A Historical Interdependency based Differential Grouping Algorithm for Large Scale Global Optimization

An Chen
Autocontrol Institute, Xi'an Jiaotong University
Xi'an, Shaanxi, 710049, P.R. China
chenan123@stu.xjtu.edu.cn

Zhigang Ren
Autocontrol Institute, Xi'an Jiaotong University
Xi'an, Shaanxi, 710049, P.R. China
renzg@mail.xjtu.edu.cn

Yang Yang*
Xi'an Jiaotong University Shenzhen Research School
Shenzhen, Guangdong, 518057, P.R. China
yyang@mail.xjtu.edu.cn

Yongsheng Liang, Bei Pang
Autocontrol Institute, Xi'an Jiaotong University
Xi'an, Shaanxi, 710049, P.R. China
liangyongsheng, beibei@stu.xjtu.edu.cn

ABSTRACT

Cooperative co-evolution (CC) is a powerful evolutionary computation framework for solving large scale global optimization (LSGO) problems via the strategy of "divide-and-conquer", but its efficiency highly relies on the decomposition result. Existing decomposition algorithms either cannot obtain correct decomposition results or require a large number of fitness evaluations (FEs). To alleviate these limitations, this paper proposes a new decomposition algorithm named historical interdependency based differential grouping (HIDG). HIDG detects interdependency from the perspective of vectors. By utilizing historical interdependency information, it develops a novel criterion which can directly deduce the interdependencies among some vectors without consuming extra FEs. Coupled with an existing vector-based decomposition framework, HIDG further significantly reduces the total number of FEs for decomposition. Experiments on two sets of LSGO benchmark functions verified the effectiveness and efficiency of HIDG.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; *Search methodologies*

KEYWORDS

Large scale global optimization, cooperative coevolution, decomposition algorithm, historical interdependency.

1 INTRODUCTION

Large-scale global optimization (LSGO) has become an active research field over the last decade due to the higher and higher dimensions of optimization problems involved in scientific research and engineering practice. Owing to the black-box characteristics of LSGO problems, the gradient-free evolutionary algorithms (EAs) are the major approaches for solving them. However, conventional EAs inevitably encounter "the curse of dimensionality" [1], which means that they can hardly get desirable solutions since they cannot adequately explore the solution space of a LSGO problem within acceptable computation time.

To alleviate this limitation, a special algorithmic framework named cooperative coevolution (CC) was proposed [2]. By taking the idea of "divide-and-conquer", CC provides a natural and efficient approach for solving LSGO problems. It first decomposes the original LSGO problem into some lower dimensional sub-problems and then cooperatively optimizes these sub-problems with conventional EAs. As such, the decomposition strategy plays a vital role in CC. Ideally, a given LSGO problem should be decomposed such that the interaction between the resultant sub-problems is minimized. Moreover, the number of solutions evaluated during the decomposition process should be as few as possible since the total number of fitness evaluations (FEs) provided to CC is very limited.

In recent years, great efforts were put on decomposition method and several kinds of decomposition strategies have been developed [3], where the learning-based decomposition strategy is an excellent one. It groups the variables according to the detected interdependency information, thus has the potential to make near-optimal decomposition. As a typical learning-based decomposition algorithm, different grouping (DG) [4] focuses on detecting the additive separability which is the most common type of separability. DG shows superior performance over other decomposition algorithms. In DG, the interaction between two variables is defined as follows [4]:

Definition 1: *Let $f(\vec{x})$ be an additively separable function. For $\forall a, b_1 \neq b_2, \delta \in \mathbb{R}, \delta \neq 0$, if the following condition holds:*

$$\Delta_{\delta, x_p}[f](\vec{x})|_{x_p=a, x_q=b_1} \neq \Delta_{\delta, x_p}[f](\vec{x})|_{x_p=a, x_q=b_2} \quad (1)$$

then x_p and x_q are non-separable, where

$$\Delta_{\delta, x_p}[f](\vec{x}) = f(..,x_p+\delta,..) - f(..,x_p,..) \quad (2)$$

*Corresponding author

GECCO'18, July 15-19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07…$15.00
https://doi.org/10.1145/3205651.3208278

refers to the forward difference of $f(\vec{x})$ *with respect to variable* x_p *with interval* δ.

According to Definition 1, if the change of f caused by a perturbation to x_p varies with different values of x_q, then we say x_p interacts with x_q. Generally, a threshold parameter (ε) is required in practice to handle the computational roundoff errors on floating-point numbers. Concretely, let the left side and right side of (1) be denoted as Δ_1 and Δ_2, respectively, and let $\tau = \Delta_1 - \Delta_2$. Then only if $|\tau| > \varepsilon$, DG adjudges x_p and x_q interact with each other.

Despite its success, DG also shows some drawbacks. To be specific, it is likely to omit indirect interactions and requires too many FEs on some functions. To improve the performance of DG, global DG (GDG) [5] and extended DG (XDG) [6] were developed. These algorithms can detect both direct and indirect interactions, but also require more FEs. Recently, a new version of DG named DG2 [7] tries to reduce the requirement for FEs by systematically generating solution samples, but its effect is relatively limited. Besides, it also adopts a new threshold setting method to improve the final decomposition accuracy.

Instead of directly analyzing the interactions among decision variables, some researchers attempted to perform decomposition operations from the view of decision vectors. The interaction between two vectors can be defined as follows [8]:

Definition 2: *For an additively separable continuous function* $f(\vec{x})$ *and a partition of* $\vec{x}$ ($S = \{S_1, S_2, ..., S_m\}$), *two vectors* S_p *and* S_q ($p, q = 1, 2, ..., m; p \neq q$) *are non-separable if the following formula holds*:

$$\Delta_{\Gamma, S_p}[f](\vec{x})|_{S_p=A, S_q=B_1} \neq \Delta_{\Gamma, S_p}[f](\vec{x})|_{S_p=A, S_q=B_2} \quad (3)$$

where

$$\Delta_{\Gamma, S_p}[f](\vec{x}) = f(..., S_p + \Gamma, ...) - f(..., S_p, ...) \quad (4)$$

refers to the forward difference of $f(\vec{x})$ *with respect to* S_p *with an interval* Γ. Γ, A, B_1, *and* B_2 *can be arbitrary values as long as they ensure the feasibility of* S_p *and* S_q.

The vector-based decomposition idea has been adopted by several algorithms, including fast interdependency identification (FII) [9], two kinds of vector-growth decomposition algorithms (VGDA-S and VGDA-D) [8], and recursive DG (RDG) [10]. Since there is no need to detect the interaction between each pair of variables, the number of FEs required by these three algorithms is reduced to a great extent. However, when decomposing functions with indirect interdependency, such as *Rosenbrock*'s function, all of them still need lots of FEs.

To improve the efficiency of existing DGs, this paper proposes an efficient decomposition algorithm called Historical Interdependency based Differential Grouping (HIDG). HIDG investigates the interdependencies among decision vectors in a more systematical way. By utilizing the historical interdependency information, HIDG develops a novel criterion which can deduce the interactions among some vectors without consuming extra FEs. Concretely, for two interacting vectors whose τ value has been investigated, when we investigated the interaction between S_p and a sub-vector of S_q, the interaction for S_p and the corresponding complementary sub-vector of S_q can be directly deduced based on the criterion. When integrating HIDG into an existing vector-based decomposition framework, two special rules are designed to facilitate the implement of the developed criterion. As a result, HIDG further significantly reduces the total number of FEs for decomposition.

2 HISTORICAL INTERDEPENDENCY BASED DIFFERENTIAL GROUPING

This section first presents and proves the new criterion for detecting interactions among vectors, then describes HIDG in detail.

2.1 New Criterion based on Historical Interdependency

By utilizing historical interdependency information, the following criterion can deduce the interactions among part of vectors without consuming any FEs.

Criterion: *For an additively separable function* $f(\vec{x})$, *suppose the vectors* S_p *and* S_q *interact with each other with* $\tau_{p,q}$ *being their interdependency value,* S_{q1} and S_{q2} *are two complementary sub-vectors of* S_q, *and* $\tau_{p,q1}$ *is the interdependency value detected for* S_p *and* S_{q1} *with the same perturbations on the corresponding variables as those for* $\tau_{p,q}$. *If* $\tau_{p,q} = \tau_{p,q1}$, *then* S_p *and* S_{q2} *are separable*; *otherwise, they are non-separable.*

The criterion can be strictly proved as follows:

Proof: *According to definition2,* $\tau_{p,q}$ *and* $\tau_{p,q1}$ *can be formulated as*

$$\tau_{p,q} = \Delta_{\Gamma, S_p}[f](\vec{x})|_{S_p=A, S_{q1}=B_1, S_{q2}=B_1} - \Delta_{\Gamma, S_p}[f](\vec{x})|_{S_p=A, S_{q1}=B_2, S_{q2}=B_2} \quad (5)$$

$$\tau_{p,q1} = \Delta_{\Gamma, S_p}[f](\vec{x})|_{S_p=A, S_{q1}=B_1, S_{q2}=B_1} - \Delta_{\Gamma, S_p}[f](\vec{x})|_{S_p=A, S_{q1}=B_2, S_{q2}=B_1} \quad (6)$$

Since they are obtained with the same perturbations on the corresponding variables, then

$$\tau_{p,q} - \tau_{p,q1} = \Delta_{\Gamma, S_p}[f](\vec{x})|_{S_p=A, S_{q1}=B_2, S_{q2}=B_1} - \Delta_{\Gamma, S_p}[f](\vec{x})|_{S_p=A, S_{q1}=B_2, S_{q2}=B_2} \quad (7)$$

which can be exactly defined as the interdependency value for S_p *and* S_{q2}, *i.e.,* $\tau_{p,q} - \tau_{p,q1} = \tau_{p,q2}$. *Then we can get the conclusion given above.*

When considering the roundoff errors introduced by floating-point numbers, the criterion should be modified as follows: For two non-separable vectors S_p and S_q with $\tau_{p,q} > \varepsilon$, if $|\tau_{p,q} - \tau_{p,q1}| < \varepsilon$, then S_p and S_{q2} are separable; otherwise, S_p and S_{q2} are non-separable. This means that, for two interacting vectors S_p and S_q whose τ value has been investigated, the interactions between S_p and the two complementary vectors of S_q can be obtained by detecting only a new interdependency, thus half of FEs can be saved.

2.2 Description of HIDG

HIDG is implemented by integrating the criterion given above into VGDA-D [8] which is an efficient vector-based decomposition framework.

VGDA-D concerns an iterative process. At each iteration, it initializes the vectors S_p and S_q with an untreated variable and all the other untreated variables, respectively. If S_p and S_q interact

with each other, it tries to find the variables in S_q interacting with S_p by equally dividing S_q into two complementary sub-vectors and detecting the interaction between S_p and each sub-vector. The division process is repeated unless the current sub-vector does not interact with S_p or contains only one variable. This process can be illustrated by a binary tree shown in Fig.1, where each node denotes a vector and two child nodes denote the complementary sub-vectors of their corresponding parent node.

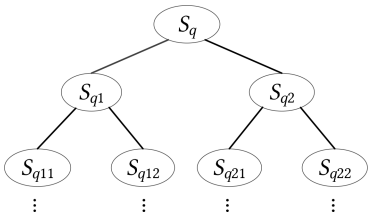


Figure 1: The Binary Tree Generated for S_q.

It seems natural to further accelerate VGDA-D with the new proposed criterion. Unfortunately, the combination of VGDA-D and the criterion is non-trivial. For example, when investigating the interactions between S_p and the variables in S_q described by the binary tree in Fig. 1, if the interdependency values $\tau_{p,q}$ and $\tau_{p,q2}$ have been detected, we can deduce the interaction between S_p and S_{q1} according to $\tau_{p,q1}=\tau_{p,q}-\tau_{p,q2}$. However, it is infeasible to directly deduce the interaction between S_p and S_{q11} by just detecting $\tau_{p,q12}$ and calculating $\tau_{p,q11}$ with $\tau_{p,q11}=\tau_{p,q1}-\tau_{p,q12}$ since the usage of $\tau_{p,q1}$ will lead to the accumulation of calculation errors, which may reduce the final decomposition accuracy to a large extent. HIDG tackles this issue according to the following two rules:

1) When investigating the interactions between S_p and the variables in S_q, HIDG follows the depth-first strategy. Without loss of generality, HIDG preferentially investigates the left nodes in the binary tree generated for S_q.

2) For the two complementary sub-vectors (child nodes) of the current node, HIDG constructs different virtual parent vectors for them instead of employing the current node. Concretely, for the current sub-vector, HIDG takes the union of the sub-vector itself and the untreated sub-vector as its parent vector.

Take the situation shown in Fig. 1 as an example, the sub-vector S_{q11} will be investigated first at the 3rd level according to rule 1) and its parent vector will be set as $\{S_{q11}, S_{q12}, S_{q2}\}$ according to rule 2), then $\{S_{q12}, S_{q2}\}$ will become its complementary sub-vector. This sub-vector is exactly the parent vector of S_{q12} who takes S_{q2} as its complementary sub-vector. We define this sub-vector as the connection vector of S_{q11} and S_{q12}, and denote it as S_v. According to rule 1), HIDG has calculated the interdependency values between S_p and each of $\{S_{q11}, S_{q12}, S_{q2}\}$ and S_{q2} at higher levels, then we can deduce the interaction between S_p and each of S_{q11} and S_{q12} by just calculating the interdependency value between S_p and S_v and applying the new criterion. Other sub-vectors shown in Fig. 1 can be treated in the similar way.

Algorithm 1: *groups* = HIDG ($\vec{x}$, ε)

```
Initialize: groups = ∅; S_q = x⃗ ; H = { };
While |S_q| > 1          //H stores sub-vectors and corresponding τ values
    select ∀x_r ∈ S_q; set S_p = {x_r}, S_q = S_q \ S_p;
    While |S_q| > 0    //There are still untreated variables in S_q
        set H = { }, V = [ ];      //V stores the variables interacted with
        calculate the interdependency value τ between S_p and S_q;
        if |τ| < ε   break;   //S_p does not interact with S_q
        store S_q, ∅ and their corresponding τ values into H;
        equally divide S_q = S_q1 ∪ S_q2; VS = {S_q1, S_q2}; //VS is a stack
        For each two top vectors S_a, S_b in VS
            VS = VS \ {S_a, S_b};
            generate S_v and investigate its τ with S_p;
            find S_v ∪ S_a and S_v \ S_b in H and set their τ values as τ_1, τ_2;
            if | τ_2 − τ | > ε
                if |S_b| = 1;
                    V = V ∪ S_b
                else
                    divide S_b = S_b1 ∪ S_b2; VS = VS ∪ {S_b1, S_b2};
            if | τ_1 − τ | > ε
                if |S_a| = 1;
                    V = V ∪ S_a;
                else
                    divide S_a = S_a1 ∪ S_a2; VS = VS ∪ {S_a1, S_a2};
            store S_v and τ into H;
        S_p = S_p ∪ V; S_q = S_q \ S_p;
    groups = groups ∪ S_p;
groups = groups ∪ S_p;
```

The whole procedure of HIDG is show in Algorithm 1. Here, the interdependency value of the empty vector ∅ is set as 0. Based on Definition 2, when a vector interacts with S_p, the absolute value of its interdependency value with S_p is less than threshold ε, so when we set the interdependency value of ∅ with S_p as 0, the developed criterion can be also used to detect the vectors whose parent vectors are themselves and the complementary sub-vectors are ∅.

As for the setting of the threshold ε, HIDG adopts a simplified method of the one developed by DG2 [7]. Concretely, it first calculates the greatest lower bound and the least upper bound of the current roundoff error and then chooses the mid-value of these two bounds as the final threshold.

3 EXPERIMENTS

We briefly evaluated the performance of HIDG by experimentally comparing it with several state-of-the-art decomposition algorithms, including DG, DG2, and VGDA-D. The experiments were conducted on the functions of 1000 dimensions in CEC2010 and CEC2013 benchmark suites [11, 12], which contain 20 and 15 functions, respectively. Note that the results of the three existing algorithms reported below are obtained from corresponding original papers.

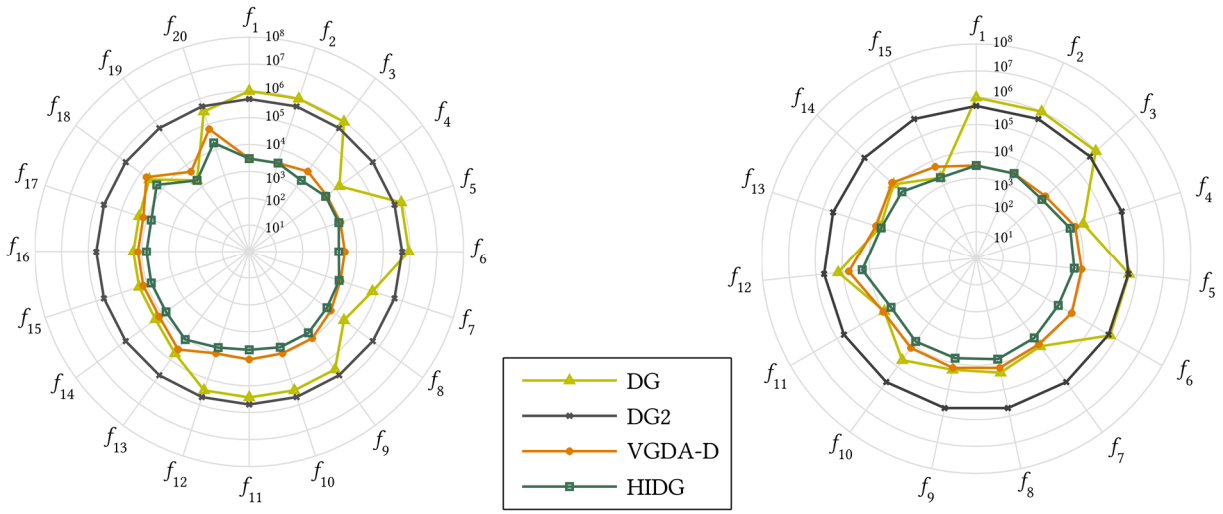
Table 1: Functions Improperly Decomposed by the Four Algorithms

Algorithm	CEC2010 suite	CEC2013 suite
DG	f_4, f_7, f_8, f_{11}, f_{13}, f_{16}, f_{18}, f_{20} (8)	f_4-f_8, f_{11}-f_{15} (10)
DG2	f_3, f_6, f_{11} (3)	f_3, f_6, f_8 (3)
VGDA-D	f_3, f_{11} (2)	f_3, f_5, f_6, f_8, f_{10}, f_{11} (6)
HIDG	f_3, f_6, f_{11} (3)	f_3, f_6, f_8, f_{11} (4)

Table 1 lists the functions which are improperly decomposed by DG, DG2, VGDA-D, and HIDG. It can be observed that HIDG achieves similar decomposition accuracy with DG2 and VGDA-D, since they only make mistakes on 7, 6 and 8 out of total 35 functions, respectively. Compared with DG, HIDG demonstrates much greater performance since DG improperly decomposes about a half of functions.

As for the decomposition cost, Fig.2(a) and Fig.2(b) provide the number of FEs consumed by the four algorithms on the functions in CEC2010 and CEC2013 benchmark suites, respectively. From the two charts, it can be observed that HIDG is an extremely efficient decomposition algorithm. Among the four algorithms, it consumes the fewest FEs on all of the 35 functions. It can decompose most of the test functions within 10,000 FEs. Even on the *Rosenbrock*'s function, which is f_{20} in CEC2010 suite and f_{12} in CEC2013 suite, the number of FEs consumed by HIDG is just slightly larger than 10,000, while the best one (VGDA-D) of the other three algorithms requires nearly 100,000 FEs.

In summary, HIDG can achieve similar decomposition accuracy with the state-of-the-art algorithms by consuming much fewer FEs. This is of great significance for CC in facilitating it concentrating a limited number of FEs on the optimization process.



(a) CEC2010 (b) CEC2013

Figure 2: The Radar Chart of the FEs Consumed by DG, DG2, VGDA-D, and HIDG on the Functions in CEC2010 and CEC2013 Benchmark Suites.

4 CONCLUSIONS

This paper presents an efficient decomposition algorithm named HIDG for LSGO problems. By utilizing the historical interdependency information, HIDG develops a novel criterion to detect the interactions among some vectors without consuming extra FEs. Then HIDG is implemented by integrating the criterion into a vector-based decomposition framework via two well-designed rules. As a result, HIDG can accurately decompose LSGO problems with much fewer FEs. The superiority of HIDG was verified through the comparison with several state-of-the-art decomposition methods on benchmark functions.

Our future work will focus on integrating HIDG into a complete CC framework and developing an excellent algorithm for solving LSGO problems.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 61105126, in part by the Postdoctoral Science Foundation of China under Grants 2014M560784 and 2016T90922, in part by the Postdoctoral Science Foundation of Shaanxi Province, and in part by the project of Shenzhen Technology Plan (No. JCYJ20170816100724089).

REFERENCES

[1] R. E. Bellman. 1957. *Dynamic Programming*. ser. Dover Books on Mathematics. Princeton University Press.
[2] M. A. Potter and K. A. De Jong. 1994. A cooperative coevolutionary approach to function optimization. In *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, 2, 249–257.
[3] S. Mahdavi, M. E. Shiri and S. Rahnamayan. 2015. Metaheuristics in large-scale global continues optimization: a survey. Information Sciences, 295, 407-428.
[4] M. N. Omidvar, X. Li, Y. Mei, and X. Yao. 2014. Cooperative co-evolution with differential grouping for large scale optimization. IEEE Transactions on Evolutionary Computation, 18(3), 378-393.
[5] Y. Mei, M. N. Omidvar, X. Li, and X. Yao. 2016. A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. Acm Transactions on Mathematical Software, 42(2), 13.
[6] Y. Sun, M. Kirley, and S. K. Halgamuge. 2015. Extended differential grouping for large scale global optimization with direct and indirect variable interactions. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'15). ACM, Madrid, Spain 26(2), 313-320.
[7] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao. 2017. DG2: a faster and more accurate differential grouping for large-scale black-box optimization. IEEE Transactions on Evolutionary Computation, 21(6), 929-942.
[8] Z. Ren, A. Chen, L. Wang, Y. Liang, and B. Pang. 2017. An efficient vector-growth decomposition algorithm for cooperative coevolution in solving large

scale problems. In *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO'2017). ACM, Berlin, Germany. 41-42.

[9] X. M. Hu, F. L. He, W. N. Chen, and J. Zhang. 2016. Cooperation coevolution with fast interdependency identification for large scale optimization. Information Sciences an International Journal, 381(C), 142-160.

[10] Y. Sun, M. Kirley, and S. K. Halgamuge. 2017. A Recursive Decomposition Method for Large Scale Continuous Optimization. *IEEE Transactions on Evolutionary Computation*, 99, 1-1.

[11] K. Tang, X. Li, P. N. Suganthan, Y. Zhang, and T. Weise. 2009, *Benchmark functions for the CEC'2010 special session and competition on largescale global optimization*, Technical Report. Nature Inspired Computation and Applications Laboratory, USTC, China, (1), 1-23.

[12] X. Li, K. Tang, M. N. Omidvar, Y. Zhang, and K. Qin. 2013, *Benchmark functions for the CEC'2013 special session and competition on large scale global optimization*, Technical Report. Evolutionary Computation and Machine Learning Group, RMIT University, Australia, 1-10.