A Recursive Decomposition Method for Large Scale Continuous Optimization

Yuan Sun, Michael Kirley, and Saman K. Halgamuge

***Abstract*—Cooperative co-evolution (CC) is an evolutionary computation framework that can be used to solve high-dimensional optimization problems via a "divide-and-conquer" mechanism. However, the main challenge when using this framework lies in problem decomposition. That is, deciding how to allocate decision variables to a particular subproblem, especially interacting decision variables. Existing decomposition methods are typically computationally expensive. In this paper, we propose a new decomposition method, which we call recursive differential grouping (RDG), by considering the interaction between decision variables based on nonlinearity detection. RDG recursively examines the interaction between a selected decision variable and the remaining variables, placing all interacting decision variables into the same subproblem. We use analytical methods to show that RDG can be used to efficiently decompose a problem, without explicitly examining all pairwise variable interactions. We evaluated the efficacy of the RDG method using large scale benchmark optimization problems. Numerical simulation experiments showed that RDG greatly improved the efficiency of problem decomposition in terms of time complexity. Significantly, when RDG was embedded in a CC framework, the optimization results were better than results from seven other decomposition methods.**

***Index Terms*—Continuous optimization problem, cooperative co-evolution (CC), decomposition method, large scale global optimization (LSGO).**

I. Introduction

LARGE-SCALE (high-dimensional) optimization problems are ubiquitous in the real-world, occurring in domains spanning the sciences, engineering, and multidisciplinary design problems [1]–[3]. Such problems are very difficult to solve when using evolutionary algorithms (EAs), and in many cases cannot be solved when using traditional mathematical approaches. This in part may be attributed to the fact that: 1) the search space of an optimization problem grows exponentially as the dimensionality increases [4]; 2) the complexity of an optimization problem usually grows as the dimensionality increases [5]; and 3) the computational cost of using some EAs (e.g., estimation of distribution algorithms) when solving very high-dimensional problems is extremely high [6].

There has been significant recent interest within the evolutionary computation community focussed specifically on tackling large scale global optimization (LSGO) problems. This is best illustrated by the introduction of special sessions held at the leading conferences and the special issues published in related journals. The review papers by Mahdavi *et al.* [7] and LaTorre *et al.* [8] highlight recent developments in this exciting field.

Cooperative co-evolution (CC) [9] has been used with some success when "scaling up" EAs to tackle very high-dimensional search and optimization problems. For example, CC has been applied to large scale continuous [10], combinatorial [11], constrained [12], multiobjective [13], and dynamic [14] optimization problems. The CC framework divides the LSGO problem into a number of subcomponents, and uses an (several) EA(s) to solve each subcomponent cooperatively. When optimizing each subcomponent, representatives (typically the best subsolutions found) from the other subcomponents are combined with individuals in the optimized subcomponent, to form complete candidate solutions that can be evaluated. A number of studies have shown that the problem decomposition can have a significant impact on the performance of a CC framework (e.g., [10] and [15]–[18]).

The existing decomposition methods can be classified into two very different approaches (see Section II-B for a brief review): in the *manual decomposition* method, the structure of the subcomponents is manually designed (e.g., random grouping [19]). This method does not take the underlying structure of variable interactions (see Section II-B1 for formal definition) into consideration. In the second method, *automatic decomposition*, the structure of the subcomponents is determined by the identified decision variable interactions (e.g., differential grouping [10]). However, this approach can be computationally expensive—decomposing an n-dimensional problem typically consumes $\mathcal{O}(n^2)$ function evaluations (FEs). This high computational complexity results in an inappropriate allocation of computational resources to the decomposition stage rather than the optimization stage.

In this paper, we propose a recursive differential grouping (RDG) method, which can decompose an n-dimensional

Manuscript received August 15, 2016; revised November 30, 2016, March 20, 2017, and July 29, 2017; accepted November 20, 2017. Date of publication November 28, 2017; date of current version September 28, 2018. *(Corresponding author: Yuan Sun.)*

Y. Sun is with the Department of Mechanical Engineering, University of Melbourne, Parkville, VIC 3010, Australia (e-mail: yuans2@student.unimelb.edu.au).

M. Kirley is with the Department of Computing and Information Systems, University of Melbourne, Parkville, VIC 3010, Australia (e-mail: mkirley@unimelb.edu.au).

S. K. Halgamuge is with the Research School of Engineering, Australian National University, Canberra, ACT 2601, Australia (e-mail: saman.halgamuge@anu.edu.au).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2017.2778089

problem using $\mathcal{O}(n\log(n))$ FEs. The RDG method examines the interaction between a selected decision variable x_i and the remaining decision variables based on nonlinearity detection. If any interaction is identified, the remaining decision variables will be divided into two equally sized groups, and the interaction between x_i and each group is checked. This process is carried out recursively until all of the individual decision variables interacting with x_i have been identified and have been placed into the same subcomponent as x_i.

We have evaluated the efficacy of the RDG method using benchmark LSGO problems (problems from the special sessions on LSGO at CEC'2010 [20] and CEC'2013 [21]). Comprehensive numerical simulations showed that the RDG method can decompose the benchmark problems efficiently in terms of time complexity. We verified that the subcomponent groupings dictated by RDG were in fact useful for optimization. When embedded into a CC framework, the optimization results generated using the RDG method were statistically better in terms of solution quality when compared against seven other decomposition methods across the benchmark suite.

The remainder of this paper is organized as follows. Section II describes the state-of-the-art algorithms and decomposition methods within the context of LSGO. Section III describes the proposed RDG method in detail. Section IV describes experiments to evaluate the proposed RDG method. Section V presents and analyzes the experimental results. Section VI concludes this paper and shows future directions.

II. Related Work

A. Algorithms for LSGO

In this section, we briefly describe the state-of-the-art techniques that can be used to "scale up" EAs for LSGO problems.

1) Cooperative Co-Evolution: The CC [9] framework tackles an LSGO problem using a divide-and-conquer strategy. It divides the problem into a number of low-dimensional subcomponents that are solved cooperatively. A standard CC algorithm consists of two stages: 1) *decomposition* and 2) *optimization*.

In the decomposition stage, an optimization problem is decomposed into several subcomponents. For example, a decomposition of a 6-D optimization problem ($f : \mathbb{R}^6 \to \bar{\mathbb{R}}$) could possibly be $\{(x_1, x_2), (x_3, x_4), (x_5, x_6)\}$, as shown in Fig. 1. When the structure of the underlying decision variable interactions are considered, this allocation to subcomponents may in fact be different. Recent studies have shown that the performance of a CC algorithm relies heavily on the way the optimization problem is decomposed [7], [10], [15], [16], [18]. See Sections II-B and III for further details.

In the optimization stage, an EA can be used to optimize each subcomponent based on a context vector. The context vector is a complete candidate solution, typically consisting of the best subsolutions from each subcomponent. When optimizing the *i*th subcomponent, the context vector (excluding the *i*th subsolution) is used to combine with the individuals in the *i*th subcomponent, to form complete candidate solutions that can be evaluated, as shown in Fig. 1. It has been found that using only one context vector may be too greedy [22]. Therefore, the adaptive multicontext CC [22] framework is proposed, which employs more than one context vector to co-evolve subcomponents. The original CC framework [9] optimizes each subcomponent in a round-robin fashion. Recently, a contribution-based CC framework [23] has been proposed to efficiently allocate the computational resources. In each cycle, it selects and optimizes the subcomponent that makes the greatest historical contribution to the fitness improvement.

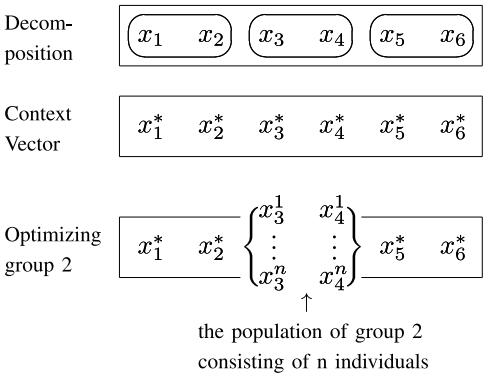


Fig. 1. Decomposition and optimization of a 6-D problem using a CC algorithm. The problem has been decomposed into three subcomponents, each with two decision variables. When optimizing the second subcomponent, the context vector (excluding the second subsolution) is used to combine with the individuals in the second subcomponent, to form complete candidate solutions that can be evaluated.

2) Other Techniques: In addition to CC, there are other techniques that can be used to address the additional challenges inherent in LSGO problems. Representative techniques include the model complexity control [6] and random projection [24] methods for estimation of distribution algorithms; the multiple strategies [25] and generalized opposition-based learning [26] methods for differential evolution; the social learning [27] and pairwise competition [28] methods for particle swarm optimization; and the multiple trajectory search [29] as well as multiple offspring sampling [30] methods for algorithm hybridization. Due to page limits, we cannot describe these techniques in detail.

B. Decomposition Methods

1) Interacting Decision Variables: In this section, we start by presenting a formal definition of "variable interactions," as some of the decomposition methods discussed below rely on variable interactions.

In an optimization problem, two decision variables interact if they cannot be optimized independently to find the global optimum. It is important to note that the interaction between given decision variables may be complicated. Take the following objective function as an example:

$$f(\mathbf{x}) := x_1^2 + (x_2 - x_3)^2 + (x_3 - x_4)^2, \quad \mathbf{x} \in [-1, 1]^4 \quad (1)$$

where ":=" denotes "defined as." Decision variable x_2 interacts with x_3, and x_3 interacts with x_4. Therefore, x_2 and x_4 are linked by x_3. If the optimal value of x_3 is known, x_2 and x_4

are independent, as they can be optimized separately. However, if x_3 needs to be optimized, x_2 and x_4 will influence each other. Therefore, x_2 conditionally interacts with x_4.[1] Note that conditional interaction only exists in overlapping problems e.g., Rosenbrock's function [21]. The formal definitions of *interaction* and *conditional interaction* are consistent with the definitions of *direct interaction* and *indirect interaction* described in [15] and [31].

Definition 1: Let $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ be a differentiable function. Decision variables x_i and x_j interact if a candidate solution $\mathbf{x}^*$ exists, such that

$$\frac{\partial^2 f(\mathbf{x}^*)}{\partial x_i \partial x_j} \neq 0 \tag{2}$$

denoted by $x_i \leftrightarrow x_j$. Decision variables x_i and x_j conditionally interact if for any candidate solution $\mathbf{x}^*$

$$\frac{\partial^2 f(\mathbf{x}^*)}{\partial x_i \partial x_j} = 0 \tag{3}$$

and a set of decision variables $\{x_{k1}, \ldots, x_{kt}\} \subset X$ exists, such that $x_i \leftrightarrow x_{k1} \leftrightarrow \ldots \leftrightarrow x_{kt} \leftrightarrow x_j$. Decision variables x_i and x_j are independent if for any candidate solution $\mathbf{x}^*$, (3) holds and a set of decision variables $\{x_{k1}, \ldots, x_{kt}\} \subset X$ does not exist, such that $x_i \leftrightarrow x_{k1} \leftrightarrow \ldots \leftrightarrow x_{kt} \leftrightarrow x_j$.

2) Manual Decomposition: In these methods, the number of subcomponents and the size of each subcomponent are manually designed. These methods work well when combined with algorithms to solve fully separable problems. However, the performance deteriorates quickly when applied on partially separable problems or fully nonseparable problems. The main reason is that it does not take the underlying structure of variable interactions into consideration.

Probably the first and simplest decomposition is the uni-variable grouping [9] method, which decomposes an n-dimensional problem into n 1-D subcomponents. The uni-variable grouping method improved the performance of a genetic algorithm (GA) when solving benchmark separable problems, however it degraded the GA's performance when solving a benchmark nonseparable problem—the Rosenbrock's function [9]. This performance difference may be attributed to the fact that the uni-variable grouping method decomposes an optimization problem without considering the interaction between decision variables.

The S_k grouping [32] method is more flexible than the uni-variable grouping method when used to decompose an optimization problem. It decomposes an n-dimensional problem into k s-dimensional subcomponents, $s < n$. The S_k grouping method has been shown to be able to improve the performance of a particle swarm optimization [32] algorithm and a biogeograph-based optimization [33] algorithm. However, like the uni-variable grouping method, the S_k grouping method does not take variable interactions into consideration.

The random grouping (RG) method is proposed within the context of a differential evolution CC (DECC) framework [19]. It randomly assigns decision variables to predetermined number of subcomponents before each evolutionary cycle. The RG method has been successfully applied to improve the performance of a particle swarm optimization [34] algorithm and an artificial bee colony [35] algorithm. However, it has been shown that the probability of assigning more than two interacting decision variables into one subcomponent using the RG method is low [36]. Another limitation of RG is the requirement of setting an appropriate subcomponent size. To address this issue, the multilevel CC [37] algorithm takes the subcomponent size as a parameter, and selects an appropriate subcomponent size according to the historical performance.

A more sophisticated method—delta grouping [38] identifies variable interactions based on the averaged difference in a certain decision variable across the whole population. It generally outperforms the RG method when incorporated with the DECC framework to solve the CEC'2010 benchmark problems [38]. However on benchmark problems with more than one nonseparable subcomponent, the performance of the delta grouping method is low [10].

The k-means grouping [39] method uses a k-means clustering algorithm to construct decision variable groups. The decision variables with similar effects on the fitness value are placed into the same subcomponent. The subcomponent with the greatest contribution to the fitness value will be optimized with more iterations (FEs). The idea is similar to the contribution-based CC [23], [40] framework. Unlike most decomposition methods which group decision variables based on variable interactions, the k-means grouping method groups decision variables based on their contribution to the fitness improvement. Therefore, it is specifically tailored for problems with unbalanced subcomponents [4], [39].

3) Automatic Decomposition: In these methods, the interacting decision variables are identified and automatically placed into the same subcomponent. It is important to note that automatic decomposition considers the underlying variable interaction structure encapsulated within the search landscape. The decomposition method proposed in Section III falls into this category.

A representative automatic decomposition method—CC with variable interaction learning (CCVIL) [41]—identifies the pairwise interaction between decision variables by the nonmonotonicity detection. If the monotonicity of the fitness function with respect to x_i does not change for different value of x_j, x_i and x_j are independent. Otherwise, decision variables x_i and x_j interact. The rationale behind the CCVIL method is consistent with the linkage identification by nonmonotonicity detection [42] method. The CCVIL method is more accurate than most of the manual decomposition methods when identifying variable interactions. However, it still cannot obtain acceptable results when used to decompose some benchmark problems [10].

The statistical variable interdependence learning (SL) [43] method identifies variable interactions based on the nonmonotonicity detection as well. Unlike the CCVIL method, the SL method detects the monotonicity relationship between x_i and x_j multitimes. The probability (p_{ij}) of the observation of nonmonotonicity is calculated. If the probability p_{ij} is greater

[1] In [4], the relationships between $\{x_2, x_4\}$ and $\{x_1, x_4\}$ are both defined as additively separable. In this paper, we use *conditional interaction* to differentiate these two relationships.

than a given threshold, decision variables x_i and x_j interact. The main issue of the SL method is the high computational complexity. The number of FEs needed to decompose an n-dimensional optimization problem is $4mn^2$, where m is the number of nonmonotonicity detection conducted for each pair of decision variables.

To address the high computational complexity of the SL method, a fast variable interdependence searching [44] method is proposed. It identifies interactions between two subsets of decision variables, instead of two decision variables, by nonmonotonicity detection. Therefore, it speeds up the decomposition process. The computational cost to decompose an n-dimensional problem can be reduced to $4mn\log(n)$ in the worst case.

The differential grouping (DG) [10] method identifies variable interactions by detecting the fitness changes when perturbing the decision variables. If the fitness change induced by perturbing decision variable x_i varies for different value of x_j, x_i and x_j interact. The rationale behind the DG method is consistent with the linkage identification by nonlinearity check [45] method. The DG method outperformed the CCVIL method when used to decompose the CEC'2010 benchmark problems [10]. However, it has been shown not to be able to completely identify interacting decision variables in overlapping problems [15], [46].

Subsequently, the extended DG (XDG) [15] method was proposed to address this issue, by placing all the linked (interacting and conditionally interacting) decision variables into one subcomponent. It employs the same technique as DG to identify interacting decision variables. Then the overlapping between subcomponents are checked to identify conditional interactions. However, the number of FEs used by XDG to decompose an n-dimensional problem is usually around n^2. The high complexity of the XDG method results in an inappropriate allocation of computational budget between decomposition and optimization, and prevents it from being applied to solve even higher dimensional problems.

The computational cost in the decomposition stage can be reduced to $(n^2+3n+2)/2$ by using the global DG (GDG) [46] method. It employs the same technique as DG to identify the pairwise interactions between decision variables. The variable interaction matrix is calculated, which is regarded as the adjacency matrix of a graph. Then the depth-first search or breadth-first search can be used to identify the connected components. Note that both the interacting and conditionally interacting decision variables will also be placed into one connected component (subcomponent).

Recently, it has been shown that the minimal number of FEs used to identify the complete variable interaction matrix based on DG is $(n^2+n+2)/2$ [47]. However, it may not need the entire variable interaction matrix to identify the connected components (subcomponents). For example, if decision variable x_1 interacts with x_2 and x_3, the interaction between x_2 and x_3 needs not to be checked, as they belong to the same connected component.

The fast interdependency identification (FII) [48] method can further improve the efficiency of problem decomposition by avoiding the need to identify the complete variable interaction matrix. FII first identifies the separable decision variables by examining the interaction between one decision variable and the other variables. Then the interaction between nonseparable decision variables is examined, and all the linked (connected) decision variables are placed into the same subcomponent. The FII method is efficient when used to decompose benchmark problems with a large portion of separable decision variables. However on benchmark problems with conditional variable interactions, the number of FEs used by FII may still be in the magnitude of n^2 ($\Theta(n^2)$). In the next section, we will propose an efficient and robust method that can decompose any n-dimensional problem using less than $6n\log_2(n)$ FEs.

III. Recursive Differential Grouping

In this section, the proposed decomposition method—RDG—is described in detail. Then, the computational complexity of the RDG method is presented.

Notation: Let X be the set of decision variables $\{x_1, \ldots, x_n\}$ and U_X be the set of unit vectors in the decision space R^n. Let X_1 be a subset of decision variables $X_1 \subset X$ and U_{X_1} be a subset of U_X such that any unit vector $\mathbf{u} = (u_1, \ldots, u_n) \in U_{X_1}$, we have

$$u_i = 0, \qquad \text{if } x_i \notin X_1. \tag{4}$$

Directional Derivative: Let $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ be a differentiable function, and $\mathbf{u} = (u_1, \ldots, u_n)$ be a vector from U_X. The directional derivative of f in the direction $\mathbf{u}$, denoted $\mathcal{D}_{\mathbf{u}} f(\mathbf{x})$, is given by

$$\mathcal{D}_{\mathbf{u}} f(\mathbf{x}) = \sum_{i=1}^{n} \frac{\partial f(\mathbf{x})}{\partial x_i} u_i. \tag{5}$$

Proposition 1: Let $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ be a differentiable function; $X_1 \subset X$ and $X_2 \subset X$ be two mutually exclusive subsets of decision variables: $X_1 \cap X_2 = \emptyset$. If there exist two unit vectors $\mathbf{u}_1 \in U_{X_1}$ and $\mathbf{u}_2 \in U_{X_2}$, and a candidate solution $\mathbf{x}^*$ in the decision space such that

$$\mathcal{D}_{\mathbf{u}_1} \mathcal{D}_{\mathbf{u}_2} f(\mathbf{x}^*) \neq 0 \tag{6}$$

there is some interaction between decision variables in X_1 and X_2.

Proof: Without loss of generality, we assume that $X_1 = \{x_{1,1}, \ldots, x_{1,p}\}$, $X_2 = \{x_{2,1}, \ldots, x_{2,q}\}$, where p and q are the number of decision variables in X_1 and X_2, respectively; $\mathbf{u}_1 = (u_1^1, \ldots, u_n^1)$ and $\mathbf{u}_2 = (u_1^2, \ldots, u_n^2)$. According to directional derivative

$$\mathcal{D}_{\mathbf{u}_1} \mathcal{D}_{\mathbf{u}_2} f(\mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} u_i^1 u_j^2. \tag{7}$$

As $\mathbf{u}_1$ and $\mathbf{u}_2$ are two unit vectors from U_{X_1} and U_{X_2}, respectively, we can obtain that

$$u_i^1 = 0, \qquad \text{if } x_i \notin X_1 \tag{8}$$

$$u_j^2 = 0, \qquad \text{if } x_j \notin X_2. \tag{9}$$

Therefore,

$$\mathcal{D}_{\mathbf{u}_1}\mathcal{D}_{\mathbf{u}_2}f(\mathbf{x}) = \sum_{i=1}^{p}\sum_{j=1}^{q}\frac{\partial^2 f(\mathbf{x})}{\partial x_{1,i}\partial x_{2,j}}u^1_{1,i}u^2_{2,j}. \tag{10}$$

If (6) holds

$$\sum_{i=1}^{p}\sum_{j=1}^{q}\frac{\partial^2 f(\mathbf{x}^*)}{\partial x_{1,i}\partial x_{2,j}}u^1_{1,i}u^2_{2,j} \neq 0. \tag{11}$$

Therefore, there exists at least one pair of (i, j), such that

$$\frac{\partial^2 f(\mathbf{x}^*)}{\partial x_{1,i}\partial x_{2,j}} \neq 0. \tag{12}$$

Based on Definition 1, at least one pair of decision variables $x_{1,i} \in X_1$ and $x_{2,j} \in X_2$ interact. ■

Corollary 1: Let $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ be an objective function; $X_1 \subset X$ and $X_2 \subset X$ be two mutually exclusive subsets of decision variables: $X_1 \cap X_2 = \emptyset$. If there exist two unit vectors $\mathbf{u}_1 \in U_{X_1}$ and $\mathbf{u}_2 \in U_{X_2}$, two real numbers $l_1, l_2 > 0$, and a candidate solution $\mathbf{x}^*$ in the decision space, such that

$$f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_2) - f(\mathbf{x}^* + l_2\mathbf{u}_2) \neq f(\mathbf{x}^* + l_1\mathbf{u}_1) - f(\mathbf{x}^*) \tag{13}$$

there is some interaction between decision variables in X_1 and X_2.

Proof: With Proposition 1, we only need to prove the following statement.

Statement 1: If there exist two unit vectors $\mathbf{u}_1 \in U_{X_1}$ and $\mathbf{u}_2 \in U_{X_2}$, two real numbers $l_1, l_2 > 0$, and a candidate solution $\mathbf{x}^*$ in the decision space, such that (13) holds, then (6) is true.

It is equivalent to prove its contraposition.

Statement 2: If for any two unit vectors $\mathbf{u}_1 \in U_{X_1}$ and $\mathbf{u}_2 \in U_{X_2}$, and for any candidate solution $\mathbf{x}^*$ in the decision space, the following condition holds:

$$\mathcal{D}_{\mathbf{u}_1}\mathcal{D}_{\mathbf{u}_2}f(\mathbf{x}^*) = 0 \tag{14}$$

then

$$f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_2) - f(\mathbf{x}^* + l_2\mathbf{u}_2) = f(\mathbf{x}^* + l_1\mathbf{u}_1) - f(\mathbf{x}^*) \tag{15}$$

for any $l_1, l_2 > 0$.

In order to prove Statement 2, we first introduce *line integral*.

Line Integral: Let L be a curve with end points A and B in the decision space R^n and the arc length of L be l. Let C be any point on L and the coordinate of C ($\mathbf{x}$) can be uniquely determined by the length of the arc AC (s) : $\mathbf{x} = \mathbf{x}(s)$, $s \in [0, l]$. The integral of a function $g : \mathbb{R}^n \to \bar{\mathbb{R}}$ along the curve L is given by

$$\int_L g(\mathbf{x})\mathrm{d}s = \int_0^l g(\mathbf{x}(s))\mathrm{d}s. \tag{16}$$

Let A_2 ($\mathbf{x}^*$) be any point in R^n, and B_2 be $\mathbf{x}^* + l_2\mathbf{u}_2$, where $\mathbf{u}_2$ is any vector in U_{X_2} and l_2 is any positive real number. Let C_2 be any point on the segment A_2B_2. Therefore, the length of the segment A_2B_2 is l_2, and the coordinate of C_2 ($\mathbf{x}$) can be uniquely determined by the length of the segment A_2C_2 (s_2): $\mathbf{x}(s_2) = \mathbf{x}^* + s_2\mathbf{u}_2$, $s_2 \in [0, l_2]$. If (14) holds for any candidate solution in the decision space, then

$$\mathcal{D}_{\mathbf{u}_1}\mathcal{D}_{\mathbf{u}_2}f(\mathbf{x}) = 0. \tag{17}$$

As $\mathcal{D}_{\mathbf{u}_1}\mathcal{D}_{\mathbf{u}_2}f(\mathbf{x}) = \mathcal{D}_{\mathbf{u}_2}\mathcal{D}_{\mathbf{u}_1}f(\mathbf{x})$, by integrating both sides of (17) along the segment A_2B_2, we can obtain that

$$\int_0^{l_2}\mathcal{D}_{\mathbf{u}_1}\mathcal{D}_{\mathbf{u}_2}f(\mathbf{x})\mathrm{d}s_2 = \int_0^{l_2}\mathcal{D}_{\mathbf{u}_2}\mathcal{D}_{\mathbf{u}_1}f(\mathbf{x})\mathrm{d}s_2 = 0. \tag{18}$$

As

$$\int_0^{l_2}\mathcal{D}_{\mathbf{u}_2}\big(\mathcal{D}_{\mathbf{u}_1}f(\mathbf{x}(s_2))\big)\,\mathrm{d}s_2 = \mathcal{D}_{\mathbf{u}_1}f(\mathbf{x}(s_2))\Big|_{s_2=0}^{s_2=l_2} \tag{19}$$

thus,

$$\mathcal{D}_{\mathbf{u}_1}f(\mathbf{x}(s_2))\Big|_{s_2=0}^{s_2=l_2} = 0 \tag{20}$$

and

$$\mathcal{D}_{\mathbf{u}_1}f(\mathbf{x}^* + l_2\mathbf{u}_2) - \mathcal{D}_{\mathbf{u}_1}f(\mathbf{x}^*) = 0. \tag{21}$$

As A_2 ($\mathbf{x}^*$) is any point in R^n, therefore

$$\mathcal{D}_{\mathbf{u}_1}f(\mathbf{x} + l_2\mathbf{u}_2) = \mathcal{D}_{\mathbf{u}_1}f(\mathbf{x}). \tag{22}$$

Let A_1 ($\mathbf{x}^*$) be any point in R^n, and B_1 be $\mathbf{x}^* + l_1\mathbf{u}_1$, where $\mathbf{u}_1$ is any vector in U_{X_1} and l_1 is any positive real number. Let C_1 be any point on the segment A_1B_1. Therefore, the length of the segment A_1B_1 is l_1, and the coordinate of C_1 ($\mathbf{x}$) can be uniquely determined by the length of the segment A_1C_1 (s_1) : $\mathbf{x}(s_1) = \mathbf{x}^* + s_1\mathbf{u}_1$, $s_1 \in [0, l_1]$. Similarly, by integrating both sides of (22) along the segment A_1B_1, we can obtain

$$\int_0^{l_1}\mathcal{D}_{\mathbf{u}_1}f(\mathbf{x}(s_1) + l_2\mathbf{u}_2)\,\mathrm{d}s_1 = \int_0^{l_1}\mathcal{D}_{\mathbf{u}_1}f(\mathbf{x}(s_1))\,\mathrm{d}s_1. \tag{23}$$

Therefore,

$$f(\mathbf{x}^* + l_1\mathbf{u}_1 + l_2\mathbf{u}_2) - f(\mathbf{x}^* + l_2\mathbf{u}_2) = f(\mathbf{x}^* + l_1\mathbf{u}_1) - f(\mathbf{x}^*). \tag{24}$$

Thus, Statement 2 is proved, and Statement 1 and Corollary 1 are true. ■

With Corollary 1, the interaction between two subsets of decision variables (X_1 and X_2) can be identified using the following procedures.

1) Set all the decision variables to the lower bounds (**lb**) of the search space ($\mathbf{x}_{l,l}$).
2) Perturb the decision variables X_1 of $\mathbf{x}_{l,l}$ from the lower bounds to the upper bounds (**ub**), denoted by $\mathbf{x}_{u,l}$.
3) Calculate the fitness difference (δ_1) between $\mathbf{x}_{l,l}$ and $\mathbf{x}_{u,l}$.
4) Perturb the decision variables X_2 of $\mathbf{x}_{l,l}$ and $\mathbf{x}_{u,l}$ from the lower bounds to the middle between the lower bounds and upper bounds, denoted by $\mathbf{x}_{l,m}$ and $\mathbf{x}_{u,m}$, respectively.
5) Calculate the fitness difference (δ_2) between $\mathbf{x}_{l,m}$ and $\mathbf{x}_{u,m}$.
6) If the difference between δ_1 and δ_2 is greater than a threshold ϵ, there is some interaction between X_1 and X_2.

The two subscripts of $\mathbf{x}$ denote the values of X_1 and X_2, respectively: "l" means lower bounds, "u" means upper bounds, and

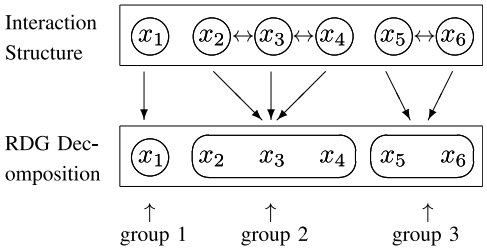


Fig. 2. Variable interaction structure and the RDG decomposition of the objective function given in (26). The notation $x_i \leftrightarrow x_j$ denotes that decision variable x_i interacts with x_j.

"*m*" means the middle between the lower bounds and upper bounds. The threshold ϵ is estimated based on the magnitude of the objective space [46]

$$\epsilon = \alpha \cdot \min\{|f(\mathbf{x}_1)|, \ldots, |f(\mathbf{x}_k)|\} \tag{25}$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_k$ are k randomly generated candidate solutions, and α is the control coefficient [46].

Based on Corollary 1, we propose the RDG (Algorithm 1) method to efficiently decompose an optimization problem. The decomposition by the RDG method considers the underlying structure of variable interactions. Taking the following objective function as an example:

$$f(\mathbf{x}) \coloneqq x_1^2 + (x_2 - x_3)^2 + (x_3 - x_4)^2 + (x_5 - x_6)^2, \mathbf{x} \in [-1, 1]^6 \tag{26}$$

the decision variables (x_2, x_3, x_4) interact, as well as (x_5, x_6). Therefore, the decomposition by the RDG method is $\{(x_1), (x_2, x_3, x_4), (x_5, x_6)\}$, as shown in Fig. 2.

The inputs to the RDG method are the fitness function (f), the upper bounds (**ub**), the lower bounds (**lb**), and the threshold (ϵ) which is estimated using (25). The outputs are the separable variable group (seps) and the nonseparable variable groups (nonseps). The *seps* contains *one* group of all separable decision variables. The *nonseps* contains *several* groups of nonseparable decision variables. Each group of decision variables will form a subcomponent.

The RDG method begins by identifying the interaction between the first decision variable x_1 and the remaining decision variables. If no interaction is detected, x_1 will be placed in the separable decision variable group, and the algorithm will move on to the next decision variable x_2. If any interaction is detected, the remaining decision variables will be divided into two (nearly) equally sized groups G_1 and G_2. Then the interaction between x_1 and G_1, x_1 and G_2 will be identified, respectively. This process is recursively conducted until all the individual decision variables that interact with x_1 are identified and placed in the decision variable subset X_1 with x_1.

Then, the RDG method examines the interaction between X_1 and the remaining decision variables (excluding the decision variables in X_1) to identify the individual decision variables that conditionally interact with x_1 (linked by other decision variables). If any interaction is identified, the interacting decision variables will be placed into X_1. This process is repeated until no interaction can be further detected between X_1 and the remaining decision variables (exclusive X_1). The decision variables in X_1 will be placed in a nonseparable group.

The RDG method moves on to the next decision variable that has not been grouped (x_i). The interaction between x_i and the remaining decision variables will be examined, and both the interacting and conditionally interacting (linked) decision variables will be placed into one group with x_i. This process is repeated until all of the decision variables are grouped. It returns the separable (seps) and the nonseparable (nonseps) decision variable groups as the outputs.

Algorithm 1 RDG

Require: f, **ub**, **lb**, ϵ
Initialize *seps* and *nonseps* as *empty* groups
Set all decision variables to the lower bounds: $\mathbf{x}_{l,l} = \mathbf{lb}$
Calculate the fitness: $y_{l,l} = f(\mathbf{x}_{l,l})$
Assign the first variable x_1 to the variable subset X_1
Assign the rest of variables to the variable subset X_2
while X_2 is not *empty* **do**
  $[X_1^*]$ = INTERACT(X_1, X_2, $\mathbf{x}_{l,l}$, $y_{l,l}$, ϵ)
  if X_1^* is the same with X_1 **then**
    if X_1 contains one decision variable **then**
      Add X_1 to *seps*
    else
      Add X_1 to *nonseps*
    end if
    Empty X_1 and X_1^*
    Assign the first variable of X_2 to X_1
    Delete the first variable in X_2
  else
    $X_1 = X_1^*$
    Delete the variables of X_1 from X_2
  end if
end while
return *seps* and *nonseps*

function INTERACT(X_1, X_2, $\mathbf{x}_{l,l}$, $y_{l,l}$, ϵ)
  $\mathbf{x}_{u,l}$=$\mathbf{x}_{l,l}$; $\mathbf{x}_{u,l}(X_1) = \mathbf{ub}(X_1)$ //Set X_1 to the **ub**
  Calculate the fitness change: $\delta_1 = y_{l,l} - f(\mathbf{x}_{u,l})$
  $\mathbf{x}_{l,m} = \mathbf{x}_{l,l}$; $\mathbf{x}_{l,m}(X_2) = \big(\mathbf{lb}(X_2) + \mathbf{ub}(X_2)\big)/2$
  $\mathbf{x}_{u,m} = \mathbf{x}_{u,l}$; $\mathbf{x}_{u,m}(X_2) = \big(\mathbf{lb}(X_2) + \mathbf{ub}(X_2)\big)/2$
  Calculate the fitness change: $\delta_2 = f(\mathbf{x}_{l,m}) - f(\mathbf{x}_{u,m})$
  if $|\delta_1 - \delta_2| > \epsilon$ **then**
    if X_2 contains one variable **then**
      $X_1 = X_1 \cup X_2$
    else
      Divide X_2 into equally-sized groups G_1, G_2
      $[X_1^1]$=INTERACT(X_1, G_1, $\mathbf{x}_{l,l}$, $y_{l,l}$, ϵ)
      $[X_1^2]$=INTERACT(X_1, G_2, $\mathbf{x}_{l,l}$, $y_{l,l}$, ϵ)
      $[X_1]$= $X_1^1 \cup X_1^2$
    end if
  end if
return X_1
end function

The computational complexity of the RDG method when used to decompose an n-dimensional problem is $\mathcal{O}(n\log(n))$, which is analyzed as follows.

1) When decomposing an n-dimensional fully separable problem, the computational complexity of the RDG method is $\Theta(n)$ in terms of the number of FEs. For each decision variable, three FEs are used to determine its separability. Therefore, totally about $3n$ FEs are needed.
2) When decomposing an n-dimensional fully nonseparable problem with one subcomponent, the computational complexity of the RDG method is $\Theta(n)$. When grouping the n interacting decision variables, the function "INTERACT" is executed about $\sum_{i=0}^{k}(n/2^i)$ times, where $k=\log_2(n)$

$$\sum_{i=0}^{k}\frac{n}{2^i}=n\left(2-\left(\frac{1}{2}\right)^k\right)<2n. \tag{27}$$

It consumes three FEs each time the function INTERACT is executed. Therefore, totally about $6n$ FEs are used.
3) When decomposing an n-dimensional partially separable problem with n/m subcomponents, the computational complexity of the RDG method is $\Theta(n\log(n))$, where n is the dimensionality and m is the number of decision variables in each subcomponent. When grouping m interacting decision variables into one subcomponent, the function INTERACT is executed less than $2m\times\log_2(n)$ times, each time consuming three FEs. The number of subcomponents is n/m. Therefore, totally less than $3\times 2m\times\log_2(n)\times n/m=6n\log_2(n)$ FEs are used.
4) When decomposing an n-dimensional partially separable problem with an m-dimensional nonseparable subcomponent, the computational complexity of the RDG method is $\mathcal{O}(\max\{n, m\log(n)\})$. The RDG algorithm consumes about $3(n-m)$ FEs to identify the $n-m$ separable decision variables, and consumes less than $6m\times\log_2(n)$ FEs to identify the m interacting decision variables. Therefore, totally less than $3(n-m)+6m\times\log_2(n)$ FEs are used.
5) When decomposing an n-dimensional overlapping problem (e.g., Rosenbrock's function [21]), the computational complexity of the RDG method is $\Theta(n\log(n))$. Starting from x_1, it consumes about $3\times 2\times 2\log_2(n)=12\log_2(n)$ FEs to identify the two decision variables (x_p and x_q) that interact with x_1. Then it also consumes about $12\log_2(n)$ FEs to identify the two decision variables that interact with (x_p, x_1, x_q). Therefore, totally about $n/2\times 12\log_2(n)=6n\log_2(n)$ FEs are used.

IV. EXPERIMENTAL METHODOLOGY

In this section, comprehensive numerical experiments are designed to evaluate the proposed RDG method. Two research questions guide the experimental design to evaluate the efficacy of the proposed RDG method.

Q1: Can the proposed RDG method decompose the CEC'2010 and CEC'2013 benchmark problems more efficiently when compared against other well-known decomposition methods.

TABLE I
PARAMETER SETTINGS FOR ALL THE DECOMPOSITION METHODS USED IN THE EXPERIMENTS. DG2 IS A PARAMETER-FREE METHOD

Decomposition Methods	Parameter Settings
RDG	control coefficient $\alpha=10^{-12}$ and $k=10$
GDG	control coefficient $\alpha=10^{-12}$ and $k=10$
XDG	threshold $\epsilon=10^{-1}$
DG	threshold $\epsilon=10^{-3}$
DG2	parameter free
FII	threshold $\epsilon=10^{-2}$
D (delta grouping)	sub-component size $sub\_dim=100$
RG	sub-component size $sub\_dim=100$

Q2: Can the proposed RDG method outperform other well-known decomposition methods when embedded into a CC framework to solve the CEC'2010 and CEC'2013 benchmark problems.

To answer Q1, the proposed RDG method was used to decompose the CEC'2010 [20] and CEC'2013 [21] benchmark problems.[2] Two metrics were employed to evaluate the performance of a decomposition method: 1) the number of FEs used to decompose the problem and 2) the percentage of interacting decision variables that are correctly grouped, defined as follows.

Decomposition Accuracy: Let $G=\{g_1,\ldots,g_m\}$ denote the groups of interacting decision variables in a problem f, and $\tilde{G}=\{\tilde{g}_1,\ldots,\tilde{g}_n\}$ denote the groups of interacting decision variables that are identified by a decomposition method. Let $G_i=\{g_{i,1},\ldots,g_{i,m}\}$ denote the ith permutation of G, where $1\le i\le m!$, and $\tilde{G}_j=\{\tilde{g}_{j,1},\ldots,\tilde{g}_{j,n}\}$ denote the jth permutation of $\tilde{G}$, where $1\le j\le n!$. The decomposition accuracy (DA) of the decomposition method on f is defined as

$$\mathrm{DA}=\frac{\max\limits_{i,j}\left\{\sum\limits_{k=1}^{\min\{m,n\}}|g_{i,k}\cap\tilde{g}_{j,k}|\right\}}{\sum\limits_{i=1}^{m}|g_i|} \tag{28}$$

where $|g_i|$ denotes the number of decision variables in g_i.

The performance of the RDG method was then compared to the GDG [46], XDG [15], DG [10], as well as two recently published methods—DG2 [47] and FII [48]. The parameter settings for all the decomposition methods used in the experiments are shown in Table I. The threshold values (ϵ) estimated by the RDG (or GDG) method for each problem were recorded. Note that the RDG and GDG methods use the same approach (25) to estimate the threshold values.

To answer Q2, the proposed RDG method was embedded into the DECC [19]/CMAESCC [46] framework to solve the CEC'2010 and CEC'2013 benchmark problems. The DECC is the most widely used CC framework, which employs a variant of differential evolution—SaNSDE [49]—to solve each subcomponent cooperatively. The CMAESCC framework uses the well-known CMA-ES [50] algorithm to solve each subcomponent. It performs well when used to solve the CEC'2010 and CEC'2013 benchmark problems. The parameter settings

[2]The MATLAB implementation of the RDG method can be accessed from the following link: https://bitbucket.org/yuans/rdg.

for the DECC and CMAESCC frameworks were consistent with the original papers. The maximum number of FEs was set to 3×10^6, divided between the decomposition stage and optimization stage. For each benchmark problem, the median, mean, and standard deviation of the best solutions found by the DECC/CMAESCC-RDG algorithm based on 25 independent runs were recorded. The performance of the RDG method was compared against the performance of the XDG, GDG, DG, DG2, FII, as well as two manual decomposition methods—D (delta grouping [38]) and RG [19] methods, when embedded in each CC framework.

The performance of DECC/CMAESCC-RDG was also compared to the performance of two state-of-the-art hybrid algorithms—MOS [30] and MA-SW-Chains [51] with default parameter settings. The MOS algorithm evaluates the constituent algorithms in each generation, and the better performed constituent algorithm will be used to generate more offspring. MOS achieved the best performance in the 2011 special issue of the Soft Computing journal. The MA-SW-Chains algorithm assigns to each individual a local search intensity that depends on its features, by chaining different local search applications. MA-SW-Chains achieved the best performance in the CEC 2010 special session and competition on LSGO.

The Kruskal–Wallis nonparametric one-way ANOVA test [52] with 95% confidence interval was used to determine whether the performance of at least one algorithm was significantly different from the others. Then a series of Wilcoxon rank-sum tests (significance level $\alpha = 0.05$) with Holm p-value correction [52] was conducted in a pairwise fashion to find the better performing algorithm.

V. Experimental Results

Comprehensive experimental results are presented and discussed in this section. Section V-A presents the decomposition comparison between the RDG method and five other methods, thus addressing Q1. Section V-B presents the optimization comparison between the RDG method and seven other methods when embedded into the DECC/CMAESCC framework to solve the benchmark problems, thus addressing Q2.

A. Decomposition Comparison

Table II lists the decomposition results of the RDG, GDG, XDG, and DG methods on the CEC'2010 and CEC'2013 benchmark problems. The parameter settings for the four decomposition methods are consistent with Table I. In Table II, "DA" represents the decomposition accuracy—the percentage of interacting decision variables that correctly grouped; "FEs" represents the number of FEs used in the decomposition stage; and "ϵ" represents the threshold used to identify interactions between decision variables. Note that the threshold values used by RDG are the same as those used by GDG. The entries with the best DA achieved using the smallest number of FEs are highlighted in bold. Different categories of benchmark problems are divided by lines.

The RDG and GDG methods obtain nearly the same DA across all the benchmark problems. The reason for this is that RDG uses the same approach (25) as GDG to estimate the threshold values. However, RDG is much more efficient than GDG in terms of FEs used. Note that the number of FEs used by GDG to decompose an n-dimensional problem is fixed: $(n^2 + 3n + 2)/2$.

The first three problems ($f_1 - f_3$) from each benchmark suite are fully separable. Therefore, DA is not applicable to these problems. On f_1 and f_2 (CEC'2010 and CEC'2013), the RDG method successfully identifies all the decision variables as separable, using a small number of FEs. However on f_3, the RDG and GDG methods identify all the separable variables as nonseparable, and place them into one subcomponent. The reason for this is that the threshold value is underestimated by the RDG and GDG methods on f_3.

The CEC'2013 f_{13} and f_{14} are benchmark problems with overlapping (conforming or conflicting) subcomponents. It is not clear yet what is the best approach to decompose these problems [4], [21]. The RDG, GDG, and XDG methods place all the overlapped subcomponents into one group. On the other benchmark problems, where the subcomponents are independent with each other, the "ideal" decomposition can possibly be achieved (see [4], [10], [20], [21] for more information). Note that the 100% DA in Table II corresponds to the ideal decomposition.

On the CEC'2010 partially separable problems ($f_4 - f_{18}$), the RDG method consistently achieves the best results when compared against GDG, XDG, and DG. The RDG, GDG, and XDG methods achieve the ideal decomposition on all of these benchmark problems. However, the number of FEs used by the GDG and XDG methods is usually several magnitude larger than that used by the RDG method. The DG method performs well on the benchmark problems without conditional variable interactions. On f_{19}, the DG method uses the smallest number of FEs to decompose the problem. However on problems with conditional variable interactions (overlapping problems, e.g., CEC'2010 f_{13}, f_{18}, and f_{20}), the DA of the DG method is low. The reason for this is that DG is unable to completely identify variable interactions in overlapping problems [15], [46].

On the CEC'2013 partially separable problems ($f_4 - f_{11}$), the RDG method achieves the best results on 4 out of 8 benchmark problems. We observe that it is generally more difficult to identify the variable interactions in the CEC'2013 than the CEC'2010 benchmark problems. None of the four methods can perfectly decompose the CEC'2013 f_8, f_{10}, and f_{11} problems. On CEC'2013 f_{11}, the threshold value is underestimated by the RDG and GDG methods (25), resulting in placing all the decision variables into a single nonseparable group. While on CEC'2013 f_8 and f_{10}, the threshold value is overestimated, resulting in some omissions of variable interactions being identified.

The threshold values estimated by the GDG and RDG methods vary significantly across the CEC'2010 and CEC'2013 benchmark problems. Therefore, it is very difficult to find a single threshold value to accurately identify variable interactions across all the problems. Although the threshold value $\epsilon = 10^{-1}$ works well for the XDG method on the CEC'2010 benchmark problems, it fails on some of the CEC'2013 benchmark problems, e.g., $f_5 - f_8$. On f_5, the XDG algorithm (with $\epsilon = 10^{-1}$) identifies all the interacting decision variables as

TABLE II
EXPERIMENTAL RESULTS OF THE PROPOSED RDG METHOD WHEN USED TO DECOMPOSE THE CEC'2010 AND CEC'2013 BENCHMARK PROBLEMS. DA IS THE DECOMPOSITION ACCURACY; FES IS THE FES USED; ϵ IS THE THRESHOLD. NOTE THAT THE THRESHOLD VALUES USED BY RDG ARE THE SAME AS THOSE USED BY GDG. THE PERFORMANCE OF THE RDG METHOD IS COMPARED WITH THE PERFORMANCES OF THE GDG, XDG, AND DG METHODS. THE ENTRIES WITH THE BEST DA ACHIEVED USING THE LOWEST FES ARE HIGHLIGHTED IN BOLD

Bench-	Func	RDG			GDG		XDG ($\epsilon = 10^{-1}$)		DG ($\epsilon = 10^{-3}$)	
marks	Num	DA	FEs	ϵ	DA	FEs	DA	FEs	DA	FEs
CEC'2010	f_1	–	3.00e+03	4.11e-01	–	5.01e+05	–	1.00e+06	–	1.00e+06
	f_2	–	3.00e+03	2.49e-08	–	5.01e+05	–	1.00e+06	–	1.00e+06
	f_3	–	6.00e+03	2.15e-11	–	5.01e+05	–	1.00e+06	–	1.00e+06
	f_4	**100%**	**4.20e+03**	1.03e+04	100%	5.01e+05	100%	8.05e+04	100%	1.45e+04
	f_5	**100%**	**4.15e+03**	1.14e-03	100%	5.01e+05	100%	9.98e+05	100%	9.05e+05
	f_6	**100%**	**5.00e+04**	2.13e-05	100%	5.01e+05	100%	9.98e+05	100%	9.06e+05
	f_7	**100%**	**4.23e+03**	5.17e+00	100%	5.01e+05	100%	9.98e+05	68.0%	6.77e+04
	f_8	**100%**	**5.60e+03**	2.62e+05	100%	5.01e+05	100%	1.21e+05	90.0%	2.32e+04
	f_9	**100%**	**1.40e+04**	4.88e-01	100%	5.01e+05	100%	9.77e+05	100%	2.70e+05
	f_{10}	**100%**	**1.40e+04**	2.52e-08	100%	5.01e+05	100%	9.77e+05	100%	2.72e+05
	f_{11}	**100%**	**1.36e+04**	2.36e-10	100%	5.01e+05	100%	9.78e+05	99.8%	2.70e+05
	f_{12}	**100%**	**1.43e+04**	4.26e-05	100%	5.01e+05	100%	9.77e+05	100%	2.71e+05
	f_{13}	**100%**	**2.92e+04**	3.71e+00	100%	5.01e+05	100%	1.00e+06	31.8%	5.03e+04
	f_{14}	**100%**	**2.05e+04**	4.15e-01	100%	5.01e+05	100%	9.53e+05	100%	2.10e+04
	f_{15}	**100%**	**2.05e+04**	2.53e-08	100%	5.01e+05	100%	9.53e+05	100%	2.10e+04
	f_{16}	**100%**	**2.09e+04**	4.30e-10	100%	5.01e+05	100%	9.56e+05	99.6%	2.11e+04
	f_{17}	**100%**	**2.07e+04**	1.10e-04	100%	5.01e+05	100%	9.53e+05	100%	2.10e+04
	f_{18}	**100%**	**4.98e+04**	8.19e+00	100%	5.01e+05	100%	9.99e+05	23.0%	3.96e+04
	f_{19}	100%	6.00e+03	6.14e-04	100%	5.01e+05	100%	3.99e+03	**100%**	**2.00e+03**
	f_{20}	**100%**	**5.08e+04**	8.53e+00	100%	5.01e+05	100%	1.00e+06	28.7%	1.55e+05
CEC'2013	f_1	–	3.00e+03	4.20e-01	–	5.01e+05	–	1.00e+06	–	1.00e+06
	f_2	–	3.00e+03	1.31e-07	–	5.01e+05	–	1.00e+06	–	1.00e+06
	f_3	–	6.00e+03	2.16e-11	–	5.01e+05	–	1.00e+06	–	1.00e+06
	f_4	**100%**	**9.84e+03**	7.22e+01	100%	5.01e+05	33.3%	3.97e+05	95.3%	1.56e+04
	f_5	**100%**	**1.01e+04**	8.03e-05	100%	5.01e+05	0.00%	1.00e+06	0.00%	1.00e+06
	f_6	**100%**	**1.32e+04**	1.07e-06	100%	5.01e+05	50.0%	9.90e+05	82.6%	5.79e+05
	f_7	**100%**	**9.82e+03**	5.82e+05	100%	5.01e+05	33.3%	2.66e+04	39.6%	1.14e+04
	f_8	80.0%	1.95e+04	1.20e+06	80.0%	5.01e+05	10.0%	6.83e+04	**85.6%**	**2.26e+04**
	f_9	100%	1.92e+04	6.07e-03	100%	5.01e+05	99.9%	9.35e+05	**100%**	**1.76e+04**
	f_{10}	82.7%	1.91e+04	9.80e-05	**90.0%**	**5.01e+05**	79.6%	9.52e+05	79.8%	4.86e+04
	f_{11}	10.0%	1.06e+04	1.52e+06	10.0%	5.01e+05	10.0%	2.20e+04	**37.7%**	**9.10e+03**
	f_{12}	**100%**	**5.08e+04**	8.57e+00	100%	5.01e+05	100%	1.00e+06	39.0%	1.49e+05
	f_{13}	–	8.39e+03	1.83e+06	–	4.10e+05	–	1.29e+04	–	5.86e+03
	f_{14}	–	1.61e+04	5.45e+06	–	4.10e+05	–	3.57e+04	–	1.39e+04
	f_{15}	100%	6.16e+03	2.70e+06	100%	5.01e+05	100%	3.99e+03	**100%**	**2.00e+03**

separable. The reason for this is that the threshold value 10^{-1} is too large compared with the computational error, which is equal to 8.03×10^{-5} estimated by the RDG and GDG methods. When using the estimated computational error as the threshold value, the XDG method achieves equal DA with the RDG method.

To further show the efficacy of the RDG method, we compare the performance of RDG against two recently published methods—DG2 and FII. The average number of FEs used by RDG to decompose the CEC'2010 and CEC'2013 benchmark problems is 1.47×10^4, which is less than that used by the DG2 and FII methods: 4.95×10^5 and 4.94×10^4, respectively. The detailed decomposition results of the DG2 and FII methods are presented in the supplementary material.

The DG2 method uses a fixed number of FEs to decompose an n-dimensional problem: $(n^2 + n + 2)/2$, which has been shown to be the lower bound of identifying the complete variable interaction matrix. With the complete variable interaction matrix being identified, it is possible to generate an effective decomposition for the problems with overlapping subcomponents, e.g., CEC'2013 f_{13} and f_{14} [47]. However, the existing automatic decomposition methods place all the linked decision variables into one subcomponent.

DG2 is a parametric-free method, which automatically estimates the threshold values in the decomposition process. The DA of DG2 on the CEC'2010 and CEC'2013 benchmark problems is high. It achieves 100% DA for the CEC'2013 f_{10} and f_{11} problems. The variable interactions in these two problems are difficult for the other decomposition methods to identify. However on CEC'2013 f_7 and f_8, the DA of DG2 is less than that of RDG.

The FII method performs well when used to decompose the benchmark problems with a large portion of separable decision variables. For example on CEC'2010 f_4, where there are 950 separable and 50 nonseparable decision variables, the number of FEs used by FII (3.69×10^3) is slightly less than that used by RDG (4.20×10^3). However, on some benchmark problems especially those with conditional variable interactions,

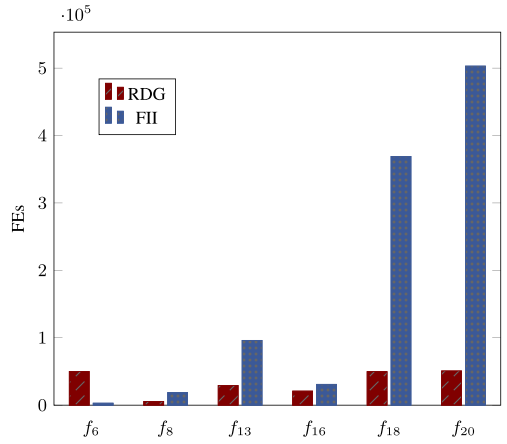


Fig. 3. Benchmark problems from the CEC'2010 test suite on which the RDG and FII methods generate significant different results (the difference between the number of FEs used is greater than 10^4).

TABLE III
EXTENDED CEC'2010 f_{12}, f_{15}, AND f_{18} PROBLEMS. FOR EACH PROBLEM, THE NUMBER OF DECISION VARIABLES IN EACH NONSEPARABLE SUBCOMPONENT IS FIXED TO 50, WHICH IS CONSISTENT WITH THE ORIGINAL BENCHMARK SET [20]

Func	Dim	Sep Variables	Non-Sep Groups
f_{12}	1000	500	10
	2000	1000	20
	3000	1500	30
	4000	2000	40
	5000	2500	50
f_{15}, f_{18}	1000	0	20
	2000	0	40
	3000	0	60
	4000	0	80
	5000	0	100

e.g., CEC'2010 f_{18} and f_{20}, RDG is much more efficient than FII, as shown in Fig. 3.

In fact, the number of FEs used by the FII method to decompose the CEC'2010 f_{18} and f_{20} problems is in $\Theta(n^2)$. It has been shown that FII uses $3n + kn_n + k$ FEs when decomposing an n-dimensional problem with equally sized nonseparable subcomponents, where n_n is the number of nonseparable decision variables, and k is the number of nonseparable subcomponents [48]. As CEC'2010 f_{18} and f_{20} are Rosenbrock's functions, n_n is equal to n and each decision variable interacts with at most two other decision variables. Therefore, the total number of FEs used by FII is around $3n+n^2/3+n/3 \in \Theta(n^2)$.

On CEC'2010 f_6, the number of FEs used by RDG (5.00×10^4) is greater than that used by FII (3.05×10^3). The reason for this is that the threshold estimated by the RDG method ($\epsilon = 2.13 \times 10^{-5}$) is too small, resulting in identifying some separable decision variables as nonseparable. If RDG employs the same threshold with FII ($\epsilon = 0.01$), the FEs used by RDG will decrease to 5.06×10^3. However, if FII employs the same threshold with RDG, the FEs used by FII will increase to 3.12×10^5.

To test the scalability of the RDG and FII methods, we extend some of the CEC'2010 benchmark problems from 1000 to 5000 dimensions (see Table III for details). When tested on the extended benchmark problems, we observe that the FEs used by the FII method increases more quickly than that used by the RDG method as the dimensionality increases, as shown in Fig. 4.

TABLE IV
AVERAGE RANKING OF EACH DECOMPOSITION METHOD WHEN EMBEDDED INTO THE DECC OR CMAESCC FRAMEWORK TO SOLVE THE CEC'2010 AND CEC'2013 BENCHMARK PROBLEMS. THE RDG METHOD CONSISTENTLY ACHIEVES THE SMALLEST AVERAGE RANKING

CCs	RDG	GDG	XDG	DG	DG2	FII	D	RG
DECC	2.0	4.9	4.0	4.4	3.9	2.5	4.6	4.7
CMAESCC	1.7	3.1	3.3	4.0	2.9	2.2	6.7	5.7

B. Optimization Comparison

The performances of the RDG, GDG, XDG, DG, DG2, FII, D (delta grouping), and RG methods when embedded in the DECC/CMAESCC framework to solve the CEC'2010 and CEC'2013 benchmark problems are presented in Figs. 5 and 6, respectively. On each benchmark problem, the eight methods are ranked from 1 to 8 (as labeled on the concentric circles in the radar charts) based on the results from 25 independent runs. The average ranking of each decomposition method across all the benchmark problems is presented in Table IV. The detailed optimization results from DECC-RDG and CMAESCC-RDG are presented in Tables V and VI. The results from the other seven methods are placed in the supplementary material due to page limits.

When embedded into the DECC/CMAESCC framework, the RDG method achieves the best solution quality when used to solve 16/15 out of 23 partially separable benchmark problems (CEC'2010 $f_4 - f_{18}$ and CEC'2013 $f_4 - f_{11}$). It obtains the smallest average ranking across all the benchmark problems investigated. The RDG method generally uses the smallest number of FEs in the decomposition stage, assigning more computational resources to optimize the problems, when compared against the other automatic decomposition methods.

On the fully separable and some fully nonseparable problems, the RDG method is outperformed by the two manual decomposition methods—D (delta grouping) and RG, as RDG does not actually perform any decomposition for these problems. However, the performance of the D (delta grouping) and RG methods deteriorates quickly on the partially separable problems.

On some benchmark problems with separable decision variables, e.g., CEC'2010 f_{10}, the GDG method generates better solution quality than RDG. The reason for this is that GDG further divides the separable decision variables into several subcomponents. Interestingly, the GDG method achieves the first place when embedded into CMAESCC, however, the last place when embedded into DECC, to solve the CEC'2010 f_9.

The DG method performs poorly on overlapping benchmark problems (e.g., CEC'2010 f_8, f_{13}, f_{18}, and f_{20}). The reason for this is that the DG method can not completely identify interaction between decision variables in an overlapping problem. Once all the variable interactions are identified and

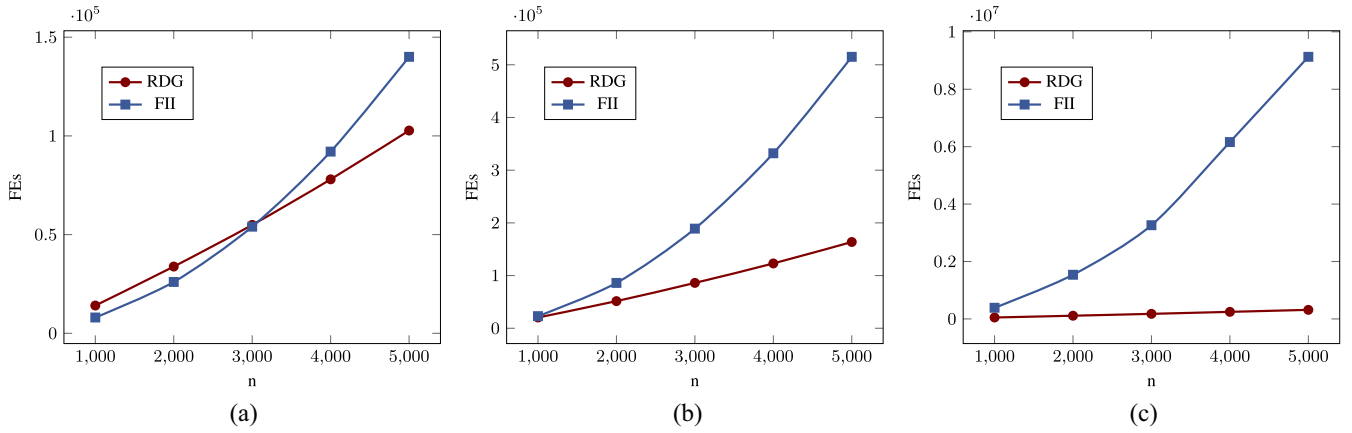


Fig. 4. Number of FEs used by RDG and FII methods when used to decompose the extended benchmark problems (CEC'2010 f_{12}, f_{15}, and f_{18}) with dimensionality equal to 1000, 2000, 3000, 4000, and 5000.

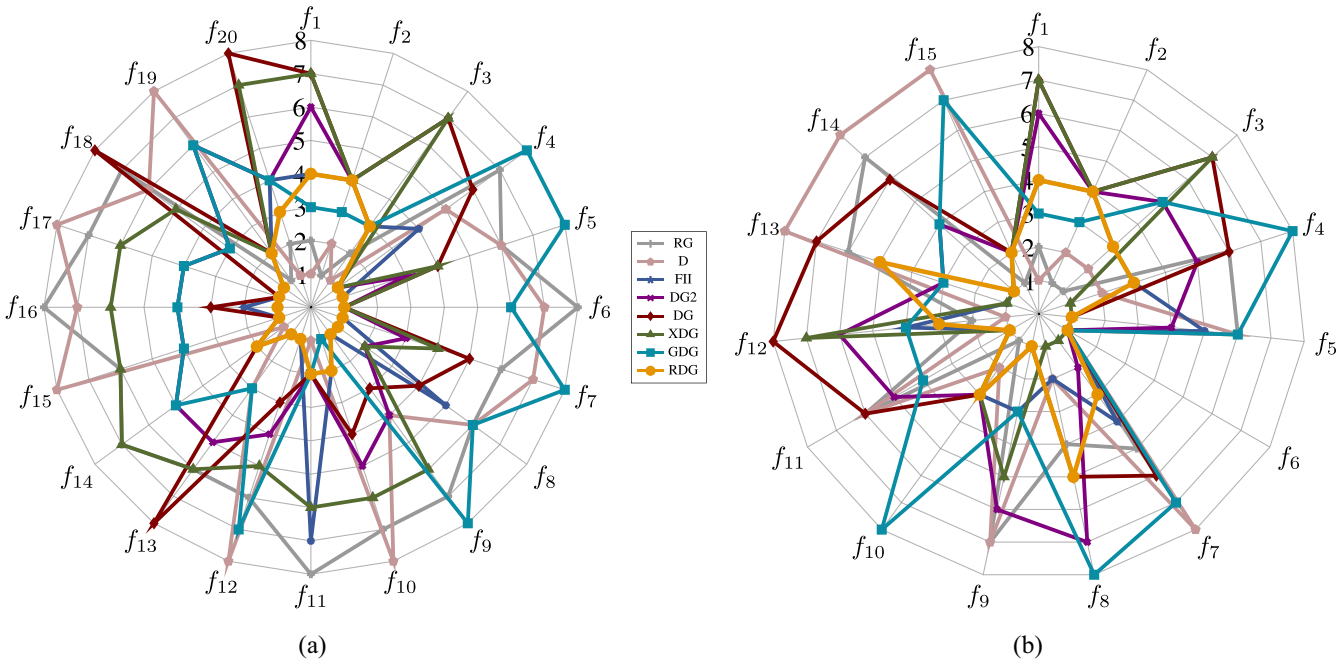


Fig. 5. Radar chart of the performance of RDG, GDG, XDG, DG, DG2, FII, D (delta grouping), and RG when embedded in the DECC framework to solve the CEC'2010 and CEC'2013 benchmark problems. On each benchmark problem, the eight methods are ranked from 1 to 8 (as labeled on the concentric circles) based on the results from 25 independent runs [Wilcoxon rank-sum tests ($\alpha = 0.05$) with Holm p-value correction].

all the linked decision variables are placed into one subcomponent, the solution quality can be greatly improved by several magnitudes.

The DG2 method generates the best solution quality when embedded into the CMAESCC framework to solve the CEC'2013 f_{11} problem. The reason for this is that DG2 achieves 100% accuracy when used to decompose this problem, which is higher than RDG. However on the other benchmark problems, the RDG method generally obtains equally well or statistically better solution quality than DG2.

The FII method performs well across the benchmark problems investigated. It achieves the second place according to the average ranking. However, on the benchmark problems where the decomposition by FII is less efficient, e.g., CEC'2010 f_8, the RDG method can generate significantly better solution quality than FII.

The DECC-D (with delta grouping) algorithm achieves much better results than the other DECC-based algorithms when used to solve the CEC'2010 f_3 and f_{11} benchmark problems. An interesting observation is that both f_3 and f_{11} are Ackley's functions [21]. Moreover, the DECC-D (with delta grouping) algorithm also performs well when used to solve the other Ackley's functions: CEC'2010 f_6, f_{16} and CEC'2013 f_3, f_6, f_{10}. So far, we do not know the reason why the DECC-D (with delta grouping) algorithm performs well on the Ackley's functions.

The convergence curves of the eight decomposition methods when embedded into the DECC/CMAESCC framework to solve the CEC'2010 f_{18} problem are shown in Fig. 7. The RDG method uses the smallest number of FEs in the decomposition stage, therefore can generate better solution quality when compared against the other automatic decomposition methods.

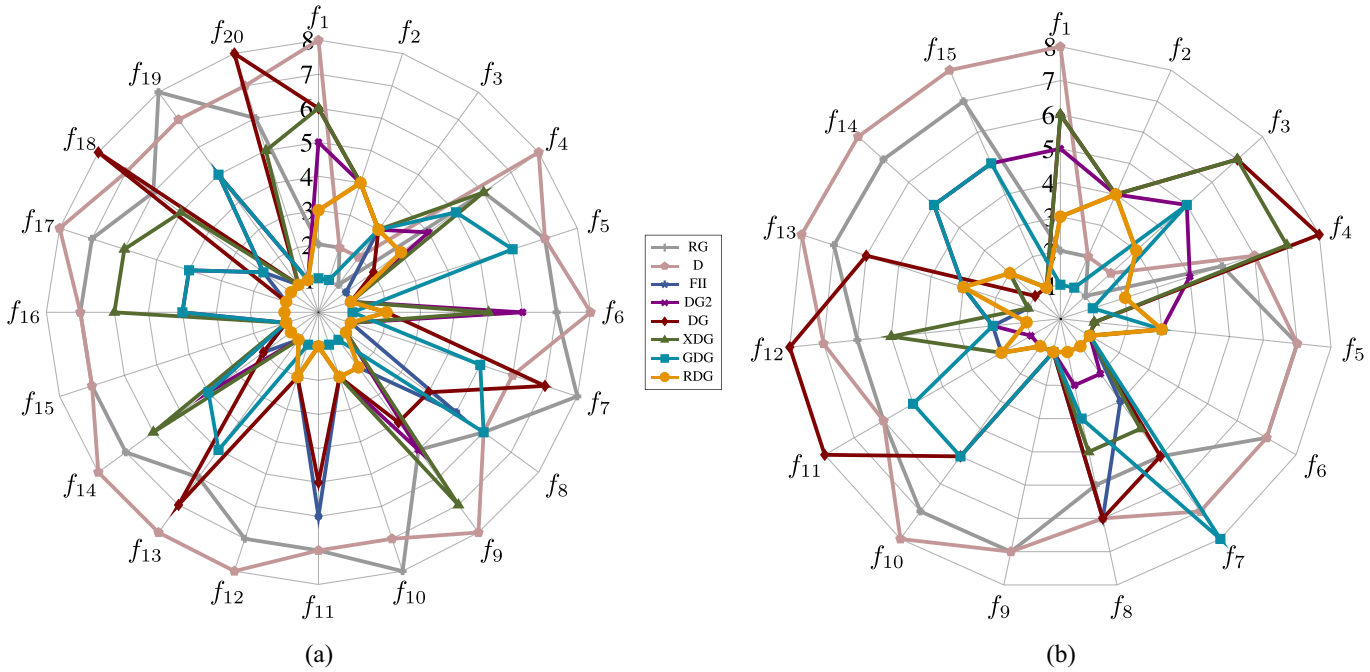


Fig. 6. Radar chart of the performance of RDG, GDG, XDG, DG, DG2, FII, D (delta grouping), and RG when embedded in the CMAESCC framework to solve the CEC'2010 and CEC'2013 benchmark problems. On each benchmark problem, the eight methods are ranked from 1 to 8 (as labeled on the concentric circles) based on the results from 25 independent runs [Wilcoxon rank-sum tests ($\alpha = 0.05$) with Holm p-value correction].

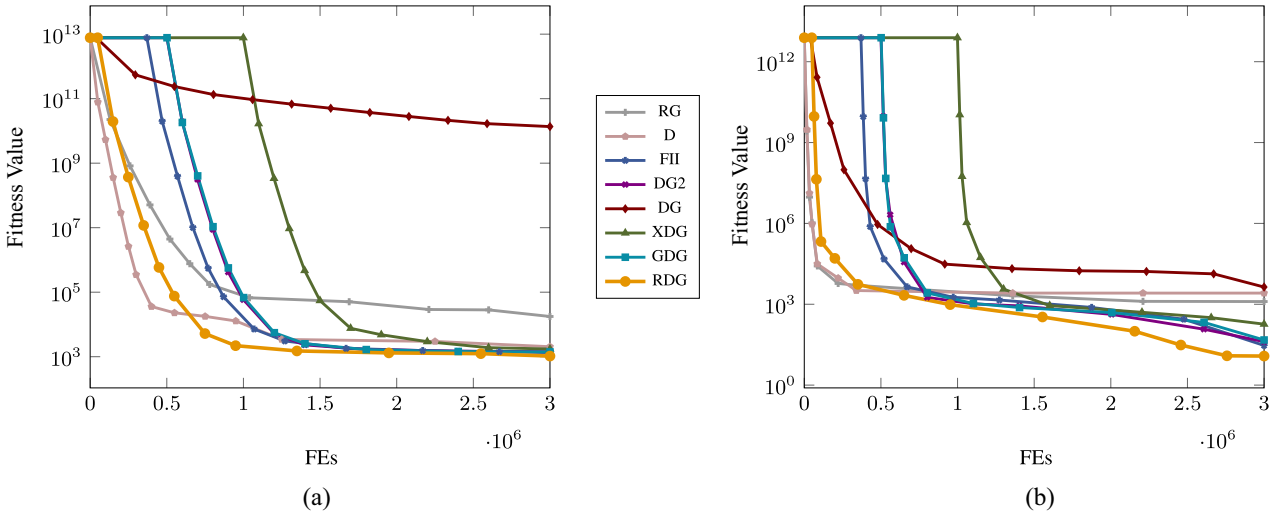


Fig. 7. Convergence curves of the RDG, GDG, XDG, DG, DG2, FII, D (delta grouping), and RG methods when embedded into the DECC and CMAESCC frameworks to solve the CEC'2010 f_{18} problem. The horizontal axis represents the number of FEs used in the evolutionary process. The vertical axis represents the median of the best fitness found.

In the next phase of the experimental study, we compare the performance of DECC-RDG and CMAESCC-RDG against the performance of two state-of-the-art algorithms—MOS and MA-SW-Chains. The experimental results of each algorithm when used to solve the CEC'2010 and CEC'2013 benchmark problems are presented in Tables V and VI, respectively.

The CMAESCC-RDG algorithm achieves the best solution quality on 22 out of 35 benchmark problems when compared against DECC-RDG, MOS, and MA-SW-Chains. It does not perform well on the fully separable problems (CEC'2010 f_1–f_3 and CEC'2013 f_1–f_3). However on partially separable and fully nonseparable problems, it generally achieves comparable or statistically better solution quality than the other algorithms (e.g., on CEC'2010 f_4–f_9 and f_{11}–f_{19}).

The DECC-RDG algorithm achieves the best solution quality when used to solve the CEC'2010 f_{16} problem. On the other 34 benchmark problems, the DECC-RDG is outperformed by the CMAESCC-RDG. It may indicate that the subcomponent optimizer used by the CMAESCC framework is more effective than that used by the DECC framework.

The MOS algorithm achieves the best results when used to solve the fully separable problems (CEC'2010 f_1–f_3, and

TABLE V
RESULTS OF THE DECC-RDG, CMAESCC-RDG, MOS, AND MA-SW-CHAINS ALGORITHMS WHEN USED TO SOLVE THE CEC'2010 BENCHMARK PROBLEMS. THE BEST PERFORMANCES ARE HIGHLIGHTED IN BOLD [WILCOXON RANK-SUM TESTS ($\alpha = 0.05$) WITH HOLM p-VALUE CORRECTION]

Func	Stats	DECC-RDG	CMAESCC-RDG	MOS	MA-SW-Chains
f_1	Median	1.50e-01	2.86e+05	**0.00e+00**	2.67e-14
	Mean	2.07e+00	2.84e+05	**1.50e-28**	3.80e-14
	Std	6.75e+00	2.28e+04	**5.55e-28**	4.91e-14
f_2	Median	4.35e+03	4.43e+03	**0.00e+00**	8.47e+02
	Mean	4.38e+03	4.42e+03	**0.00e+00**	8.40e+02
	Std	1.72e+02	1.76e+02	**0.00e+00**	4.88e+01
f_3	Median	1.65e+01	1.12e+00	**0.00e+00**	5.16e-13
	Mean	1.65e+01	1.05e+00	**0.00e+00**	5.76e-13
	Std	3.35e-01	3.49e-01	**0.00e+00**	2.73e-13
f_4	Median	5.75e+11	**9.97e+05**	4.94e+11	3.10e+11
	Mean	6.68e+11	**1.01e+06**	5.16e+11	2.97e+11
	Std	3.33e+11	**9.37e+04**	1.85e+11	6.19e+10
f_5	Median	1.31e+08	**9.05e+07**	5.00e+08	2.30e+08
	Mean	1.28e+08	**9.52e+07**	4.93e+08	2.18e+08
	Std	1.92e+07	**2.22e+07**	6.93e+07	5.75e+07
f_6	Median	1.61e+01	**1.04e+00**	1.97e+07	2.45e+00
	Mean	1.61e+01	**9.17e-01**	1.97e+07	1.42e+05
	Std	3.64e-01	**4.23e-01**	1.15e+05	3.96e+05
f_7	Median	2.46e+00	**7.41e-19**	2.27e+07	7.94e-03
	Mean	2.16e+01	**7.41e-19**	3.54e+07	1.17e+02
	Std	7.56e+01	**8.35e-20**	3.22e+07	2.37e+02
f_8	Median	3.66e+00	**1.83e-17**	2.14e+06	2.76e+06
	Mean	1.59e+05	**6.37e+05**	3.75e+06	6.90e+06
	Std	7.97e+05	**1.49e+06**	4.40e+06	1.90e+07
f_9	Median	4.65e+07	**4.80e+06**	1.18e+07	1.48e+07
	Mean	4.69e+07	**4.82e+06**	1.13e+07	1.49e+07
	Std	5.21e+06	**3.88e+05**	1.61e+06	1.61e+06
f_{10}	Median	4.33e+03	2.78e+03	6.35e+03	**2.02e+03**
	Mean	4.33e+03	2.79e+03	6.28e+03	**2.01e+03**
	Std	1.39e+02	1.17e+02	3.12e+02	**1.59e+02**
f_{11}	Median	1.03e+01	**1.51e-12**	2.84e+01	3.77e+01
	Mean	1.03e+01	**3.58e-02**	3.08e+01	3.86e+01
	Std	8.50e-01	**1.79e-01**	6.07e+00	8.06e+00
f_{12}	Median	1.38e+03	**4.30e-22**	3.46e+03	3.09e-06
	Mean	1.53e+03	**4.22e-22**	4.39e+03	3.24e-06
	Std	4.66e+02	**8.38e-23**	2.92e+03	5.78e-07
f_{13}	Median	6.12e+02	**3.98e+00**	3.19e+02	8.61e+02
	Mean	7.12e+02	**4.78e+00**	3.32e+02	9.83e+02
	Std	2.52e+02	**3.98e+00**	1.19e+02	5.66e+02
f_{14}	Median	3.47e+08	**3.90e-20**	2.04e+07	3.23e+07
	Mean	3.47e+08	**3.91e-20**	2.05e+07	3.25e+07
	Std	2.31e+07	**2.11e-20**	3.60e+06	2.46e+06
f_{15}	Median	5.82e+03	**1.92e+03**	1.29e+04	2.67e+03
	Mean	5.84e+03	**1.94e+03**	1.29e+04	2.68e+03
	Std	1.01e+02	**1.10e+02**	3.48e+02	9.95e+01
f_{16}	Median	**2.66e-13**	8.41e-13	3.97e+02	9.32e+01
	Mean	**2.67e-13**	8.43e-13	3.96e+02	9.95e+01
	Std	**9.81e-15**	2.10e-14	3.47e+00	1.53e+01
f_{17}	Median	4.08e+04	**6.89e-24**	7.30e+03	1.28e+00
	Mean	4.07e+04	**6.90e-24**	8.45e+03	1.27e+00
	Std	2.55e+03	**2.05e-25**	5.04e+03	1.24e-01
f_{18}	Median	1.19e+03	**1.55e+01**	7.78e+02	1.41e+03
	Mean	1.20e+03	**1.50e+01**	8.96e+02	1.57e+03
	Std	1.07e+02	**7.19e+00**	4.03e+02	6.73e+02
f_{19}	Median	1.71e+06	**5.63e+03**	5.71e+05	3.75e+05
	Mean	1.71e+06	**5.46e+03**	5.49e+05	3.80e+05
	Std	8.91e+04	**7.07e+02**	8.38e+04	2.34e+04
f_{20}	Median	3.70e+03	8.55e+02	**7.40e+01**	1.04e+03
	Mean	6.96e+03	8.26e+02	**9.23e+01**	1.06e+03
	Std	1.27e+04	6.35e+01	**8.99e+01**	9.38e+01

CEC'2013 f_1–f_3). However, on partially separable problems and fully nonseparable problems, it is generally outperformed by the CMAESCC-RDG algorithm.

TABLE VI
RESULTS OF THE DECC-RDG, CMAESCC-RDG, MOS, AND MA-SW-CHAINS ALGORITHMS WHEN USED TO SOLVE THE CEC'2013 BENCHMARK PROBLEMS. THE BEST PERFORMANCES ARE HIGHLIGHTED IN BOLD [WILCOXON RANK-SUM TESTS ($\alpha = 0.05$) WITH HOLM p-VALUE CORRECTION]

Func	Stats	DECC-RDG	CMAESCC-RDG	MOS	MA-SW-Chains
f_1	Median	5.32e-01	2.84e+05	**1.34e-30**	7.12e-13
	Mean	3.73e+01	2.89e+05	**3.10e-29**	1.34e-12
	Std	1.24e+02	3.27e+04	**4.53e-29**	2.45e-12
f_2	Median	1.29e+04	4.66e+03	**1.90e+01**	1.24e+03
	Mean	1.27e+04	4.68e+03	**1.83e+01**	1.25e+03
	Std	6.40e+02	1.77e+02	**4.65e+00**	1.05e+02
f_3	Median	2.13e+01	2.03e+01	**1.49e-13**	6.83e-13
	Mean	2.13e+01	2.03e+01	**1.65e-13**	6.85e-13
	Std	1.64e-02	4.96e-02	**1.02e-13**	2.12e-13
f_4	Median	4.01e+10	**5.83e+06**	1.23e+10	2.75e+09
	Mean	4.44e+10	**5.90e+06**	1.40e+10	3.81e+09
	Std	1.77e+10	**6.56e+05**	7.65e+09	2.73e+09
f_5	Median	5.09e+06	**2.19e+06**	1.12e+07	**2.03e+06**
	Mean	5.09e+06	**2.20e+06**	1.15e+07	**2.25e+06**
	Std	4.81e+05	**3.76e+05**	1.82e+06	**1.30e+06**
f_6	Median	1.06e+06	9.95e+05	9.78e+05	**6.33e+02**
	Mean	1.06e+06	9.95e+05	9.83e+05	**1.86e+04**
	Std	1.21e+03	2.88e+01	8.22e+03	**2.54e+04**
f_7	Median	5.41e+07	**2.94e-20**	1.30e+07	4.03e+06
	Mean	6.42e+07	**8.12e-17**	2.33+07	3.85e+06
	Std	2.97e+07	**2.17e-16**	3.62e+07	6.34e+05
f_8	Median	4.74e+15	**8.71e+06**	1.15e+15	4.60e+13
	Mean	5.04e+15	**9.74e+06**	1.65e+15	4.62e+13
	Std	1.86e+15	**5.83e+06**	1.76e+15	9.02e+12
f_9	Median	4.85e+08	1.57e+08	9.08e+08	**1.42e+08**
	Mean	4.82e+08	1.65e+08	9.02e+08	**1.44e+08**
	Std	3.06e+07	4.16e+07	1.04e+08	**1.55e+07**
f_{10}	Median	9.44e+07	9.04e+07	8.82e+07	**3.34e+02**
	Mean	9.44e+07	9.12e+07	6.66e+07	**3.72e+04**
	Std	2.06e+05	1.53e+06	3.01e+07	**6.25e+04**
f_{11}	Median	5.31e+08	**1.64e+07**	1.82e+09	2.10e+08
	Mean	5.38e+08	**1.62e+07**	3.87e+10	2.10e+08
	Std	1.34e+08	**6.11e+05**	1.06e+11	2.35e+07
f_{12}	Median	3.77e+03	1.01e+03	**6.89e+01**	1.25e+03
	Mean	4.85e+03	9.81e+02	**8.64e+01**	1.24e+03
	Std	3.06e+03	7.30e+01	**7.82e+01**	8.33e+01
f_{13}	Median	3.16e+09	**2.49e+06**	8.45e+08	1.91e+07
	Mean	3.06e+09	**2.47e+06**	1.09e+09	3.58e+07
	Std	6.68e+08	**3.83e+05**	7.69e+08	4.30e+07
f_{14}	Median	2.50e+09	**2.74e+07**	2.27e+09	1.43e+08
	Mean	2.87e+09	**2.76e+07**	6.65e+09	1.45e+08
	Std	1.73e+09	**1.49e+06**	1.62e+10	1.60e+07
f_{15}	Median	9.67e+06	**2.18e+06**	1.24e+08	5.80e+06
	Mean	9.75e+06	**2.19e+06**	1.33e+08	5.98e+06
	Std	1.91e+06	**2.28e+05**	6.05e+07	1.42e+06

The MA-SW-Chains algorithm performs well on the CEC'2013 benchmark problems. It achieves the best results when used to solve the CEC'2013 f_5, f_6, f_9, and f_{10} problems. However, on the other partially separable and fully nonseparable benchmark problems, it is consistently outperformed by the CMAESCC-RDG algorithm. In some cases, the best solution found by the CMAESCC-RDG algorithm is much better than that found by the MA-SW-Chains algorithm.

VI. Conclusion

In this paper, we have investigated the influence of problem decomposition on the performance of CC algorithms when used to solve LSGO problems. A robust decomposition method—RDG—was proposed, which can decompose an n-dimensional problem using $\mathcal{O}(n \log(n))$ FEs based on a measure of nonlinearity between decision variables. Significantly, RDG outperformed seven other decomposition methods when embedded into the DECC/CMAESCC framework and tested across a suite of benchmark LSGO problems. When compared against two other state-of-the-art hybrid algorithms, the CMAESCC-RDG algorithm achieved statistically significantly better results.

In future work, we plan to apply the CMAESCC-RDG algorithm to solve real-world LSGO problems. Another direction worth pursuing is focused on the modification of the RDG method to the combinatorial and multiobjective spaces.

Acknowledgment

The authors would like to thank M. A. Muñoz and W. Wang for their valuable comments.

References

[1] P. Benner, "Solving large-scale control problems," *IEEE Control Syst.*, vol. 24, no. 1, pp. 44–59, Feb. 2004.

[2] S. Shan and G. G. Wang, "Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions," *Struct. Multidisciplinary Optim.*, vol. 41, no. 2, pp. 219–241, 2010.

[3] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 4, pp. 491–502, Apr. 2005.

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

[5] T. Weise, R. Chiong, and K. Tang, "Evolutionary optimization: Pitfalls and booby traps," *J. Comput. Sci. Technol.*, vol. 27, no. 5, pp. 907–936, 2012.

[6] W. Dong, T. Chen, P. Tino, and X. Yao, "Scaling up estimation of distribution algorithms for continuous optimization," *IEEE Trans. Evol. Comput.*, vol. 17, no. 6, pp. 797–822, Dec. 2013.

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

[8] A. LaTorre, S. Muelas, and J.-M. Peña, "A comprehensive comparison of large scale global optimizers," *Inf. Sci.*, vol. 316, pp. 517–549, Sep. 2015.

[9] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving From Nature PPSN III*. Heidelberg, Germany: Springer, 1994, pp. 249–257.

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

[11] Y. Mei, X. Li, and X. Yao, "Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 435–449, Jun. 2014.

[12] E. Sayed, D. Essam, R. Sarker, and S. Elsayed, "Decomposition-based evolutionary algorithm for large scale constrained problems," *Inf. Sci.*, vol. 316, pp. 457–486, Sep. 2015.

[13] K. C. Tan, Y. J. Yang, and C. K. Goh, "A distributed cooperative coevolutionary algorithm for multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 527–549, Oct. 2006.

[14] C.-K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 1, pp. 103–127, Feb. 2009.

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

[16] W. Chen and K. Tang, "Impact of problem decomposition on cooperative coevolution," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, Mexico, 2013, pp. 733–740.

[17] H. Liu, Y. Wang, X. Liu, and S. Guan, "Empirical study of effect of grouping strategies for large scale optimization," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Vancouver, BC, Canada, 2016, pp. 3433–3439.

[18] B. Kazimipour, M. N. Omidvar, X. Li, and A. Qin, "A sensitivity analysis of contribution-based cooperative co-evolutionary algorithms," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Sendai, Japan, 2015, pp. 417–424.

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

[20] K. Tang, X. Yao, P. Suganthan, Z. Yang, and T. Weise "Benchmark functions for the CEC'2010 special session and competition on large scale global optimization," Nat. Inspired Comput. Appl. Lab., USTC, Hefei, China, Tech. Rep., 2010, pp. 1–23.

[21] X. Li, K. Tang, M. N. Omidvar, Z. Yang, and K. Qin, "Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization," *Gene*, vol. 7, no. 33, p. 8, 2013.

[22] R.-L. Tang, Z. Wu, and Y.-J. Fang, "Adaptive multi-context cooperatively coevolving particle swarm optimization for large-scale problems," *Soft Comput.*, vol. 21, no. 16, pp. 4735–4754, 2016.

[23] M. Yang *et al.*, "Efficient resource allocation in cooperative co-evolution for large-scale global optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 493–505, Aug. 2017.

[24] A. Kabán, J. Bootkrajang, and R. J. Durrant, "Toward large-scale continuous EDA: A random matrix theory perspective," *Evol. Comput.*, vol. 24, no. 2, pp. 255–291, Jun. 2016.

[25] J. Brest and M. S. Maučec, "Self-adaptive differential evolution algorithm using population size reduction and three strategies," *Soft Comput.*, vol. 15, no. 11, pp. 2157–2174, 2011.

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

[27] R. Cheng and Y. Jin, "A social learning particle swarm optimization algorithm for scalable optimization," *Inf. Sci.*, vol. 291, pp. 43–60, Jan. 2015.

[28] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, Feb. 2015.

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

[30] A. LaTorre, S. Muelas, and J.-M. Peña, "A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: A scalability test," *Soft Comput.*, vol. 15, no. 11, pp. 2187–2199, 2011.

[31] Y. Sun, M. Kirley, and S. K. Halgamuge, "Quantifying variable interactions in continuous optimization problems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 249–264, Apr. 2017, doi: 10.1109/TEVC.2016.2599164.

[32] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.

[33] X.-W. Zheng, D.-J. Lu, X.-G. Wang, and H. Liu, "A cooperative coevolutionary biogeography-based optimizer," *Appl. Intell.*, vol. 43, no. 1, pp. 95–111, 2015.

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

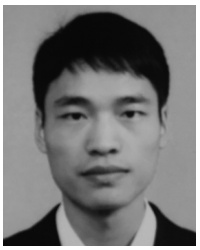
[35] Y. Ren and Y. Wu, "An efficient algorithm for high-dimensional function optimization," *Soft Comput.*, vol. 17, no. 6, pp. 995–1004, 2013.

[36] M. N. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative co-evolution for large scale optimization through more frequent random grouping," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Barcelona, Spain, 2010, pp. 1–8.

[37] Z. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in *Proc. IEEE Congr. Evol. Comput. CEC (IEEE World Congr. Comput. Intell.)*, Hong Kong, 2008, pp. 1663–1670.

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

[39] S. Mahdavi, S. Rahnamayan, and M. E. Shiri, "Multilevel framework for large-scale global optimization," *Soft Comput.*, vol. 21, no. 14, pp. 4111–4140, 2017.
[40] M. N. Omidvar, X. Li, and X. Yao, "Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms," in *Proc. 13th Annu. Conf. Genet. Evol. Comput.*, Dublin, Ireland, 2011, pp. 1115–1122.
[41] W. Chen, T. Weise, Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *Parallel Problem Solving From Nature, PPSN XI*. Heidelberg, Germany: Springer, 2010, pp. 300–309.
[42] M. Munetomo and D. E. Goldberg, "Linkage identification by non-monotonicity detection for overlapping functions," *Evol. Comput.*, vol. 7, no. 4, pp. 377–398, 1999.
[43] L. Sun, S. Yoshida, X. Cheng, and Y. Liang, "A cooperative particle swarm optimizer with statistical variable interdependence learning," *Inf. Sci.*, vol. 186, no. 1, pp. 20–39, 2012.
[44] H. Ge, L. Sun, X. Yang, S. Yoshida, and Y. Liang, "Cooperative differential evolution with fast variable interdependence learning and cross-cluster mutation," *Appl. Soft Comput.*, vol. 36, pp. 300–314, Nov. 2015.
[45] M. Tezuka, M. Munetomo, and K. Akama, "Linkage identification by nonlinearity check for real-coded genetic algorithms," in *Genetic and Evolutionary Computation–GECCO 2004*. Heidelberg, Germany: Springer, 2004, pp. 222–233.
[46] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Trans. Math. Softw.*, vol. 42, no. 2, p. 13, 2016.
[47] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, "DG2: A faster and more accurate differential grouping for large-scale black-box optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 6, pp. 929–942, Dec. 2017.
[48] X.-M. Hu, F.-L. He, W.-N. Chen, and J. Zhang, "Cooperation coevolution with fast interdependency identification for large scale optimization," *Inf. Sci.*, vol. 381, pp. 142–160, Mar. 2017.
[49] Z. Yang, K. Tang, and X. Yao, "Self-adaptive differential evolution with neighborhood search," in *Proc. IEEE Congr. Evol. Comput. CEC (IEEE World Congr. Comput. Intell.)*, Hong Kong, 2008, pp. 1110–1116.
[50] N. Hansen, "The CMA evolution strategy: A tutorial," Mach. Learn. Optim. Group, Inria, France, Rep., 2011.
[51] D. Molina, M. Lozano, and F. Herrera, "MA-SW-Chains: Memetic algorithm based on local search chains for large scale continuous global optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Barcelona, Spain, 2010, pp. 1–8.
[52] D. J. Sheskin, *Handbook Of Parametric and Nonparametric Statistical Procedures*. Boca Raton, FL, USA: CRC Press, 2003.



Yuan Sun received the B.Sc. degree in theoretical and applied mechanics from Peking University, Beijing, China, in 2013. He is currently pursuing the Ph.D. degree with the University of Melbourne, Parkville, VIC, Australia.

His doctoral research focuses on the analysis of interaction between decision variables in optimization and classification problems. His current research interests include exploratory landscape analysis, large scale optimization, multiobjective optimization, and feature selection.

Michael Kirley received the B.Ed. degree from Deakin University, Burwood, VIC, Australia, in 1998 and the Ph.D. degree from Charles Sturt University, Bathurst, NSW, Australia, in 2003.

He is currently an Associate Professor with the Department of Computing and Information Systems, University of Melbourne, Parkville, VIC, Australia. His current research interests include theory and application of evolutionary computation, evolutionary game theory, multiagent system, and complex systems science. He has published over 100 papers in the above areas.

Saman K. Halgamuge received the B.Sc.Eng. degree in electronic and telecommunication engineering from the University of Moratuwa, Moratuwa, Sri Lanka, and the Dr.-Ing and Dipl.-Ing degrees in electrical engineering (data engineering) from TU Darmstadt, Darmstadt, Germany, in 1990 and 1995, respectively.

He is a Professor and the Director of the Research School of Engineering, Australian National University, Canberra, ACT, Australia. He held appointments as a Professor with the Department of Mechanical Engineering and the Associate Dean International with the Melbourne School of Engineering, University of Melbourne, Parkville, VIC, Australia. He also holds the Professor V. K. Samaranayake Endowed Visiting Chair with the University of Colombo, Colombo, Sri Lanka. His current research interests include data engineering, optimization, smart grids, mechatronics, and bioinformatics. He has published over 250 papers and graduated 32 Ph.D. students in the above areas and obtained substantial research grants from the ARC, NHMRS, and industry.