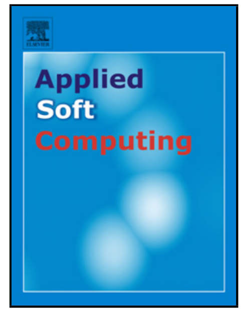
Mixed second order partial derivatives decomposition method for large scale optimization

Li, Lin; Jiao, Licheng; Stolkin, Rustam; Liu, Fang

DOI:
10.1016/j.asoc.2017.08.025



Title: Mixed Second Order Partial Derivatives Decomposition Method for Large Scale Optimization

Author: Lin Li Licheng Jiao Rustam Stolkin Fang Liu

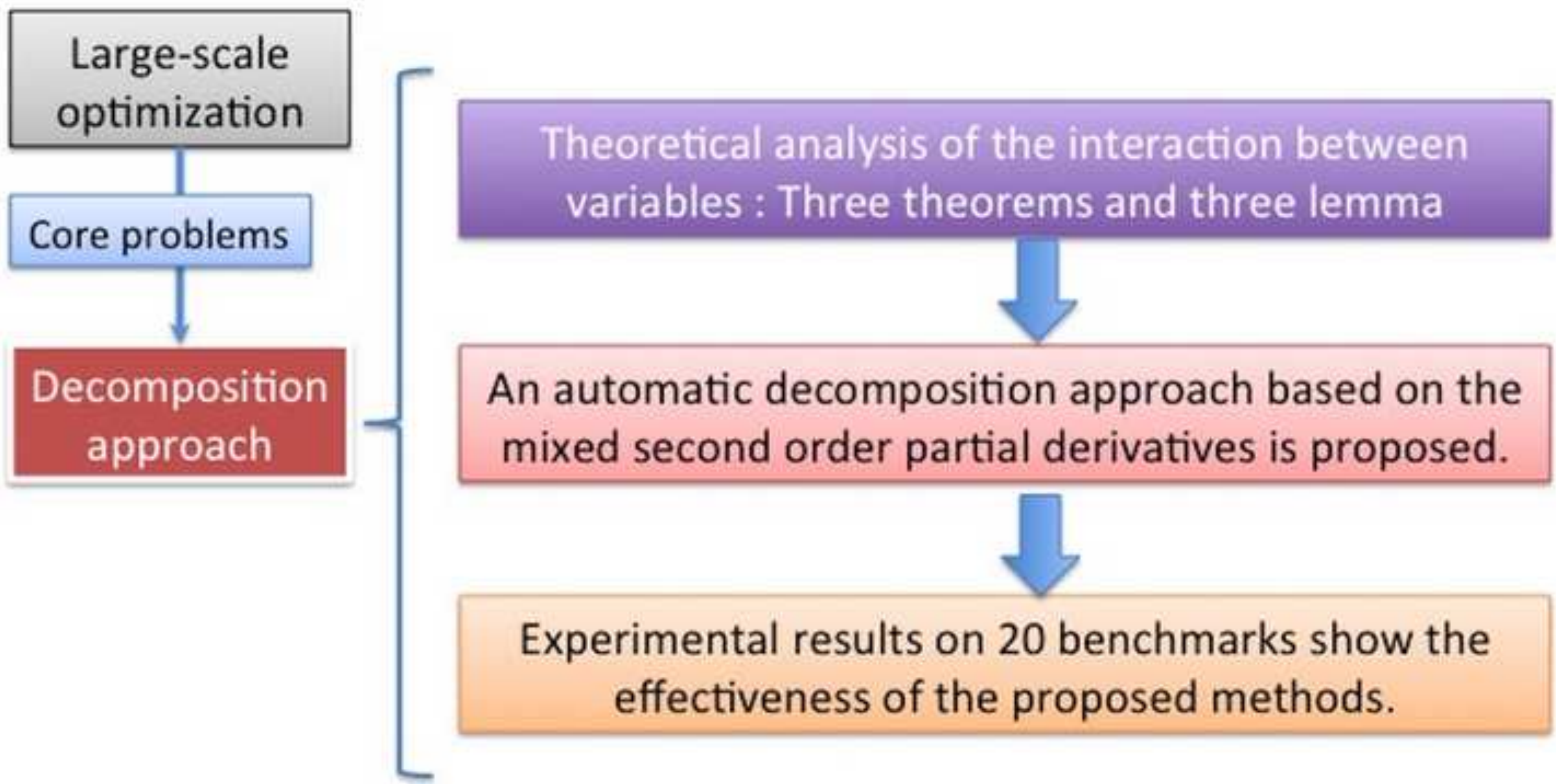
PII: S1568-4946(17)30507-0
DOI: http://dx.doi.org/doi:10.1016/j.asoc.2017.08.025
Reference: ASOC 4415

To appear in: *Applied Soft Computing*

Received date: 11-12-2015
Revised date: 16-5-2017
Accepted date: 8-8-2017

1. A theoretical analysis of the interaction between variables is developed.
2. Three theorems and three lemma are presented, as well as a theoretical explanation of overlapping subcomponents.
3. A decomposition approach based on the mixed second order partial derivatives of the analytic expression of the optimization problems is proposed.

*Graphical abstract (for review)



Mixed Second Order Partial Derivatives Decomposition Method for Large Scale Optimization

Lin Li[a,*], Licheng Jiao[b,**], Rustam Stolkin[c], Fang Liu[b]

[a] *Key Laboratory of Information Fusion Technology of Ministry of Education, School of Automation, Northwestern Polytechnical University, Xi'an, Shaanxi Province, 710072, PR China*
[b] *Key Lab of Intelligent Perception and Image Understanding of Ministry of Education, Xidian University, Xi'an, 710071, China*
[c] *Extreme Robotics Lab, University of Birmingham, Edgbaston, Birmingham B152TT, U.K.*

Abstract

This paper focuses on decomposition strategies for large-scale optimization problems. The cooperative co-evolution approach improves the scalability of evolutionary algorithms by decomposing a single high dimensional problem into several lower dimension sub-problems and then optimizing each of them individually. However, the dominating factor for the performance of these algorithms, on large-scale function optimization problems, is the choice of the decomposition approach employed. This paper provides a theoretical analysis of the interaction between variables in such approaches. Three theorems and three lemma are introduced to investigate the relationship between decision variables, and we provide theoretical explanations on overlapping subcomponents. An automatic decomposition approach, based on the mixed second order partial derivatives of the analytic expression of the optimization problem, is presented. We investigate the advantages and disadvantages of the differential grouping (DG) automatic decomposition approach, and we propose one enhanced version of differential grouping to deal with problems which the original differential grouping method is unable to resolve. We compare the performance of three different grouping strategies and provide the results of empirical evaluations using 20 benchmark data sets.

Keywords: Large-scale optimization, Evolutionary Algorithm, Cooperative

*Corresponding author: Tel.: +86 02988431307; Fax: +86 02988431306
**Corresponding author: Tel.: +86 029 8820 9786; Fax: +86 029 8820 1023.
Email addresses: linli@nwpu.edu.cn (Lin Li), lchjiao@mail.xidian.edu.cn (Licheng Jiao)

Preprint submitted to Applied Soft Computing *August 17, 2017*

Co-evolution, Divide-and-Conquer, Decomposition Method, Nonseparability, Curse of Dimensionality.

1. Introduction

1.1. Overview

The solution of large optimization problems has attracted increasing attention from the evolutionary computation community in recent years [1, 2, 3]. A wide variety of metaheuristic optimization algorithms have been proposed during the past few decades, such as Genetic Algorithms [4, 5], Evolutionary Algorithms (EAs) [6, 7, 8, 9, 10], Particle Swarm Optimization (PSO) [11, 12], Differential Evolution (DE) [13, 14], Simulated Annealing [15, 16], Ant Colony Optimization [17, 18], Evolutionary Programming (EP). While these methods have been successfully applied to theoretical and real-world optimization problems, their application to problems of large dimension (e.g. problems with more than one hundred decision variables) remain problematic. This paper discusses techniques for solving such large-scale optimization (LSO) problems.

The performance of many metaheuristic methods deteriorates rapidly with the increase in dimension of the decision variables, referred to as the “curse of dimensionality” in much of the literature [19, 20]. There are two reasons for this phenomenon [21]. Firstly, the search space grows exponentially with dimension, engendering much greater computation time in algorithms which performed well on low dimensional spaces. Secondly, the complexity of an optimization problem may change with high dimensions, making the search for optimal solutions more difficult. For example, Rosenbrocks function is unimodal when there are only two variables but it becomes multimodal when the dimension is larger than four [22, 23]. An intuitive yet efficient way to deal with this predicament is to decompose the original large-scale optimization problem into a group of smaller and less complex sub-problems, and then handle each sub-problems separately. This is known as a “divide-and-conquer” strategy and it has been successfully applied in many areas [19, 24, 25, 26, 27].

The cooperative co-evolution (CC) method, proposed by Potter and De Jong in [28], provided a new way to solve more complex structures such as neural networks and rule sets, and its performance has since been tested on well-studied optimization problems. The scalability of CC to large-scale decision variables was explored in [29], which suggested that the CC framework for large-scale problems is very sensitive to the choice of decomposition strategy for grouping the different subcomponents. This paper therefore focuses on the decomposition problem.

1.2. Motivation of this paper

The main contributions of this paper can be summarized as follows:

- A theoretical analysis of the interaction between variables is developed. Three theorems and three lemma are presented, as well as a theoretical explanation of overlapping subcomponents.
- A decomposition approach based on the mixed second order partial derivatives of the analytic expression of the optimization problems is proposed.
- An investigation and discussion of the advantages and disadvantages of the automatic decomposition approach DG [20] is presented, and we also propose an enhanced version of DG to address problems which the original DG method is not capable of solving.
- Experimental results on 20 benchmarks are presented, which show the effectiveness of the proposed decomposition methods.

1.3. Layout of this paper

Section 2 surveys the various decomposition techniques employed within the CC framework in the literature, and the techniques most related to our proposed method (CCVIL [30] and DG [20]) are explained in detail. In section 3, the variable interaction problems and the proposed theory and approaches are introduced in detail. Experimental results and discussion are presented in section 4. Section 5 summarizes and provides concluding remarks.

2. CC decomposition methods

According to [20], decomposition strategies can be classified into four categories: random methods, perturbation methods, interaction adaptation, and model building. In contrast, we suggest dividing CC grouping approaches into three decomposition methods, based on their respective strategies for deciding the total number and size of the sub-groups:

1) fixed-size grouping methods, e.g. CCGA, CCGA-1 [28], FEPCC [29], and the random grouping strategy used by DECC-G [31];

2) adaptive-size grouping methods, e.g. correlation based adaptive variable partitioning technique (CCEA-AVP) [32], delta grouping [33], MLCC [34];

3) automatic grouping methods, e.g. CCVIL [30] and differential grouping (DG) [20].

Table 1: Comparison of grouping strategies between different algorithms based on CC framework

Decomposition Categories	Algorithms	Grouping method
Fixed-size grouping	CCGA,CCGA1 [28], FEPCC [29]	1-D decomposition
	DECC-G [31]	Random groping (RG)
	DECC-D [33]	Delta grouping (DLG)
Adaptive-size grouping	CCEA-AVP [32]	Correlation based adaptive variable partitioning technique
	MLCC [34]	RG with performance based self-adaptive subgroup size
	DECC-ML [35]	More frequent RG with random self-adaptive subgroup size
	DECC-DML [33]	DLG with random self-adaptive subgroup size
Automatic grouping	CCVIL [30]	Variable interaction learning
	DECC-DG,CBCC-DG [20]	Differential grouping

2.1. Fixed-size grouping

Fixed-size grouping methods are those which divide an n-dimensional problem into k modules with m dimensions ($m << n$) and then solve each module with a particular optimizer (such as, GAs, EAs, EP, PSO) separately and cooperatively. We refer to such methods as m-D decomposition throughout the remainder of this paper. Algorithms CCGA, CCGA1 [28] and FEPCC [29] adopt a 1-D decomposition strategy, which decomposes the original optimization problems into n one dimension sub-problems and then optimize each sub-problem with GA and Fast EP, respectively. CCGA and CCGA1 have shown poor performance on non-separable problems with maximum of 30 decision variables [28]. FEPCC [29] has previously been scaled successfully to 1000 dimension problems with separable functions, but the performance on non-separable optimization problems remains unclear.

m-D decomposition with $m << n$ was employed in [36], which applied PSO as the optimizer within a CC framework, known as the cooperative particle swarm optimizer (CPSO). CPSO has shown significant improvement over traditional PSO on several benchmark optimization problems. However, CPSO was not tested on large-scale problems. In [37], the cooperative co-evolutionary differential evolution (CCDE) was proposed. $\frac{n}{2}$-D decomposition method was applied and the optimizer in the CC framework was DE. However, splitting up the decision variables into two equally sized sub-groups arbitrarily does not improve the scalability of the proposed method.

The main drawback for of the above m-D decomposition methods are that they are static decomposition strategies. If such a method does not correctly identify the appropriate subgroups, then it can never find the right subcomponents of the problems. This is one of the reasons why such m-D decomposition strategies have difficulty solving non-separable optimization problems. Here we refer to such m-D decomposition methods as static m-D decomposition.

In contrast, a decomposition strategy known as random grouping was proposed by Yang et al. [31] to improve the ability of CC framework for optimization problems with interaction decision variables. Similar to the static m-D decomposition, random grouping decomposes the problem into k m-dimensional subcomponents, but the m decision variables are randomly selected in each cycle. It can be shown that random grouping increaseses the probability of grouping two non-separable variables into the same subcomponent for several cycles.

The proposed method (DECC-G) in [31] adopted random grouping and adaptive weighting for dividing the original optimization problems, and each subcomponent was then optimized by a DE algorithm. The experimental results on a set of benchmark problems up to 1000 dimensions, showed that random grouping achieved good performance on detecting interacting variables. In [38], Li and Xin proposed algorithm CCPSO2 by employing the random grouping within the CC framework with PSO as the optimizer. CCPSO2 was tested on problems of up to 2000 decision variables to show the scalability of PSO. Although random grouping has shown advantages over previous proposed decomposition methods, it has limited performance on problems with more than five interacting variables [33].

Delta grouping was proposed in [33] for identifying larger numbers of interacting variables. This method measures the delta value (the amount of change) of every variable in each iteration. Decision variables with smaller delta values are considered likely to be interacting with other decision variables. The delta values are ranked and the decision variables with smaller delta values were put into a common sub-group. Experimental results in [33] suggest that delta grouping can deliver good performance on finding interactive variables. However, the main drawback of delta grouping is that it can only group all the nonseparable variables into a single sub-group and it has difficulty handling problems with more than two nonseparable sub-groups.

2.2. Adaptive-size grouping

In algorithm CCEA-AVP [32], a correlation based adaptive variable partitioning technique (AVP) was proposed. In AVP, a correlation matrix is calculated based on the top 50 percent individuals of the current population after every M iterations (M was set to five in [32]). Then the correlation coefficient of each variable is obtained by the correlation matrix and the decision variables with a correlation coefficient greater than a user defined value (0.6 in [32]) are grouped together in one sub-population. The main advantage of AVP is that it increases the possibility to handle problems where separability of variables might vary with different sub-regions of the overall decision space.

In [34], a multilevel cooperative coevolution (MLCC) for large scale optimization was proposed. The main motivation for MLCC was to deal with the hard-to-determine parameter, group size, in DECC-G. MLCC makes use of a decomposer set S with different sizes of subcomponents instead of a specific decomposer. At the beginning of each cycle, one decomposer is selected from the decomposer set based on their previous performance. The selected decomposer is used to partition the original optimization problem into several sub-problems each of which is optimized by an EA. At the end of each cycle, the performance record of this chosen decomposer is then updated according to its performance in the current cycle. Experimental results in [34] suggest that MLCC can self-adapt to appropriate interaction levels during the evolution stage.

DECC-DML [33] employs delta grouping to decompose the original problems into different subgroups but the size of each sub-group is decided by a random self-adaptive subgroup size technique. Different from the self-adaptation mechanism in MLCC, a simpler and more efficient technique is used to decide the size of each sub-component. Similar to MLCC, a decomposer set S is designed to choose a specific decomposer. Instead of using the sophisticated formula based on the historical performance of each decomposer, a uniform random generator is used to choose a decomposer from the set S when there is no improvement performance between the current and the previous cycles.

Compared to fixed-size grouping, adaptive-size grouping methods are more likely to find the interaction among decision variables. However, these techniques are less efficient at decomposing problems with different sizes of sub-problems, and more work is still needed to underpin such methods with a sound theoretical basis.

2.3. Automatic grouping

Literature [30] proposed a new CC framework named cooperative coevolution with variable interaction learning (CCVIL). This algorithm begins by treating all decision variables as independent and puts all of them into a single separate group. Then it determines the relation between pairs of variables iteratively and merges the groups if the condition for interaction holds. The main contribution of CCVIL is the interaction criterion it used for identifying the interaction between two variables:

CCVIL criterion: If two decision variables x_i and x_j are interactive, then there exists $\vec{\boldsymbol{x}}_1 = (..., x_{i-1}, a, ..., x_{j-1}, b, ...)$, $\vec{\boldsymbol{x}}_2 = (..., x_{i-1}, a + \delta_a, ..., x_{j-1}, b, ...)$, $\vec{\boldsymbol{x}}_3 = (..., x_{i-1}, a, ..., x_{j-1}, b + \delta_b, ...)$, $\vec{\boldsymbol{x}}_4 = (..., x_{i-1}, a + \delta_a, ..., x_{j-1}, b + \delta_b, ...)$ such that, the following equation (1) holds.

$$f(\vec{\boldsymbol{x}}_1) - f(\vec{\boldsymbol{x}}_2) < 0 \wedge f(\vec{\boldsymbol{x}}_3) - f(\vec{\boldsymbol{x}}_4) > 0 \tag{1}$$

CCVIL identifies the interactions between variables based on theoretical facts. However, the interaction criterion in equation (1) is a sufficient but not necessary condition for detecting two interacting variables, which means it is incapable of finding all the possible interactions. We will explain this issue in more detail later in section 3.5.

An automatic decomposition approach called differential grouping (DG) was proposed in [20], which can automatically identify the interactive decision variables and partition the original problems into several sub-problems according to the independence between variables. The interaction criterion of DE is derived from the definition of partially additively separable problems and it provides a theoretical foundation for determining interacting decision variables. The experimental results show that this near-optimal decomposition is beneficial for handling large-scale global optimization problems.

DG criterion: For a partially additively separable function $f(\vec{\boldsymbol{x}}), \forall a, \quad b, \quad \delta_a \neq 0, \quad \delta_b \neq 0 \in R$, such that the following condition holds:

$$f(\vec{\boldsymbol{x}}_1) - f(\vec{\boldsymbol{x}}_2) \neq f(\vec{\boldsymbol{x}}_3) - f(\vec{\boldsymbol{x}}_4), \tag{2}$$

where $\vec{\boldsymbol{x}}_1 = (..., x_{i-1}, a, ..., x_{j-1}, b, ...)$, $\vec{\boldsymbol{x}}_2 = (..., x_{i-1}, a + \delta_a, ..., x_{j-1}, b, ...)$, $\vec{\boldsymbol{x}}_3 = (..., x_{i-1}, a, ..., x_{j-1}, b + \delta_b, ...)$, $\vec{\boldsymbol{x}}_4 = (..., x_{i-1}, a + \delta_a, ..., x_{j-1}, b + \delta_b, ...)$, then variables x_i and x_j interact with each other.

3. Mixed second order partial derivatives decomposition method

3.1. Problem definitions

Definition 1 A global numerical optimization problem can be formulated as follows,

$$arg\min_{\vec{\boldsymbol{x}}} f(\vec{\boldsymbol{x}}), \quad such \quad that \quad \vec{\boldsymbol{L}} \leq \vec{\boldsymbol{x}} \leq \vec{\boldsymbol{U}} \tag{3}$$

where $\vec{\boldsymbol{x}} = (x_1, ..., x_i, ..., x_n)$, $\vec{\boldsymbol{L}} = (l_1, ..., 1_i, ...l_n)$, $\vec{\boldsymbol{U}} = (u_1, ..., u_i, ...u_n) \in R^n$. $\vec{\boldsymbol{x}}$ is called the decision variable vector and the domain of each variable is defined by its lower and upper bounds respectively $l_i \leq x_i \leq u_i$. The space $S \in R^n$ formed by $l_i \leq x_i \leq u_i$, is called the decision space. The problem is called a large scale global optimization problem when the dimensionality of the decision variable is very high, such as problems with more than one hundred variables.

Definition 2 A optimization function $f(\vec{\boldsymbol{x}})$ is called fully-separable *iff*

$$\begin{aligned} arg\min_{\vec{\boldsymbol{x}}} f(\vec{\boldsymbol{x}}) =&(arg\min_{x_1} f(x_1, ...), ..., \\ &arg\min_{x_i} f(.., x_i, ...), ..., \\ &arg\min_{x_n} f(..., x_n)). \end{aligned} \tag{4}$$

It is obvious that a fully-separable function $f(\vec{\boldsymbol{x}})$ defined by equation (4) can be divided into n subcomponents and optimized respectively to obtain a globally optimal solution. The n variables are referred to as independent, i.e. a fully-separable function consists of n subcomponents, each of them with one independent variable.

Definition 3 A function $f(\vec{\boldsymbol{x}})$ is a partially separable function with m independent subcomponents *iff*

$$\begin{aligned} arg\min_{\vec{\boldsymbol{x}}} f(\vec{\boldsymbol{x}}) =&(arg\min_{\vec{\boldsymbol{x}}_1} f(..., \vec{\boldsymbol{x}}_1, ...), ..., \\ &arg\min_{\vec{\boldsymbol{x}}_i} f(..., \vec{\boldsymbol{x}}_i, ...), ..., \\ &arg\min_{\vec{\boldsymbol{x}}_m} f(..., \vec{\boldsymbol{x}}_m, ...)). \end{aligned} \tag{5}$$

Note that, each vector $\vec{\boldsymbol{x}}_i = (x_1, ..., x_{d_i}), i = 1, 2, ...m$ in equation (5) is a dis-joint sub-vector of $\vec{\boldsymbol{x}}$ with d_i dimensions and denotes a subcomponent of the original function. The variables in each vector $\vec{\boldsymbol{x}}_i$ interact with each other. Variables from different vectors, such as $\vec{\boldsymbol{x}}_i$ and $\vec{\boldsymbol{x}}_j, i \neq j$, are independent. The total number of independent subcomponents is m. Also note that the fully-separable function is a special example of partially separable functions with n independent subcomponents, each of which has only one decision variable.

Definition 4 A function $f(\vec{\boldsymbol{x}})$ is called fully-nonseparable *iff*, every pair of its variables $\forall i \neq j \in \{1, ..., n\}, x_i, x_j$ are not independent of each other.

Definition 4 is also a special case of Definition 3 with one subcomponent of d-dimensions.

Definition 5 A function $f(\vec{\boldsymbol{x}})$ is called partially additively separable with m subcomponents *iff* it can be written in the following form:

$$arg \min_{\vec{\boldsymbol{x}}} f(\vec{\boldsymbol{x}}) = arg \min_{\vec{\boldsymbol{x}}_i} \sum_{i=1}^{m} f_i(\vec{\boldsymbol{x}}_i), \tag{6}$$

where $\vec{\boldsymbol{x}}_i \in R^{d_i}$ are mutually exclusive decision vectors of f_i and $\sum_{i=1}^{m} d_i = n$. Partially additively separable functions are commonly found in real-world practice and they can represent the modular nature [39] of many real-world optimization problems. For this reason, most of the literature has focused on solving these types of optimization problems.

Here, a specific example is given to explain the partially additively separable fuction. Consider an optimization fuction, $arg \min_{\vec{\boldsymbol{x}}} f(\vec{\boldsymbol{x}}) = {x_1}^2 + {x_2}^2 + x_1 x_2 + {x_3}^2 + {x_4}^2 + {x_5}^2 + 2x_3x_4x_5$, which is a partially additively separable function with 2 subcomponents. It can be written as $arg \min_{\vec{\boldsymbol{x}}} f(\vec{\boldsymbol{x}}) = arg \min_{\vec{\boldsymbol{x}}_i} \sum_{i=1}^{2} f_i(\vec{\boldsymbol{x}}_i)$, where $f_1(\vec{\boldsymbol{x}}_1) = {x_1}^2 + {x_2}^2 + x_1x_2, \vec{\boldsymbol{x}}_1 = (x_1, x_2)$ and $f_2(\vec{\boldsymbol{x}}_2) = {x_3}^2 + {x_4}^2 + {x_5}^2 + 2x_3x_4x_5, \vec{\boldsymbol{x}}_2 = (x_3, x_4, x_5)$.

For the sake of convenience and clarity but without loss of generality, we assume that the function $f(\vec{\boldsymbol{x}})$ has m independent subcomponents denoted as $\{S_1, ..., S_m, \quad m = \{1, ..., n\}\}$. Each subcomponent S_i has d_i variables.

Definition 6 A function $f(\vec{\boldsymbol{x}})$ has overlapping subcomponents *iff*, $\exists i \neq j \in \{1, .., m\}$, *such that*, S_i and S_j have the same subset S_{ij}.

In other words, variables in S_{ij} interact with any other variables in subcomponents S_i and S_j. But other variables in S_i and S_j (not included in S_{ij}) are independent with each other (see Fig. 1). The elements in S_{ij} are denoted as overlapping decision variables.

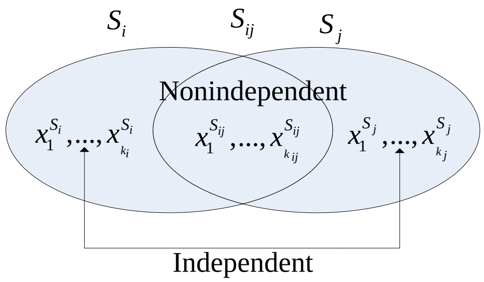


Figure 1: Illustration of two subcomponents S_i and S_j with overlapping variables, where $S_i = \{x_1^{S_i}, ..., x_{k_i}^{S_i}, x_1^{S_{ij}}, ..., x_{k_{ij}}^{S_{ij}}\}$ and $S_j = \{x_1^{S_j}, ..., x_{k_j}^{S_j}, x_1^{S_{ij}}, ..., x_{k_{ij}}^{S_{ij}}\}$. $d_i = k_i + k_{ij}$ is the dimension of subcomponent S_i. S_j is d_j-dimensional with $d_j = k_j + k_{ij}$. S_{ij} is the set with overlapping variables from S_i and S_j. It is evident that: 1. every pair of decision variables in S_i interact with each other; 2. any two decision variables in S_j also interact with each other; 3. $S_{ij} \in S_i$ and $S_{ij} \in S_j$. Any pair of elements in S_{ij} also interact with each other; 4. however, variables in set $S_i - S_{ij}$ are independent of variables in set $S_j - S_{ij}$.

3.2. Theoretical foundation for interaction and independence of variables

Theorem 1: For a partially additively separable function $f(\vec{\boldsymbol{x}})$, if $\frac{\partial^2 f(\vec{\boldsymbol{x}})}{\partial x_i \partial x_j} = 0$, then x_i and x_j are separable. (For clarity, we assume that the functions are continuous and smooth and $\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_i \partial x_j}$).

Proof: $f(\vec{\boldsymbol{x}}) = \sum_{k=1}^{m} f(..., \vec{\boldsymbol{x}}_k, ...)$. Let $x_i \in S_{k_0}, k_0 \in [1, ..., m]$.

$\Rightarrow \frac{\partial f(\vec{\boldsymbol{x}})}{\partial x_i} = \sum_{k=1}^{m} \frac{\partial f(..., \vec{\boldsymbol{x}}_k, ...)}{\partial x_i}$

$\vec{\boldsymbol{x}}_k, k = (1, ..., m)$ are mutually exclusive decision vectors of $f(\vec{\boldsymbol{x}})$. So $\frac{\partial f(..., \vec{\boldsymbol{x}}_k, ...)}{\partial x_i} = 0, k \neq k_0$.

$\Rightarrow \frac{\partial f(\vec{\boldsymbol{x}})}{\partial x_i} = \frac{\partial f(..., \vec{\boldsymbol{x}}_{k_0}, ...)}{\partial x_i}$

$\Rightarrow \frac{\partial^2 f(\vec{\boldsymbol{x}})}{\partial x_i \partial x_j} = \frac{\partial^2 f(..., \vec{\boldsymbol{x}}_{k_0}, ...)}{\partial x_i \partial x_j} = 0$

$\Rightarrow x_j \notin S_{k_0}$

$\Rightarrow x_i$ and x_j are separable.

Theorem 1 can be rewritten as the following Lemma 1:

Lemma 1: If $\forall a \in [l_i, u_i], \quad b \in [l_j, u_j]$, *such that* $\frac{\partial^2 f(\vec{\boldsymbol{x}})}{\partial x_i \partial x_j}|_{x_i=a, x_j=b} = 0$, then x_i and x_j are separable with each other.

It is evident that the first order partial derivative in the direction of x_i, $f_{x_i} = \frac{\partial f(\vec{\boldsymbol{x}})}{\partial x_i}$, is very important for finding those variables that interact with x_i from the proof of Theorem 1. In fact, to detect all of the variables that interact with x_i, we only need to find out which variables are involved in the partial derivative in direction x_i or affect the value of f_{x_i}. This observation can be formulated as the following lemma:

Lemma 2: $\forall \quad x_i \in [l_i, u_i]$, if $\forall \quad b, \quad b+\delta_b \in [l_j, u_j]$, $f_{x_i}|_{x_i,x_j=b} - f_{x_i}|_{x_i,x_j=b+\delta_b} = 0 \quad (j \neq i, \quad \delta_b \neq 0)$, then x_j is separate with x_i; however, if $\exists \quad b, \quad \delta_b$, such that $f_{x_i}|_{x_i,x_j=b} - f_{x_i}|_{x_i,x_j=b+\delta_b} \neq 0 \quad (j \neq i, \quad \delta_b \neq 0)$, then x_j belongs to the group of variables intact with x_i.

The above theorem and lemmas mainly show how to identify the independence of two decision variables. In the following we will study the properties of interaction between two variables from the perspective of the second order partial derivatives of the functions.

Theorem 2: If $\frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j} \neq 0$, then x_i and x_j interact with each other.

Proof: Theorem 2 is the contrapositive of Theorem 1. Theorem 1 holds $\Rightarrow$ Theorem 2 holds.

Lemma 3: $\exists a \in [l_i, u_i], \quad b \in [l_j, u_j]$, *such that* $\frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j}|_{x_i=a,x_j=b} \neq 0$, then x_i and x_j interact with each other.

The theoretical analysis described above mainly explains the relationship of interaction and independence between two decision variables. From section 3.1, two independent variables can interact with the same variables, which are known as overlapping variables. We will now show theoretically how an overlapping variable is identified.

Theorem 3: x_k is an overlapping variable, if $\exists i, j \in \{1, ..., n\} \neq k$, *such that*, $\frac{\partial^2 f(\vec{x})}{\partial x_k \partial x_i} \neq 0$, and $\frac{\partial^2 f(\vec{x})}{\partial x_k \partial x_j} \neq 0$, but $\frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j} = 0$.

Proof: $\frac{\partial^2 f(\vec{x})}{\partial x_k \partial x_i} \neq 0 \Rightarrow x_k$ and x_i are nonseparate.

$\frac{\partial^2 f(\vec{x})}{\partial x_k \partial x_j} \neq 0 \Rightarrow x_k$ and x_j are nonseparate.

$\frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j} = 0 \Rightarrow x_i$ and x_j are independent.

$\Rightarrow$ From the definition for overlapping variables, we can obtain that x_k is an overlapping variable of subcomponents including elements x_i and x_j respectively.

Definition 7 *Degree of interaction between two variables*

For two decision variables x_i and x_j, $\frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j}$ shows the degree of interaction between them.

$\frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j}$ shows the strength of non-separability between the two variables. The larger $|\frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j}|$ is, the stronger the interaction between these two variables; otherwise, the two are more likely to be independent. In the extreme case, when $\frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j} = 0$, then x_i and x_j are separable. Otherwise, x_i and x_j are nonseparable and $\frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j}$ indicts how strongly they interact with each other.

We now give a specific example to show how the above theorems and lemmas can be used. Consider an optimization problem $f(\vec{x}) = x_1^2 + \lambda_1 x_1 x_2 + \lambda_2 x_2 x_3 +$

$x_2^2 + x_3^2, \lambda_1 \neq 0, \lambda_2 \neq 0$.

The first order partial derivative in each direction is $f_{x_1} = 2x_1 + \lambda_1 x_2$, $f_{x_2} = 2x_2 + \lambda_1 x_1 + \lambda_2 x_3$, $f_{x_3} = 2x_3 + \lambda_2 x_2$ respectively. From *lemma 2*, we can draw the following conclusions: 1) Because there are only two variables x_1 and x_2 involved in f_{x_1}, x_1 interacts with x_2 and x_1 is separate with x_3; 2) From the expression of f_{x_2}, x_2 interacts with both x_1 and x_3; 3) From the expression of f_{x_3}, x_3 interacts with x_2 but separates with x_1.

The mixed second order derivatives are $\frac{\partial^2 f(\vec{\boldsymbol{x}})}{\partial x_1 \partial x_2} = \lambda_1$, $\frac{\partial^2 f(\vec{\boldsymbol{x}})}{\partial x_1 \partial x_3} = 0$, and $\frac{\partial^2 f(\vec{\boldsymbol{x}})}{\partial x_2 \partial x_3} = \lambda_2$ respectively. By using *theorem 1* and *theorem 2* we can reach the same conclusions that were derived from *lemma 1*: 1) x_1 and x_3 are separate; 2) x_2 interacts with x_1 and x_3. Moreover, according to *theorem 3*, x_2 is an overlapping variable to x_1 and x_3. The degree of interaction between x_1 and x_2 is λ_1 and the degree of interaction between x_2 and x_3 is λ_2. If we set $\lambda_1 = 0$ then x_1 and x_2 become separate; if $\lambda_2 = 0$, then x_2 and x_3 are separate; if both $\lambda_1 = 0$ and $\lambda_2 = 0$, then $f(\vec{\boldsymbol{x}})$ becomes a fully-separable function.

3.3. Derived interaction criterion

The previous section 3.2 provided a theoretical foundation with respect to the non-separability of optimization problems. In this section, we introduce an interaction criterion based on the above mentioned theorems and lemmas, and some decomposition algorithms for detecting the interactive subcomponents of the optimization function.

In the previous section 3.2, we showed how $\frac{\partial^2 f(\vec{\boldsymbol{x}})}{\partial x_i \partial x_j}$ is of great importance in determining the relationship between two variables x_i and x_j. Here we show how an interaction criterion can be derived by considering such second mixed partial derivatives.

Interaction and separability criterion (IS criterion):

If $\forall$ a, $a+$ $\delta_a \in [l_i, u_i]$, b, $b + \delta_b \in [l_j, u_j]$, such that, $(f(\vec{\boldsymbol{x}})|_{x_i=a+\delta_a, x_j=b+\delta_b} - f(\vec{\boldsymbol{x}})|_{x_i=a, x_j=b+\delta_b}) - (f(\vec{\boldsymbol{x}})|_{x_i=a+\delta_a, x_j=b} - f(\vec{\boldsymbol{x}})|_{x_i=a, x_j=b}) = 0$, then x_i and x_j are separate with each other; If $\exists a, a + \delta_a \in [l_i, u_i], b, b + \delta_b \in [l_j, u_j]$, such that, $(f(\vec{\boldsymbol{x}})|_{x_i=a+\delta_a, x_j=b+\delta_b} - f(\vec{\boldsymbol{x}})|_{x_i=a, x_j=b+\delta_b}) - (f(\vec{\boldsymbol{x}})|_{x_i=a+\delta_a, x_j=b} - f(\vec{\boldsymbol{x}})|_{x_i=a, x_j=b}) \neq 0$, then x_i and x_j interact with each other. Here we denote $(f(\vec{\boldsymbol{x}})|_{x_i=a+\delta_a, x_j=b+\delta_b} - f(\vec{\boldsymbol{x}})|_{x_i=a, x_j=b+\delta_b}) - (f(\vec{\boldsymbol{x}})|_{x_i=a+\delta_a, x_j=b} - f(\vec{\boldsymbol{x}})|_{x_i=a, x_j=b})$ as $\bigtriangledown_{x_i, x_j}$.

Proof: We first prove that $\frac{\partial^2 f(\vec{\boldsymbol{x}})}{\partial x_i \partial x_j} \Rightarrow \bigtriangledown_{x_i, x_j}$. Then according to the theorems and lemmas in the previous section 3.2, we can obtain the conclusions in the interaction criterion.

$$
\begin{aligned}
\frac{\partial^2 f(\vec{\boldsymbol{x}})}{\partial x_i \partial x_j} &\Rightarrow \int_b^{b+\delta_b} \int_a^{a+\delta_a} \frac{\partial^2 f(\vec{\boldsymbol{x}})}{\partial x_i \partial x_j} \, dx_i \, dx_j \\
&\Rightarrow \int_b^{b+\delta_b} \frac{\partial f(\vec{\boldsymbol{x}})}{\partial x_j} |_a^{a+\delta_a} \, dx_j \\
&\Rightarrow (f(\vec{\boldsymbol{x}})|_{x_i=a+\delta_a, x_j=b+\delta_b} - f(\vec{\boldsymbol{x}})|_{x_i=a, x_j=b+\delta_b}) - \\
&\quad\ (f(\vec{\boldsymbol{x}})|_{x_i=a+\delta_a, x_j=b} - f(\vec{\boldsymbol{x}})|_{x_i=a, x_j=b})
\end{aligned}
$$

The above derived criterion is useful in that it only requires the difference of two decision variables, and does not require explicit knowledge of the derivatives of the objective function, which will often be unavailable, e.g. in many real-world problems for which there is no obvious overall analytic function.

3.4. Proposed algorithms based on IS criterion

Section 3.2, presented a theoretical foundation for understanding interaction and this was then used in section 3.3, to derive a useful interaction criterion. The main advantage of the derived criterion is that it only requires the difference of two decision variables and does not require the derivatives of the objective function to be explicitly known. This makes it more convenient and suitable for implementation, especially for problems without obvious analytical functions. In this section, we propose a algorithm, random DG (RDG), for identifying the interactive variables and subcomponents according to the *IS criterion* given in section 3.3. RDG is introduced in algorithm 2. For comparison, we also show the pseudocode for the DG grouping method, based on the DG criterion, in algorithm 1.

Algorithm 2 finds the subcomponents of a function by detecting the interaction between two variables x_i and x_j. Once these two variables are determined as non-separable according to *IS criterion*, they are grouped into a single subcomponent and x_j is then deleted from the selection pool. After all dimensions in the selection pool $dims$ have been compared with the ith variable for interaction detection, the sub-component for all variables interacted with x_i is formed. If no interaction is detected, x_i is considered to be separable and is placed into the $Seps$ set. This process is repeated until there is no element left in the selection pool. Note that the main difference between the DG and RDG grouping algorithms is the method for generating the two pairs of decision variable vectors $\vec{\boldsymbol{x}}_1$, $\vec{\boldsymbol{x}}_2$, and $\vec{\boldsymbol{x}}_3$, $\vec{\boldsymbol{x}}_4$ for the interaction detection between ith and jth variables. In RDG, $\vec{\boldsymbol{x}}_1$ is randomly generated from the decision space. Then $\vec{\boldsymbol{x}}_2$ is obtained through replacing the ith dimension in vector $\vec{\boldsymbol{x}}_1$ by a random number $tempi$ inside the boundaries of the ith variable. $\vec{\boldsymbol{x}}_3$ and $\vec{\boldsymbol{x}}_4$ are obtained by replacing the value of the jth variable with a randomly generated $tempj$ form $[\vec{\boldsymbol{L}}(j), \vec{\boldsymbol{U}}(j)]$. Then $\bigtriangledown_{x_i,x_j}$ is calculated to determine if these two variables interact with each other.

Algorithm 1 DG grouping for detecting the subcomponents of an optimization problem according to DG criterion

Input: optimization function *func*, dimension number n, upper and low bounds $\vec{\boldsymbol{U}}$ and $\vec{\boldsymbol{L}}$
Initialization: $dims = \{1, 2, ...n\}$, $Seps = \{\}$, *allgroups*=$\{\}$
for $i \in dims$ **do**
    $group = i$;
    $\vec{\boldsymbol{x}}_1 = \vec{\boldsymbol{L}} \times ones(1, n)$
    $\vec{\boldsymbol{x}}_2 = \vec{\boldsymbol{x}}_1$
    $\vec{\boldsymbol{x}}_2(i) = \vec{\boldsymbol{U}}(i)$
    for $j \in dims \wedge i \neq j$ **do**
        $\vec{\boldsymbol{x}}_3 = \vec{\boldsymbol{x}}_1$
        $\vec{\boldsymbol{x}}_4 = \vec{\boldsymbol{x}}_2$
        $\vec{\boldsymbol{x}}_3(j) = 0$
        $\vec{\boldsymbol{x}}_4(j) = 0$
        if $\mid \bigtriangledown_{x_i,x_j} \mid > \epsilon$ **then**
            $gruop = group \cup j$
        end if
    end for
    $dims = dims - group$
    if $length(group) = 1$ **then**
        $Seps = Seps \cup group$
    else
        $allgroups = allgroups \cup \{group\}$
    end if
end for
Output: $allgroups = allgroups \cup \{Seps\}$

3.5. Relationship between the proposed method, DG and CCVIL

In section 2.3, we provided a detailed explanation of two automatic grouping methods, CVIL [30] and DG [20]. In this section, we discuss the properties of these methods and their relationships with the criterion and theorems proposed in this paper.

Algorithm 2 Random DG approach for detecting the subcomponents of a optimization problem according to *IS* criterion

Input: optimization function *func*, dimension number n, upper and low bounds $\vec{\boldsymbol{U}}$ and $\vec{\boldsymbol{L}}$

Initialization: $dims = \{1, 2, ...n\}$, $Seps = \{\}$, *allgroups*=$\{\}$

for $i \in dims$ **do**
    $group = i$
    $tempi = \vec{\boldsymbol{L}}(i) + rand(1,1) \times (\vec{\boldsymbol{U}}(i) - \vec{\boldsymbol{L}}(i))$
    $\vec{\boldsymbol{x}}_1 = \vec{\boldsymbol{L}} + rand(1,n) \times (\vec{\boldsymbol{U}} - \vec{\boldsymbol{L}})$
    $\vec{\boldsymbol{x}}_2 = \vec{\boldsymbol{x}}_1$
    $\vec{\boldsymbol{x}}_2(i) = tempi$
    for $j \in dims \wedge i \neq j$ **do**
        $tempj = \vec{\boldsymbol{L}}(j) + rand(1,1) \times (\vec{\boldsymbol{U}}(j) - \vec{\boldsymbol{L}}(j))$
        $\vec{\boldsymbol{x}}_3 = \vec{\boldsymbol{x}}_1$
        $\vec{\boldsymbol{x}}_4 = \vec{\boldsymbol{x}}_2$
        $\vec{\boldsymbol{x}}_3(j) = tempj$
        $\vec{\boldsymbol{x}}_4(j) = tempj$
        if $\mid \bigtriangledown_{x_i,x_j} \mid > \epsilon$ **then**
            $gruop = group \cup j$
        end if
    end for
    $dims = dims - group$
    if $length(group) = 1$ **then**
        $Seps = Seps \cup group$
    else
        $allgroups = allgroups \cup \{group\}$
    end if
end for

Output: $allgroups = allgroups \cup \{Seps\}$

Notation $rand(1,n)$ stands for a 1-by-n matrix with random values drawn on the open interval $(0,1)$. $rand(1,1)$ is a random value from $(0,1)$.

Chen *et al.* proposed a variable interaction learning algorithm CCVIL in [30]. The interaction is determined by the interaction criterion in equation (1). However, note that the interaction criterion in equation (1) only encodes one of the possible scenarios defined by *IS criterion*. More specifically, if (1) holds, then variables x_i and x_j interact with each other. But if (1) does not hold, there is no guarantee that variables x_i and x_j are necessarily separate with each other. We now provide proofs that equation (1) $\Rightarrow$ *IS criterion* for iteration holds and *IS criterion* for iteration holds $\nRightarrow$ equation (1).

Denote $\vec{\boldsymbol{x}}_1 = (..., x_{i-1}, a, ..., x_{j-1}, b, ...)$, $\vec{\boldsymbol{x}}_2 = (..., x_{i-1}, a+\delta_a, ..., x_{j-1}, b, ...)$, $\vec{\boldsymbol{x}}_3 = (..., x_{i-1}, a, ..., x_{j-1}, b+\delta_b, ...)$, $\vec{\boldsymbol{x}}_4 = (..., x_{i-1}, a+\delta_a, ..., x_{j-1}, b+\delta_b, ...)$. Then $\bigtriangledown_{x_i,x_j} = (f(\vec{\boldsymbol{x}}_2) - f(\vec{\boldsymbol{x}}_1)) - (f(\vec{\boldsymbol{x}}_4) - f(\vec{\boldsymbol{x}}_3))$. Therefore, the IS criterion can be rewritten as $\forall$ a, $a+\ \delta_a \in [l_i, u_i]$, b, $b+\delta_b \in [l_j, u_j]$, such that $f(\vec{\boldsymbol{x}}_1) - f(\vec{\boldsymbol{x}}_2) = f(\vec{\boldsymbol{x}}_3) - f(\vec{\boldsymbol{x}}_4)$, then x_i and x_j are separate with each other; $\exists$ a, $a+\ \delta_a \in [l_i, u_i]$, b, $b+\delta_b \in [l_j, u_j]$, such that $f(\vec{\boldsymbol{x}}_1) - f(\vec{\boldsymbol{x}}_2) \neq f(\vec{\boldsymbol{x}}_3) - f(\vec{\boldsymbol{x}}_4)$, then x_i and x_j interact each other.

If the condition for CCVIL holds then the *IS criterion* for identifying interaction also holds. (equation (1) $\Rightarrow \bigtriangledown_{x_i,x_j} \neq 0$)

Proof: Because the condition for CCVIL holds, $\exists a, \quad b, \quad \delta_a \neq 0, \quad \delta_b \neq 0 \in R$, such that $f(\vec{\boldsymbol{x}}_1) - f(\vec{\boldsymbol{x}}_2) < 0 \wedge f(\vec{\boldsymbol{x}}_3) - f(\vec{\boldsymbol{x}}_4) > 0 \Rightarrow f(\vec{\boldsymbol{x}}_1) - f(\vec{\boldsymbol{x}}_2) \neq f(\vec{\boldsymbol{x}}_3) - f(\vec{\boldsymbol{x}}_4)$. So if equation (1) holds, we can derive that equation $\bigtriangledown_{x_i,x_j} \neq 0$ holds.

If the condition for $\bigtriangledown_{x_i,x_j} \neq 0$ holds, there is no guarantee that CCVIL criterion also holds. (equation $\bigtriangledown_{x_i,x_j} \neq 0 \nRightarrow$ equation (1))

Proof: If equation $\bigtriangledown_{x_i,x_j} \neq 0$ holds, there are four scenarios for the relation between $f(\vec{\boldsymbol{x}}_1) - f(\vec{\boldsymbol{x}}_2)$ and $f(\vec{\boldsymbol{x}}_3) - f(\vec{\boldsymbol{x}}_4)$: 1. $f(\vec{\boldsymbol{x}}_1) - f(\vec{\boldsymbol{x}}_2) < 0$ and $f(\vec{\boldsymbol{x}}_3) - f(\vec{\boldsymbol{x}}_4) > 0$; 2. $f(\vec{\boldsymbol{x}}_1) - f(\vec{\boldsymbol{x}}_2) \leqq 0, f(\vec{\boldsymbol{x}}_3) - f(\vec{\boldsymbol{x}}_4) \leqq 0$ and $f(\vec{\boldsymbol{x}}_1) - f(\vec{\boldsymbol{x}}_2) \neq f(\vec{\boldsymbol{x}}_3) - f(\vec{\boldsymbol{x}}_4)$; 3. $f(\vec{\boldsymbol{x}}_1) - f(\vec{\boldsymbol{x}}_2) \geqq 0, f(\vec{\boldsymbol{x}}_3) - f(\vec{\boldsymbol{x}}_4) \geqq 0$, and $f(\vec{\boldsymbol{x}}_1) - f(\vec{\boldsymbol{x}}_2) \neq f(\vec{\boldsymbol{x}}_3) - f(\vec{\boldsymbol{x}}_4)$; 4. $f(\vec{\boldsymbol{x}}_1) - f(\vec{\boldsymbol{x}}_2) > 0$ and $f(\vec{\boldsymbol{x}}_3) - f(\vec{\boldsymbol{x}}_4) < 0$. So equation $\bigtriangledown_{x_i,x_j} \neq 0 \nRightarrow$ equation (1). In other word, if two variables are nonseparable, equation (1) does not necessarily hold. Therefore the interaction criterion used in CCVIL cannot detect all nonseparable variables.

Omidvar *et al.* investigated large scale optimization problems from a theoretical perspective and proposed the automatic grouping method DG in [20] to detect interacting variables with high accuracy. The DG criterion to identify the interaction between two variables is derived from the mathematical definition of partially additively separable optimization problems (Definition 5 in section 3.1).

The equation (2) in the DG criterion is actually equivalent to the *IS criterion* in terms of identifying the interaction $\bigtriangledown_{x_i,x_j} \neq 0$. However, in the DG criterion, if two variables are interactive, equation (2) holds for all a, $a+\ \delta_a \in [l_i, u_i]$, b,

$b + \delta_b \in [l_j, u_j]$. In contrast, in the *IS criterion*, if there exits one a, $a + \delta_a \in [l_i, u_i]$, b, $b + \delta_b \in [l_j, u_j]$, such that $\bigtriangledown_{x_i,x_j} \neq 0$, then we can determine that the two variables are nonseparable. It is obvious that the DG criterion holds $\Rightarrow$ *IS criterion* for variable interaction holds. However, *IS criterion* for variable interaction holds $\nRightarrow$ DG criterion holds. Therefore, the DG criterion is a sufficient but not necessary condition for detecting two interactive variables. As an example, consider a function $f(\vec{\boldsymbol{x}}) = x_1 x_2 (x_1 - 1)(x_2 - 1)$, where x_1 and x_2 are nonseparable variables. However, not all a, $a + \delta_a \in [l_i, u_i]$, b, $b + \delta_b \in [l_j, u_j]$ such that equation (2) holds. When $\vec{\boldsymbol{x}}_1 = (0, 0)$, $\vec{\boldsymbol{x}}_2 = (1, 0)$, $\boldsymbol{x}_3 = (0, 1)$, $\vec{\boldsymbol{x}}_4 = (1, 1)$, equation (2) does not hold.

4. Experimental results and discussion

The comparison results of our proposed grouping algorithm with two automatic grouping methods from the literature, DG and CCVIL, are shown in Table 2 on 20 benchmark functions [21]. Optimization functions $G01 - G03$ are completely separable. $G04 - G08$ have only one nonseparable subcomponent comprising 50 variables, and the other 950 variables are separate. $G09 - G13$ have 10 nonseparable groupings and each of them has 50 interacting variables. $G14 - G18$ are nonseparable functions with 20 subcomponents. $G19$ and $G20$ are nonseparable functions with one subcomponent.

For separate functions $G01 - G03$, both RDG and DG found all the 1000 separate variables correctly. CCVIL erroneously placed some separable variables into one group as nonseparable variables for $G03$. Moreover, CCVIL required a greater number of fitness evaluations (FEs) than RDG and DG.

For $G04 - G08$, RDG and DG both achieved good results on $G05$ and $G06$. For $G07$, RDG found the correct nonseparable groups and separable groups. CCVIL also achieved good result on $G07$, only one separable variable was misplaced into nonseparable group. However, DG was unable to correctly find the separable or nonseparable groups for $G07$. For $G04$ and $G08$, both DG and RDG performed poorly, while CCVIL found 43 nonseparable variables and only misplaced 7 nonseparable variables.

RDG and DG show similar performance on $G09 - G13$. They classified correctly on functions $G09 - G12$. For $G13$, RDG identified 537 separable variables and 463 nonseparable variables were grouped into 80 subcomponents. DG only identified 131 separable variables and the non-separable variables were grouped into 40 subgroups. CCVIL grouped all the variables into one non separable group on $G13$.

Table 2: Comparison of grouping results by algorithms RDG, DG and CCVIL respectively.

Fun	Sep Vars	Non-sep Vars	Non-sep Groups	$RDG(\epsilon = 10^{-3})/DG(\epsilon = 10^{-3})/CCVIL$ Captured Sep Var	Formed Non-sep Groups	FE
G01	1000	0	0	1000/1000/1000	0/0/0	1001000/1001000/69990
G02	1000	0	0	1000/1000/1000	0/0/0	1001000/1001000/69990
G03	1000	0	0	1000/1000/938	0/0/1	1001000/1001000/1798666
G04	950	50	1	3/33/957	9/10/1	3490/14564/1797614
G05	950	50	1	950/950/950	1/1/1	905450/905450/1795705
G06	950	50	1	950/950/910	1/1/22	906332/906332/1796370
G07	950	50	1	950/247/951	1/4/1	906822/7410/1796475
G08	950	50	1	10/135/1000	12/5/0	8630/23608/69842
G09	500	500	10	500/500/583	10/10/33	270802/270802/1792212
G10	500	500	10	500/500/508	10/10/10	272958/272958/1774642
G11	500	500	10	502/501/476	10/10/26	271662/270640/1774565
G12	500	500	10	500/500/516	10/10/11	271390/271390/1777344
G13	500	500	10	537/131/1000	80/40/0	468696/48470/69990
G14	0	1000	20	0/0/150	20/20/63	21000/21000/1785975
G15	0	1000	20	0/0/18	20/20/20	21000/21000/1751241
G16	0	1000	20	0/4/11	20/20/20	21000/21128/1751647
G17	0	1000	20	0/0/25	20/20/20	21000/21000/1752340
G18	0	1000	20	0/85/1000	20/49/0	21000/34230/69990
G19	0	1000	20	0/0/0	1/1/1	2000/2000/48212
G20	0	1000	20	10/42/972	22/16/14	8630/22206/17908708

Table 3: Comparison of different parameter ϵ on the grouping results.

Fun	Sep Vars	Non-sep Vars	Non-sep Groups	$RDG(\epsilon = 10^{-1})/RDG(\epsilon = 10^{-6})$ Captured Sep Var	Formed Non-sep Groups	FE
G01	1000	0	0	1000/13	0/9	1001000/4302
G02	1000	0	0	1000/1000	0/0	1001000/1001000
G03	1000	0	0	1000/3	0/10	1001000/4910
G04	950	50	1	2/5	13/8	8840/3704
G05	950	50	1	950/950	1/1	905450/905450
G06	950	50	1	950/2	1/8	906332/3342
G07	950	50	1	950/2	1/13	906822/7410
G08	950	50	1	5/3	12/18	5570/9324
G09	500	500	10	500/2	10/11	270802/5994
G10	500	500	10	502/500	10/10	274972/272958
G11	500	500	10	509/1	10/15	315094/15228
G12	500	500	10	500/500	10/10	271390/271390
G13	500	500	10	550/5	173/23	636686/9990
G14	0	1000	20	0/3	20/11	21000/5254
G15	0	1000	20	1/0	20/20	21038/21000
G16	0	1000	20	20/0	74/20	53066/21000
G17	0	1000	20	0/0	20/20	21000/21000
G18	0	1000	20	79/4	359/18	383540/6844
G19	0	1000	20	0/0	1/1	2000/2000
G20	0	1000	20	0/11	500/18	501000/9202

Table 4: Comparison of optimization results against other five algorithms on the CEC's 2010 benchmark functions using 25 independent trials

Functions		DECC-RDG	DECC-DG	MLCC	DECC-D	DECC-DML	DECC-I
G01	Mean	8.26E+03	5.47E+03	1.53E-27	**1.01E-25**	1.93E-25	1.73E+00
	Std	3.20E+04	2.02E+04	7.66E-27	1.40E-25	1.86E-25	2.55E+00
G02	Mean	4.44E+03	4.39E+03	**5.57E-01**	2.99E+02	2.17E+02	4.40E+03
	Std	1.52E+02	1.97E+02	2.21E+00	1.92E+01	2.98E+01	1.90E+02
G03	Mean	1.67E+01	1.67E+01	9.88E-13	**1.81E-13**	**1.18E-13**	1.67E+01
	Std	3.04E-01	3.34E-01	3.70E-01	6.68E-15	8.22E-15	3.75E-01
G04	Mean	4.35E+12	4.79E+12	9.61E+12	3.99E+12	3.58E+12	**6.13E+11**
	Std	1.04E+12	1.44E+12	3.43E+12	1.30E+12	1.54E+12	2.08E+07
G05	Mean	**1.49E+08**	1.55E+08	3.84E+08	4.16E+08	2.98E+08	1.34E+08
	Std	1.83E+07	2.17E+07	6.93E+07	1.01E+08	9.31E+07	2.31E+07
G06	Mean	**1.64E+01**	**1.64E+01**	1.62E+07	1.36E+07	7.93E+05	1.64E+01
	Std	3.27E-01	2.71E-01	4.97E+06	9.20E+06	3.97E+06	2.66E-01
G07	Mean	8.02E+08	**1.16E+04**	6.89E+05	6.58E+07	1.39E+08	2.97E+01
	Std	8.24E+08	7.41E+03	7.37E+05	4.06E+07	7.72E+07	8.59E+01
G08	Mean	1.03E+08	3.04E+07	4.38E+07	5.39E+07	3.46E+07	**3.19E+05**
	Std	8.71E+07	2.11E+07	3.45E+07	2.93E+07	3.56E+07	1.10E+06
G09	Mean	**5.63E+07**	5.96E+07	1.23E+08	6.19E+07	5.92E+07	4.84E+07
	Std	5.77E+06	8.18E+06	1.33E+07	6.43E+06	4.71E+06	6.56E+06
G10	Mean	4.54E+03	4.52E+03	**3.43E+03**	1.16E+04	1.25E+04	4.34E+03
	Std	1.42E+02	1.41E+02	8.72E+02	2.68E+03	2.66E+02	1.46E+02
G11	Mean	1.03E+01	1.03E+01	1.98E+02	4.76E+01	**1.80E-13**	1.02E+01
	Std	8.00E-01	1.01E+00	6.98E-01	9.53E+01	9.88E-15	1.13E+00
G12	Mean	2.65E+03	**2.52E+03**	3.49E+04	1.53E+05	3.79E+06	1.47E+03
	Std	7.65E+02	4.86E+02	4.92E+03	1.23E+04	1.50E+05	4.28E+02
G13	Mean	3.56E+06	4.54E+06	2.08E+03	**9.87E+02**	1.14E+03	7.51E+02
	Std	6.08E+05	2.13E+06	7.27E+02	2.41E+02	4.31E+02	3.70E+02
G14	Mean	3.47E+08	3.41E+08	3.16E+08	**1.98E+08**	1.89E+08	3.38E+08
	Std	2.83E+07	2.41E+07	2.77E+07	1.45E+07	1.49E+07	2.40E+07
G15	Mean	**5.85E+03**	5.88E+03	7.11E+03	1.53E+04	1.54E+04	5.87E+03
	Std	9.00E+01	1.03E+02	1.34E+03	3.92E+02	3.59E+02	9.89E+01
G16	Mean	**7.01E-13**	7.39E-13	3.76E+02	1.88E+02	5.08E-02	2.47E-13
	Std	4.88E-14	5.70E-14	4.71E+01	2.16E+02	2.54E-01	9.17E-15
G17	Mean	4.11E+04	**4.01E+04**	1.59E+05	9.03E+05	6.54E+06	3.91E+04
	Std	2.89E+03	2.85E+03	1.43E+04	5.28E+04	4.63E+05	2.75E+03
G18	Mean	6.73E+07	1.11E+10	7.09E+03	2.12E+03	2.47E+03	**1.17E+03**
	Std	2.86E+07	2.04E+09	4.77E+03	5.18E+02	1.18E+03	9.66E+01
G19	Mean	1.82E+06	1.74E+06	**1.36E+06**	1.33E+07	1.59E+07	1.74E+06
	Std	8.52E+04	9.54E+04	7.35E+04	1.05E+06	1.72E+06	9.54E+04
G20	Mean	1.28E+09	4.87E+07	2.05E+03	**9.91E+02**	**9.91E+02**	4.14E+03
	Std	3.57E+08	2.27E+07	1.80E+02	2.61E+01	3.51E+01	8.14E+02

For $G14 - G18$, RDG achieved the best performance and it correctly grouped all the variables. DG was unable to find all the nonseparable variables (85 nonseparable variables were misplaced as separable variables) on $G18$ and the number of the subcomponents was not correctly chosen either. CCVIL cannot correctly identity all the nonseparable groups on these functions compared to RDG and DG.

All three algorithms obtained good results on $G19$. However, none of the algorithms were able to correctly group all variables into one nonseparable subcomponent for $G20$. The main difference between the results with function $G19$ and $G20$ is that the non-separable variables in $G20$ are overlapping variables. A detailed explanation of why the current algorithms fail to capture the non-separable variables of functions with overlapping variables is given later in this section.

Table 3 shows the effect of the parameter ϵ on the grouping performance of the proposed method. It is apparent that a larger ϵ helps in finding the separable variables, while some separable variables were misclassified as interacting variables with very small ϵ ($\epsilon = 10^{-6}$) values (which might be due to the precision error in calculating $\bigtriangledown_{x_i,x_j}$). However, compared to CCVIL with different ϵ, RDG had better performance on most of the 20 functions. In other words, RDG is not very sensitive to the parameter ϵ as long as it is sufficiently small.

It is evident that RDG outperforms approaches DG and CCVIL on most of the test problems and RDG can identify the separable subcomponents that DG and CCVIL fail to find, which shows the advantages of the proposed decomposition method. But, both RDG and DG had poor performance on $G08, G13$ and $G20$. Moreover, it is very interesting to find that $G08, G13$ and $G20$ are instances of the Rosenbrock function. So here we make an insight on the reason of the poor performer on these functions. Based on the theritical analysis on the overlapping variables, it is easy to know that these three optimization problems all contain overlapping variables. Here we analyse the behaviour of RDG and DG when handling problems with overlapping variables.

Once RDG or DG determines that x_j interacts with x_i, x_j is then deleted from the selection pool and grouped into a subcomponent with x_i. This means that x_j is not compared with other variables that are dependent with x_i. Consider a function $f(\vec{\boldsymbol{x}})$ with $\vec{\boldsymbol{x}} = \{x_1, ..., , x_n\}$. Let $x_p, 1 < p < n$ is a overlapping variable and x_{p-1} interacts with x_p, x_p interacts with x_{p+1}, but x_{p-1} and x_{p+1} are independent. If we apply RDG to find the subcomponent of this problem. x_{p-1} and x_p is put into one subcomponent. Because x_p is deleted after the interaction detection with x_p, x_p never get the chance to detect the relationship between x_{p+1} and x_{p+1} is grouped into another subcomponent as it is dependent with x_{p-1}. In a conclusion, these approaches cannot correctly identify the nonseparable groups when deal with problems with overlapping variables.

In table 4, the optimization results under CC framework with RDG and other algorithms with different grouping methods. The decomposition strategies used are DG (applied in algorithm DECC-DG), random grouping (DECC-G, MLCC)[31], delta grouping (DECC-D, DECC-DML) [33], and an ideal grouping that can be derived by Theorem 1 and Theorem 2. DECC-RDG and DECCDG outperform other algorithms on functions G05$\sim$G09, G12, G15$\sim$G17. DECC-RDG outperforms DECC-DG on functions G04, G05, G09, G16, and G18. (Note that the results of the comparison algorithms were from [20].)

5. Concluding remarks

In this paper, we have set out a theoretical foundation for understanding decomposition of LSO problems and we have proposed an automatic grouping algorithm, RDG, for identifying separable and nonseparable groups automatically. Experimental results also show that the proposed methods outperform other grouping algorithms on the 20 benchmark problems. We conclude with remarks on two more specific issues.

1) We have analyzed the behaviour of the proposed decomposition approach on optimization problems with overlapping variables and the experimental results also show that the proposed method cannot correctly group all the independent variables when dealing with overlapping variables. Neither the proposed method nor other decomposition methods have carefully investigated this issue. So, how to deal problems with overlapping variables is one of our future works.

2) Although accurate decomposition to identify interacting decision variables is very important for optimization, even a perfect decomposition strategy can not guarantee a successful optimization stage. Future work will investigate which decomposition methods most benefit the optimization stage, even if they do not necessarily yield an accurate grouping result.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No.61603305), the China Postdoctoral Science Foundation (Grant No. 2016M602857).

References

[1] X. Li, K. Tang, M. N. Omidvar, Z. Yang, K. Qin, H. China, Benchmark functions for the cec 2013 special session and competition on large-scale global optimization, gene 7 (2013) 33.

[2] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y.-P. Chen, C.-M. Chen, Z. Yang, Benchmark functions for the cec2008 special session and competition on large scale global optimization, Nature Inspired Computation and Applications Laboratory, USTC, China.

[3] S. Rahnamayan, G. G. Wang, Solving large scale optimization problems by opposition-based differential evolution (ode), WSEAS Transactions on Computers 7 (10) (2008) 1792–1804.

[4] M. Srinivas, L. Patnaik, Genetic algorithms: a survey, Computer 27 (6) (1994) 17–26.

[5] D. E. Goldberg, K. Deb, A comparative analysis of selection schemes used in genetic algorithms, in: Foundations of Genetic Algorithms, 1991, pp. 69–93.

[6] Z. Michalewicz, M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, Evolutionary computation 4 (1) (1996) 1–32.

[7] A. Eiben, Z. Michalewicz, M. Schoenauer, J. Smith, Parameter Control in Evolutionary Algorithms, in: F. G. Lobo, C. F. Lima, Z. Michalewicz (Eds.), Parameter Setting in Evolutionary Algorithms, Vol. 54 of Studies in Computational Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, Ch. 2, pp. 19–46.

[8] J. Vesterstrom, R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, Congress on Evolutionary Computation (CEC2004) 2 (2004) 1980–1987 Vol.2.

[9] L. Li, X. Yao, R. Stolkin, M. Gong, S. He, An evolutionary multi-objective approach to sparse reconstruction, IEEE Transactions on Evolutionary Computation 18 (6) (2014) 827–844.

[10] L. Jiao, L. Li, R. Shang, F. Liu, R. Stolkin, A novel selection evolutionary strategy for constrained optimization, Information Sciences 239 (2013) 122–141.

[11] J. Kennedy, Particle swarm optimization, Springer US, 2010.

[12] Z.-H. Zhan, J. Zhang, Y. Li, Y. hui Shi, Orthogonal learning particle swarm optimization, IEEE Transactions on Evolutionary Computation 15 (6) (2011) 832–847.

[13] J. Brest, S. Greiner, B. Boskovic, M. Mernik, V. Zumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, IEEE Transactions on Evolutionary Computation 10 (6) (2006) 646–657.

[14] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, Journal of global optimization 11 (4) (1997) 341–359.

[15] E. Aarts, J. Korst, Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing, Wiley, 1988.

[16] P. J. Van Laarhoven, E. H. Aarts, Simulated annealing, Springer, 1987.

[17] M. Dorigo, M. Birattari, Ant colony optimization, in: Encyclopedia of Machine Learning, Springer, 2010, pp. 36–39.

[18] M. Dorigo, T. Stützle, The ant colony optimization metaheuristic: Algorithms, applications, and advances, in: Handbook of metaheuristics, Springer, 2003, pp. 250–285.

[19] R. Descartes, J. Veitch, Discourse on method, Blue Unicorn Editions, 2002.

[20] M. Omidvar, X. Li, Y. Mei, X. Yao, Cooperative co-evolution with differential grouping for large scale optimization, IEEE Transactions on Evolutionary Computation 18 (3) (2014) 378–393.

[21] K. Tang, X. Li, P. N. Suganthan, Z. Yang, T. Weise, Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization, Tech. rep., University of Science and Technology of China

(USTC), School of Computer Science and Technology, Nature Inspired Computation and Applications Laboratory (NICAL): Héféi, Ānhuī, China (2010).
URL http://www.it-weise.de/documents/files/TLSYW2009BFFTCSSACOLSGO.pdf

[22] H. H. Rosenbrock, An automatic method for finding the greatest or least value of a function, The Computer Journal 3 (3) (1960) 175–184.

[23] Y.-W. Shang, Y.-H. Qiu, A Note on the Extended Rosenbrock Function, Evolutionary Computation 14 (1) (2006) 119–126.

[24] L. Li, R. Stolkin, L. Jiao, F. Liu, S. Wang, A compressed sensing approach for efficient ensemble learning, Pattern Recognition 47 (10) (2014) 3451 – 3465.

[25] G. B. Dantzig, P. Wolfe, Decomposition principle for linear programs, Operations research 8 (1) (1960) 101–111.

[26] A. van der Vaart, K. M. Merz, Divide and conquer interaction energy decomposition, The Journal of Physical Chemistry A 103 (17) (1999) 3321–3329.

[27] M. Reimann, K. Doerner, R. F. Hartl, D-ants: Savings based ants divide and conquer the vehicle routing problem, Computers & Operations Research 31 (4) (2004) 563–591.

[28] M. A. Potter, K. A. De Jong, A cooperative coevolutionary approach to function optimization, in: Parallel Problem Solving from NaturePPSN III, Springer, 1994, pp. 249–257.

[29] Y. Liu, X. Yao, Q. Zhao, T. Higuchi, Scaling up fast evolutionary programming with cooperative coevolution, in: Proceedings of the 2001 Congress on Evolutionary Computation, Vol. 2, IEEE, 2001, pp. 1101–1108.

[30] W. Chen, T. Weise, Z. Yang, K. Tang, Large-scale global optimization using cooperative coevolution with variable interaction learning, in: R. Schaefer, C. Cotta, J. Koodziej, G. Rudolph (Eds.), Parallel Problem Solving from Nature, PPSN XI, Vol. 6239 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 300–309.

[31] Z. Yang, K. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Information Sciences 178 (15) (2008) 2985 – 2999, nature Inspired Problem-Solving.

[32] T. Ray, X. Yao, A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning, in: Evolutionary Computation, 2009. CEC'09. IEEE Congress on, IEEE, 2009, pp. 983–989.

[33] M. N. Omidvar, X. Li, X. Yao, Cooperative co-evolution with delta grouping for large scale non-separable function optimization, in: 2010 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2010, pp. 1–8.

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

[35] M. Omidvar, X. Li, Z. Yang, X. Yao, Cooperative co-evolution for large scale optimization through more frequent random grouping, in: Evolutionary Computation (CEC), 2010 IEEE Congress on, 2010, pp. 1–8.

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

[37] Y.-j. Shi, H.-f. Teng, Z.-q. Li, Cooperative co-evolutionary differential evolution for function optimization, in: L. Wang, K. Chen, Y. Ong (Eds.), Advances in Natural Computation, Vol. 3611 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005, pp. 1080–1088.

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

[39] P. L. Toint, Test problems for partially separable optimization and results for the routine pspmin, Technical report, The university of Namur, Department of mathematics, Belgium.