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

YI MEI, 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 unconstrained large scale black-box optimization, which has an unavailable algebraic model for the problem and thousands of decision variables. Assuming that the overall problem is composed of a number of smaller independent sub-problems, the proposed algorithm addresses two important issues in solving large scale black-box optimization: (1) identifying the independent sub-problems and (2) solving the identified black-box sub-problems. First, a Global Differential Grouping (GDG) method is proposed based upon the differential grouping method to find near optimal decomposition of the decision variables and the corresponding sub-problems. Then, a variant of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is adopted to solve the nonseparable sub-problems due to 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, GDG manages to obtain an ideal decomposition of the variables, and CC-GDG-CMAES outperformed the state-of-the-art results. Besides, 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. Competitive Divide-and-Conquer Algorithm for Unconstrained Large Scale Black-Box Optimization *ACM Trans. Math. Softw.* V, N, Article A (January YYYY), 28 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. It is often inevitable to encounter black-boxes in many research and industrial fields. For example, in cognitive science, the human mind can be seen as a black box ([Friedenberg

This work was supported by an ARC Discovery grant (No. DP120102205), an NSFC grant (No. 61329302), and an EPSRC grant (No. EP/I010297/1). Xin Yao was supported by a Royal Society Wolfson Research Merit Award.
Author's addresses: Y. Mei, 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

and Silverman 2006], pp. 85–88, "Mind as a Black Box: The Behaviorist Approach"). 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 computing area, a program may have to be considered as a black box if it is a closed source program and the users have no access to its inner workings. In artificial intelligence, heuristic algorithms such as artificial neural network and random search algorithms are often referred to as black boxes because their working mechanisms are so complicated that it is hardly possible to completely analyze them.

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 spread 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, the 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 Quasi-Newton method [Zhu et al. 1997] and 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 this case, derivative-free algorithms are promising methods for solving black-box optimization problems as they take only the input and output of function into account. There have been plenty of derivative-free algorithms developed in literature, including various local search algorithms (e.g., direct search method [Hooke and Jeeves 1961], Nelder-Mead simplex method [Nelder and Mead 1965] [Tseng 1999], generalized pattern search [Torczon 1997], mesh adaptive direct search methods [Audet and Dennis Jr 2006], pattern search methods using simplex gradients [Custódio and Vicente 2007], implicit filtering method [Gilmore and Kelley 1995] and trust-region methods [Conn et al. 1997]) and global search algorithms (e.g., divide a hyperrectangle algorithm [Jones 2001], multilevel coordinate search [Huyer and Neumaier 1999], efficient global optimization [Jones et al. 1998], simulated annealing [Kirkpatrick et al. 1983], genetic algorithms [Holland 1975], Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [Hansen 2006], particle swarm optimization [Eberhart and Kennedy 1995], tabu search [Glover et al. 2000], differential evolution [Storn and Price 1997], fast evolutionary programming [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 solution space increases exponentially. For example, in a binary optimization problem, when the problem size increases from 10 to 20, the size of 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 solve 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. 2013] is a generalized framework of the coordinate descent and block coordinate descent methods. It keeps a set of candidates for each coordinate or block instead of the single best-so-far value to relax the way of selecting the collaborative values. With divide-and-conquer strategies, the decision variables are decomposed into smaller subsets, which are called *subcomponents*, and optimized separately. This way, the solution space is much reduced and can be explored more effectively. In this paper, we focus on the direct decomposition methods without model approximation.

When solving large scale black-box optimization problems with a divide-and-conquer strategy, two major issues should be addressed: (1) Identifying the optimal decomposition of decision variables and (2) selecting competent optimizers for the nonseparable subcomponents. The former ensures that optimizing the subcomponents separately is equivalent to optimizing the overall decision vector. For example, assume that we are dealing with an n-dimensional minimization problem $\min f(\mathbf{x}) = f(x_1, \ldots, x_n) = \sum_{i=1}^{g} f_i(\mathbf{x}_i)$, where $(\mathbf{x}_1, \ldots, \mathbf{x}_g)$ is a partition of the decision vector $\mathbf{x}$ (g is the number of subcomponents), i.e., $\cup_{i=1}^{g} \mathbf{x}_i = \mathbf{x}$ and $\mathbf{x}_i \cap \mathbf{x}_j = \emptyset$, $\forall j \neq i$. It is obvious that minimizing $f(\mathbf{x})$ is equivalent to minimizing $f_i(\mathbf{x}_i)$'s ($\forall i = 1, \ldots, g$) simultaneously. To guarantee the optimality of the final solution, it is necessary to find the decomposition $(\mathbf{x}_1, \ldots, \mathbf{x}_g)$. This task is trivial if the formula of the objective function is known. In black-box optimization, nevertheless, it is often challenging to find the optimal decomposition due to the black-box nature of the objective function. The latter (selecting a competent optimizer) enhances the capability of the algorithm to obtain optimal solution for each subcomponent.

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. 2013] has been demonstrated to have a high accuracy of grouping interacting decision variables together without knowing any information of 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 algorithm for nonseparable black-box optimization problems [Rios and Sahinidis 2012]. It conducts a rotation-invariant search which is independent from 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 20) 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 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. 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*, 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 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

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

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) is 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 $\lambda_i \leftarrow \lambda_i - \mu_k g_i(\mathbf{x}_k^*)$ 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 straightforwardly. Thus, we focus on solving unconstrained

black-box optimization problems, which are stated as follows:

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

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

In our work, large scale black-box optimization is a special case of black-box optimization, where the number of decision variables n is large (e.g., $n \geq 1000$).

2.2. Divide-and-Conquer Strategies

A divide-and-conquer strategy decomposes the original large set of decision variables into smaller subcomponents and optimizes them separately. It typically includes the following two tasks: (1) Decompose the decision variables to obtain the subcomponents and (2) coordinate the optimization of the subcomponents. First, there are two types of decompositions depending on different relationships between subcomponents [Shan and Wang 2010]. One is called the *ideal decomposition*, which decomposes the decision variable into independent subcomponents without any interaction between them (i.e., no variable belong to more than one subcomponent). The other one is called the *coordination-based decomposition*, which leads to subcomponents sharing common variables due to the interaction between them. In this paper, we focus on the simpler former case as a starting point of the investigation in the challenging area of large scale black-box optimization.

With regard to the effectiveness, we formally define *ideal decomposition* as follows:

Definition 2.1 (*Ideal Decomposition*). Given a function $f(\mathbf{x})$ and a non-overlapping decomposition $\{\mathbf{x}_1, \ldots, \mathbf{x}_g\}$ of the decision vector $\mathbf{x}$ (g is the number of subcomponents) so that $\cup_{i=1}^{g} \mathbf{x}_i = \mathbf{x}$ and $\mathbf{x}_i \cap \mathbf{x}_j = \emptyset$ ($\forall i \neq j$), $\{\mathbf{x}_1, \ldots, \mathbf{x}_g\}$ is called the *ideal decomposition* of $\mathbf{x}$ if there exist functions $f_1, \ldots, f_g$ of $\mathbf{x}_1, \ldots, \mathbf{x}_g$, which are called the *ideal component functions*, so that the optimum (minimum in this case) $\mathbf{x}^* = \arg\min\{f(\mathbf{x})\}$ of the function f is the combination of the optima $\mathbf{x}_i^* = \arg\min\{f_i(\mathbf{x}_i)\}$ of the functions f_i ($i = 1, \ldots, g$), denoted as $\mathbf{x}^* = (\mathbf{x}_1^*, \ldots, \mathbf{x}_g^*)$.

According to the above definition, it is obvious that if $\{\mathbf{x}_1, \ldots, \mathbf{x}_g\}$ is the ideal decomposition of $\mathbf{x}$ and the corresponding functions $f_1, \ldots, f_g$ have been found, then solving the large scale problem $\min f(\mathbf{x})$ is equivalent to solving the following series of smaller-sized sub-problems: $\min f_i(\mathbf{x}_i)$, ($\forall i = 1, \ldots, g$). In this case, the function $f(\mathbf{x})$ is called a *separable function*. The formal definition of separability of function is given as follows:

Definition 2.2 (*Separability*). Given a function $f(\mathbf{x})$, if there exists at least an ideal decomposition $\{\mathbf{x}_1, \ldots, \mathbf{x}_g\}$ (defined in Definition 2.1) along with the corresponding functions $f_1, \ldots, f_g$, then $f(\mathbf{x})$ is called a *partially separable* function. If there exists an ideal single-variable decomposition $(\mathbf{x}_1, \ldots, \mathbf{x}_g) = (x_1, \ldots, x_n)$, then $f(\mathbf{x})$ is called a *fully separable* function. If no ideal decomposition exists, then $f(\mathbf{x})$ is called a *non-separable* function. A class of separable functions is the so-called *additively separable* functions, which can be written as $f(\mathbf{x}) = \sum_{i=1}^{g} f_i(\mathbf{x}_i)$.

Given a large scale black-box optimization problem, its decomposition consists of two tasks. One is to find the ideal decomposition $(\mathbf{x}_1, \ldots, \mathbf{x}_g)$ of $\mathbf{x}$, and the other is to identify the corresponding functions $f_1(\mathbf{x}_1), \ldots, f_g(\mathbf{x}_g)$. Since $f(\mathbf{x})$ is a black box, it is impossible to exactly obtain $f_1(\mathbf{x}_1), \ldots, f_g(\mathbf{x}_g)$. Cooperative co-evolution is a suitable framework to address this issue.

A typical cooperative co-evolution framework divides the entire optimization process into several *cycles*. At each cycle, the subcomponents are evolved in a round-robin fashion. Given the g subcomponents $(\mathbf{x}_1, \ldots, \mathbf{x}_g)$, a sub-population $\{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{iN}\}$ is maintained for $\mathbf{x}_i$ ($\forall i = 1, \ldots, g$, N is the size of the sub-population), where each $\mathbf{x}_{ij}$ is called

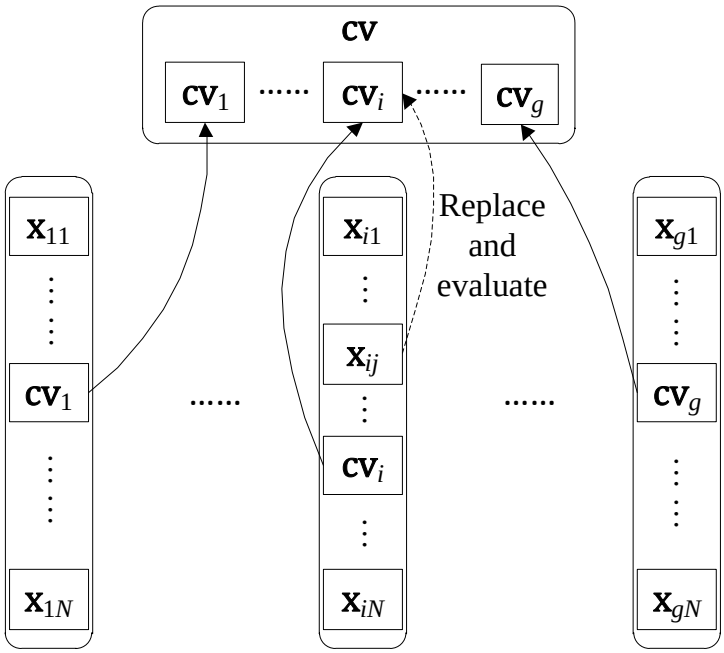

Fig. 1. Sub-populations and the context vector $\mathbf{cv}$ of the cooperative co-evolution framework, where $\mathbf{cv}_i$ is the best-fit individual in the sub-population $\{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{iN}\}$. When evaluating $\mathbf{x}_{ij}$, it first replaces $\mathbf{cv}_i$ in $\mathbf{cv}$. Then, its fitness value is set to $fit(\mathbf{x}_{ij}) = f(\mathbf{cv}_1, \ldots, \mathbf{cv}_{i-1}, \mathbf{x}_{ij}, \mathbf{cv}_{i+1}, \ldots, \mathbf{cv}_g)$

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 $\mathbf{x}_{ij}$, a temporary vector $\mathbf{v}_{ij} = (\mathbf{cv}_1, \ldots \mathbf{cv}_{i-1}, \mathbf{x}_{ij}, \mathbf{cv}_{i+1}, \ldots, \mathbf{cv}_g)$ is first obtained by replacing $\mathbf{cv}_i \in \mathbf{cv}$ with $\mathbf{x}_{ij}$. Then, the fitness value of $\mathbf{x}_{ij}$ is assigned to the objective value of $\mathbf{v}_{ij}$ ($fit(\mathbf{x}_{ij}) = f(\mathbf{v}_{ij})$). The context vector is randomly initialized and then updated after each cycle. A commonly used way 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.

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 divides the decision variables in their natural order before the optimization and no longer change the grouping throughout the optimization process. In this case, the ideal decomposition of the decision variables can hardly be obtained 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 grouping at the beginning of each cycle. Suppose that in each cycle, there is a probability p $(0 < p < 1)$ of obtaining the ideal decomposition, then the probability of obtaining the ideal decomposition 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 1000–2000 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. 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] and delta grouping [Omidvar et al. 2010]).

Although the proposed dynamic grouping methods significantly increase the chance of obtaining the ideal decomposition, such probability is still low, especially for the problems whose ideal decomposition has complicated structure (e.g., consisting of many imbalanced subcomponents). In addition, the computational resource assigned to the optimization under the ideal decomposition of the decision variables is not adequate to reach the global optimum, or at least a good local optimum. Recently, Omidvar *et al.* [Omidvar et al. 2013] 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, and fixes the grouping throughout the optimization. Differential grouping has been empirically verified to be the state-of-the-art decomposition method based on the results obtaind from the CEC'2010 large scale global optimization (LSGO) benchmark functions [Omidvar et al. 2013].

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

The advantage of cooperative co-evolution is that if the objective function is additively separable and the ideal decomposition can be found by some grouping method, then the global optimum can be guaranteed by optimizing the subcomponents separately without identifying the corresponding ideal component functions $f_i(\mathbf{x}_i)$ $(i = 1, \ldots, g)$. First, we give Lemma 3.1 as follows:

LEMMA 3.1. *If* $f(\mathbf{x}) = \sum_{i=1}^{g} f_i(\mathbf{x}_i)$*, then* $\forall\ i = 1, \ldots, g$*, we have*

$$\arg\min_{\mathbf{x}_i}\{f_i(\mathbf{x}_i)\} = \arg\min_{\mathbf{x}_i}\{f(\mathbf{a}_1, \ldots, \mathbf{a}_{i-1}, \mathbf{x}_i, \mathbf{a}_{i+1}, \ldots, \mathbf{a}_g)\}, \forall\ \mathbf{a}_j \in \Omega(\mathbf{x}_j), j \neq i,$$

where $\Omega(\mathbf{x}_j)$ *stands for the solution space of* $\mathbf{x}_j$*.*

PROOF. Let $\mathbf{x}_i^* = \arg\min_{\mathbf{x}_i}\{f(\mathbf{a}_1, \ldots, \mathbf{a}_{i-1}, \mathbf{x}_i, \mathbf{a}_{i+1}, \ldots, \mathbf{a}_g)\}$. Then we have

$$f(\mathbf{a}_1, \ldots, \mathbf{a}_{i-1}, \mathbf{x}_i^*, \mathbf{a}_{i+1}, \ldots, \mathbf{a}_g) \leq f(\mathbf{a}_1, \ldots, \mathbf{a}_{i-1}, \mathbf{x}_i, \mathbf{a}_{i+1}, \ldots, \mathbf{a}_g),\ \forall\ \mathbf{x}_i \neq \mathbf{x}_i^*.$$

Since $f(\mathbf{x}) = \sum_{i=1}^{g} f_i(\mathbf{x}_i)$, we have

$$\begin{aligned} f_1(\mathbf{a}_1) + \cdots + f_{i-1}(\mathbf{a}_{i-1}) + f_i(\mathbf{x}_i^*) + f_{i+1}(\mathbf{a}_{i+1}) + \cdots + f_g(\mathbf{a}_g) \leq \\ f_1(\mathbf{a}_1) + \cdots + f_{i-1}(\mathbf{a}_{i-1}) + f_i(\mathbf{x}_i) + f_{i+1}(\mathbf{a}_{i+1}) + \cdots + f_g(\mathbf{a}_g), \forall\ \mathbf{x}_i \neq \mathbf{x}_i^* \\ \Longrightarrow f_i(\mathbf{x}_i^*) \leq f_i(\mathbf{x}_i), \forall\ \mathbf{x}_i \neq \mathbf{x}_i^* \\ \Longrightarrow \mathbf{x}_i^* = \arg\min_{\mathbf{x}_i}\{f_i(\mathbf{x}_i)\}. \end{aligned}$$

□

Lemma 3.1 implies that optimizing the ideal component function $f_i(\mathbf{x}_i)$ is equivalent to optimizing $f(\mathbf{x})$ by setting all the other $\mathbf{x}_j$'s to any fixed vectors $\mathbf{a}_j$'s. Based on Lemma 3.1, we have Theorem 3.2.

THEOREM 3.2. *If* $f(\mathbf{x}) = \sum_{i=1}^{g} f_i(\mathbf{x}_i)$*, where* $(\mathbf{x}_1, \ldots, \mathbf{x}_g)$ *is the ideal decomposition and* g *is the number of subcomponents, then the cooperative co-evolution framework that optimizes the subcomponents in terms of* $f_i^{(t)}(\mathbf{x}_i) = f\big(\mathbf{cv}_1^{(t-1)}, \ldots, \mathbf{cv}_{i-1}^{(t-1)}, \mathbf{x}_i, \mathbf{cv}_{i+1}^{(t-1)}, \ldots, \mathbf{cv}_n^{(t-1)}\big)$ *must converge to the global optimum* $\mathbf{x}^* = \arg\min\{f(\mathbf{x})\}$*, given that each subcomponent is converged to its own global optimum* $\mathbf{x}_i^{*(t)} = \arg\min\{f_i^{(t)}(\mathbf{x}_i)\}$.

PROOF. Based on Lemma 3.1, by setting $\mathbf{a}_i = \mathbf{cv}_i^{(t-1)}$, we have

$$\arg\min_{\mathbf{x}_i}\{f_i(\mathbf{x}_i)\} = \arg\min_{\mathbf{x}_i}\{f_i^{(t)}(\mathbf{x}_i)\}, \forall\, t \in \mathbb{N}^+,\ i = 1, \ldots, g.$$

That is, $\mathbf{x}_i^* = \mathbf{x}_i^{*(t)}, \forall\, t \in \mathbb{N}^+,\ i = 1, \ldots, g$. Then,

$$\mathbf{x}^* = (\mathbf{x}_1^*, \ldots, \mathbf{x}_g^*) = (\mathbf{x}_1^{*(t)}, \ldots, \mathbf{x}_g^{*(t)}).$$

This means the global optimum $\mathbf{x}^*$ can be obtained by combining the global optima $\mathbf{x}_i^{*(t)}$ ($\forall\, i = 1, \ldots, g$) obtained by the cooperative co-evolution framework. □

Theorem 3.2 indicates that if the ideal decomposition is identified and the global optimum for each subcomponent in the ideal decomposition is obtained, the optimality of the cooperative co-evolution framework is guaranteed. In our work, a Global Differential Grouping (GDG) is proposed to identify the ideal decomposition, and a variant of CMA-ES is designed to find the global optimum of each subcomponent.

The proposed algorithm, named CC-GDG-CMAES, is described in Alg. 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 $\overline{\mathbf{x}} = (\overline{x}_1, \ldots, \overline{x}_n)$ and $\underline{\mathbf{x}} = (\underline{x}_1, \ldots, \underline{x}_n)$ 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 grouping phase, initialization phase, and optimization phase. During the grouping phase (line 1), the decomposition $(\mathbf{x}_1, \ldots, \mathbf{x}_g)$ is obtained by `GDG()` along with the number of subcomponents g and the number of fitness evaluations Γ that has been consumed. It is then fixed throughout the subsequent phases. During the initialization phase (lines 2–7), the parameters $\mathcal{A}$ of CMA-ES are initialized. The initialization of $\mathcal{A}$ will be described in details in Section 3.2. Then, the context vector $\mathbf{cv} = (\mathbf{cv}_1, \ldots, \mathbf{cv}_g)$ is randomly initialized. The optimization phase (lines 8–12) is divided into a number of cycles. Within each cycle, CMA-ES is applied to the subcomponents sequentially, and the corresponding dimensions of $\mathbf{cv}$ is replaced by the latest best-fit individual (line 10). The CMA-ES parameters $\mathcal{A}$ and number of fitness evaluations Γ are updated as well. The algorithm stops when the total number of fitness evaluations exceeds the maximal number of fitness evaluation Γ_{max}. Finally, the best-so-far solution $\mathbf{x}^*$ is set to $\mathbf{cv}$ and its objective value f^* is obtained.

Next, the details of GDG will be given in Section 3.1. Then, CMA-ES components, including the initialization of parameters $\mathcal{A}$ and the function `CMAES()` will be described in details 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. 2013], which groups the variables based on the in-

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

```
(g, x_1, ..., x_g, Γ) = GDG(f, x, n, x̄, x, ε);
Initialize the CMA-ES parameters A;
for i = 1 → g do
    Γ_i = 0;
    cv_i = randinit(A, x_i);
end
cv = (cv_1, ..., cv_g);
while Σ_{i=1}^{g} Γ_i < Γ_max − Γ do
    for i = 1 → g do
        (cv, A, Γ_i) = CMAES(cv, x_i, A, Γ_i);
    end
end
x* = cv, f* = f(x*);
return (x*, f*);
```

teraction between them. To facilitate the description of GDG, two complementary relationships between variables are first defined in Definitions 3.3 and 3.4.

Definition 3.3 (*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 3.4 (*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 3.5 [Omidvar et al. 2013].

THEOREM 3.5. *If $f(\mathbf{x})$ is a partially additively separable function, and $\forall a, b_1 \neq b_2, \delta \in \mathbb{R}, \delta \neq 0$, the following condition holds*

$$\Delta_{\delta, x_p}[f](\mathbf{x})|_{x_p=a, x_q=b_1} \neq \Delta_{\delta, x_p}[f](\mathbf{x})|_{x_p=a, x_q=b_2}, \tag{6}$$

where

$$\Delta_{\delta, x_p}[f](\mathbf{x}) = f(\ldots, x_p + \delta, \ldots) - f(\ldots, x_p, \ldots) \tag{7}$$

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 3.5, one can detect the interaction between any two different variables x_i and x_j by checking whether the condition Eq. (6) holds. In Alg. 2, this is done by setting $a = \underline{x}_i$, $b_1 = \underline{x}_j$, $b_2 = 0$, $\delta = \overline{x}_i - \underline{x}_i$ and $x_k = \underline{x}_k$ for all the other variables $x_k \in \{x_1, \ldots, x_n\} \setminus \{x_i, x_j\}$. 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 placed in the same subcomponent. If a variable is independent from all the other variables, then it is the only variable in its subcom-

ALGORITHM 2: $(g, \mathbf{x}_1, \ldots, \mathbf{x}_g, \mathbf{x}_{sep}, \Gamma) = \mathtt{DG}(f, \mathbf{x}, n, \overline{\mathbf{x}}, \underline{\mathbf{x}}, \epsilon)$

```
1  D = {1, ..., n}, x_sep = {}, g = 0, Γ = 0;
2  for i ∈ D do
3      y = (x_i);
4      for j ∈ D ∧ i ≠ j do
5          p_1 = x_, p_2 = p_1, p_2(i) = x̄_i;
6          Δ_1 = f(p_1) − f(p_2);
7          p_1(j) = 0, p_2(j) = 0;
8          Δ_2 = f(p_1) − f(p_2);
9          Γ ← Γ + 4;
10         if |Δ_1 − Δ_2| > ε then
11             y ← (y, x_j), D ← D \ {j};
12         end
13     end
14     if |y| = 1 then
15         x_sep ← (x_sep, y);
16     else
17         g ← g + 1;
18         x_g = y;
19     end
20 end
21 return (g, x_1, ..., x_g, x_sep, Γ);
```

ponent and the cardinality of the subcomponent ($|\mathbf{y}|$ in line 14) is 1. All such separable variables are placed in a separable subcomponent $\mathbf{x}_{sep}$.

Although Alg. 2 has been demonstrated to be effective in placing the interacting variables in the same subcomponent 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 completed and many interactions between variables may be missed. For example, suppose that there are three variables (x_1, x_2, x_3). 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 Alg. 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 decomposition will be $\mathbf{x}_1 = (x_1, x_2)$ and $\mathbf{x}_2 = (x_3)$, which is different from the ideal decomposition $\mathbf{x}_1 = (x_1, x_2, x_3)$. This phenomenon happens on the Rosenbrock function [Rosenbrock 1960], where each variable interacts with at most two of the other variables. Second, 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. 2013], and three values of 10^{-1}, 10^{-3} and 10^{-6} are empirically compared. The results show that the best ϵ value is different for different test functions, which indicates that ϵ needs to be set adaptively as per the test function rather than fixed. Third, it cannot deal with fully separable functions or the functions with large portion of separable variables. It only places all the separable variables in the subcomponent $\mathbf{x}_{sep}$, but does not further decompose them. As a result, it cannot optimize the subcomponent of separable variables $\mathbf{x}_{sep}$ effectively. For example, it optimizes all the variables of the fully separable functions without any decomposition. It is 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 straightforwardly 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 $y = x_1x_2 + x_1x_4 + x_2x_4 + x_3x_5x_6 + x_5x_6x_7$ with unique lower bound of -1 and upper bound of 1 for all the variables, the Λ matrix obtained by GDG is shown in Eq. (8). With a sufficiently small ϵ, the Θ matrix is then shown in Eq. (9). Considering Eq. (9) 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).

$$\Lambda = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \\ \begin{pmatrix} 0 & 2 & 0 & 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 2 & 0 \\ 2 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 4 & 2 \\ 0 & 0 & 2 & 0 & 4 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 0 \end{pmatrix} \end{array}, \tag{8}$$

$$\Theta = \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}. \tag{9}$$

It should be noted that the design structure matrix is necessary for the functions with overlapping interactions, in which different subcomponents share common variables. This type of functions is more general in practice, and more challenging. Given the design structure matrix, various decompositions can be devised in order to deal with overlapping subcomponents in future.

To address the second issue, an intuitive method is used to approximately 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, the value of ϵ is set as follows:

$$\epsilon = \alpha \cdot \min\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_K)\}, \tag{10}$$

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

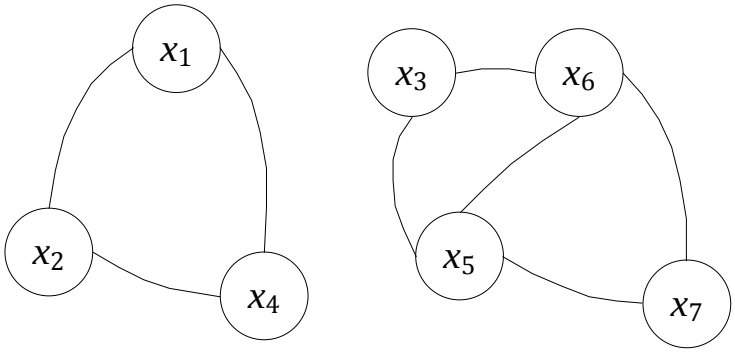

Fig. 2. The graph corresponding to the adjacency matrix Eq. (9).

Finally, a simple decomposition scheme is used to address the third issue. It is known that if the original decomposition is ideal [Omidvar et al. 2014], then any further decomposition of the separable subcomponent $\mathbf{x}_{sep}$ will lead to another ideal decomposition. Here, the separable subcomponent is further decomposed according to a rule of thumb for selecting a subcomponent size, which is to choose it small enough so that it is within the capacity of the subcomponent optimizer, but it should not be made any smaller [Omidvar et al. 2014]. Based on the previous empirical studies [Hansen et al. 2010], CMA-ES performs well for the functions with dimension of up to 20. 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]. Therefore, if the separable subcomponent is still large, it is further decomposed into a number of 20-dimensional or smaller subcomponents. For example, if the separable subcomponent consists of 950 variables, then it is decomposed into 47 20-dimensional subcomponents plus a 10-dimensional subcomponent.

The pseudocode of GDG is described in Alg. 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 subcomponents $\mathbf{y}_1, \ldots, \mathbf{y}_k$ are 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 subcomponent, since it is an isolated node in the graph. Then, all the separable variables are combined into $\mathbf{x}_{sep}$ (lines 24 – 31) and further divided into 20-dimensional or smaller subcomponents (lines 32 – 40). It can be seen that GDG takes a fixed number of fitness evaluations $\Gamma = 1 + 2n + n(n-1)/2 = (n^2 + 3n + 2)/2$.

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].

The minimalistic implementation of CMA-ES (`purecmaes.m`) given in the official website [1] is adopted here. It is referred to as pure CMA-ES hereafter. Its parameters include $\langle \mathbf{x} \rangle_{\mathbf{w}}$, $\boldsymbol{\sigma}$, λ, μ, $\mathbf{w}$, μ_{eff}, c_σ, c_1, c_μ c_c, $\mathbf{B}$, $\mathbf{D}$, $\mathbf{C}$, $\mathbf{p}_\sigma$, $\mathbf{p}_c$ and d_σ. Since there are multiple subcomponents to be optimized in CC-GDG-CMAES, there

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

ALGORITHM 3: $(g, \mathbf{x}_1, \ldots, \mathbf{x}_g, \Gamma) = \texttt{GDG}(f, \mathbf{x}, n, \overline{\mathbf{x}}, \underline{\mathbf{x}}, \epsilon)$

$\mathbf{x}_1 = \underline{\mathbf{x}}$, $F_1 = f(\mathbf{x}_1)$, $\Gamma = 1$;
$\mathbf{F}_2 = \mathbf{0}_{1\times n}$, $\mathbf{F}_3 = \mathbf{0}_{1\times n}$, $\mathbf{F}_4 = \mathbf{0}_{n\times n}$;
for $i = 1 \rightarrow n$ **do**
    $\mathbf{x}_2 = \mathbf{x}_1$, $\mathbf{x}_2(i) = \overline{x}_i$, $\mathbf{F}_2(i) = f(\mathbf{x}_2)$, $\Gamma \leftarrow \Gamma + 1$;
    $\mathbf{x}_3 = \mathbf{x}_1$, $\mathbf{x}_3(i) = 0$, $\mathbf{F}_3(i) = f(\mathbf{x}_3)$, $\Gamma \leftarrow \Gamma + 1$;
end
for $i = 1 \rightarrow n - 1$ **do**
    for $j = i + 1 \rightarrow n$ **do**
        $\mathbf{x}_4 = \mathbf{x}_1$, $\mathbf{x}_4(i) = \overline{x}_i$, $\mathbf{x}_4(j) = 0$;
        $\mathbf{F}_4(i, j) = \mathbf{F}_4(j, i) = f(\mathbf{x}_4)$, $\Gamma \leftarrow \Gamma + 1$;
    end
end
$\Lambda = \mathbf{0}_{n\times n}$, $\Theta = \mathbf{0}_{n\times n}$;
for $i = 1 \rightarrow n - 1$ **do**
    for $j = i + 1 \rightarrow n$ **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
$(k, \mathbf{y}_1, \ldots, \mathbf{y}_k) = \texttt{ConnComp}(\Theta)$; `// Computing the connected components`
$\mathbf{x}_{sep} = \{\}$, $g = 0$;
for $i = 1 \rightarrow k$ **do**
    if $|\mathbf{y}_i| = 1$ **then**
        $\mathbf{x}_{sep} \leftarrow (\mathbf{x}_{sep}, \mathbf{y}_i)$;
    else
        $g \leftarrow g + 1$, $\mathbf{x}_g = \mathbf{y}_i$;
    end
end
while $|\mathbf{x}_{sep}| > 0$ **do**
    $g \leftarrow g + 1$, $\mathbf{x}_g = \{\}$;
    for $x_i \in \mathbf{x}_{sep}$ **do**
        $\mathbf{x}_g \leftarrow (\mathbf{x}_g, x_i)$, $\mathbf{x}_{sep} \leftarrow \mathbf{x}_{sep} \setminus \{x_i\}$;
        if $|\mathbf{x}_{sep}| = 0 \vee |\mathbf{x}_g| = 20$ **then**
            break;
        end
    end
end
return $(g, \mathbf{x}_1, \ldots, \mathbf{x}_g, \Gamma)$;

is a parameter set $\langle\mathbf{x}\rangle_{\mathbf{w}}^k$, $\boldsymbol{\sigma}^k$, λ^k, μ^k, $\mathbf{w}^k$, μ_{eff}^k, c_σ^k, c_1^k, c_μ^k, c_c^k, $\mathbf{B}^k$, $\mathbf{D}^k$, $\mathbf{C}^k$, $\mathbf{p}_\sigma^k$, $\mathbf{p}_c^k$ and d_σ^k for each subcomponent $\mathbf{x}_k$ ($\forall k = 1, \ldots, g$). Then, the parameter set $\mathcal{A} = \{\langle\mathbf{x}\rangle_{\mathbf{w}}^k, \boldsymbol{\sigma}^k, \lambda^k, \mu^k, \mathbf{w}^k, \mu_{eff}^k, c_\sigma^k, c_1^k, c_\mu^k, c_c^k, \mathbf{B}^k, \mathbf{D}^k, \mathbf{C}^k, \mathbf{p}_\sigma^k, \mathbf{p}_c^k, d_\sigma^k | k = 1, \ldots, g\}$. The parameters are initialized in a standard way and described as follows:

$$\langle\mathbf{x}\rangle_{\mathbf{w}}^k = \underline{\mathbf{x}}_k + \frac{\overline{\mathbf{x}}_k - \underline{\mathbf{x}}_k}{2}, \quad \boldsymbol{\sigma}^k = 0.3 \cdot (\overline{\mathbf{x}}_k - \underline{\mathbf{x}}_k), \tag{11}$$

$$\lambda^k = 4 + \lfloor 3\log(|\mathbf{x}_k|)\rfloor, \quad \mu^k = \lfloor \lambda^k/2 \rfloor, \tag{12}$$

ALGORITHM 4: $\mathbf{cv}_i = \texttt{randinit}(\mathcal{A}, \mathbf{x}_i)$

1 Obtain $\langle \mathbf{x} \rangle_{\mathbf{w}}^i$, $\boldsymbol{\sigma}^i$, $\mathbf{B}^i$ and $\mathbf{D}^i$ from $\mathcal{A}$ and $\mathbf{x}_i$;
2 $\mathbf{z}_i \sim \mathcal{N}(0, \mathbf{I}_{|\mathbf{x}_i| \times |\mathbf{x}_i|})$;
3 $\mathbf{cv}_i = \langle \mathbf{x} \rangle_{\mathbf{w}}^i + \boldsymbol{\sigma}^i \circ \mathbf{B}^i \mathbf{D}^i \mathbf{z}_i$;
4 **return** $\mathbf{cv}_i$;

$$w_i^k = \frac{\log(\lambda^k/2 + 1/2) - \log(i)}{\sum_{i=1}^{\mu^k} (\log(\lambda^k/2 + 1/2) - \log(i))}, \quad i = 1, \ldots, \mu^k, \tag{13}$$

$$\mu_{\mathit{eff}}^k = \frac{\left(\sum_{i=1}^{\mu^k} w_i^k\right)^2}{\sum_{i=1}^{\mu^k} (w_i^k)^2}, \quad c_\sigma^k = \frac{\mu_{\mathit{eff}}^k + 2}{|\mathbf{x}_k| + \mu_{\mathit{eff}}^k + 5}, \tag{14}$$

$$c_1^k = \frac{2}{(|\mathbf{x}_k| + 1.3)^2 + \mu_{\mathit{eff}}^k}, \tag{15}$$

$$c_\mu^k = \min \left\{ 1 - c_1^k, \quad \frac{2(\mu_{\mathit{eff}}^k - 2 + 1/\mu_{\mathit{eff}}^k)}{(|\mathbf{x}_k| + 2)^2 + \mu_{\mathit{eff}}^k} \right\}, \tag{16}$$

$$c_c^k = \frac{4 + \mu_{\mathit{eff}}^k / |\mathbf{x}_k|}{|\mathbf{x}_k| + 4 + 2\mu_{\mathit{eff}}^k / |\mathbf{x}_k|}, \tag{17}$$

$$\mathbf{B}^k = \mathbf{I}_{|\mathbf{x}_k| \times |\mathbf{x}_k|}, \quad \mathbf{D}^k = \mathbf{1}_{|\mathbf{x}_k| \times 1}, \quad \mathbf{C}^k = \mathbf{I}_{|\mathbf{x}_k| \times |\mathbf{x}_k|}, \tag{18}$$

$$\mathbf{p}_c^k = \mathbf{0}_{|\mathbf{x}_k| \times 1}, \quad \mathbf{p}_\sigma^k = \mathbf{0}_{|\mathbf{x}_k| \times 1}, \tag{19}$$

$$d_\sigma^k = 1 + 2 \max \left\{ 0, \sqrt{(\mu_{\mathit{eff}}^k - 1)/(|\mathbf{x}_k| + 1)} - 1 \right\} + c_\sigma^k. \tag{20}$$

After the initialization of the parameters $\mathcal{A}$, the context vector $\mathbf{cv}$ is then initialized by randomly sampling the values of the subcomponents $\mathbf{x}_i$ ($\forall i = 1, \ldots, g$) with the function `randinit()`. The function is shown in Alg. 4, in which the initial values of the parameters are $\langle \mathbf{x} \rangle_{\mathbf{w}}^i = \underline{\mathbf{x}}_i + (\overline{\mathbf{x}}_i - \underline{\mathbf{x}}_i)/2$, $\boldsymbol{\sigma}^i = 0.3 \cdot (\overline{\mathbf{x}}_i - \underline{\mathbf{x}}_i)$, $\mathbf{B}^i = \mathbf{I}_{|\mathbf{x}_i| \times |\mathbf{x}_i|}$ and $\mathbf{D}^i = \mathbf{1}_{|\mathbf{x}_i| \times 1}$.

Within each cycle of CC-GDG-CMAES, a generation of the pure CMA-ES is directly applied to the subcomponents in a round-robin fashion. The pseudocode of the pure CMA-ES for each subcomponent $\mathbf{x}_k$ is described in Alg. 5. First, a population of λ^k individuals of the subcomponent $\mathbf{x}_k$ are randomly sampled based on the current covariance matrix and evaluated together with the collaborators in $\mathbf{x}^*$ (lines 2 – 6). Then, the generated individuals are sorted from best to worst in terms of their fitness values. Finally, all the parameters relevant to the covariance matrix are updated based on the distribution of the first half of the sorted individuals (lines 8 – 15). Compared with the original algorithm, the only modification of Alg. 5 is on the fitness evaluation (line 4), which changes from the direct evaluation $f(\mathbf{x})$ to a collaborative evaluation $f(\mathbf{cv}_1, \ldots, \mathbf{cv}_{k-1}, \mathbf{v}_i^k, \mathbf{cv}_{k+1}, \ldots, \mathbf{cv}_g)$ in the cooperative co-evolution framework. Now we prove that such modification does not change the performance of CMA-ES if $f(\mathbf{x})$ is additively separable and $(\mathbf{x}_1, \ldots, \mathbf{x}_g)$ is the ideal decomposition of $\mathbf{x}$.

THEOREM 3.6. *If $f(\mathbf{x}) = \sum_{i=1}^g f_i(\mathbf{x}_i)$, where $(\mathbf{x}_1, \ldots, \mathbf{x}_g)$ is the ideal decomposition and g is the number of subcomponents, then applying the pure CMA-ES shown in Alg.*

ALGORITHM 5: $(\mathbf{cv}, \mathcal{A}, \Gamma_k) = \texttt{CMAES}(\mathbf{cv}, \mathbf{x}_k, \mathcal{A}, \Gamma_k)$

1 Obtain $\langle \mathbf{x} \rangle_{\mathbf{w}}^k$, $\boldsymbol{\sigma}^k$, λ^k, μ^k, $\mathbf{w}^k$, μ_{eff}^k, c_σ^k, c_1^k, c_μ^k c_c^k, $\mathbf{B}^k$, $\mathbf{D}^k$, $\mathbf{C}^k$, $\mathbf{p}_\sigma^k$, $\mathbf{p}_c^k$ and d_σ^k from $\mathcal{A}$ and $\mathbf{x}_k$;
2 **for** $i = 1 \rightarrow \lambda^k$ **do**
3 $\quad \mathbf{z}_i^k \sim \mathcal{N}(0, \mathbf{I}_{|\mathbf{x}_k| \times |\mathbf{x}_k|})$, $\mathbf{v}_i^k = \langle \mathbf{x} \rangle_{\mathbf{w}}^k + \boldsymbol{\sigma}^k \circ \mathbf{B}^k \mathbf{D}^k \mathbf{z}_i^k$;
4 $\quad \mathbf{F}_i^k = f(\mathbf{x}_1^*, \ldots, \mathbf{x}_{k-1}^*, \mathbf{v}_i^k, \mathbf{x}_{k+1}^*, \ldots, \mathbf{x}_g^*)$;
5 $\quad \Gamma_k \leftarrow \Gamma_k + 1$;
6 **end**
7 $(s_1, \ldots, s_{\lambda^k}) = \texttt{sort}(\mathbf{F}^k)$;
8 $\langle \mathbf{x} \rangle_{\mathbf{w}}^k = \sum_{i=1}^{\mu^k} w_i^k \mathbf{v}_{s_i}^k$, $\langle \mathbf{z} \rangle_{\mathbf{w}}^k = \sum_{i=1}^{\mu^k} w_i^k \mathbf{z}_{s_i}^k$;
9 $\mathbf{p}_\sigma^k \leftarrow (1 - c_\sigma^k)\mathbf{p}_\sigma^k + \sqrt{c_\sigma^k(2 - c_\sigma^k)\mu_{\mathit{eff}}^k}\mathbf{B}^k \langle \mathbf{z} \rangle_{\mathbf{w}}^k$;
10 $H_\sigma = \begin{cases} 1, & \text{if } \frac{|\mathbf{x}_k| \cdot \|\mathbf{p}_\sigma^k\|^2}{1 - (1 - c_\sigma^k)^{2\Gamma_k / \lambda^k}} < 2 + \frac{4}{|\mathbf{x}_k| + 1} \\ 0, & \text{otherwise.} \end{cases}$;
11 $\mathbf{p}_c^k \leftarrow (1 - c_c^k)\mathbf{p}_c^k + H_\sigma \sqrt{c_c^k(2 - c_c^k)\mu_{\mathit{eff}}^k}\mathbf{B}^k \mathbf{D}^k \langle \mathbf{z} \rangle_{\mathbf{w}}^k$;
12 $\mathbf{C}^k \leftarrow$
$(1 - c_1^k - c_\mu^k)\mathbf{C}^k + c_1^k(\mathbf{p}_c^k(\mathbf{p}_c^k)^T + (1 - H_\sigma)c_c^k(2 - c_c^k)\mathbf{C}^k) + c_\mu^k \sum_{i=1}^{\mu^k} w_i^k (\mathbf{B}^k \mathbf{D}^k \mathbf{z}_{id_i}^k)(\mathbf{B}^k \mathbf{D}^k \mathbf{z}_{id_i}^k)^T$;
13 $\boldsymbol{\sigma}^k \leftarrow \boldsymbol{\sigma}^k \exp\left(\frac{c_\sigma^k}{d_\sigma^k}\left(\frac{\|\mathbf{p}_\sigma^k\|}{E(\|\mathcal{N}(0, \mathbf{I})\|)} - 1\right)\right)$;
14 $(\mathbf{B}^k, \mathbf{D}^k) = \texttt{eig}(\mathbf{C}^k)$;
15 $\mathbf{cv}_k = \mathbf{v}_{s_1}^k$, $\mathbf{cv} = (\mathbf{cv}_1, \ldots, \mathbf{cv}_g)$;
16 **return** $(\mathbf{cv}, \mathcal{A}, \Gamma_k)$;

5 to optimize the fitness function of $f(\mathbf{cv}_1, \ldots, \mathbf{cv}_{k-1}, \mathbf{x}_k, \mathbf{cv}_{k+1}, \ldots, \mathbf{cv}_g)$ is the same as applying it to optimize the fitness function of $f_k(\mathbf{x}_k)$, $\forall \mathbf{cv}_i \in \Omega(\mathbf{x}_i)$, $i = 1, \ldots, k-1, k+1, \ldots, g$.

PROOF. According to the pseudocode shown in Alg. 5, the change of fitness evaluation has influence on only the sorting process (line 7), which sorts the fitness values $\mathbf{F}^k$ of all the λ^k generated individuals from the best to worst, and returns the corresponding ordered indices $(s_1, \ldots, s_{\lambda^k})$. Since $f(\mathbf{x}) = \sum_{i=1}^g f_i(\mathbf{x}_i)$, then $\forall \mathbf{cv}_i \in \Omega(\mathbf{x}_i)$, $i = 1, \ldots, k-1, k+1, \ldots, g$, $\mathbf{a}_k \neq \mathbf{b}_k$, we have

$$\begin{aligned} & f(\mathbf{cv}_1, \ldots, \mathbf{cv}_{k-1}, \mathbf{a}_k, \mathbf{cv}_{k+1}, \ldots, \mathbf{cv}_g) < f(\mathbf{cv}_1, \ldots, \mathbf{cv}_{k-1}, \mathbf{b}_k, \mathbf{cv}_{k+1}, \ldots, \mathbf{cv}_g) \\ \Longleftrightarrow & \sum_{i=1}^{k-1} f_i(\mathbf{cv}_i) + f_k(\mathbf{a}_k) + \sum_{i=k+1}^{g} f_i(\mathbf{cv}_i) < \sum_{i=1}^{k-1} f_i(\mathbf{cv}_i) + f_k(\mathbf{b}_k) + \sum_{i=k+1}^{g} f_i(\mathbf{cv}_i) \\ & \Longleftrightarrow f_k(\mathbf{a}_k) < f_k(\mathbf{b}_k). \end{aligned}$$

Therefore, the objective functions of $f(\mathbf{cv}_1, \ldots, \mathbf{cv}_{k-1}, \mathbf{x}_k, \mathbf{cv}_{k+1}, \ldots, \mathbf{cv}_g)$ and $f_k(\mathbf{x}_k)$ leads to the same ordered indices $(s_1, \ldots, s_{\lambda^k})$ in line 7. As a result, the update of all the parameters of the pure CMA-ES remains unchanged throughout the whole optimization process. □

Theorem 3.6 indicates that when employing the pure CMA-ES as the subcomponent optimizer of a large scale black-box optimization problem within the cooperative co-evolution framework, its performance is expected to be competitive due to its competitiveness in solving low-dimensional optimization problems.

Note that in CMA-ES, the population size λ^k for the subcomponent $\mathbf{x}_k$ is proportional to its problem size $|\mathbf{x}_k|$ ($\lambda^k = 4 + \lfloor 3 \log(|\mathbf{x}_k|) \rfloor$). Such an adaptive setting of population size addresses the issue of contributions of different subcomponents mentioned

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

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

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

Algorithm	Description
DECC-I [Omidvar et al. 2013]	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

in [Omidvar et al. 2011] [Omidvar et al. 2013] 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 consists of 20 1000-dimensional benchmark functions ($f_1 \sim f_{20}$) 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 structure of the obtained decomposition with the ideal decomposition. Then, the final results obtained by CC-GDG-CMAES are compared with that of DECC-I [Omidvar et al. 2013], 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-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 the ideal decomposition obtained manually using the knowledge of the benchmark functions. 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. In contrast, the comparison with CC-IG-CMAES shows the effect on the final solution when GDG cannot find the ideal decomposition. The compared algorithms in the experimental studies are summarized in Table II.

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

4.1. Experimental Settings

The parameter settings of CC-GDG-CMAES is given in Table III. All the parameters of CMA-ES are set in the standard way suggested on the official website [2], as shown in Eqs (11)–(20). Thus, there are only two parameters K and α related to GDG (Eq. (10)) besides the stopping criterion Γ_{max}. Here, $\Gamma_{max} = 3 \times 10^6$ is a commonly used setting which has been adopted by most of 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

First, the accuracy of GDG is evaluated on $f_1 \sim f_{20}$. 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 follows:

$$\rho_1 = \frac{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\Theta \circ \Theta_{ideal})_{i,j}}{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\Theta_{ideal})_{i,j}} \times 100\%, \tag{21}$$

$$\rho_2 = \frac{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}((\mathbf{1}_{n\times n} - \Theta) \circ (\mathbf{1}_{n\times n} - \Theta_{ideal}))_{i,j}}{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\mathbf{1}_{n\times n} - \Theta_{ideal})_{i,j}} \times 100\%, \tag{22}$$

$$\rho_3 = \frac{\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}(\mathbf{1}_{n\times n} - |\Theta - \Theta_{ideal}|)_{i,j}}{n(n-1)/2} \times 100\%, \tag{23}$$

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 $\Theta_{i,j} = (\Theta_{iedal})_{i,j}$, and 1 otherwise. The operator "$\circ$" is the Hadamard product or entrywise product of two matrices. Given two $n \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$, we have

$$\mathbf{A} \circ \mathbf{B} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \circ \begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & \cdots & a_{1n}b_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1}b_{n1} & \cdots & a_{nn}b_{nn} \end{pmatrix}. \tag{24}$$

Table IV shows the accuracies of identifying interaction, independence and both types of relationships of GDG on the CEC'2010 LSGO benchmark functions. Note that there is no ρ_1 for $f_1 \sim f_3$, 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 is seen that $\rho_1 = 100\%$ for all the test functions, which implies that all the interactions between variables were perfectly identified by GDG. Besides, the accuracy of identifying the independence is also high, showing by 100% detection precision on

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

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%

all the test functions except f_3 and f_{11}. In other words, for f_3 and f_{11}, most of the computational errors between independent variables are larger than ϵ. When looking further into the details of f_3 and f_{11}, it is found that the parts of the objective functions including the separable variables are both the Ackley function, which is stated as follows:

$$f_{Ackley}(\mathbf{x}) = 20 - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + e. \tag{25}$$

One can see that the Ackley function includes the exponential function. This is consistent with our intuition that exponential function can induce larger computational errors due to its higher complexity than the polynomial functions. The observation suggests a possibility of more adaptive setting of ϵ based not only on magnitude of objective space, but also on the complexity level of the objective function.

As a result, GDG obtained ideal decomposition 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, the high accuracy of GDG on almost all the test functions demonstrated that GDG is effective in identifying the ideal decomposition of the variables.

Tables V – VI show the accuracies achieved by $\alpha = 10^{-9}$ and 10^{-8}. It can be seen that when α becomes larger, the accuracy of identifying independence for f_3 and f_{11} increases. However, when $\alpha = 10^{-8}$, some of the interactions in f_{20} cannot be identified, and the accuracy decreased to 99.8%. In terms the final grouping of variables, 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, $\alpha = 10^{-10}$ is a proper choice in terms of the accuracies of both types of relationships.

Table VII shows 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 significance level of 0.05 is used to find out if there is any significance difference between the algorithms. If a significant difference is detected, then a series of pair-wise Wilcoxon rank sum tests [Wilcoxon 1945] with significance level of 0.05 are conducted with Bonferroni correction [Sheskin

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%

2003] in order to find the best performing algorithm. Bonferroni correction is a simple technique for controlling the family-wise error rate. 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 all the instances except f_3 and f_{11}, CC-GDG-CMAES manages to obtain the ideal grouping. For these instances, CC-GDG-CMAES and CC-IG-CMAES have the same final results. If they perform significantly better than all the other algorithms, both 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, obtained significantly best results on 12 and 13 functions, respectively. In contrast, DECC-I, MA-SW-Chains and CMA-ES perform statistically the best on 2, 3 and 3 functions. For f_5 and f_6, none of the algorithm perform significantly better than all the other algorithms. This is due to the large standard deviation of some of the 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 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 Table VII, it is seen that CC-GDG-CMAES obtains better mean values than CMA-ES on 13 out of the total 20 test functions (f_1, f_2, f_4, $f_7 \sim f_{11}$, $f_{13} \sim f_{16}$ and f_{18}). For median, CC-GDG-CMAES performs better than CMA-ES on one more function (f_6). CC-GDG-CMAES obtains the same results as CMA-ES on f_3, f_{19} and f_{20} since it obtains a single nonseparable group for them, and thus equivalent to CMA-ES which has no decomposition. On f_{12} and f_{17}, both CC-GDG-CMAES and CMA-ES obtain the global optimum of 0. CC-GDG-CMAES is outperformed by CMA-

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. 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
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

	Median	5.87e+03	2.71e+03	5.16e+03	**2.01e+03**	**2.01e+03**
f_{15}	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
	Median	**2.49e-13**	9.39e+01	4.33e+02	8.46e+01	8.46e+01
f_{16}	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
	Median	3.93e+04	1.26e+00	**0.00e+00**	**0.00e+00**	**0.00e+00**
f_{17}	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
	Median	1.19e+03	1.19e+03	2.51e+03	**5.79e+01**	**5.79e+01**
f_{18}	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
	Median	1.75e+06	**2.85e+05**	2.81e+06	2.81e+06	2.81e+06
f_{19}	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
	Median	3.87e+03	1.06e+03	**8.29e+02**	**8.29e+02**	**8.29e+02**
f_{20}	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	3	3	13	12

ES on only f_5, with both worse mean and median. CC-GDG-CMAES obtains the same results as CC-IG-CMAES on all the test functions except f_3 and f_{11}, where the two algorithms are essentially the same due to exactly the same groupings of variables. As shown in Table IV, CC-GDG-CMAES obtains ideal decomposition for all the test functions except f_3 and f_{11}.

Besides, 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 outperforms DECC-I on 14 out of the total 20 test functions (f_1, f_2, f_4, f_5, f_7, f_9, f_{10}, $f_{12} \sim f_{15}$, f_{17}, f_{18} and f_{20}) in terms of mean, and 15 functions in terms of median (on additional f_6). Note that DECC-I keeps all the separable variables in a single subcomponent, while CC-GDG-CMAES further divided them into at most 20-dimensional subcomponents. Therefore, CC-GDG-CMAES employs different decomposition for f_1, f_2, $f_4 \sim f_{10}$, f_{12} and f_{13}, where there are $500 \sim 1000$ separable variables. However, for $f_{14} \sim f_{20}$, there is no separable variable and the ideal decompositions are unique for both algorithms. CC-GDG-CMAES shows better performance than DECC-I on 5 out of these 7 functions, which indicates that CMA-ES is a powerful subcomponent optimizer for large scale black-box optimization. For f_3 and f_{11}, CC-GDG-CMAES does not obtain the ideal decomposition, 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 stably reaches the global optimum on f_3.

Table VIII shows the best fitness values obtained by the 25 independent runs of the compared algorithm on the CEC'2010 LSGO benchmark functions. For each function, the best algorithm is marked in bold. It can be seen that CC-GDG-CMAES performs the best on 13 out of the total 20 test functions. In contrast, DECC-I, MA-SW-Chains and CMA-ES perform the best on only 3 functions. Since DECC-I and MA-SW-Chains are the state-of-the-art algorithms for the CEC'2010 LSGO benchmark functions, their best results are considered to be the best known results. In this sense, CC-GDG-CMAES is able to update the best known results on most of the benchmark functions.

Table VIII. 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

Figures 3-7 show the convergence curves of CMA-ES, CC-IG-CMAES and CC-GDG-CMAES on f_1, f_5, f_{11}, f_{18} and f_{19}. 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}, CC-IG-CMAES and CC-GDG-CMAES have the same convergence curves. For f_{19}, all the three algorithms have the same convergence curve.

From the figures, it can be seen that the CC-IG-CMAES and CC-GDG-CMAES converge quickly on f_1, f_5 and f_{11}. This is because that 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 converge quickly. The early convergence allows more fitness evaluations to identify the ideal decomposition if needed in the future. 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 did 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 even for the nonseparable functions to increase convergence of algorithm.

5. CONCLUSION

In this paper, the two important issues in solving large scale black-box optimization with decomposition are addressed. First, the Global Differential Grouping (GDG) is derived from the differential grouping [Omidvar et al. 2013] to identify ideal decompo-

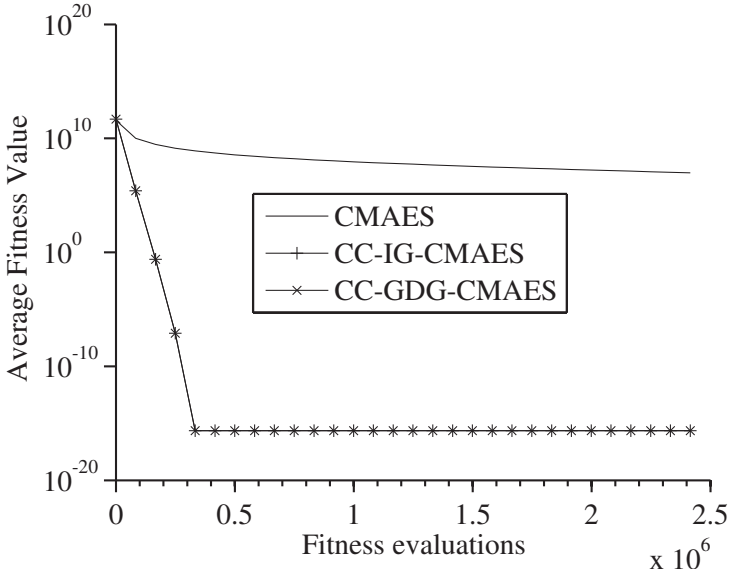

Fig. 3. Convergence curves of the compared algorithms on f_1.

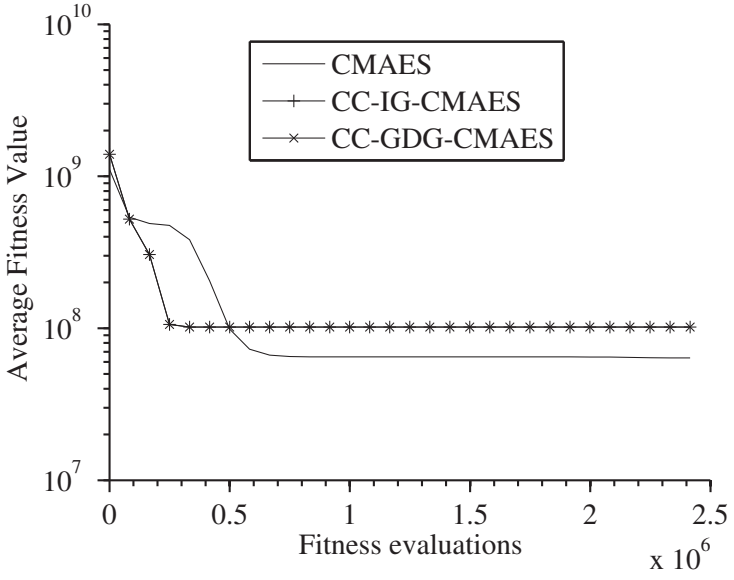

Fig. 4. Convergence curves of the compared algorithms on f_5.

sition of variables. Then, CMA-ES is employed and modified to adapt to the Cooperative Co-evolution (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, the ideal decompositions have been identified by GDG.

In the future, the assumption of separability of the objective function is to be relaxed. Furthermore, many of the theorems in this paper require additive separability of the

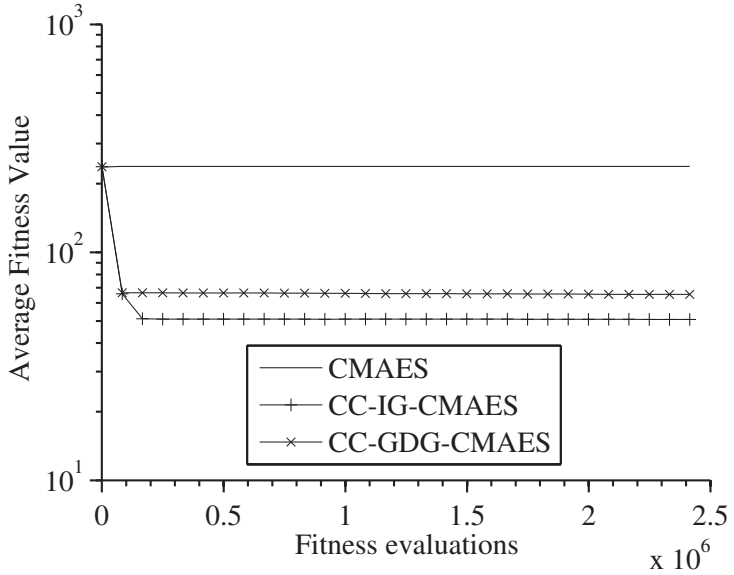

Fig. 5. Convergence curves of the compared algorithms on f_{11}.

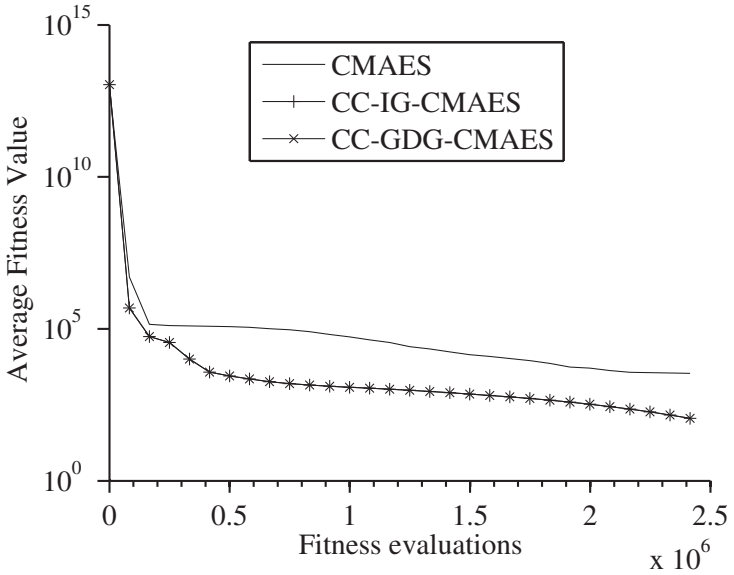

Fig. 6. Convergence curves of the compared algorithms on f_{18}.

function. Therefore, the functions with other types of separability or some nonseparable functions with weak interaction matrices such as Rosenbrock function cannot be decomposed and thus cannot be solved effectively. Effective decomposition methods are to be designed for such kind of nonseparable functions to explore the solution space more effectively. Besides, the strength of interaction between variables is also to be taken into account to further improve the efficiency of the search, e.g., the vari-

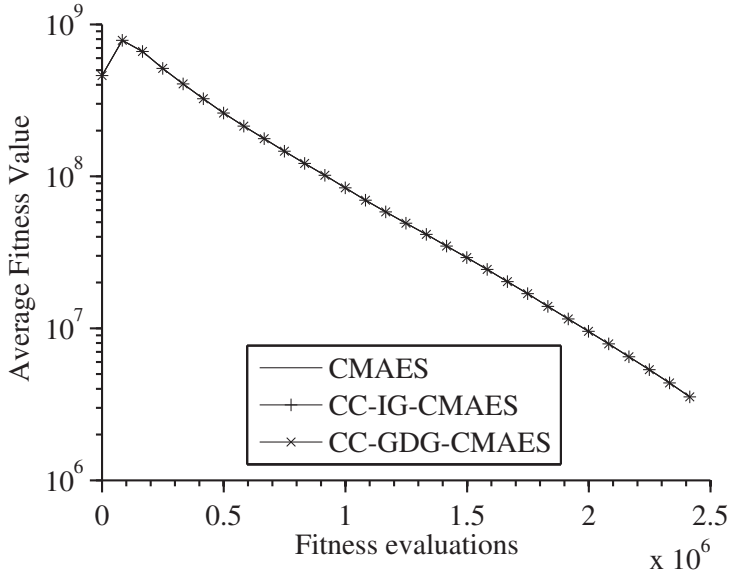

Fig. 7. Convergence curves of the compared algorithms on f_{19}.

ables with stronger interactions are optimized together more often than the ones with weaker interactions.

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.

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

Donald WK Andrews and Yoon-Jae Whang. 1990. Additive interactive regression models: circumvention of the curse of dimensionality. *Econometric Theory* 6, 4 (1990), 466–479.

Charles Audet and JE Dennis Jr. 2006. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on optimization* 17, 1 (2006), 188–217.

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

JC Bezdek, RJ Hathaway, RE Howard, CA Wilson, and MP Windham. 1987. Local convergence analysis of a grouped variable version of coordinate descent. *Journal of Optimization Theory and Applications* 54, 3 (1987), 471–477.

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

Tyson 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.

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

Andrew R Conn, Katya Scheinberg, and Ph L Toint. 1997. Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical programming* 79, 1-3 (1997), 397–414.

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

George B Dantzig and Mukund N Thapa. 1997. *Linear programming: 1: Introduction*. Vol. 1. Springer.

Russell Eberhart and James 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.

Jay Friedenberg and Gordon Silverman. 2006. *Cognitive science: An introduction to the study of mind*. Sage.

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

Paul Gilmore and Carl T Kelley. 1995. An implicit filtering algorithm for optimization of functions with many local minima. *SIAM Journal on Optimization* 5, 2 (1995), 269–285.

Fred Glover, Manuel Laguna, and Rafael Martí. 2000. Fundamentals of scatter search and path relinking. *Control and cybernetics* 39, 3 (2000), 653–684.

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

Nikolaus Hansen. 2006. The CMA evolution strategy: a comparing review. In *Towards a new evolutionary computation*. Springer, 75–102.

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

Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr 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.

Nikolaus Hansen and Andreas 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.

Georges Raif 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.

John 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.

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

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

Waltraud Huyer and Arnold Neumaier. 1999. Global optimization by multilevel coordinate search. *Journal of Global Optimization* 14, 4 (1999), 331–355.

Donald R Jones. 2001. Direct global optimization algorithm. In *Encyclopedia of optimization*. Springer, 431–440.

Donald R Jones, Matthias Schonlau, and William J Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13, 4 (1998), 455–492.

S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. 1983. Optimizing by simulated annealing. *Science* 220, 4598 (1983), 671–680.

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

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

Xiaodong Li and Xin 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.

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

Daniel Molina, Manuel Lozano, and Francisco 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.

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

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.

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

Mohammad Nabi Omidvar, Xiaodong Li, Yi Mei, and Xin Yao. 2013. Cooperative Co-evolution with Differential Grouping for Large Scale Optimization. *IEEE Transactions on Evolutionary Computation, Preprinted, DOI: 10.1109/TEVC.2013.2281543* (2013).

Mohammad Nabi Omidvar, Xiaodong Li, and Xin 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.

Mohammad Nabi Omidvar, Yi Mei, and Xiaodong Li. Accepted, Mar 2014. Optimal Decomposition of Large-Scale Separable Continuous Functions for Cooperative Co-evolutionary Algorithms. In *2014 IEEE Congress onEvolutionary Computation (CEC2014)*. IEEE.

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

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

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

Raymond Ros and Nikolaus 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.

Songqing Shan and G Gary 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.

David 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.

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

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

Rainer Storn and Kenneth Price. 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11, 4 (1997), 341–359.

Ke Tang, Xiaodong Li, P. N. Suganthan, Zhenyu Yang, and Thomas 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.

Virginia Torczon. 1997. On the convergence of pattern search algorithms. *SIAM Journal on optimization* 7, 1 (1997), 1–25.

Paul Tseng. 1999. Fortified-descent simplicial search method: A general approach. *SIAM Journal on Optimization* 10, 1 (1999), 269–288.

Paul 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.

Jan Weglarz, Jacek Blazewicz, Wojciech Cellary, and Roman 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.

Karsten Weicker and Nicole 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.

Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge 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.