Information Sciences 381 (2017) 142–160

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Cooperation coevolution with fast interdependency identification for large scale optimization

Xiao-Min Hu[a,*], Fei-Long He[a], Wei-Neng Chen[b], Jun Zhang[b]

[a] Sun Yat-sen University, Guangzhou, 510006, PR China
[b] South China University of Technology, Guangzhou, 510006, PR China

ARTICLE INFO

Article history:
Received 21 July 2016
Revised 19 October 2016
Accepted 24 November 2016
Available online 25 November 2016

Keywords:
Cooperative coevolution (CC)
Large scale global optimization (LSGO)
Problem decomposition
Differential evolution

ABSTRACT

Cooperative coevolution (CC) provides a powerful divide-and-conquer architecture for large scale global optimization (LSGO). However, its performance relies highly on decomposition. To make near-optimal decomposition, most developed decomposition strategies either cannot obtain the correct interdependency information or require a lot of fitness evaluations (FEs) in the identification. To alleviate the limitations in previous works, in this paper we propose a fast interdependency identification (FII) algorithm for CC in LSGO. The proposed algorithm firstly identifies separable and nonseparable variables efficiently. Then, the interdependency information of nonseparable variables is further investigated. To make near-optimal decomposition for CC, our algorithm avoids the necessity of obtaining the full interdependency information of nonseparable variables. Therefore, a significant number of FEs can be saved. Extensive experiments have been conducted on two suites of LSGO benchmark functions with up to 2000 variables. FII correctly identified the interdependency information on most benchmark functions with much fewer FEs in comparison with three state-of-the-art algorithms. Furthermore, combined with CC and coupled with a differential evolution variant serving as the optimizer, FII has shown its promising performance in LSGO.

1. Introduction

In real world, there are many large scale global optimization (LSGO) problems that arise in science or engineering [11,19,40,44,47,51]. LSGO problems are extremely difficult, mainly because the search space is large and the properties of the problem may change as the number of decision variables increases [25]. In order to address the problem of high-dimensional variables, Potter and De Jong [27] proposed a cooperative coevolution (CC) architecture, which provides a divide-and-conquer strategy for LSGO. Before the evolution, CC decomposes a problem into a certain number of sub-problems. Each sub-problem maintains a subpopulation, which is evolved by a traditional evolutionary algorithm (EA) in a round robin fashion [27]. Essentially, each subpopulation searches only a part of the variables, i.e. a projection of the whole search space [6]. By combing with different EAs, a variant of algorithms have been developed under CC [10], such as the cooperative particle swarm optimizer (CPSO) [1], cooperative coevolving PSO (CCPSO2) [12], covariance matrix adaptation evolution strategy with CC (CC-CMA-ES) [13], CC orthogonal artificial bee colony (CCOABC) [30], and fast evolutionary programming with CC (FEPCC) [14], etc.

* Corresponding author.
E-mail address: xiaomin.hu@aliyun.com (X.-M. Hu).

http://dx.doi.org/10.1016/j.ins.2016.11.013
0020-0255/

Although CC has achieved great success, its weakness in tackling nonseparable problems still exists. Based on whether a variable interacts with other variables, the variables in a problem can be divided into separable and nonseparable variables. If the function value of the problem follows a monotonic change in the value of a variable while the values of other variables are not changed, the function is separable and the variable is termed a separable variable. If no variables can be separated from the function, there must be interactions among the remaining variables, and thus the rest variables are nonseparable variables. In a nonseparable problem, changing one subcomponent will result in a "deforming" or "warping" of the respective fitness landscape of other interdependent subcomponents [9,28]. The strong interdependency in the subcomponents of nonseparable problems reduces the search capabilities of CC [6,8,27,34,45]. The reason may be due to its collaboration strategy, in which an individual is only evaluated by combining itself with the current best individuals in other subpopulations [26]. Moreover, the weakness of CC can be alleviated by making near-optimal decomposition which leads to minimum interdependency between the subcomponents [20]. Problem decomposition plays a key role in CC [6].

Dynamic decomposition strategies have been proposed, e.g. random grouping [48], multilevel CC (MLCC) [49], Delta grouping [23], and CC algorithm with correlation based adaptive variable partitioning (CCEA-AVP) [29]. However, these dynamic decomposition strategies were incapable of making near-optimal decomposition for CC on many problems [20,22,15,31]. Later, detection-based static decomposition strategies were developed, e.g. differential grouping (DG) [20], extended DG (XDG) [36], and global DG (GDG) [16]. DG [20] focused on uncovering the interdependency information of variables to make near-optimal decomposition. It improved the performance of CC on nonseparable problems with direct nonseparability [20], but it ignored indirect nonseparability, which was more common because two variables without direct interaction could be linked both by interacting with the other variables [36]. XDG [36] and GDG [16] were proposed to cover both direct and indirect nonseparability and they improved the identification performance especially on problems with indirect nonseparability. However, both of them required heavy computational cost in the identification process, resulting in unsatisfactory performance in the subsequent optimization.

If the interdependency information can be identified, near-optimal decomposition can be achieved for CC. However, it is unknown beforehand in real-world optimization problems [20]. Most detection-based algorithms in the literature use a pairwise fashion [2,20,36,16], in which heavy computational budget is required in the identification. Even though the interdependency information in a problem can be uncovered by these algorithms under the computational constraints, the advantages of making near-optimal decomposition are diminished. A more efficient identification algorithm is needed.

In this paper, a fast interdependency identification (FII) algorithm is proposed for CC. As a kind of detection-based static decomposition, the decomposition is performed based on the interdependency information identified by the algorithm. FII firstly identifies separable and nonseparable variables efficiently. If nonseparable variables are found, their interdependency information will be investigated further. Because identifying the interdependency between any pair of variables is avoided in our algorithm, there is no need to obtain full interdependency information of nonseparable variables in the proposed algorithm to make near-optimal decomposition for CC. Hence, a significant number of fitness evaluations (FEs) can be saved for the subsequent optimization. The near-optimal decomposition for CC can be achieved at the cost of small computational effort.

The performance of the algorithm is tested by the 1000-dimensional CEC'2010 benchmark functions [38]. FII can obtain correct identification results on most functions with much smaller computational effort compared with the previous work. In the experiment on LSGO, combined with CC and a differential evolution variant, FII shows competitive performance in comparison with six state-of-the-art algorithms. To verify the scalability of FII, a test suite with 2000-dimensional benchmark functions is developed based on CEC'2010 benchmark functions. FII also shows promising performance on solving functions.

The rest of this paper is organized as follows. Section 2 introduces the CC architecture and briefly reviews the related work. The theoretical basis is also presented in this section. Section 3 describes the proposed algorithm in detail. The experimental studies are presented and discussed in Section 4. At last, Section 5 makes the conclusion.

2. Background

This section describes the CC architecture and briefly reviews the decomposition strategies proposed for CC in the literature. At last, theoretical basis for interaction identification in this paper is presented.

2.1. Cooperative coevolution (CC)

CC adopts the notion of modularity to address complex problems. It is quite similar to some numerical optimization methods which optimize one function variable at a time while holding others constant [27]. Before the evolution, CC decomposes a problem into a certain number of sub-problems. With fewer variables, the sub-problems can be tackled comparatively easier [6]. Each sub-problem maintains a subpopulation, which is evolved in independently by a traditional EA. Therefore, each subpopulation only searches projections of the whole problem space [6]. During the evolutionary process, i.e. in a round-robin fashion, individuals within the same subpopulation compete with each other and they collaborate with the best members in the other subpopulations [8]. Communications among subpopulations occur when the fitness of an individual is needed to be evaluated. The evaluation is made by combining the individual with the current best ones from the

other subpopulations. In this way, CC introduces a kind of evolutionary pressure with regard to competition and collaboration. It provides an inspiring divide-and-conquer framework and gains great popularity in research [1,3,12,14,24,33,42,48] and applications [4,7,32,46,56].

2.2. Problem decomposition

Various problem decomposition strategies have been proposed to improve the performance of CC. Most decomposition strategies fall into two categories, i.e. static decomposition and dynamic decomposition [15]. The algorithms belonging to the two categories are introduced in the following two parts.

2.2.1. Static decomposition

In static decomposition strategies, the decomposition is performed only in the initiation of CC. In CC genetic algorithm (CCGA) [27], an N-dimensional problem is divided into N one-dimensional sub-problems. With this static decomposition strategy, CCGA performs better than the GA with a standard evolutionary strategy on problems without variable interaction [27]. However, the decomposition strategy has two drawbacks. First, if there are interdependencies between subcomponents, the search capacities of CC may be damaged drastically. As reported in [27], CCGA performs much worse than the GA with a standard evolutionary strategy on the Rosenbrock function due to the high interdependency between variables in the function [27,5,45]. Second, in CCGA each subpopulation focuses on a one-dimensional sub-problem. Such a one-variable partition method may waste computational resources in the evolution [21]. In [28], a splitting-into-half decomposition strategy is proposed but it does not scale up well in LSGO. In CPSO-S_K [1], variables are partitioned into K subcomponents, but its effectiveness is also reduced on nonseparable problems [12].

In recent years, various detection-based static decomposition methods are developed and they focus on making near-optimal decomposition by exploring the interdependency information in the problem. Variable interaction is identified mostly by perturbation methods such as the linkage identification by non-monotonicity detection (LIMD) [17], linkage identification by nonlinearity check (LINC) [18], LINC for real-coded GAs (LINC-R) [41], DG [20], XDG [36] and GDG [16]. CC with variable interaction learning (CCVIL) was put forward by Chen et al. [2] similar to the methods in [43,5,35]. The basis behind CCVIL is that the fitness landscape of one variable will be changed when its interacting variable is changed [43]. So if two variables x_i and x_j are interacting, the following predict formula can be satisfied in a minimization problem [2,43].

$$\begin{aligned}&\exists \vec{X}^*, x'_i, x'_j((f(\vec{X}^*) < f(x_1^*, \cdots, x'_i, \cdots, x_j^*, \cdots, x_n^*))\\ &\quad \wedge (f(x_1^*, \cdots, x'_i, \cdots, x'_j, \cdots, x_n^*) < f(x_1^*, \cdots, x'_i, \cdots, x_j^*, \cdots, x_n^*)))\end{aligned} \tag{1}$$

Here $\vec{X}^* = (x_1^*, x_2^*, \cdots, x_n^*)$ is the best solution found so far. Two variables x_i and x_j are randomly selected and their inferior variable values are denoted by x'_i and x'_j. It can be observed that $f(\vec{X}^*) < f(x_1^*, \cdots, x'_i, \cdots, x_j^*, \cdots, x_n^*)$. But if two variables are interacting, there will exist x'_i and x'_j such that $f(x_1^*, \cdots, x'_i, \cdots, x'_j, \cdots, x_n^*) < f(x_1^*, \cdots, x'_i, \cdots, x_j^*, \cdots, x_n^*)$, i.e. a solution with two inferior variable values may be better than the one with only one inferior variable value. The existential quantifier in the predict formula indicates that the algorithm may need to sample many times to verify the satisfiability [20], using a considerable number of FEs.

In the DG proposed by Omidvar et al. [20], in order to investigate whether x_i interacts with x_j, two values Δ_1 and Δ_2 are calculated as

$$\Delta_1 = f(l_1, \cdots, l_i, \cdots, l_j, \cdots, l_n) - f(l_1, \cdots, u_i, \cdots, l_j, \cdots, l_n) \tag{2}$$

$$\Delta_2 = f(l_1, \cdots, l_i, \cdots, c, \cdots, l_n) - f(l_1, \cdots, u_i, \cdots, c, \cdots, l_n) \tag{3}$$

The lower bound and upper bound of the ith variable are denoted as l_i and u_i respectively. Compared with (2), (3) is just modified by replacing l_j by an arbitrary value c ($c \neq l_j$). According to the analysis in [20], if the value of $|\Delta_1 - \Delta_2|$ is larger than a predefined small value, x_i and x_j interacts mutually. DG has achieved success in identifying the interdependency information for the CEC'2010 benchmark functions [38], and the experiments have also shown its benefits in making near-optimal decomposition in the optimization [20]. However, DG is not efficient in identification. It has a time complexity of $O(N^2)$ with respect to FEs on fully separable N-dimensional functions. More than 1 million FEs are required on the 1000-dimensional functions. Moreover, DG only considers direct nonseparability, but ignores indirect nonseparability which is more common in real-world problems. As the experimental results in [20] show, DG is incapable of correctly identifying the interdependency information in Rosenbrock functions which have indirect nonseparability [20,38].

Based on DG, Sun et al. [36] proposed XDG to cover both direct and indirect nonseparability forms. XDG not only identifies the variable interactions by a similar method as that in DG but also detects indirect nonseparability information. XDG has an improved identification performance on the CEC'2010 benchmark functions especially on Rosenbrock functions. However, XDG is based on a pairwise fashion to identify the interdependency information, which is computationally expensive. As reported in [36], XDG requires about 1 million FEs on the most CEC'2010 benchmark functions.

Similarly, a GDG method [16] was proposed. It maintains global information by an interaction matrix [16]. GDG acquires the interaction information between any pair of variables with a time complexity of $O(N^2)$ with respect to FEs for N-dimensional problems [16]. However, in order to make near-optimal decomposition for CC, it is unnecessary to obtain the

full interaction information. The advantages of XDG and GDG in LSGO are diminished since a lot of computational budget is consumed in identifying interdependency information.

2.2.2. Dynamic decomposition

In dynamic decomposition strategies, variables are partitioned during the evolutionary process. Random grouping represents a classical dynamic decomposition strategy. It was firstly introduced in CC-based EA with random grouping (EACC-G) proposed by Yang et al. [48,50] and it was also adopted in CCPSO2 [12] and CCOABC [30]. At the beginning of each evolutionary cycle in random grouping, all variables are randomly assigned to a certain number of groups (e.g., 10 groups) [48,12]. Two interacting variables are more likely to be grouped into a subcomponent during the evolutionary process [48]. But if there are many interacting variables in a problem, the random grouping scheme is incapable of capturing all of them into one subcomponent [20,22]. Based on random grouping, Yang et al. proposed a MLCC framework [49]. In MLCC, the subcomponent size is selected from a decomposer pool according to the probabilities measured by historical performance. To some extent, MLCC adapts the subcomponent size to different interaction levels. As reported in [49], MLCC performs better than EACC-G on the CEC'2008 benchmark functions [37]. An improved variant [22] of MLCC suggests increasing the frequency of random grouping so that the probability of grouping interacting variables into one subcomponent can be increased. However, even though some improvement has been made, the above stochastic decomposition methods are unable to make near-optimal decomposition for problems with many nonseparable variables [20].

Other kinds of dynamic decomposition, such as Delta grouping [23] and CCEA-AVP [29], were developed in which the historical information was utilized. Delta grouping [23] was proposed based on the empirical studies which found that the movements of interacting variables were limited during the evolutionary process if they were grouped into different subcomponents. In Delta grouping, the average change (termed a delta value) of each variable between two consecutive cycles is calculated and all variables are sorted in a descending order according to their average delta values. The ordered variables are partitioned into a predefined number of subcomponents. In this way, the interacting variables with limited average delta value can be grouped together [23], but Delta grouping may lose its efficacy on some partially additive separable functions with more than one nonseparable subcomponent [20]. In CCEA-AVP [29], after a predefined number of generations, a correlation matrix is created and variables with a correlation coefficient greater than a threshold are grouped into one subcomponent. The correlation based variable partition is operated at every cycle. Although CCEA-AVP uses heavy computation cost but it is just capable of detecting linear variable dependencies without covering nonlinear interdependency [15,31].

2.3. Theoretical basis

The theoretical basis for variable interdependency identification can be described as follows [20,36]. Suppose $F(\vec{x})$ is a given problem. If x_i interacts with x_j, then we have a function

$$\frac{\partial F}{\partial x_i} = g(\vec{x}_{\text{sub}}) \tag{4}$$

in which $x_j \in \vec{x}_{\text{sub}}$, $\vec{x}_{\text{sub}} \subseteq \vec{x}$.

According to the Newton-Leibniz formula, we have

$$F(\vec{x})|_{x_i=b} - F(\vec{x})|_{x_i=a} = \int_a^b \frac{\partial F}{\partial x_i} dx_i \tag{5}$$

So based on (4) and (5), we can get

$$F(\vec{x})|_{x_i=b} - F(\vec{x})|_{x_i=a} = \int_a^b g(\vec{x}_{\text{sub}}) dx_i \tag{6}$$

Since $x_j \in \vec{x}_{\text{sub}}$, for x_i its difference value obtained from the left part of this equation is affected under the perturbation of x_j. If x_i is separable, $g(\vec{x}_{\text{sub}})$is a function of x_i or a constant. So the perturbation of all other variables cannot affect the left part of (6). Therefore, by simply investigating a variable's difference value with a perturbation method, we can obtain the variable interdependency information of a problem [20,36].

3. Algorithm

In this section, the proposed fast interdependency identification (FII) algorithm is described in detail. Then the basic framework of combining FII with CC is presented. At last, the number of FEs required by FII and the time complexity are analyzed.

3.1. Fast interdependency identification (FII)

Generally, the interdependency information of the variables keeps consistent in the solution space. So we can just choose a sample solution and then make an investigation according to the theoretical basis in Section 2. If all variables are correctly identified as separable or nonseparable, and all nonseparable variables are correctly partitioned into the smallest independent subcomponents, the interdependency information is correctly uncovered. Therefore, FII has the following two identification stages.

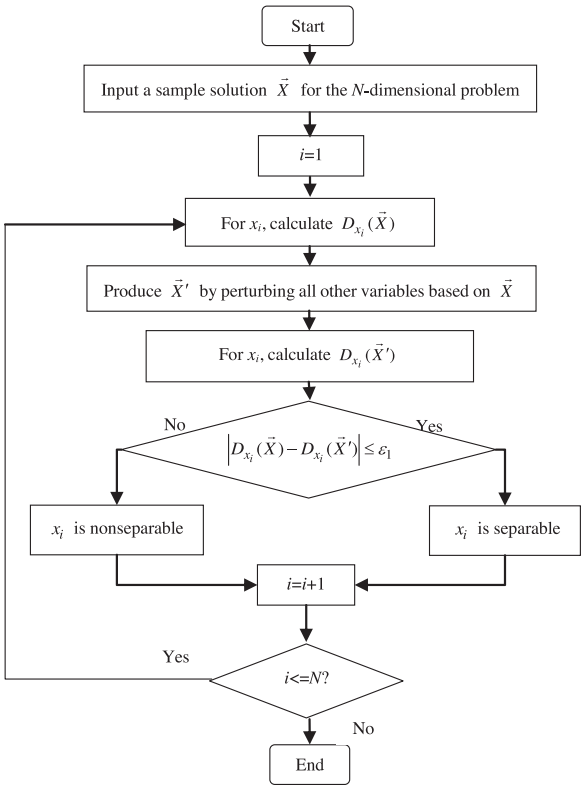


Fig. 1. Flow chart of the first identification stage.

3.1.1. Stage 1 – identification of separable and nonseparable variables

In this stage, separable and nonseparable variables are identified. For x_i, its difference value $D_{x_i}(\vec{X})$ is calculated according to

$$D_{x_i}(\vec{X}) = F(\vec{X})\big|_{x_i=a+\Delta} - F(\vec{X})|_{x_i=a}, \tag{7}$$

where $\vec{X}$ is a sample solution and Δ is an increment for x_i based on $x_i = a$. According to (4)–(6), if x_i is a separable variable, $D_{x_i}(\vec{X})$ is not changed under the perturbation of all other variables. In our algorithm, by perturbing all other variables, a new solution $\vec{X}$ is produced. Then a new difference value $D_{x_i}(\vec{X}')$ is calculated for x_i by

$$D_{x_i}(\vec{X}') = F(\vec{X}')\big|_{x_i=a+\Delta} - F(\vec{X}')|_{x_i=a}, \tag{8}$$

where a and Δ are the same as in (7). If x_i is a separable variable, then $D_{x_i}(\vec{X})$ equals to $D_{x_i}(\vec{X}')$. In the algorithm, a predefined threshold ε_1 is used as an allowable error for the equality, i.e., if

$$\left|D_{x_i}(\vec{X}) - D_{x_i}(\vec{X}')\right| \le \varepsilon_1, \tag{9}$$

the values of $D_{x_i}(\vec{X})$ and $D_{x_i}(\vec{X}')$ are equal, and thus x_i is captured as a separable variable. The flow chart of this identification stage is shown in Fig. 1. If all decision variables are separable, the interdependency information is uncovered just in this stage. If a certain number of nonseparable variables are identified, their interdependency information is further explored in the next stage.

3.1.2. Stage 2 – identification of the interdependency information of nonseparable variables

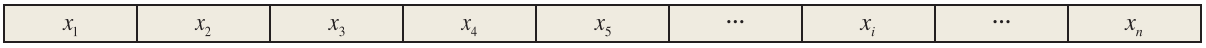
In real-world problems, there are two kinds of nonseparability, i.e., direct and indirect nonseparability [36]. They are defined as follows.

Definition 1. If x_i and x_j satisfy (4), they have direct nonseparability.

Definition 2. If x_i and x_j do not satisfy (4) but they are both directly nonseparable with x_k, they have indirect nonseparability.

For a problem with only direct nonseparability, any pair of variables belonging to a nonseparable subcomponent satisfies (4). If indirect nonseparability exists, (4) is not satisfied. In order to make near-optimal decomposition, all directly or indirectly nonseparable variables should be grouped into one subcomponent.

Nonseparable variables discovered:

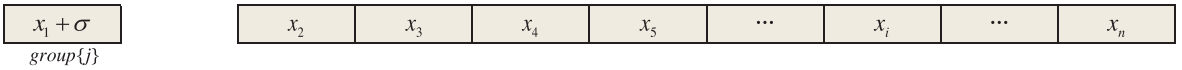


Identification process for nonseparable variables:

Initially j=1.

Step1. Choose a nonseparable variable and perturb it to produce a perturbed solution $\vec{X}'$ based on a sample solution $\vec{X}$.

Suppose x_1 is perturbed.



Step2. Put the variable that satisfies $\left|D_{x_i}(\vec{X})-D_{x_i}(\vec{X}')\right|>\varepsilon_2$ into the subcomponent $group\{j\}$ with the perturbed variable(s).

Suppose x_2, x_3 and x_4 satisfy the condition.

Step3. The variables newly captured are perturbed together based on $\vec{X}$ to find more variables belonging to $group\ \{j\}$.

Step4. Repeat **Step2- Step3** till no variable can be added into $group\{j\}$. Then $group\{j\}$ is formed as a subcomponent. Set j =j+1.

Step5. Repeat **Step1- Step4** till no nonseparable variable is remaining for forming groups.

Fig. 2. An example of identifying interdependencies of nonseparable variables.

In the first step, a nonseparable variable x_p in a solution $\vec{X}$ is perturbed to produce a perturbation vector $\vec{X}'$. For each of the remaining nonseparable variables, the difference value $D_{x_i}(\vec{X}')$ is calculated by (8). According to (6), if x_i interacts with x_p (i.e., $x_p \in \vec{x}_{\text{sub}}$ in (4)), the left part of (6) is affected under the perturbation of x_p. Hence, x_i interacts with x_p, if

$$\left|D_{x_i}(\vec{X})-D_{x_i}(\vec{X}')\right|>\varepsilon_2, \tag{10}$$

where ε_2 is a predefined threshold. In the second step, the variable satisfies (10) is captured and put into the nonseperable subcomponent which is denoted as $group\{j\}$ with x_p, initially $j=1$.

All indirect nonseparability should be considered, because there may be variables that do not interact with x_p but interact with the newly captured variables. In order to identify more nonseparable variables and add them to $group\{j\}$, in the third step the captured variables are then perturbed together based on $\vec{X}$ to produce a new perturbed solution.

Under the perturbation of all newly captured variables, if a variable interacts with at least one of the perturbed variables, it will satisfy (10) and be put into $group\{j\}$. In this way, there is no need to explore the specific interaction information between a remaining variable and each of the perturbed variables, and thus the proposed algorithm avoids a lot of extra FEs in the identification.

As long as there are nonseparable variables identified into $group\{j\}$, these newly captured variables should be perturbed together next time (produce $\vec{X}'$ based on the solution $\vec{X}$) to discover more interactions. This process is repeated until no more variables satisfies (10) and finally the nonseparable subcomponent $group\{j\}$ is established. The process of the second identification stage is illustrated by an example in Fig. 2. In Fig. 2, a nonseparable variable x_1 is perturbed and three variables (x_2, x_3 and x_4) are captured and put into $group\{j\}$. These three newly captured variables are then perturbed together for finding more nonseparable variables belonging to $group\{j\}$.When the nonseparable subcomponent $group\{j\}$ is formed, $j=j+1$ and the interdependency information of the remaining variables will be investigated until no nonseparable variable is remaining for forming groups.

3.2. Implementation of CC with FII

The pseudocode of FII is described in Fig. 3, whereas Fig. 4 is an operation used in FII. In Fig. 4, a given variable index set is passed to this method as *dim* to calculate the difference value (recorded as $\vec{v}$ in Fig. 4) for the corresponding variables according to (7) and (8). The number of FEs used is recorded every time this method is invoked. In Fig. 3, a sample solution $\vec{X}$ is randomly generated (line 05). The first identification stage based on the condition given by (9) (line 13) is presented in lines 06–18. For every variable, its difference value before the perturbation is calculated based on $\vec{X}$ (line 06). The difference

```
[ group , seps , FEs_used ] = Identify (F, σ, ε1, ε2)
Require: F, σ, ε1, ε2          Output: group, seps, FEs_used
01: FEs_used = 0 ;                                  // to record FEs
02: seps ← [ ];                                     // separable variable index set
03: nonseps ← [ ];                                  // nonseparable variable index set
04: all_dims ← [1,2,3,⋯,N]                          // indexes of all decision variables
05: Randomly generate a sample solution X⃗ ;
06: [FEs, dv⃗] = Diff(F, X⃗, Δ, all_dims) ;           // Fig. 4
07: FEs_used = FEs_used + FEs ;
08: For i in all_dims                               // Identification Stage 1
09:     X⃗′ = X⃗ + σ ;                               // σ is added for each dimension
10:     X⃗′(i) = X⃗(i) ;                             // keep ith dimension in X⃗′ consistent with that in X⃗
11:     [FEs, pv⃗] = Diff(F, X⃗′, Δ, i) ;            // Fig. 4
12:     FEs_used = FEs_used + FEs ;
13:     If | pv⃗(i) − dv⃗(i) | ≤ ε1
14:         seps ← seps ∪ i ;
15:     Else
16:         nonseps ← nonseps ∪ i ;
17:     End
18: End
19: group ← {} ;    // nonseparable subcomponents
20: j = 1 ;
21: While nonseps is not empty                      // Identification Stage 2
22:     pdim ← nonseps(1) ;      // nonseps(1) is chosen to be perturbed
23:     group{j} ← nonseps(1) ;
24:     nonseps ← nonseps − group{j} ;
25:     While both pdim and nonseps are not empty
26:         X⃗′ = X⃗ ;
27:         X⃗′(pdim) = X⃗′(pdim) + σ ;  // σ is added for each dimension belonging to pdim
28:         [FEs, pv⃗] = Diff(F, X⃗′, Δ, nonseps) ;   // Fig. 4
29:         FEs_used = FEs_used + FEs ;
30:         pdim ← [ ] ;          // perturbation index set containing newly captured variable
31:         For i in nonseps
32:             If ( | pv⃗(i) − dv⃗(i) | > ε2 )
33:                 group{j} ← group{j} ∪ i ;
34:                 pdim ← pdim ∪ i ;
35:             End
36:         End
37:         nonseps ← nonseps − group{j} ;
38:     End
39:     If | group{j} | = 1
40:         seps ← seps ∪ group{j} ;
41:     Else
42:         j = j + 1 ;
43:     End
44: End
```

Fig. 3. Pseudocode of the fast interdependency identification (FII).

value $D_{x_i}(\vec{X}')$ for x_i is calculated (line 11) after all other variables are perturbed together (lines 09–10). The separable and nonseparable variable indexes are respectively put into *seps* and *nonseps* (lines 14 and 16). Lines 19–44 describe the second identification stage. Firstly, a nonseparable variable is selected to be perturbed (lines 22 and 27). After the perturbation, for each of the remaining nonseparable variables its difference value is calculated in line 28. If (10) is satisfied (line 32), the nonseparable variable is put into a group with the perturbation variable(s) (line 33). The newly captured variables also form a new perturbations variable set *pdim* (lines 30 and 34). All variables in this set are perturbed together next time (line 27) to discover more interactions.

$[FEs, \vec{v}] = \text{Diff}(\text{F}, \vec{X}, \Delta, dim)$

Input: $\text{F}, \vec{X}, \Delta, dim$ **Output**: $FEs, \vec{v}$

01: $FEs = 0$;
02: $value1 = \text{F}(\vec{X})$;
03: $FEs = FEs + 1$;
04: **For** i in 1 to dim // $|dim|$ times loop
05: $\overrightarrow{Xd} = \vec{X}$;
06: $\overrightarrow{Xd}(i) = \overrightarrow{Xd}(i) + \Delta$; // Δ is an increment for x in $\vec{X}$
07: $value2 = \text{F}(\overrightarrow{Xd})$;
08: $\vec{v}(i) = value2 - value1$;
09: $FEs = FEs + 1$;
10: **End**

Fig. 4. Pseudocode of the computation the difference value $\vec{v}$.

Require: F, $FEs\_max, S_{\max}, pop$

01: $subcmpt \leftarrow \{\}$; // a problem's subcomponents
02: $[group, seps, FEs\_used] = \text{Identify}(\text{F}, \sigma, \varepsilon_1, \varepsilon_2)$;
03: **If** $group$ is not empty
04: $subcmpt \leftarrow group$;
05: **End**
06: **If** $seps$ is not empty
07 $j = |subcmpt| + 1$;
08: **If** $|seps| \geq S_{\max}$
09: $n = \text{floor}(|seps| / S_{\max})$;
10: divide $seps$ into n parts: $seps\{1\}, seps\{2\}, \ldots, seps\{n\}$;;
11: **For** $i = 1$ to n
12: $subcmpt\{j\} \leftarrow seps\{i\}$;
13: $j = j + 1$;
14: **End**
15: **Else**
16: $subcmpt\{j\} \leftarrow seps$;
17: **End**
18: **End**

19: $FEs\_max = FEs\_max - FEs\_used$;
20: $FEs = 0$;
21: **While** ($FEs < FEs\_max$)
22: **For** $i = 1$ to $|subcmpt|$
23: $iter = 1$;
24: optimizer($subcmpt\{i\}, pop\{i\}, iter$);
25 $FEs = FEs + popsize * iter$;
26: **End**
27: **End**

Fig. 5. Pseudocode of the CC with FII.

It should be noted that the perturbation vector $\vec{X}'$ is always produced based on the sample solution $\vec{X}$ (lines 26 and 27). This identification process for forming a nonseparable subcomponent is operated repeatedly till no nonseparable variable is captured or remaining (line 25). Thus a nonseparable subcomponent is formed. Then the above process is operated on the remaining variables in *nonseps* (lines 37 and 21). If a subcomponent formed has just one variable (line 39 in Fig. 3), i.e. no variable is captured interacting with it in the second stage, it is finally regarded as a separable variable (line 40).

The framework of combing CC with FII is presented in Fig. 5. Firstly the interdependency information is identified by FII (line 02). Nonseparable variables are partitioned into a certain number of subcomponents (line 04). Intuitively, each separable variable can form a subcomponent respectively. However, the decomposition may lead to wasting computational resources [21]. It is also inappropriate that all separable variables form just one subcomponent as that in [20]. If there are a large number of separable variables (e.g., fully separable problems), the search space of such subcomponent is too large and

it is difficult for the optimizer to tackle. In our algorithm, the size of the separable variable subcomponent is controlled. If the number of separable variables exceeds $S_{\max}$, the separable subcomponent will be divided into smaller ones (lines 08–14). After the decomposition, the remaining FEs are used in the optimization (line 19). The optimizer can be any traditional EA algorithm. In every cycle, each subpopulation is evolved just once by the optimizer (line 24).

3.3. Fitness evaluations and time complexity

In Fig. 4, the number of FEs used is

$$1 + |dim| \tag{11}$$

In the first identification stage of Fig. 3, the invocation of Fig. 4 occurs on lines 06 and 11. On line 06 in Fig. 3, using N as the dimension of the problem and $|all\_dims| = N$, the number of FEs on this step is $(1+N)$. Line 11 in Fig. 3 is in a loop which executes N times and each time only one variable is passed to Fig. 4, so it requires $2N$ FEs. Hence, in the first stage the number of FEs (S_1) used is

$$S_1 = 3N + 1, \tag{12}$$

and the time complexity with respect to FEs is

$$O(S_1) = O(3N + 1) = O(N). \tag{13}$$

If n nonseparable variables are discovered ($n \le N$), their interdependency information is explored in the second stage (lines 19–44 in Fig. 3). The method described by Fig. 4 is invoked on line 28 where *nonseps* is the index set of the remaining nonseparable variables not belonging to *group* (lines 24 and 37). According to (11), the invocation of this method on line 28 consumes $(1+|nonseps|)$ FEs. If only direct nonseparability exists in the problem, lines 26–37 execute two times to form a nonseparable subcomponent. The exception is forming the last nonseparable group and lines 26–37 execute one time because no nonseparable variable remains after that). So the number of FEs (S_2) used in the second stage is calculated as

$$\begin{aligned} S_2 = {} & (1 + n - 1) + (1 + n - d_1) + (1 + n - d_1 - 1) + (1 + n - d_1 - d_2) + (1 + n - d_1 - d_2 - 1) \\ & + \cdots + (1 + n - d_1 - d_2 - \cdots - d_{k-1}) + (1 + n - d_1 - d_2 - \cdots - d_{k-1} - 1), \end{aligned} \tag{14}$$

where k is the number of nonseparable subcomponents and $d_{k\text{-}1}$ represents the size of the (k−1)th nonseparable group (subcomponent). If all nonseparable subcomponents have an equal size d, i.e.

$$d_1 = d_2 = \cdots = d_k = d, \tag{15}$$

Hence, (14) can be expressed by

$$S_2 = n + (1 + n - d) + (1 + n - d - 1) + (1 + n - 2d) + \cdots + [1 + n - (k-1)d] + [1 + n - (k-1)d - 1]. \tag{16}$$

So with $n = k^* d$,

$$S_2 = kn + k - 1 \tag{17}$$

With (12) and (17), the total number of FEs (S) for such problem is

$$S = S_1 + S_2 = 3N + kn + k \tag{18}$$

So the time complexity with respect to FEs is

$$O(S) = O(3N + kn + k) = O(N + kn) \tag{19}$$

Both of XDG [36] and GDG [16] have the time complexity of $O(N^2)$ on such problems. The time complexity of DG on such problems is $O(kn + (N-n)^2)$. Intuitively, DG seems more efficient than XDG and GDG which were developed based on DG. But it should be noted that DG has not covered indirect nonseparability form, which can lead to improvement in time complexity. As an algorithm that covers both of nonseparability forms, FII is more efficient than XDG and GDG.

4. Experimental studies

In this section, firstly a suite of 1000-dimensional benchmark functions is introduced to study the efficacy of FII. The identification results of FII are compared with that of DG, XDG and GDG. To show the advantages of combining FII with CC in LSGO, the optimization results are presented and compared with that of 6 decomposition algorithms: EACC-G [48], MLCC [49], Delta grouping [23], DG [20], XDG [36] and GDG [16]. To further verify the scalability of FII, the empirical sudies are conducted on a suite of 2000-dimensional benchmark functions.

4.1. Parameter settings

In the proposed FII, each variable is classified as a separable variable in the first stage or is included in a nonseparable subcomponent in the second stage. Therefore, the values of the two thresholds ε_1 and ε_2 are generally set to be the same. Based on our empirical study, the performance of the proposed method is not sensitive to the parameters. The parameters are set as $\varepsilon_1 = \varepsilon_2 = 10^{-2}$, $\sigma = 10$, $\Delta = 10$, $S_{\max} = 200$. For the tested algorithms, the subpopulations are evolved once in every evolutionary cycle. For FII, DG, XDG and GDG, the separable variables identified are divided into smaller subcomponents if the number of them exceeds threshold $S_{\max}$ as Fig. 5 describes. The parameters in EACC-G, MLCC, Delta grouping, DG, XDG and GDG are set according to the respective papers [48,49,23,20,36,16].

In recent years, various works focused on the research and application of EAs [[39,52,53] [54,55]]. In this paper, we chose a newly-developed variant of differential evolution (IDE) with the individual-dependent mechanism [39] as the optimizer for each algorithm. The population size is set to 50. In each of 25 independent runs, for the CEC'2010 benchmark functions the maximal number of FEs is set to 3×10^6 and for the 2000-dimensional benchmark functions it is set to 6×10^6. Other parameters in IDE are set according to the suggestions in [39].

4.2. CEC'2010 benchmark functions

The CEC'2010 [38] benchmark functions are chosen in our experiments, because they are widely tested in the field of LSGO and we can extend them to higher-dimensional functions to conduct scalability studies. This suite includes 20 functions as follows [38].

Category 1: Separable functions (f_1-f_3);
Category 2: Single-group m-nonseparable functions (f_4-f_8);
Category 3: $\frac{D}{2m}$−group m-nonseparable functions (f_9-f_{13});
Category 4: $\frac{D}{m}$−group m-nonseparable functions (f_{14}-f_{18});
Category 5: Nonseparable function (f_{19},f_{20}).

In the above description, D is the number of decision variables and m is the size of nonseparable subcomponent ($D = 1000$, $m = 50$). For most nonseparable problems except for F_8, F_{13}, F_{18} and F_{20}, the variable nonseparability derives from the coordinate rotation [38] which results in a kind of direct nonseparability.

4.2.1. Identification results and analysis

Table 1 shows the identification results of FII, DG, XDG and GDG on the CEC'2010 benchmark functions. In Table 1, it can be observed that FII achieved 100% identification accuracy on 16 functions except for f_4, f_8, f_{11} and f_{16}. DG, XDG and GDG respectively correctly identify the interdependency information on 12, 18 and 18 functions. DG obtains poor identification results on f_8, f_{13}, f_{18} and f_{20} in each of which there is indirect nonseparability. The capability of FII in identifying indirect nonseparability is testified by the experimental results on most Rosenbrock functions (f_{13}, f_{18} and f_{20}).

To show the drawback of DG on problems with indirect nonseparability, consider a Rosenbrock function with 3 variables.

$$F(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + 100(x_2^2 - x_3)^2 + (x_2 - 1)^2 \tag{20}$$

According to (4), there is no interaction between x_1 and x_3 in (20). But it is obvious that x_1 interacts with x_2 and x_2 interacts with x_3. Based on Definition 2, x_1 and x_3 are indirectly nonseparable. Hence, x_1, x_2 and x_3 should be grouped into one subcomponent together. However, DG just considers the situation where there is only direct nonseparability. In the above example, if x_1 is perturbed, DG can identify x_2 interacting with x_1, so it put x_1 and x_2 into one subcomponent. Then DG will investigate the interdependency information of the remaining variable (x_3) but ignore that x_2 interacts with x_3.

Table 1 also shows the number of FEs each algorithm used in the identification. GDG explores the interaction information for each pair of variables. For each function in this suite, GDG requires 501511 FEs in the identification. For fully-separable functions (f_1-f_3), both DG and XDG consume 1001000 FEs. In contrast, FII can fast identify separable variables in its first stage. It just used 3001 FEs which is consistent with (12) on f_1-f_3 and has a linear time complexity with respect to FEs according to (13). For partially additive separable functions (f_5-f_7, f_9, f_{10}, f_{12}, f_{13}) with separable variables, FII still uses the smallest FEs for correctly identifying the information. For partially additive separable functions without separable variable (f_{14}-f_{18}), FII uses several thousand FEs more than DG and obtains correct identification results on f_{14}, f_{15}, f_{17}. The number of FEs used by FII on f_{14}, f_{15}, f_{17} is consistent with (18). DG performs a bit competitively on f_{14}-f_{18}. Because only direct nonseparability exists in these functions, DG can get correct results and the ignorance of indirect nonseparability means the consumption of a lot of extra FEs is avoided. FII consumes a bit more FEs than DG or GDG for making correct identification on nonseparable function f_{19} and f_{20}. With a covering indirect nonseparability form, both XDG and GDG are in a pairwise fashion, so compared with FII, more FEs are required on most benchmark functions. Table 1 shows that XDG uses about 1 million FEs on most functions except for f_4, f_8 (wrongly identified) and f_{19} (just one subcomponent with only direct nonseparability). GDG maintains global information by an interaction matrix [16] for all decision variables. So GDG is also computationally expensive. In the experiment, FII has shown its capabilities of efficiently and correctly identifying interdependency information on the most CEC'2010 benchmark functions.

Table 1
Interdependency identification results on CEC'2010 benchmark functions.

Func	SVars	NVars	Nonsep Subcmpt	FII/ DG/ XDG/ GDG				
				SVars Identified	NVars Identified	Formed Nonsep Subcmpt	Identification Accuracy (%)	Fitness Evaluations
1	1000	0	0	1000/1000/1000/1000	0/ 0/ 0/ 0	0/ 0/ 0/ 0	100/ 100/ 100/ 100	3001/ 1001000 /1001000/ 501511
2	1000	0	0	1000/1000/1000/1000	0/ 0/ 0/ 0	0/ 0/ 0/ 0	100/ 100/ 100/ 100	3001/ 1001000/ 1001000/ 501511
3	1000	0	0	1000/ 1000/ 1000/ 0	0/ 0/ 0/ 1000	0 /0 /0/ 1	100/ 100/ 100/ –	3001/ 1001000/ 1001000 / 501511
4	950	50	1	429/ 33/ 0/ 950	571/ 967/ 1000/50	1/ 10/ 1/ 1	–/ –/ –/ 100	3673/ 14554/ 80526/ 501511
5	950	50	1	950/ 950/ 950/ 950	50/ 50/ 50/ 50	1/ 1/ 1/ 1	100/ 100/ 100/ 100	3051/ 905450/ 998648/ 501511
6	950	50	1	950/ 950/ 950/ 950	50/ 50/ 50/ 50	1/ 1/ 1/ 1	100/ 100/ 100/ 100	3051/ 906332/ 998648/ 501511
7	950	50	1	950/ 248/ 950/ 950	50/ 752/ 50/ 50	1/ 4/ 1/ 1	100/ –/ 100/ 100	3051/ 67742/ 998648/ 501511
8	950	50	1	479/ 134/ 0/ 950	521/ 866/ 1000/ 50	2/ 5/ 2/ 1	–/ –/ –/ 100	18850/ 23286/ 121658/ 501511
9	500	500	10	500/ 500/ 500/ 500	500/ 500/ 500/ 500	10/10/10/10	100/ 100/ 100/ 100	8010/ 270802/ 977480/ 501511
10	500	500	10	500/ 500/ 500/ 500	500/ 500/ 500/ 500	10/10/10/ 10	100/ 100/ 100/ 100	8010/ 272958/ 977480/ 501511
11	500	500	10	522/ 501/ 500/ 0	478/ 499/ 500/ 1000	10/10/10/10	–/ –/ 100/ –	9255/ 270640/ 978528/ 501511
12	500	500	10	500/ 500/ 500/ 500	500/ 500/ 500/ 500	10/10/10/10	100/ 100/ 100/ 100	8010/ 271390/ 977480/ 501511
13	500	500	10	500/ 131/ 500/ 500	500/ 869/ 500/ 500	10/34/10/10	100/ –/ 100/ 100	96183/ 50328/ 1000154/ 501511
14	0	1000	20	0/ 0/ 0/ 0	1000/1000/1000/1000	20/20/20/20	100/ 100/ 100/ 100	23020/ 21000/ 953960/ 501511
15	0	1000	20	0/ 0/ 0	1000/1000/1000/1000	20/20/20/20	100/ 100/ 100/ 100	23020/ 21000/ 953962/ 501511
16	0	1000	20	43/ 4/ 0/ 0	957/ 996/ 1000/ 1000	20/20/20/20	–/ –/ 100/ 100	30911/ 21128/ 956286/ 501511
17	0	1000	20	0/ 0/ 0/ 0	1000/1000/1000/1000	20/20/20/20	100/ 100/ 100/ 100	23020/ 21000/ 953960/ 501511
18	0	1000	20	0/ 78/ 0/ 0	1000/ 922/ 1000/ 1000	20/50/20/ 20	100/ –/ 100/ 100	369902/ 39624/ 999340/ 501511
19	0	1000	1	0/ 0/ 0/ 0	1000/1000/1000/1000	1/ 1/ 1/ 1	100/ 100/ 100/ 100	4001/ 2000/ 3998/ 501511
20	0	1000	1	0/ 33/ 0/ 0	1000/ 967/ 1000/ 1000	1/ 241/ 1/ 1	100/ –/ 100/ 100	503500/ 155430/ 1001000/ 501511

1. SVars: the number of separable variables.
2. NVars: the number of nonseparable variables;
3. Nonsep Subcmpt: the number of nonseparable subcomponents.
4. The results of FII, DG, XDG and GDG are separated by "/".
5. Identification Accuracy is 100% if all decision variables are correctly identified and partitioned, otherwise it is marked by "–".

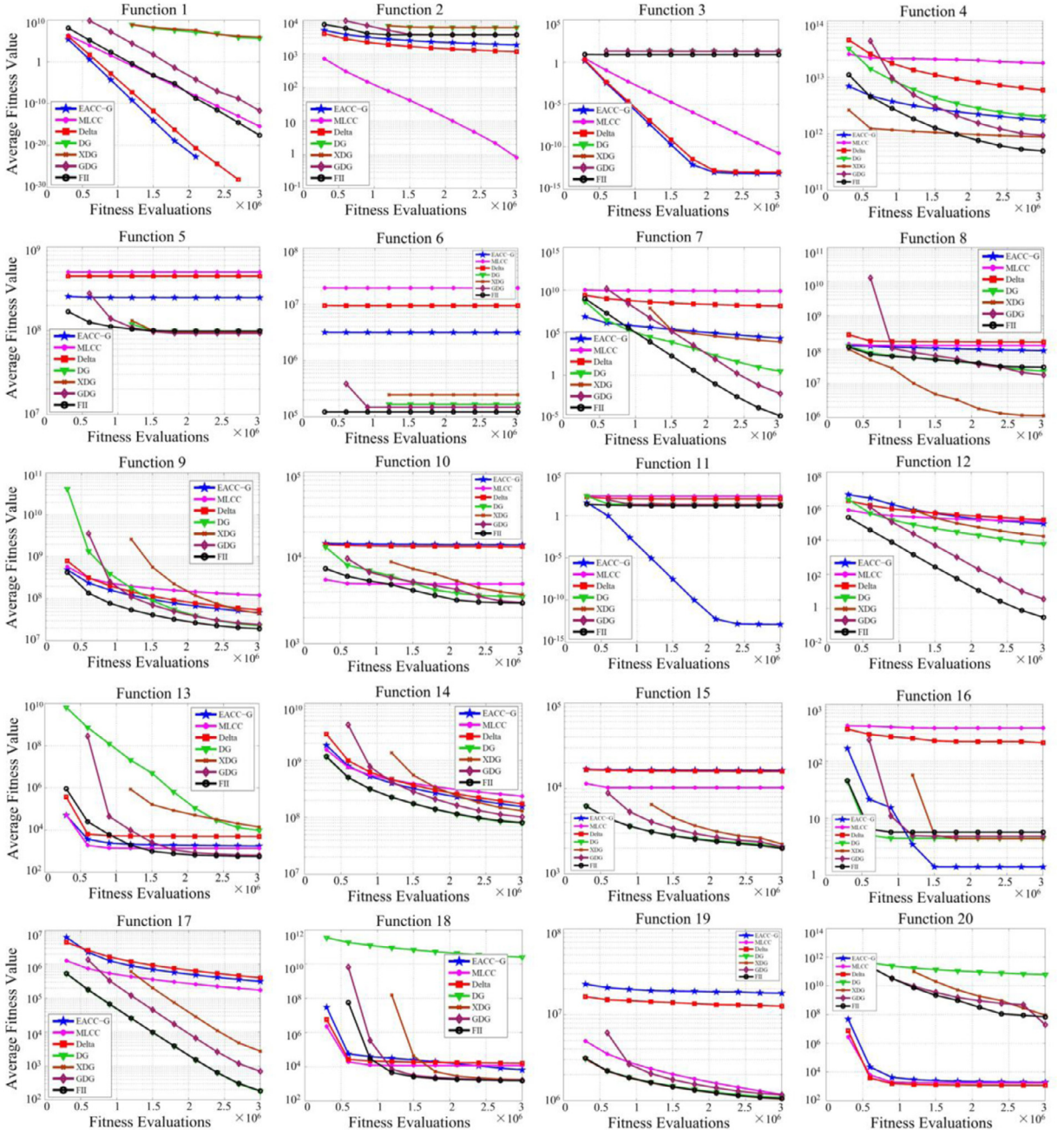


Fig. 6. Convergence curves on CEC'2010 benchmark functions.

4.2.2. Optimization results and analysis

In the optimization experiments, FII, DG, XDG and GDG respectively consume a certain number of FEs in the identification, so for them the number of FEs that is leaved for optimization is the maximal number of FEs minus the one used in the identification. Table 2 gives the statistics of 25 independent runs for each algorithm. The Wilcoxon rank sum test is conducted between FII and each of the six algorithms. FII obtains the best average optimization results on 10 out of 20 functions in the 25 independent runs. According to the Wilcoxon rank sum test, FII performs significantly better than each of the six algorithms respectively on more than half of the benchmark functions as the last two rows in Table 2 shows.

The convergence curves are displayed in Fig. 6. DG, XDG and FII consume a certain number of FEs in the identification, so some of their convergence curves start late evidently. The dynamic decomposition algorithms (EACC-G, MLCC and Delta grouping) have shown their advantages on fully separable problems (f_1-f_3). However, on most nonseparable problems, the

Table 2
Optimization results of FII and 6 algorithms (EACC-G, MLCC, Delta grouping, DG, XDG and GDG) on CEC'2010 benchmark functions.

Function		EACC-G	MLCC	Delta	DG	XDG	GDG	FII
f_1	Mean	**0.00e+00**	3.63e−16	**0.00e+00**	4.01e+05	8.33e+05	2.03e−12	2.72e−18
	Std.	0.00e+00	4.46e−17	0.00e+00	1.23e+06	2.74e+06	9.41e−12	8.44e−18
	p-value	9.73e−11#	**−1.42e−09#**	9.73e−11#	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	–
f_2	Mean	1.84e+03	**8.18e−01**	1.16e+03	6.05e+03	6.10e+03	3.71e+03	3.70e+03
	Std.	4.33e+01	5.97e−02	3.77e+01	1.51e+02	1.25e+02	1.41e+02	1.18e+02
	p-value	1.42e−09#	1.42e−09#	1.42e−09#	**−1.42e−09#**	**−1.42e−09#**	−9.46e−01	–
f_3	Mean	**7.01e−14**	1.71e−11	1.03e−13	1.89e+01	1.90e+01	1.90e+01	7.77e+00
	Std.	3.48e−15	7.50e−13	3.97e−15	1.09e−01	9.80e−02	9.66e−02	5.36e−01
	p-value	6.68e−10#	1.42e−09#	1.03e−09#	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	–
f_4	Mean	1.74e+12	1.79e+13	5.93e+12	2.02e+12	8.83e+11	9.35e+11	**4.96e+11**
	Std.	6.65e+11	1.13e+13	1.90e+12	7.18e+11	3.65e+11	4.29e+11	2.79e+11
	p-value	**−4.13e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−2.03e−09#**	**−1.13e−04#**	**−5.91e−05#**	–
f_5	Mean	2.48e+08	5.03e+08	4.49e+08	9.93e+07	**9.58e+07**	9.34e+07	9.94e+07
	Std.	5.63e+07	1.26e+08	1.17e+08	1.48e+07	1.64e+07	1.72e+07	1.75e+07
	p-value	**−1.41e−09#**	**−1.41e−09#**	**−1.41e−09#**	8.92e−01	6.41e−01	1.68e−01	–
f_6	Mean	3.17e+06	1.96e+07	9.48e+06	1.66e+05	2.47e+05	1.48e+05	**1.22e+05**
	Std.	3.58e+06	2.91e+05	1.01e+07	3.98e+05	5.10e+05	4.09e+05	3.41e+05
	p-value	**−2.87e−08#**	**−1.42e−09#**	−8.16e−01	**−2.20e−06#**	**−1.11e−06#**	−9.46e−01	–
f_7	Mean	2.12e+04	7.19e+09	1.28e+08	2.91e+00	8.11e+03	7.04e-03	**1.71e−05**
	Std.	6.80e+03	2.73e+09	5.02e+07	2.56e+00	4.55e+03	5.82e−03	1.56e−05
	p-value	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	–
f_8	Mean	9.26e+07	1.31e+08	1.65e+08	2.29e+07	**1.14e+06**	1.78e+07	3.03e+07
	Std.	4.37e+07	1.91e+08	1.88e+08	2.78e+07	1.83e+06	1.83e+07	2.96e+07
	p-value	**−3.88e−06#**	**−4.45e−04#**	**−5.85e−09#**	3.72e−01	1.16e−05#	2.44e−01	–
f_9	Mean	4.65e+07	1.21e+08	5.40e+07	2.26e+07	4.46e+07	2.42e+07	**1.93e+07**
	Std.	2.88e+06	1.40e+07	3.57e+06	3.08e+06	2.27e+07	2.97e+06	2.54e+06
	p-value	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−5.14e−04#**	**−1.42e−09#**	**−2.00e−06#**	–
f_{10}	Mean	1.41e+04	4.96e+03	1.35e+04	3.52e+03	3.73e+03	3.02e+03	**2.98e+03**
	Std.	2.92e+02	4.20e+02	2.98e+02	1.53e+02	2.58e+02	1.08e+02	1.63e+02
	p-value	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−5.21e−09#**	**−2.29e−09#**	−1.25e−01	–
f_{11}	Mean	**1.30e−13**	1.98e+02	9.30e+01	1.95e+01	1.89e+01	1.86e+01	1.38e+01
	Std.	5.40e−15	3.15e−01	1.16e+02	1.52e+00	1.59e+00	1.64e+00	1.66e+00
	p-value	1.13e−09#	**−1.42e−09#**	−2.29e−01	**−2.57e−09#**	**−2.29e−09#**	**−5.21e−09#**	–
f_{12}	Mean	9.17e+04	1.27e+05	1.56e+05	5.99e+03	1.72e+04	3.51e+00	**2.95e-01**
	Std.	1.70e+04	1.68e+04	1.49e+04	2.62e+03	5.17e+03	8.40e−01	9.45e−02
	p-value	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	–
f_{13}	Mean	1.74e+03	1.36e+03	4.98e+03	9.56e+03	1.40e+04	6.35e+02	**5.55e+02**
	Std.	8.56e+02	1.30e+02	3.26e+03	4.85e+03	5.42e+03	1.06e+02	1.15e+02
	p-value	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.53e−02#**	–
f_{14}	Mean	1.54e+08	2.35e+08	1.73e+08	**7.90e+07**	1.30e+08	1.02e+08	8.07e+07
	Std.	5.83e+06	1.99e+07	1.40e+07	7.05e+06	9.03e+06	7.31e+06	5.94e+06
	p-value	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	3.52e−01	**−1.42e−09#**	**−3.26e−09#**	–
f_{15}	Mean	1.64e+04	1.03e+04	1.59e+04	2.05e+03	2.24e+03	2.09e+03	**2.00e+03**
	Std.	2.60e+02	1.03e+03	4.24e+02	7.47e+01	7.96e+01	9.55e+01	7.04e+01
	p-value	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.23e−02#**	**−3.26e−09#**	**−2.47e−03#**	–
f_{16}	Mean	**1.39e+00**	3.81e+02	2.12e+02	4.43e+00	4.35e+00	4.81e+00	5.66e+00
	Std.	1.12e+00	6.70e+01	2.13e+02	1.44e+00	1.89e+00	2.20e+00	2.67e+00
	p-value	4.07e−07#	**−1.42e−09#**	−6.41e−01	3.44e−02#	4.16e−02#	1.51e−01	–
f_{17}	Mean	3.11e+05	1.72e+05	4.00e+05	**1.77e+02**	2.73e+03	6.90e+02	1.84e+02
	Std.	2.18e+04	1.22e+04	1.97e+04	2.42e+01	2.78e+02	7.80e+01	2.22e+01
	p-value	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	2.29e−01	**−1.42e−09#**	**−1.42e−09#**	–
f_{18}	Mean	6.50e+03	1.17e+04	1.59e+04	2.54e+10	1.76e+03	1.58e+03	**1.52e+03**
	Std.	3.46e+03	7.96e+03	5.38e+03	5.19e+09	2.04e+02	1.69e+02	1.25e+02
	p-value	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−3.90e−05#**	−2.07e−01	–
f_{19}	Mean	1.78e+07	1.18e+06	1.26e+07	1.08e+06	**1.05e+06**	1.17e+06	**1.05e+06**
	Std.	1.25e+06	5.68e+04	1.17e+06	7.17e+04	4.48e+04	5.58e+04	5.22e+04
	p-value	**−1.42e−09#**	**−9.51e−08#**	**−1.42e−09#**	−7.43e−02	5.22e−01	**−1.45e−07#**	–

(*continued on next page*)

Table 2 (*continued*)

Function		EACC-G	MLCC	Delta	DG	XDG	GDG	FII
f_{20}	Mean	1.93e+03	1.78e+03	**1.19e+03**	6.08e+10	9.03e+07	1.97e+07	6.96e+07
	Std.	1.69e+02	1.69e+02	8.60e+01	8.23e+09	2.88e+08	6.26e+07	3.25e+08
	p-value	1.42e−09#	1.42e−09#	1.42e−09#	**−1.42e−09#**	−9.62e−05#	5.48e−01	–
Signt_Better		14	17	13	14	15	12	–
Signt_Worse		6	3	4	1	3	0	–

1. The *p*-value is obtained from Wilcoxon rank sum test between FII and the algorithm in the corresponding column.
2. A *p*-value prefixed by "−" indicates that FII obtained better optimization result than algorithm in the corresponding column.
3. A *p*-value appended by "#" indicates significant difference between the two samples at level $\alpha = 0.05$.
4. Signt_Better: The number of functions on which FII performed significantly better than the algorithm in the corresponding column (*p*-value is bolded).
5. Signt_Worse: The number of functions on which FII performed significantly worse than the algorithm in the corresponding column.
6. For a function, the best mean value obtained in the experiment is bolded.

convergence curves show that FII performs better than EACC-G, MLCC and Delta grouping. FII performs obviously the best on f_6, f_7, f_9, f_{10} and f_{12}. The good optimization performance is associated with its identification performance as Table 1 shows. The interdependency information of these functions is identified correctly and most computational resources are used in the optimization due to our efficient identification technique. DG uses much fewer FEs on f_{13} but performs poorly in identifying the interdependency information, which leads to worse optimization results. DG and FII perform similarly on f_{14}, f_{15}, f_{17} because both of them get the correct identification results and consume comparative FEs as Table 1 shows. However, the ignorance of indirect nonseparability in DG leads to worse optimization performance on some Rosenbrock functions (f_{13}, f_{18}, f_{20}). As Table 1 shows, DG can not correctly identify the interdependency information on these functions with indirect nonseparability, so it obtains obviously poor optimization results as Fig. 6 shows. XDG and GDG can correctly identify the interdependency information for most functions. However, their drawback of low efficiency is shown in the convergence curves. XDG and GDG require a lot of computational resources in the identification, so in the experiment they converge slower than FII. In summary, FII has shown its competitive performance in LSGO.

4.3. Scalability studies

To verify the scalability of FII, we designed a suite of 2000-dimensional functions, which is referred as CEC'2010-Extended benchmark functions. This suite includes 12 functions as follows.

Category 1: Single-group m-nonseparable functions (f'_7-f'_8);
Category 2: $\frac{D}{2m}$−group m-nonseparable functions (f'_9-f'_{13});
Category 3: $\frac{D}{m}$−group m-nonseparable functions (f'_{14}-f'_{18});

In this suite, $D = 2000$, $m = 100$. Each function in this suite has at least one nonseparable subcomponent, which is very common in the real world.

4.3.1. Identification results and analysis

Table 3 shows the identification results of FII, DG, XDG and GDG on the CEC'2010-Extended benchmark functions. FII achieves 100% identification accuracy on 9 out of 12 functions and the exceptions are f_8, f_{11} and f_{16}. DG, XDG and GDG respectively obtain correct interdependency information on 6, 10 and 11 functions. In terms of the identification results, XDG and GDG perform a bit better than FII. As expected, DG does not perform well on Rosenbrock functions (f'_8, f'_{13} and f'_{18}) in this suite. The number of FEs required by each algorithm is also shown in Table 3. DG obtains correct identification results on f'_9, f'_{10} and f'_{12} at the cost of more than 1 million FEs and it can fast identify the interdependency information on f'_{14}, f'_{15} and f'_{17} also due to the ignorance of indirect nonseparability. XDG obtains the correct identification results at the cost of consuming around 4 million FEs and GDG requires a fixed number of FEs (2003011) for all functions in this suite. In contrast, FII requires at most a few tens or thousands of FEs on most functions. The exceptions are two Rosenbrock functions f'_{13} and f'_{18} in which the interaction structures are chain-like. In summary, FII has shown its good scalability in interdependency identification for these 2000-dimensional problems.

4.3.2. Optimization results and analysis

For the CEC'2010-Extended benchmark functions, the optimization results are shown in Table 4. FII obtains the best average optimization results on 8 out of 12 functions f'_9-f'_{15} and f'_{18}, which is associated with its identification performance. FII correctly identifies the interdependency information for most these functions as Table 3 shows (the exception is only f'_{11}). As the last two rows in Table 4 shows, FII performs significantly better than each of the six other algorithms respectively on more than half of benchmark functions. Compared with EACC-G, MLCC and Delta grouping, FII has shown its advantages on making near-optimal decomposition on nonseparable problems. Compared with DG, XDG and GDG, FII saves a considerable number of FEs for the subsequent optimization. So FII can perform better than DG, XDG and GDG.

Table 3
Interdependency identification results on CEC'2010-Extended benchmark functions.

Func	SVars	NVars	Nonsep Subcmpt	FII/DG/XDG/GDG				
				SVars Identified	NVars Identified	Formed Nonsep Subcmpt	Identification Accuracy (%)	Fitness Evaluations
7	1900	100	1	1900/ 595/ 1/ 1900	100/ 1405/ 1999/ 100	1/ 11/ 2/ 1	100/ –/ –/ 100	6101/ 396078/ 497242/ 2003011
8	1900	100	1	1761/ 220/ 0/ 1900	239/ 1780/ 2000/ 100	2/ 6/ 2/ 1	–/ –/ –/ 100	6520/ 56812/ 427370/ 2003011
9	1000	1000	10	1000/1000/1000/1000	1000/1000/1000/1000	10/10/10/10	100/ 100/ 100/ 100	16010/1043088/ 3904980/ 2003011
10	1000	1000	10	1000/1000/1000/1000	1000/1000/1000/1000	10/10/10/ 10	100/ 100/ 100/ 100	16010/1053384/ 3904980/ 2003011
11	1000	1000	10	1072/ 1005/ 1000/ 0	928/ 995/ 1000/ 2000	10/10/10/10	–/ –/ 100/ –	19298/ 1063946/ 3908916/ 2003011
12	1000	1000	10	1000/1000/1000/1000	1000/1000/1000/1000	10/10/10/10	100/ 100/ 100/ 100	16010/1047840/ 3904980/ 2003011
13	1000	1000	10	1000/ 252/ 1000/ 1000	1000/1748/1000/1000	10/49/10/10	100/ –/ 100/ 100	396309/137182/ 4000188/ 2003011
14	0	2000	20	0/ 0/ 0/ 0	2000/2000/2000/2000	20/20/20/20	100/ 100/100/ 100	46020/ 42000/ 3807960/ 2003011
15	0	2000	20	0/ 0/ 0/ 0	2000/2000/2000/2000	20/20/20/20	100/ 100/ 100/ 100	46020/ 42000/ 3807966/ 2003011
16	0	2000	20	122/ 8/ 0/ 0	1878/1992/2000/2000	20/23/20/20	–/ –/ 100/ 100	63402/ 42428/ 3815822/ 2003011
17	0	2000	20	0/ 0/ 0/ 0	2000/2000/2000/2000	20/20/20/20	100/ 100/ 100/ 100	46020/ 42000/ 3807960/ 2003011
18	0	2000	20	0/ 161/ 0/ 0	2000/1839/2000/2000	20/77/20/20	100/ –/ 100/ 100	1408338/112714/3998380/2003011

1. SVars: the number of separable variables.
2. NVars: the number of nonseparable variables;
3. Nonsep Subcmpt: the number of nonseparable subcomponents.
4. The results of FII, DG, XDG and GDG are separated by "/".
5. Identification Accuracy is 100% if all decision variables are correctly identified and partitioned, otherwise it is marked by "–".

Table 4
Optimization results of FII and 6 algorithms (EACC-G, MLCC, Delta grouping, DG, XDG and GDG) on CEC'2010-Extended benchmark functions.

Function		EACC-G	MLCC	Delta	DG	XDG	GDG	FII
f_7	Mean	1.17e+06	4.01e+10	1.48e+09	1.45e+08	**1.48e+05**	3.28e+08	1.68e+07
	Std.	1.91e+05	1.10e+10	4.28e+08	1.55e+08	2.47e+04	1.55e+08	9.12e+06
	p-value	1.42e−09#	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**1.42e−09#**	**−1.42e−09#**	–
f_8	Mean	1.60e+08	1.20e+08	3.55e+08	2.12e+07	**2.00e+07**	1.66e+08	1.56e+08
	Std.	2.81e+07	6.54e+07	7.73e+08	3.64e+07	2.99e+07	4.83e+07	1.23e+08
	p-value	−1.21e−01	4.26e−01	−5.00e−02	8.29e−07#	3.02e−07#	−8.08e−02	–
f_9	Mean	1.03e+08	2.35e+08	1.20e+08	2.15e+08	1.16e+09	4.72e+07	**2.98e+07**
	Std.	5.19e+06	2.32e+07	8.94e+06	1.51e+08	3.26e+08	3.67e+06	1.69e+06
	p-value	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	–
f_{10}	Mean	3.17e+04	1.17e+04	2.52e+04	9.18e+03	9.92e+03	6.62e+03	**6.53e+03**
	Std.	5.36e+02	9.91e+02	8.54e+03	2.44e+02	2.39e+02	1.33e+02	1.70e+02
	p-value	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−5.21e−09#**	**−2.29e−09#**	−1.25e−01	–
f_{11}	Mean	1.23e+02	1.98e+02	2.34e+02	4.36e+01	4.30e+01	4.30e+01	**3.12e+01**
	Std.	1.18e+02	1.59e−01	2.91e+00	2.26e+00	1.51e+00	1.24e+00	2.03e+00
	p-value	−8.16e−01	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	–
f_{12}	Mean	1.21e+07	5.44e+05	9.58e+05	8.60e+04	3.72e+05	2.25e+04	**4.28e+03**
	Std.	5.50e+05	3.18e+04	6.64e+04	1.63e+04	2.12e+04	3.40e+03	9.48e+02
	p-value	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	–
f_{13}	Mean	4.61e+03	3.70e+03	9.04e+03	1.52e+06	3.14e+05	2.54e+03	**1.97e+03**
	Std.	2.24e+03	3.47e+03	3.53e+03	9.95e+05	3.21e+04	5.15e+02	2.10e+02
	p-value	**−2.57e−09#**	**−4.61e−05#**	**−1.46e−08#**	**−1.42e−09#**	**−1.42e−09#**	**−1.36e−06#**	–
f_{14}	Mean	2.85e+08	4.17e+08	3.12e+08	8.87e+07	2.87e+08	1.39e+08	**8.77e+07**
	Std.	1.32e+07	3.37e+07	1.20e+07	3.49e+06	1.10e+07	8.08e+06	3.89e+06
	p-value	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	−2.77e−01	**−1.42e−09#**	**−1.42e−09#**	–
f_{15}	Mean	3.58e+04	2.50e+04	3.25e+04	**5.66e+03**	7.11e+03	5.87e+03	**5.66e+03**
	Std.	6.91e+02	1.87e+03	5.42e+03	1.36e+02	3.58e+02	2.10e+02	1.40e+02
	p-value	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	−7.12e−01	**−1.42e−09#**	**−1.13e−04#**	–
f_{16}	Mean	3.74e+02	3.97e+02	4.27e+02	4.63e+01	**4.62e+01**	**4.62e+01**	5.47e+01
	Std.	1.40e+02	2.02e−01	2.00e+00	1.81e+00	2.75e+00	2.67e+00	6.37e+00
	p-value	**−4.26e−06#**	**−1.42e−09#**	**−1.42e−09#**	1.01e−06#	1.65e−06#	1.36e−06#	–
f_{17}	Mean	1.77e+06	8.90e+05	1.97e+06	**2.94e+04**	4.18e+05	1.11e+05	2.95e+04
	Std.	2.36e+05	4.59e+04	1.13e+05	2.59e+03	1.27e+04	7.47e+03	2.56e+03
	p-value	**−1.42e−09#**	**−1.42e−09#**	**−1.42e−09#**	6.28e−01	**−1.42e−09#**	**−1.42e−09#**	–
f_{18}	Mean	7.52e+03	7.39e+03	9.65e+03	1.65e+11	1.68e+04	5.72e+03	**4.96e+03**
	Std.	3.08e+03	3.00e+03	3.98e+03	1.84e+10	5.97e+03	4.90e+02	3.94e+02
	p-value	**−8.46e−04#**	−5.72e−02	**−3.29e−05#**	**−1.42e−09#**	**−1.42e−09#**	**−3.88e−06#**	
Signt_Better		9	10	11	7	10	9	–
Signt_Worse		2	2	1	3	0	2	–

1. The *p*-value is obtained from Wilcoxon rank sum test between FII and the algorithm in the corresponding column.
2. A *p*-value prefixed by "−" indicates that FII obtained better optimization result than algorithm in the corresponding column.
3. A *p*-value appended by "#" indicates significant difference between the two samples at level $\alpha = 0.05$.
4. Signt_Better: The number of functions on which FII performed significantly better than the algorithm in the corresponding column (*p*-value is bolded).
5. Signt_Worse: The number of functions on which FII performed significantly worse than the algorithm in the corresponding column.
6. For a function, the best mean value obtained in the experiment is bolded.

The convergence curves are displayed in Fig. 7. It can be observed that FII evidently converges faster than other algorithms on f_9, f_{10}, f_{12}, f_{14}, f_{15} and f_{17}. Although DG, XDG and GDG obtain the same identification results as that of FII on these functions, their drawback of inefficiency leads to slow convergence on most of these functions. DG performs as well as FII on f_{14}, f_{15}andf_{17}. Without covering indirect nonseparability, DG can fast identify the interdependency information on this kind of functions as Table 3 shows. However, it is incapable of making near-optimal decomposition on f_{13}andf_{18} with indirect nonseparability, so it converges much worse on these two functions as Fig. 7 shows. In summary, with good scalability, FII has also shown its strong competitiveness on the 2000-dimensional benchmark functions.

5. Conclusion

In this paper, a fast interdependency identification algorithm (FII) is proposed for CC. FII has two identification stages. In the first stage, FII identifies separable and nonseparable variables with linear time complexity with respect to FEs. This identification stage is the highlight of the algorithm because the number of variables which will be taken to identify nonseparabilty is reduced. Hence FII saves a large number of FEs compared with the other state-of-the-art algorithms, especially

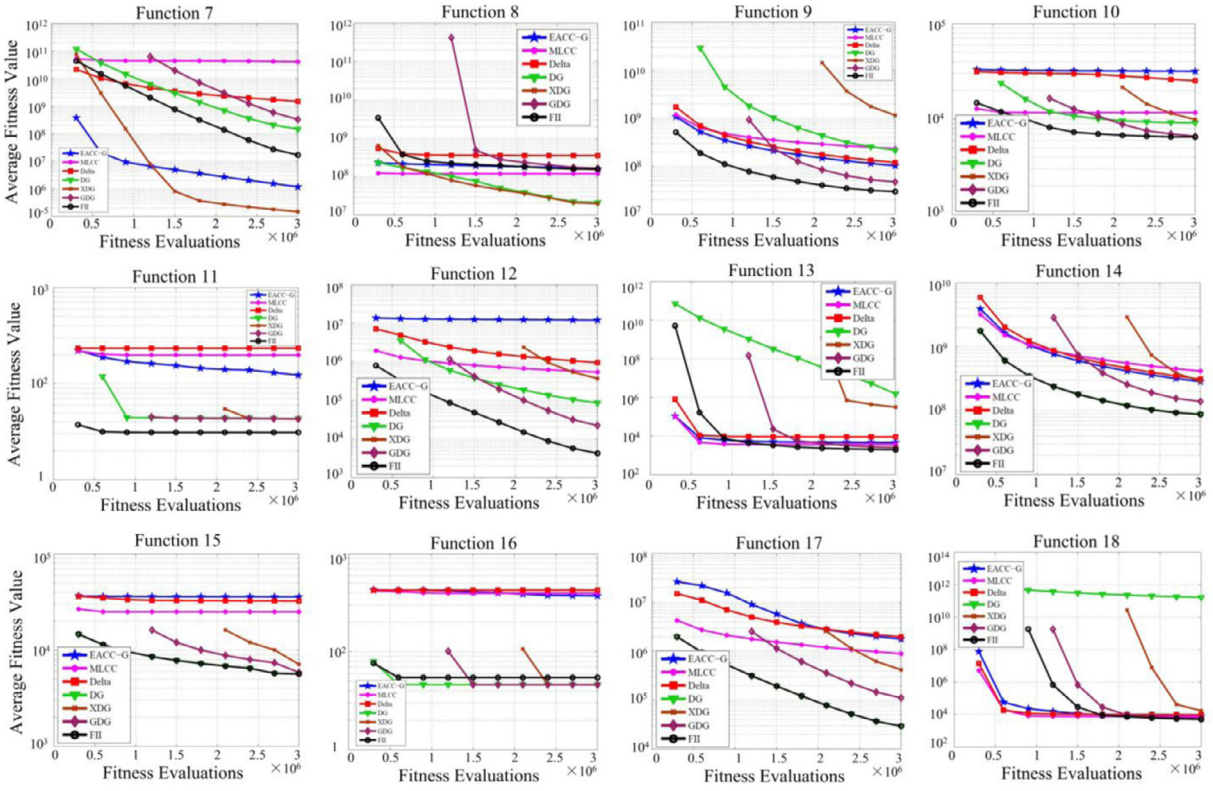


Fig. 7. Convergence curves on CEC'2010-extended benchmark functions.

in the case that the majority of the variables are separable. If nonseparable variables are identified, the interdependency information is identified further in the second stage of FII. The variables that have direct or indirect nonseparability with other variables are grouped in a subcomponent. This stage also avoids exploring the specific interaction information between a remaining variable and each of the perturbed variables, and thus saves extra FEs. Therefore, to make near-optimal decomposition for CC, FII avoids unnecessarily obtaining the full interdependency information of nonseparable variables. The speed of FII in identifying interdependency of variables is very significant.

On the other hand, the correctness of FII on the interdependency identification is also competitive. The experimental studies are firstly conducted on the CEC'2010 1000-dimensional benchmark functions. It is shown that FII obtains correct interdependency information on most test functions with much fewer FEs compared with three decomposition strategies. In LSGO experiments, the optimization results have reflected encouraging advantages of combining FII with CC architecture in comparison with six state-of-the art algorithms. To verify the scalability of FII, a suite of 2000-dimensional benchmark functions is designed based on the CEC'2010 benchmark functions. FII has also shown promising performances in the identifications and optimizations on the 2000-dimensional functions.

Note that FII decomposes a problem absolutely according to the interdependency information. It can be observed that the identification accuracy of FII is slightly lower than that of XDG and GDG for solving the benchmark functions. A more effective identification criterion is needed in future research. Moreover, if great imbalance exists among the subcomponents, the performance of CC in the round-robin fashion may be affected because it treats all subcomponents equally in optimization. The method for dispatching computational resources may play a key role in the CC-based optimization and it will be focused in the future.

Acknowledgement

This work was supported in part by NSFC projects Nos. 61202130, 61379061, 61332002, 61622206, in part by Natural Science Foundation of Guangdong for Distinguished Young Scholars No. 2015A030306024, in part by "Guangdong Special Support Program" No. 2014TQ01X550, and in part by Guangzhou Pearl River New Star of Science and Technology Project No. 201506010002, and in part by Guangdong Soft Science Research Program No. 2012B070400008.

References

[1] F.V. Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput 8 (3) (2004) 225–239.

[2] W. Chen, T. Weise, Z. Yang, K. Tang, Large-scale global optimization using cooperative coevolution with variable interaction learning, in: Parallel Problem Solving from Nature, PPSN XI, Springer, 2010, pp. 300–309.
[3] W. Ding, Z. Guan, Q. Shi, J. Wang, A more efficient attribute self-adaptive co-evolutionary reduction algorithm by combining quantum elitist frogs and cloud model operators, Inf. Sci. 293 (2015) 214–234.
[4] N. García-Pedrajas, C. Hervás-Martínez, J. Muñoz-Pérez, COVNET: a cooperative coevolutionary model for evolving artificial neural networks, IEEE Trans. Neural Netw 14 (3) (2003) 575–596.
[5] H. Ge, L. Sun, X. Yang, S. Yoshida, Y. Liang, Cooperative differential evolution with fast variable interdependence learning and cross-cluster mutation, Appl. Soft Comput. J. 36 (2015) 300–314.
[6] T. Jansen, R. Wiegand, The cooperative coevolutionary (1+1) EA, Evol. Comput. 12 (4) (2004) 405–434.
[7] B. Jiang, N. Wang, Cooperative bare-bone particle swarm optimization for data clustering, Soft Comput 18 (6) (2014) 1079–1091.
[8] K. de Jong, M.A. Potter, Evolving, complex structures via cooperative coevolution, Evol. Prog (1995) 307–317.
[9] S. Kauffman, S. Johnsen, Coevolution to the edge of chaos: Coupled fitness landscapes, poised states, and coevolutionary avalanches, J. Theor. Biol. 149 (1991) 467–505.
[10] A. LaTorre, S. Muelas, J. Peña, A comprehensive comparison of large scale global optimizers, Inf. Sci. 316 (2015) 517–549.
[11] W. Lee, Y. Hsiao, Inferring gene regulatory networks using a hybrid GA–PSO approach with numerical constraints and network decomposition, Inf. Sci. 188 (2012) 80–99.
[12] X. Li, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Trans. Evol. Comput 16 (2) (2012) 210–224.
[13] J. Liu, K. Tang, Scaling up covariance matrix adaptation evolution strategy using cooperative coevolution, in: Intelligent Data Engineering and Automated Learning–IDEAL, Springer, 2013, pp. 350–357.
[14] 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 (CEC), Piscataway, NJ, USA, 2001, pp. 1101–1108.
[15] S. Mahdavi, M.E. Shiri, S. Rahnamayan, Metaheuristics in large-scale global continues optimization: a survey, Inf. Sci. 295 (2015) 407–428.
[16] Y. Mei, M. Omidvar, X. Li, X. Yao, A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization, ACM Trans. Math. Softw. (2015) in press.
[17] M. Munetomo, D.E. Goldberg, Linkage identification by non-monotonicity detection for overlapping functions, Evol. Comput. 7 (4) (1999) 377–398.
[18] M. Munetomo, D.E. Goldberg, A genetic algorithm using linkage identification by nonlinearity check, in: Proceedings of the 1999 IEEE Conference on System, Man, and Cybernetics, 1999, pp. 595–600.
[19] M. Olhofer, Y. Jin, B. Sendhoff, Adaptive encoding for aerodynamic shape optimization using evolution strategies, in: Proc. of IEEE Congress on Evolutionary Computation, IEEE, 2001, pp. 576–583.
[20] M.N. Omidvar, X. Li, Y. Mei, X. Yao, Cooperative co-evolution with differential grouping for large scale optimization, IEEE Trans. Evol. Comput 18 (3) (2014) 378–393.
[21] M.N. Omidvar, Y. Mei, X. Li, Effective decomposition of large-scale separable continuous functions for cooperative coevolutionary algorithms, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2014, pp. 1305–1312.
[22] M.N. Omidvar, X. Li, Z. Yang, X. Yao, Cooperative co-evolution for large scale optimization through more frequent random grouping, in: 2010 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2010, pp. 1–8.
[23] 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.
[24] M.N. Omidvar, X. Li, X. Yao, Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms, in: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, ACM, 2011, pp. 1115–1122.
[25] M.N. Omidvar, X. Li, K. Tang, Designing benchmark problems for large-scale continuous optimization, Inf. Sci. 316 (2015) 419–436.
[26] L. Panait, Theoretical convergence guarantees for cooperative coevolutionary algorithms, Evol. Comput. 18 (4) (2010) 581–615.
[27] M. Potter, K. De Jong, A cooperative coevolutionary approach to function optimization, in: Proceedings of the Third Conference on Parallel Problem Solving from Nature, 2, Jerusalem, Israel, 1994, pp. 249–257.
[28] M. Potter, K. De Jong, Cooperative coevolution: an architecture for evolving coadapted subcomponents, Evol. Comput. 8 (1) (2000) 1–29.
[29] T. Ray, X. Yao, A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning, in: Proceedings of the 2009 Congress on Evolutionary Computation (CEC), Trondheim, Norway, 2009, pp. 983–989.
[30] Y. Ren, Y. Wu, An efficient algorithm for high-dimensional function optimization, Soft Comput 17 (6) (2013) 995–1004.
[31] E. Sayed, D. Essam, R. Sarker, Dependency identification technique for large scale optimization problems, in: 2012 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2012, pp. 1–8.
[32] R. Shang, Y. Wang, J. Wang, L. Jiao, S. Wang, L. Qi, A multi-population cooperative coevolutionary algorithm for multi-objective capacitated arc routing problem, Inf. Sci. 277 (2014) 609–642.
[33] Y. Shi, H. Teng, Z. Li, Cooperative co-evolutionary differential evolution for function optimization, in: Proceedings of the First International Conference on Natural Computation, Springer-Verlag, 2005, pp. 1080–1088.
[34] D. Sofge, K. De Jong, A. Schultz, A blended population approach to cooperative coevolution for decomposition of complex problems, in: Proceedings of the 2002 Congress on Evolutionary Computation (CEC), 1, 2002, pp. 413–418.
[35] L. Sun, S. Yoshida, X. Cheng, Y. Liang, A cooperative particle swarm optimizer with statistical variable interdependence learning, Inf. Sci. 186 (2012) 20–39.
[36] Y. Sun, M. Kirley, S.K. Halgamuge, Extended differential grouping for large scale global optimization with direct and indirect variable interactions, in: Proceedings 2015 Genetic and Evolutionary Computational Conference - GECCO '15, 2015, pp. 313–320.
[37] K. Tang, X. Yao, P.N. Suganthan, C. MacNish, Y.P. Chen, C.M. Chen, Z. Yang, Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007 Technical Report.
[38] 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, Nature Inspired Computation and Applications Laboratory (NICAL), USTC, China, 2010 Technical report.
[39] L. Tang, Y. Dong, J. Liu, Differential evolution with an individual-dependent mechanism, IEEE Trans. Evol. Comput 19 (4) (2015) 560–574.
[40] H.F. Teng, Y. Chen, W. Zeng, Y.-J. Shi, Q.-H. Hu, A dual-system variable-grain cooperative coevolutionary algorithm: satellite-module layout design, IEEE Trans. Evol. Comput. 14 (3) (2010) 438–455.
[41] M. Tezuka, M. Munetomo, K. Akama, Linkage identification by nonlinearity check for real-coded genetic algorithms, in: 2004 Genetic and Evolutionary Computation (GECCO), Springer, 2004, pp. 222–233.
[42] Y. Wang, J. Huang, W.S. Dong, J.C. Yan, C.H. Tian, M. Li, W.T. Mo, Two-stage based ensemble optimization framework for large scale global optimization, Eur. J. Oper. Res. 228 (2) (2013) 308–320.
[43] K. Weicker, N. Weicker, On the improvement of coevolutionary optimizers by learning variable interdependencies, in: Proceedings of the 1999 Congress on Evolutionary Computation, Washington, D.C. USA, 1999, pp. 1627–1632.
[44] X. Wen, W.-N. Chen, Y. Lin, T. Gu, H. Zhang, Y. Li, Y. Yin, J. Zhang, A Maximal Clique Based Multiobjective Evolutionary Algorithm for Overlapping Community Detection, IEEE Transactions on Evolutionary Computation, in press, DOI: 10.1109/TEVC.2016.2605501, 2016.
[45] R.P. Wiegand, W.C. Liles, K.A. De Jong, An empirical analysis of collaboration methods in cooperative coevolutionary algorithms, in: Proceedings of Genetic and Evolutionary Computational Conference, 2001, pp. 1235–1242.
[46] M. Xiao, J. Zhang, K. Cai, X. Cao, K. Tang, Cooperative co-evolution with weighted random grouping for large-scale crossing waypoints locating in air route network, in: 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 2011, IEEE, 2011, pp. 215–222.

[47] Q. Yang, W.-N. Chen, T. Gu, H. Zhang, J.D. Deng, Y. Li, J. Zhang, Segment-Based Predominant Learning Swarm Optimizer for Large-Scale Optimization, IEEE Transactions on Cybernetics, in press, DOI:10.1109/TCYB.2016.2616170, 2016.
[48] Z. Yang, K. Tang, X. Yao, Large scale evolutionary optimization using cooperative coevolution, Inf. Sci. 178 (15) (2008) 2985–2999.
[49] Z. Yang, K. Tang, X. Yao, Multilevel cooperative coevolution for large scale optimization, in: Proceedings of the 2008 Congress on Evolutionary Computation, Hong Kong, 2008, pp. 1663–1670.
[50] Z. Yang, K. Tang, X. Yao, Differential evolution for high-dimensional function optimization, in: 2007 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2007, pp. 3523–3530.
[51] B. Zhang, H.-F. Teng, Y.-J. Shi, Layout optimization of satellite module using soft computing techniques, Appl. Soft Comput. 8 (1) (2008) 507–521.
[52] X. Zhang, Y. Tian, Y. Jin, Approximate non-dominated sorting for evolutionary many-objective optimization, Inf. Sci. (2016) in press.
[53] X. Zhao, L. Shen, X. Peng, W. Zhao, Toward SLA-constrained service composition: An approach based on a fuzzy linguistic preference model and an evolutionary algorithm, Inf. Sci. 316 (2015) 370–396.
[54] J. Zhao, V.B. Fernandes, L. Jiao, I. Yevseyeva, A. Maulana, R. Li, T. Bäck, K. Tang, M.T.M. Emmerich, Multiobjective optimization of classifiers by means of 3D convex-hull-based evolutionary algorithms, Inf. Sci. (2016) 367–368 80-104.
[55] Q. Zhu, Q. Lin, Z. Du, Z. Liang, W. Wang, Z. Zhu, J. Chen, P. Huang, Z. Ming, A novel adaptive hybrid crossover operator for multiobjective evolutionary algorithm, Inf. Sci. 345 (2016) 177–198.
[56] F. Zhu, S. Guan, Cooperative co-evolution of GA-based classifiers based on input decomposition, Eng. Appl. of Artif. Intell. 21 (8) (2008) 1360–1369.