An Efficient Vector-Growth Decomposition Algorithm for Cooperative Coevolution in Solving Large Scale Problems

Zhigang Ren, An Chen
Autocontrol Institute
Xi'an Jiaotong University
Xi'an, Shaanxi, 710049, P. R. China

Lin Wang
School of Information Science and Technology, Northwest University
Xi'an, Shaanxi, 710127, P. R. China
corresponding author: wanglin@nwu.edu.cn

Yongsheng Liang, Bei Pang
Autocontrol Institute
Xi'an Jiaotong University
Xi'an, Shaanxi, 710049, P. R. China

ABSTRACT

By taking the idea of divide-and-conquer, cooperative coevolution provides a powerful architecture for large scale optimization problems, but its efficiency depends heavily on the decomposition strategy. 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 novel decomposition algorithm by exploring interdependency from the view of vectors. It is good at discovering both direct and indirect interdependency and can significantly reduce FEs, which was verified by conducting experiments on CEC'2010 large scale benchmark functions.

CCS CONCEPTS

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

KEYWORDS

Cooperative coevolution, decomposition strategy

1 INTRODUCTION

Nowadays, there are more and more large scale global optimization (LSGO) problems arising in scientific research and engineering practice. Due to the curse of dimensionality, conventional evolutionary algorithms (EAs) lose their efficiency in solving this kind of problems. Cooperative coevolution (CC) algorithms which first decompose the original LSGO problem into a certain number of lower dimensional sub-problems and then separately solve them with EAs, have been shown to be of great potential. The decomposition strategy plays a very important role in CC. It is hoped that the separable variables could be grouped into different sub-problems and all the nonseparable variables could be grouped into the same one.

Most of the existing learning-based decomposition algorithms explore interdependency from the view of variables. Some of them, such as Differential Grouping (DG) [1], just investigate part of interdependency, thus may cause incorrect decomposition results. Some other decomposition algorithms, such as Global Differential Grouping (GDG) [2], detect the interdependency between each pair of variables, thus need many fitness evaluations (FEs). Recently, a decomposition algorithm called Fast Inter-dependency Identification (FII) [3] improves DG and GDG. However, it still needs lots of FEs for discovering indirect interdependency. To tackle these issues, an efficient Vector-Growth Decomposition Algorithm (VGDA) is proposed in this paper. It explores interdependency from the view of vectors and can obtain correct decomposition results with much fewer FEs.

GECCO'17 Companion, July 15-19, 2017, Berlin, Germany

ACM ISBN 978-1-4503-4939-0/17/07.
http://dx.doi.org/10.1145/3067695.3082048

2 VECTOR-GROWTH DECOMPOSITION ALGORITHM

Definition: For an n-dimensional continuous function $f(X)$ to be optimized and a partition of X, $S = \{S_1, S_2, \dots, S_m\}$, we say the variable subsets S_p and S_q (p, q = 1, 2, … , m; $p \neq q$) are additively separable if the following formula holds:

$$\Delta_{\Gamma,S_p}[f](X)|_{S_p=A,S_q=B_1} = \Delta_{\Gamma,S_p}[f](X)|_{S_p=A,S_q=B_2}\,, \quad (1)$$

where

$$\Delta_{\Gamma,S_p}[f](X) = f(\dots,S_p+\Gamma,\dots) - f(\dots,S_p,\dots) \quad (2)$$

refers to the forward difference of $f(X)$ with respect to S_p with an interval Γ, and Γ, A, B_1 and B_2 ($B_1 \neq B_2$) can take arbitrary values as long as they ensure the feasibility of S_p and S_q.

From the above definition, it can be deduced that if S_p and S_q are separable, then $\forall x_i \in S_p$ and $\forall x_j \in S_q$ are separable. On the contrary, if S_p and S_q are nonseparable, then there definitely exists interdependency between at least a pair of variables $\{x_i \in S_p, x_j \in S_q\}$.

According to above properties, VGDA takes the following three key ideas: 1) For the current unpartitioned vector (denoted as UV), it first initializes a nonseparable vector (denoted as NV) with a randomly selected variable $x_r \in UV$ and updates UV with $UV \setminus NV$, then investigates the interdependency between NV and UV. If there is no interdependency, VGDA directly takes NV as a complete variable group. The aim of this operation is to avoid detecting interdependency between NV and each variable in UV. 2) If NV and UV are nonseparable, VGDA tries to find the concrete variables interacting with NV by splitting UV. 3) Once a variable interacting with NV is found, VGDA immediately merges it into NV and updates UV with $UV \setminus NV$, which facilitates finding the indirect interdependency between the older variables in NV and the variables in UV.

As for idea 2), VGDA provides two alternative strategies. One is to sequentially investigate the interdependency between *NV* and each variable in *UV*. The other is to equally divide *UV* into two sub-vectors first, and then detect the interdependency between *NV* and each sub-vector. We name VGDAs taking these two strategies as VGDA-S and VGDA-D, respectively. Their pseudocodes are given in **Algorithm 1** and **Algorithm 2**. Besides, it is worth mentioning that when detecting the interdependency between two entities, VGDA does not stiffly apply (1), but checks whether the absolute difference between two Δs in (1) is smaller than a small number ε. It adopts the same ε setting with GDG [2].

Algorithm 1: *groups* = VGDA-S(X)

```
Initialize groups = ∅ , UV = X;
While |UV| > 1
    select ∀x_r ∈ UV; set NV = {x_r} , UV = UV \ NV;
    While |UV| > 0
        If isSeparable (NV, UV)    break;
        For each x_i ∈ UV
            If ~isSeparable ( NV, x_i )               // NV and x_i are nonsepable
                NV = NV ∪ {x_i}; UV = UV \ NV;    // merge x_i into NV
    groups = groups ∪ NV ;   // take NV as a complete variable group
groups = groups ∪ UV ;
```

Algorithm 2: *groups* = VGDA-D(X)

```
Initialize groups = ∅ , UV = X;
While |UV| > 1
    select ∀x_r ∈ UV; set NV = {x_r} , UV = UV \ NV ;
    While |UV| > 0
        If isSeparable (NV, UV)    break;
        UVS = {UV};              // add UV to a stack
        For each UV' ∈ UVS
            If isSeparable (NV ,UV')
                UVS = UVS \ {UV'};    // exclude UV' from UVS
            Else
                If |UV'|= 1              // UV' contains a single variable
                    NV = NV ∪ UV'; UV = UV \ UV'; UVS = UVS \ {UV'};
                Else                     // UV' contains more than one variable, divide it
                    divide UV' = UV_1' ∪ UV_2'; UVS = UVS ∪ {UV_1', UV_2'} \ {UV'};
    groups = groups ∪ NV;
groups= groups ∪ UV;
```

3 EXPERIMENT AND CONCLUSIONS

We tested the efficiency of VGDA on CEC'2010 benchmark functions which are widely employed in the field of LSGO, and compared the decomposition results with those of DG [1], GDG [2] and FII [3]. From the summarized results listed in the last row of Table 1, it can be observed that the two versions of VGDA obtain the greatest decomposition accuracy which is the same as that of GDG, but they need much fewer FEs than GDG. Furthermore, the two versions of VGDA outperform DG and FII both on decomposition accuracy and the number of FEs. As for VGDA-S and VGDA-D, it can be found that VGDA-D performs more consistently on different kinds of functions, while VGDA-S is more suitable for the functions with strong interdependency.

Table 1: Decomposition Results on CEC'2010 Benchmark Functions

Func	SVars	NVars	Nonsep Groups	VGDA-S / VGDA-D / DG / GDG /FII				
				SVars	NVars	Nonsep Groups	Accuracy(%)	Number of FEs
1	1000	0	0	1000/1000/1000/1000/1000	0 /0 /0/0 /0	0/0/0/0/0	100/100/100/100/100	3009/3009/1001000/501511/3001
2	1000	0	0	1000/1000/1000/1000/1000	0 /0 /0/0 /0	0/0/0/0/0	100/100/100/100/100	3009/3009/1001000/501511/3001
3	1000	0	0	0 /0 /1000/0 /1000	1000/1000/0/1000/0	1/1/0/1/0	- / - /100/ - /100	3016/5094/1001000/501511/3001
4	950	50	1	950 /950/33 /950/429	50/50/967/50/571	1/1/10/1/1	100/100/ - /100/ -	4896 /3760/14554 /501511/3673
5	950	50	1	950/950/950/950/950	50/50/50 /50/50	1/1/1 /1/1	100/100/100/100/100	4912 /3728/905450/501511/3051
6	950	50	1	950/950/950/950/950	50/50/50 /50/50	1/1/1 /1/1	100/100/100/100/100	4894 /3736/906332/501511/3051
7	950	50	1	950/950/248/950/950	50/50/752/50/50	1/1/4 /1/1	100/100/ - /100/100	4884 /3736/67742 /501511/3051
8	950	50	1	950/950/134/950/479	50/50/866/50/521	1/1/5 /1/2	100/100/ - /100/ -	28048/5857/23286 /501511/18850
9	500	500	10	500/500/500/500/500	500 /500 /500/500 /500	10/10/10/10/10	100/100/100/100/100	17441 /10041/270802/501511/8010
10	500	500	10	500/500/500/500/500	500 /500 /500/500 /500	10/10/10/10/10	100/100/100/100/100	17397 /10035/272958/501511/8010
11	500	500	10	0 /0 /501/0 /522	1000/1000/499/1000/478	13/11/10/10/10	- / - / - / - / -	15720 /10506/270640/501511/9255
12	500	500	10	500/500/500/500/500	500 /500 /500/500 /500	10/10/10/10/10	100/100/100/100/100	17492 /9985 /271390/501511/8010
13	500	500	10	500/500/131/500/500	500 /500 /869/500 /500	10/10/34/10/10	100/100/ - /100/100	275825/32849/50328 /501511/96183
14	0	1000	20	0/0/0 /0/0	1000/1000/1000/1000/1000	20/20/20/20/20	100/100/100/100/100	22068 /14050/21000/501511/23020
15	0	1000	20	0/0/0 /0/0	1000/1000/1000/1000/1000	20/20/20/20/20	100/100/100/100/100	22068 /14286/21000/501511/23020
16	0	1000	20	0/0/4 /0/43	1000/1000/996 /1000/957	20/20/20/20/20	100/100/ - /100/100	22068 /14380/21128/501511/30911
17	0	1000	20	0/0/0 /0/0	1000/1000/1000/1000/1000	20/20/20/20/20	100/100/100/100/100	22068 /14344/21000/501511/23020
18	0	1000	20	0/0/78/0/0	1000/1000/922 /1000/1000	20/20/50/20/20	100/100/ - /100/100	372788/54875/39624/501511/369902
19	0	1000	1	0/0/0 /0/0	1000/1000/1000/1000/1000	1/1/1 /1/1	100/100/100/100/100	3011/5011 /2000 /501511/4001
20	0	1000	1	0/0/33/0/0	1000/1000/967 /1000/1000	1/1/124/1/1	100/100/ - /100/100	3011/64291/155430/501511/503500
Average				——	——	——	90/90/60/90/85	43381/14329/316883/501511/57374

1. SVars: the number of separable variables. Nvars: the number of nonseparable variables. Nonsep Groups: the number of nonseparable variable groups.
2. The results of different algorithms are separated by '/'. Accuracy value is marked by '-' if the result is not completely correct. The results of DG, GDG and FII are obtained from [3].

ACKNOWLEDGMENTS

This work was partially supported by the National Natural Science Foundation of China under grants 61105126 and 61503300, and by the Postdoctoral Science Foundation of China under grants 2014M560784 and 2016T90922.

REFERENCES

[1] 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.

[2] 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: 1–24.

[3] X.-M. Hu, F.-L. He, W.-N. Chen, and J. Zhang. 2017. Cooperation coevolution with fast interdependency identification for large scale optimization. *Information Sciences*, 381, 142–160.