. 2006. Algorithm 851: CG_DESCENT, a conjugate gradient method with guaranteed descent. *ACM Transactions on Mathematical Software (TOMS)* 32, 1 (2006), 113–137.

N. Hansen. 2011. *The CMA evolution strategy: A tutorial*. Technical Report.

N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík. 2010. Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*. ACM, 1689–1696.

N. Hansen and A. Ostermeier. 1996. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*. IEEE, 312–317.

G.R. Harik. 1997. *Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms*. Ph.D. Dissertation. The University of Michigan, Ann Arbor, MI, USA.

J.H. Holland. 1975. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* The University of Michigan Press.

R. Hooke and T.A. Jeeves. 1961. "Direct search" solution of numerical and statistical problems. *Journal of the ACM (JACM)* 8, 2 (1961), 212–229.

J.E. Hopcroft and R.E. Tarjan. 1973. Efficient algorithms for graph manipulation. *Commun. ACM* 16, 6 (1973), 372–378.

P. Larrañaga and J.A. Lozano. 2002. *Estimation of distribution algorithms: A new tool for evolutionary computation*. Vol. 2. Springer.

G. Li, C. Rosenthal, and H. Rabitz. 2001. High dimensional model representations. *The Journal of Physical Chemistry A* 105, 33 (2001), 7765–7777.

X. Li and X. Yao. 2012. Cooperatively coevolving particle swarms for large scale optimization. *Evolutionary Computation, IEEE Transactions on* 16, 2 (2012), 210–224.

Y. Liu, X. Yao, Q. Zhao, and T. Higuchi. 2001. Scaling up fast evolutionary programming with cooperative coevolution. In *Proceedings of the 2001 Congress on Evolutionary Computation*, Vol. 2. IEEE, 1101–1108.

I. Loshchilov, M. Schoenauer, and M. Sebag. 2011. Adaptive coordinate descent. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 885–892.

Y. Mei, X. Li, and X. Yao. 2014. Cooperative Co-evolution with Route Distance Grouping for Large-Scale Capacitated Arc Routing Problems. *IEEE Transactions on Evolutionary Computation* 18, 3 (2014), 435–449.

D. Molina, M. Lozano, and F. Herrera. 2010. MA-SW-Chains: Memetic algorithm based on local search chains for large scale continuous global optimization. In *2010 IEEE Congress on, Evolutionary Computation (CEC)*. IEEE, 1–8.

J.A. Nelder and R. Mead. 1965. A simplex method for function minimization. *The computer journal* 7, 4 (1965), 308–313.

M.N. Omidvar and X. Li. 2010. A comparative study of cma-es on large scale global optimisation. In *AI 2010: Advances in Artificial Intelligence*. Springer, 303–312.

M.N. Omidvar, X. Li, Y. Mei, and X. Yao. 2014. Cooperative Co-evolution with Differential Grouping for Large Scale Optimization. *IEEE Transactions on Evolutionary Computation* 18, 3 (2014), 378–393.

M.N. Omidvar, X. Li, Z. Yang, and X. Yao. 2010. Cooperative co-evolution for large scale optimization through more frequent random grouping. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation*. 1–8.

M.N. Omidvar, X. Li, and X. Yao. 2010. Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation*. 1762–1769.

M.N. Omidvar, X. Li, and X. Yao. 2011. Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 1115–1122.

M.N. Omidvar, Y. Mei, and X. Li. 2014. Optimal Decomposition of Large-Scale Separable Continuous Functions for Cooperative Co-evolutionary Algorithms. In *Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC2014)*. IEEE.

M.A. Potter and K. De Jong. 1994. A cooperative coevolutionary approach to function optimization. *Parallel Problem Solving from Nature (PPSN)* (1994), 249–257.

P. Richtárik and M. Takáč. 2012. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming* (2012), 1–38.

L.M. Rios and N.V. Sahinidis. 2012. Derivative-free optimization: A review of algorithms and comparison of software implementations. *Journal of Global Optimization* (2012), 1–47.

R. Ros and N. Hansen. 2008. A simple modification in CMA-ES achieving linear time and space complexity. In *Parallel Problem Solving from Nature–PPSN X*. Springer, 296–305.

H.H. Rosenbrock. 1960. An Automatic Method for Finding the Greatest or Least Value of a Function. *Comput. J.* 3, 3 (1960), 175–184.

S. Shan and G. Wang. 2010. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization* 41, 2 (2010), 219–241.

D.J. Sheskin. 2003. *Handbook of parametric and nonparametric statistical procedures*. crc Press.

Y. Shi, H. Teng, and Z. Li. 2005. Cooperative co-evolutionary differential evolution for function optimization. In *Proceedings of the First international conference on Advances in Natural Computation - Volume Part II*. Springer-Verlag, 1080–1088.

J. Smith and T.C. Fogarty. 1995. An adaptive poly-parental recombination strategy. In *Evolutionary Computing*. Springer, 48–61.

C.J. Stone. 1985. Additive regression and other nonparametric models. *The annals of Statistics* (1985), 689–705.

K. Tang, X. Li, P.N. Suganthan, Z. Yang, and T. Weise. 2009. *Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization*. Technical Report. Nature Inspired Computation and Applications Laboratory, USTC, China. http://nical.ustc.edu.cn/cec10ss.php.

P. Tseng. 2001. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications* 109, 3 (2001), 475–494.

F. Van den Bergh and A.P. Engelbrecht. 2004. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation* 8, 3 (2004), 225–239.

J. Weglarz, J. Blazewicz, W. Cellary, and R. Slowinski. 1977. Algorithm 520: An Automatic Revised Simplex Method for Constrained Resource Network Scheduling [H]. *ACM Transactions on Mathematical Software (TOMS)* 3, 3 (1977), 295–300.

K. Weicker and N. Weicker. 1999. On the improvement of coevolutionary optimizers by learning variable interdependencies. In *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*. IEEE.

F. Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin* 1, 6 (1945), 80–83.

Z. Yang, K. Tang, and X. Yao. 2008. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences* 178, 15 (2008), 2985–2999.

X. Yao, Y. Liu, and G. Lin. 1999. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation* 3, 2 (1999), 82–102.

C. Zhu, R.H. Byrd, P. Lu, and J. Nocedal. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)* 23, 4 (1997), 550–560.