2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
Banff Center, Banff, Canada, October 5-8, 2017

Overlapped Cooperative Co-evolution for Large Scale Optimization

An Song
Sun Yat-sen University
Guangzhou, China
safe.song@qq.com

Wei-Neng Chen, Peng-Ting Luo, Yue-Jiao Gong, and Jun Zhang
South China University of Technology
Guangzhou, China
cwnraul634@aliyun.com

***Abstract*—The cooperative co-evolution (CC) framework is one of the most efficient methods to solve large scale optimization problems. The traditional CC framework divides decision variables into several mutually-exclusive groups. In this paper, we propose the overlapped cooperative co-evolution (OCC) framework for large scale optimization problems. In OCC framework, the decision variables that have strong impacts on the optimization are overlapped by different groups. First, we devise the delta-disturbance strategy to detect the influential variables. Then the overlapped grouping strategy is proposed to overlap the influential variables. Finally, the OCC framework is proposed to allocate more computation resources to the influential decision variables. To compare the performance of CC and OCC, we combine two frameworks with the random grouping strategy and the differential grouping strategy, and the comparative experiments are conducted on the CEC2010 benchmark functions. The experimental results verify that the proposed OCC framework is promising through comparing with the CC framework.**

Keywords—Cooperative co-evolution, large scale optimization, evolutionary computation

I. Introduction

Evolutionary computation (EC) has become an effective technology to solve complex optimization problems [1, 2]. As lots of real-world problems are in high dimensions [3], studying EC in solving large scale optimization problems is necessary and important [4-8]. In large scale situations, the search space will increase exponentially when the dimension grows, which makes optimization algorithms lose their effectiveness dramatically, called "the curse of dimensionality". Such feature makes large scale optimization problems difficult to be dealt with.

Cooperative co-evolution (CC) framework [9, 10] is a classical technique for solving high-dimensional optimization problems. Based on the divide-and conquer strategy, the core idea of CC is to divide the original high-dimensional problem into several relatively low-dimensional sub-problems, and each sub-problem is optimized independently. In this way, solving sub-problems is easier than the original one.

This work was supported in part by the National Natural Science Foundation of China under Grant 61622206, Grant 61379061, and Grant 61332002, the Natural Science Foundation of Guangdong under Grant 2015A030306024, the Guangdong Special Support Program No. 2014TQ01X550, and the Guangzhou Pearl River New Star of Science and Technology No. 201506010002. *(Corresponding Author: Wei-Neng Chen).*

The key step of CC is the decomposition procedure. In abstract, the purpose of decomposition is to divide the total decision variables into several groups which compose the subcomponents. As each group of decision variables is optimized separately, the decomposition procedure needs to take the dependence of decision variables into account and classify dependent variables in the same group. As a result, the disturbance from other subcomponents can be reduced while optimizing one subcomponent. To achieve this goal, several grouping strategies were proposed recently [11, 12], such as random grouping [13], differential grouping [12].

Most grouping strategies only consider the dependence among decision variables [12, 14], while another attribute, the impact of decision variables, is ignored. For a specified problem, each dimension of decision variables might have different impacts on optimization. More precisely, some decision variables can make significant contributions to fitness values, while the contributions of other variables might be far less. For example, suppose we have the following fitness function

$$\begin{cases} \text{min: } y = x_1^2 + x_2 \\ \text{subject to: } x_1, x_2 \in [0, +\infty) \end{cases} \tag{1}$$

and the current position of x_1 and x_2 is (10, 10). Obviously, the fitness value y is $10^2+10 = 110$. If the decision variable x_1 is optimized from 10 to 9, the decrease of fitness value made from x_1 is $110-(9^2+10) = 19$. Similarly, if x_2 is optimized from 10 to 9, the decrease of fitness value made from x_2 is $110-(10^2+9) = 1$. Although x_1 and x_2 are optimized with the same degree, their contributions to fitness value are different. Therefore, in this case x_1 has a stronger impact than x_2.

As the decision variables with strong impact can make more contribution to optimization, they deserve more computation resources to get better results. However, the traditional CC framework divides decision variables into mutually-exclusive groups which means that one variable is not allowed to exist in two different groups. In this way, each decision variable will be optimized only once, which implies that computation resources are equally allocated to each variable. In this paper, to improve the optimizing capability for influential variables, we propose a general overlapped cooperative co-evolution (OCC) framework for large scale optimization. First, we devise the delta-disturbance strategy to measure the impact of different variables. Then we decompose the whole decision variables into several overlapped groups. The overlapped parts in each group are composed of the influential variables. Thus an influential variable might exist in different groups and will be optimized several times. Therefore, influential variables can obtain more computation resources.

As OCC is a general framework, different grouping strategies can be embedded, including dynamic grouping strategies and static grouping strategies. We embed the random grouping strategy in OCC, namely OCC-G and embed the differential grouping (DG) strategy in OCC, namely OCC-DG. Random grouping and DG are representative grouping strategies in dynamic grouping and static grouping fields which will be reviewed in Section II. To verify the effectiveness and superiority of proposed OCC framework relative to CC framework, we compare OCC-G with DECC-G [13] and compare OCC-DG with DECC-DG [12] on widely used large scale optimization benchmark sets CEC2010 [15].

The rest of this paper is organized as follows. Section II reviews related work about different grouping strategies and overlapped grouping methods. The proposed delta-disturbance strategy and OCC framework will be detailed in section III. In Section IV, we conduct experiments to verify the promising performance of the proposed framework. Section V concludes this paper..

II. Related Work

CC framework is a classical method to solve large scale optimization problems. The grouping procedure is the key step in the CC framework and seriously influence the performance of optimization algorithms. In this section, we firstly review the related work about different grouping strategies and then the related work about overlapped grouping framework will be introduced

A. Grouping Strategies

Recently, researcher proposed lots of grouping strategies to deal with the problem of variable dependence. In general, most grouping strategies can be classified into two categories, 1) dynamic grouping strategies and 2) static grouping strategies.

Random grouping [16] is the representative of dynamic grouping strategies. Given a group size *k*, *n* decision variables are divided into *n*/*k* groups with equal group size *k* and the content of each group is selected randomly. Delta grouping is proposed by Omidvar *et al*. [17] in 2010. In delta grouping, the average absolute difference (i.e., delta value) of each dimension in two consecutive generations is evaluated. All dimensions are sorted according to the delta values and the grouping procedure is carried out based on the order of dimensions. To effectively decompose fully-separable problem, Omidvar *et al.* [17] applied reinforcement learning to make the sub-problem as small as possible. In this way, the computation resources can be fully utilized.

Dynamic grouping strategies change the content of groups dynamically during the optimization while static grouping strategies keep the grouping results unchanged.

A famous static grouping strategy is the differential grouping (DG) [12]. Based on the differential theory, if the following inequality is satisfied, variable x_i and x_j are considered to be interdependent,

$$\begin{aligned} &f(x_1,,,x_i+\delta_i,,,x_j,,,x_n)-f(x_1,,,x_i,,,x_j,,,x_n)\neq \\ &f(x_1,,,x_i+\delta_i,,,x_j+\delta_j,,,x_n)- \\ &f(x_1,,,x_i,,,x_j+\delta_j,,,x_n) \end{aligned} \tag{2}$$

where $\delta_i,\delta_j \in \boldsymbol{R}$ and $\delta_i,\delta_j \neq 0$. (2) is a determinacy condition. As long as the inequality (2) is satisfied, the dependence of variables is detected. Therefore, the consumed number of fitness evaluations is reduced in DG compared to CCVIL.

DG is only able to find the direct dependence between decision variables while the indirect dependence cannot be detected. Following the basic idea of DG, the extensions of DG, such as XDG [11] and GDG [19], are developed. These strategies first use the traditional DG strategy to find the direct dependence between two variables. Then different variables that interact with the same variable are grouped together. Hence, the indirect dependence among variables can be found. The experiments results in [11, 19] show that XDG and GDG can obtain more accurate grouping results than DG.

B. Overlapped Grouping Framework

Original CC framework decomposes a population into several mutually-exclusive subpopulations. To further improve the optimizing capability, factored evolutionary algorithm (FEA) framework [20] was proposed to divide the population into overlapped subpopulations. To the best of our knowledge, FEA is the only framework with the overlapped grouping characteristic so far.

The overlapping idea of FEA comes from the process that polynomials are decomposed into a product of factors. Naturally, one element might exist in different factors and thus different factors can be seen as overlapping. There are three major steps in FEA, solving, competition and sharing. In the solving stage, all subpopulations are optimized one by one and different optimizers can be integrated. Due to the overlapping mechanism, one decision variable might exist in different subpopulations and thus it will get different values. In the competition stage, the overlapped decision variables will be evaluated on these different values and a better combination of overlapped values will be constructed as the global best solution. In the sharing stage, the worst solution in each subpopulation is replaced with the newly constructed best solution and thus the communications among overlapped subpopulations are realized. The experimental results in [20] show that FEA can outperform CC in some application problems and benchmark functions.

However, the selection of overlapped variables in FEA is crude and FEA is only tested on some small scale problems. To utilize the overlapping idea in solving large scale optimization problems, in this study, we propose the overlapped cooperative

Algorithm 1: delta-disturbance strategy

```
input: $\Delta$, gbest, gbest_value, influential variable list size size, dimension $n$
output: the influential variable list list
1:  impact = zeros(1,$n$);
2:  for $i$ = 1: $n$
3:      gbest' = gbest;
4:      generate random number $r$ in [0, 1];
5:      if ($r$ < 0.5)
6:          gbest'($i$) = gbest($i$)+$\Delta$;
7:      else
8:          gbest'($i$) = gbest($i$)-$\Delta$;
9:      end if
10:     impact($i$) = |func(gbest') - gbest_value|;
11: end for
12: sort decision variables vars in descending order according to impact;
13: list = vars(1: size);
```

co-evolution framework, which will be presented in the next section.

III. Overlapped Cooperative Co-evolution Framework

Decision variables have not only the attribute of dependence, but also the attribute of their impacts on fitness values. The influential decision variables are crucial to optimizing capability and they deserve more computation resources. The traditional CC framework decomposes decision variables into mutually-exclusive groups and thus the computation resources are equally allocated to each dimension. To assign more computation resources to influential variables, we propose the overlapped cooperative co-evolution (OCC) framework for solving large scale optimization problems. We first present the delta-disturbance strategy to measure the impact of different variables and then the overlapped grouping strategy is detailed. Combining above two strategies, we finally present the OCC framework.

A. Delta-disturbance Strategy

As aforementioned, decision variables can have different impacts on optimization. We devise delta-disturbance strategy to measure the impact of decision variables, which is shown in Algorithm 1.

The impact of decision variables is related to the position of variables in the search space. For example, in (1), if the current position is (10, 9), x_1 has a stronger impact on optimization than x_2. Nevertheless, if the position is (0.5, 0.5), x_2 might have a stronger impact than x_1. Therefore, to measure the impact of decision variables, it is necessary to choose a reference position. In this study, we choose the global best solution ***gbest*** found by the population as the reference position.

Besides, if all dimensions are changed simultaneously, it is difficult to estimate which dimension makes the contribution to the optimization. Therefore, we only disturb one dimension each time and the rest is fixed. We use a small real number $\Delta > 0$ to disturb each dimension. The disturbance includes the positive direction and negative direction to fairly measure the impact of each decision variable. First, a random number r in [0, 1] is generated for each dimension. If r is less than 0.5, we make the positive disturbance (line 6 in Algorithm 1). Otherwise, the negative disturbance is executed (line 7 in Algorithm 1). Then we evaluate the fitness value for the disturbed position vector, and record the absolute difference

Algorithm 2: overlapped grouping strategy

```
input: influential variable list list;
output: groups of dimensions allgroups;
1:  dims = {1, 2,…, $n$};
2:  get mutexgroups for dims with CC grouping strategy;
3:  for each group in mutexgroups
4:      duplicated elements dupli = group $\cap$ list;
5:      list' = list \ dupli;
6:      overlap_size = 0.5 $\times$ size(list);
7:      if size(list') < overlap_size then
8:          subgroup = group $\cup$ list';
9:      else
10:         select overlap_size elements randomly from list' into overlap_list;
11:         subgroup = group $\cup$ overlap_list;
12:     end if
13:     allgroups = allgroups $\cup$ subgroup;
14: end for
```

between the disturbed fitness value and the original fitness value as the estimated impact of the decision variable (line 10 in Algorithm 1 where *func*(.) represents the fitness function). A large impact value represents this decision variable might have a stronger impact on optimization. After all decision variables get their impact values, we sort the decision variables in descending order according to the impact values (line 12 in Algorithm 1). As the decision variables with small impact values are not relatively important for optimization, we select the top *size* (*size* = 1, 2, 3…) decision variable as influential variables which will be used in the overlapping procedure.

The delta-disturbance procedure consumes extra fitness evaluations in impact estimation (line 10 in Algorithm 1). The number of extra fitness evaluations for each single execution of the delta-disturbance procedure is equal to the dimension of problems. For example, suppose the problem is provided with 1000 dimensions, the consumption of fitness evaluations for each single execution of the delta-disturbance procedure is 1000.

B. Overlapped Grouping Strategy

As influential decision variables can make more contributions to optimization, they deserve more computation resources. In this study, more computation resources are allocated to these influential variables with the overlapped mechanism. The overlapped grouping strategy for OCC framework is presented in Algorithm 2.

The overlapped grouping strategy is mainly divided into two steps. The first step is to construct mutually-exclusive groups with any traditional grouping strategies reviewed in Section II (line 2 in Algorithm 2). The second step is to construct overlapped groups based on these mutually-exclusive groups. As discussed before, more computation resources should be assigned to influential variables. Thus we only overlap influential variables obtained from Algorithm 1.

For each mutually-exclusive group, there might be some duplicated dimensions in influential variable list. We first delete these duplicated elements from the influential variable list (lines 4-5 in Algorithm 2). Then the expected number of overlapped dimensions is evaluated. In this study, we expect about half of the influential variables are overlapped. As some duplicated variables are deleted from the influential variable list,

Algorithm 3: OCC framework

```
input:  the population pop;
1:  construct influential variable list list with Algorithm 1;
2:  while the termination criterion is not met
3:      divide dimensions into overlapped groups allgroups based on list using Algorithm 2;
4:      for each group in allgroups
5:          subpop = pop[:, group];
6:          subpop = optimizer(subpop, gbest);
7:          pop[:, group] = subpop;
8:      end for
9:      If the specific condition is satisfied
10:         update the influential variable list list using Algorithm 1;
11:     end if
12: end while
```

we need to check the size of the new influential variable list. If the size of the new list is less than the expected overlapping size, all decision variables in the new list are added to the mutually-exclusive group (lines 7-8 in Algorithm 2). Otherwise, we randomly select expected number of decision variables to be added to the mutually-exclusive group (lines 10-11 in Algorithm 2).

Since about half of the influential variables in the list will be added to each mutually-exclusive group, there must be some decision variables which are overlapped by several groups. The overlapping topology is dynamic and is determined by the overlapped influential variables. The overlapped influential variables are optimized by different groups. In other words, these influential variables will be optimized multiple times in the OCC framework and thus more computation resources are allocated to them.

C. *OCC Framework*

Combining the delta-disturbance and overlapped grouping strategy, we propose the OCC framework for large scale optimization problems shown in Algorithm 3. Similar to CC framework, OCC framework also follows the divide-and-conquer idea to make large scale problems easier to solve. However, OCC utilizes overlapped grouping mechanize to allocate more computation resources to influential variables, which is the fundamental difference between OCC and CC framework.

We first construct the influential variable list using Algorithm 1. Then the iterative optimization for each subpopulation is executed until the termination criterion is satisfied. At the beginning of each generation, all dimensions are divided into overlapped groups using Algorithm 2 (line 3 in Algorithm 3). As the selection of overlapped dimensions from influential variable list is random, this operation implies that the overlapping topology and the overlapped dimensions in each group are dynamically changed over the evolutionary state. Therefore, more different overlapping modes can be tried to improve the quality of solutions.

According to the grouping results, the population is divided into several subpopulations. Different optimizer can be embedded to optimize subpopulations, which is similar to the CC framework (lines 4-8 in Algorithm 3).

As aforementioned, the impact attributes of decision variables are related to the reference position of variables and we choose the best solution ***gbest*** found by the population as the reference position. During the optimization, the best solution ***gbest*** is often changed so that the influential variable list needs to be updated as well. However, the estimation of variables' impacts consumes extra fitness evaluations. If the influential variable list is updated too frequently, much more fitness evaluations are consumed in impact estimations. This might cause the problem that there is not enough number of fitness evaluations for optimization, leading to bad performance of algorithms. Hence, the influential variable list should be updated conditionally (line 9 in Algorithm 1), which will be specified in experimental studies in Section IV.

During the optimization, the influential variables overlapped in different groups are optimized multiple times, which implies that they are assigned with more computation resources. At the same time, overlapped variables might have different values in different groups. In this study, the value of overlapped variables in previous groups will be replaced with the value in current groups. In other words, the value of overlapped variables is determined by the final optimized groups. This mechanism might increase the diversity of populations and avoid falling into local optima.

IV. Experimental Studies

A. *Experiment Setup*

To verify the promising performance of the proposed OCC framework in solving large scale optimization problems, we conduct experiments on widely used benchmark functions f_1-f_{20}, which were proposed by CEC2010 special session on large scale global optimization [15]. All functions are provided with 1000 dimensions. f_1-f_3 are fully separable functions. f_4-f_8 have one group of 50 non-separable variables and one group of 950 separable variables. f_9-f_{13} have 10 groups of 50 non-separable variables and one group of 500 separable variables. f_{14}-f_{18} have 20 groups of 50 non-separable variables. f_{19}-f_{20} have one group of 1000 non-separable variables.

Similar to CC framework, OCC is also a general framework. Not only different grouping strategies, but also different optimizers can be embedded. To compare the performance of CC and OCC, we combine these two frameworks with the random grouping, namely DECC-G and OCC-G, respectively, and combine them with the differential grouping (DG), namely DECC-DG and OCC-DG.

The population size is 50 for OCC-DG and DECC-DG [12] and is 100 for OCC-G and DECC-G [13]. The maximum number of fitness evaluations (NFEs) is set to 3×10^6 for all algorithms. As the estimation of variables' impacts consumes extra NFEs, the influential variable list is updated conditionally. For OCC-G, as each subpopulation is optimized 200 times in each generation, the ***gbest*** is changed frequently. The influential variable list is updated in each generation. As for OCC-DG, each subpopulation is optimized once in each generation and thus the ***gbest*** changes slowly. The influential variable list is updated in every 10 generations in OCC-DG. The size of influential variable list *size* is set to 50 empirically.

Additionally, all experiments are executed with 30 independent runs for statistics. During comparisons, the two-

TABLE I
COMPARISON RESULTS OF DECC-G AND OCC-G WITH RANDOM GROUPING ON CEC2010 BENCHMARK FUNCTIONS

	DECC-G		OCC-G		
benchmark	mean	std	mean	std	t-test
f_1	3.381E-08	9.065E-09	6.479E-01	3.120E+00	1.12
f_2	**1.265E+03**	2.378E+01	1.420E+03	3.355E+01	20.35
f_3	**1.134E+00**	3.261E-01	1.520E+00	2.220E-01	5.27
f_4	2.547E+13	7.470E+12	**2.107E+11**	5.545E+10	-18.21
f_5	2.839E+08	7.822E+07	**2.140E+08**	1.692E+07	-4.70
f_6	4.819E+06	8.742E+05	**1.586E+06**	5.904E+05	-16.51
f_7	5.749E+08	3.334E+08	**5.600E-01**	2.355E-01	-9.29
f_8	6.947E+07	3.818E+07	**7.748E+06**	1.745E+07	-7.92
f_9	4.130E+08	5.000E+07	**2.550E+08**	3.350E+07	-14.14
f_{10}	**1.015E+04**	3.148E+02	1.053E+04	2.828E+02	4.80
f_{11}	2.575E+01	1.238E+00	**2.334E+01**	2.913E+00	-4.10
f_{12}	8.967E+04	1.033E+04	**4.639E+04**	4.706E+03	-20.54
f_{13}	7.727E+03	5.385E+03	**2.301E+03**	6.563E+02	-5.39
f_{14}	9.712E+08	5.842E+07	**7.530E+08**	5.683E+07	-14.42
f_{15}	1.248E+04	7.703E+02	**1.139E+04**	1.105E+03	-4.34
f_{16}	**6.944E+01**	7.608E+00	8.458E+01	8.293E+00	7.25
f_{17}	2.923E+05	1.315E+04	**2.481E+05**	1.495E+04	-11.96
f_{18}	3.302E+04	1.140E+04	**6.724E+03**	1.176E+03	-12.36
f_{19}	1.090E+06	5.604E+04	**1.058E+06**	6.135E+04	-2.06
f_{20}	4.396E+03	7.137E+02	**3.888E+03**	2.761E+02	-3.58
Win:	4		15		

TABLE II
COMPARISON RESULTS OF DECC-DG AND OCC-DG WITH DIFFERENTIAL GROUPING ON CEC2010 BENCHMARK FUNCTIONS

	DECC-DG		OCC-DG		
benchmark	mean	std	mean	std	t-test
f_1	N/A	N/A	N/A	N/A	N/A
f_2	N/A	N/A	N/A	N/A	N/A
f_3	N/A	N/A	N/A	N/A	N/A
f_4	2.130E+13	6.202E+12	**1.630E+12**	6.488E+11	-16.99
f_5	**2.190E+08**	1.882E+07	3.280E+08	7.415E+07	7.68
f_6	**1.674E+01**	3.979E-01	1.905E+01	4.139E-01	21.66
f_7	**6.352E+06**	2.527E+06	5.242E+10	6.190E+09	45.60
f_8	4.774E+07	2.408E+07	5.195E+07	3.337E+07	0.55
f_9	2.366E+08	1.968E+07	**1.006E+08**	9.299E+06	-33.67
f_{10}	6.549E+03	1.787E+02	**5.462E+03**	5.149E+02	-10.74
f_{11}	**1.150E+01**	8.052E-01	1.430E+01	1.458E+00	9.04
f_{12}	**6.108E+04**	4.743E+03	4.110E+06	1.699E+05	128.30
f_{13}	1.661E+06	1.274E+06	**1.033E+03**	3.435E+02	-7.02
f_{14}	8.653E+08	4.841E+07	**3.410E+08**	2.314E+07	-52.62
f_{15}	6.759E+03	8.331E+01	**5.154E+03**	4.705E+02	-18.09
f_{16}	**7.239E-05**	4.320E-06	6.436E+00	2.630E+00	13.18
f_{17}	**2.490E+05**	1.082E+04	3.747E+06	1.816E+05	103.55
f_{18}	5.880E+10	8.450E+09	**6.194E+03**	2.542E+03	-37.48
f_{19}	N/A	N/A	N/A	N/A	N/A
f_{20}	2.070E+11	2.233E+10	**2.049E+06**	2.955E+05	-49.92
Win:	7		8		

tailed t-test at significance level α =0.05 is conducted where the critical value for 30 samples is 2.042. According to the t-test, significantly best solutions are highlighted in bold. Specially, the row "win" records the number of benchmark functions where the algorithm can obtain better results than the other.

B. Simulation Results

Table I presents the average results over 30 runs of DECC-G and OCC-G, which are combined with the random grouping strategy.

From Table I, it can be seen that OCC-G significantly outperforms DECC-G on 15 out of 20 functions especially on non-separable functions (f_4-f_{20}). In these non-separable problems, the dependence among decision variables is more complex. Thus the random grouping strategy with CC framework is hard to efficiently detect the variable dependence. On the contrary, OCC utilizes the overlapping mechanism. On the one hand, the overlapped decision variables might improve the possibility of grouping interactive variables together. On the other hand, as the overlapped variables are selected from the influential variable list, more computation resources are allocated to influential variables and thus the effectiveness of the algorithm is improved.

Table II presents the average results over 30 runs of DECC-DG and OCC-DG, which are combined with the differential grouping strategy. The differential grouping strategy divides all decision variables into one group in functions f_1-f_3 ,f_{19}. However, the overlapping mechanism is valid under the situation that there are at least two groups. Therefore, the comparisons between OCC-DG and DECC-DG are meaningless on functions f_1-f_3 ,f_{19} which are marked by "N/A" in Table II.

From Table II, it can be observed that OCC-DG is significantly better than DECC-DG on 8 functions, especially on the functions f_4, f_{13}, f_{18} and f_{20}. The promising performance of OCC-DG verifies that the overlapping mechanism is still effective for static grouping strategies. However, the performance of the OCC framework with DG strategy is not as good as it with the random grouping strategy. This is because the group size and group number obtained by differential grouping are various. For example, on function f_5, there are 2 groups obtained by DG while there are 24 groups on function f_{16}. In addition, the group size is various on some functions. For example, the minimum group size is 50 while the maximum group size is 950 on function f_6. Such non-uniform grouping results might increase the difficulty of overlapping. In contrast, the random grouping strategy divides dimensions into groups with equal size and thus the group number is the same for all functions. This characteristic can make the OCC framework work efficiently. Hence, OCC framework is more appropriate to solve the groups of variables with the uniform size.

The converging curves on some functions are depicted in Fig. 1. The solid lines represent the OCC-based algorithms and the dashed lines represent the CC-based algorithms. From Fig. 1, it can be observed that the OCC-based algorithms can converge at lower fitness values than CC-based algorithms. This verifies that the proposed OCC framework is superior to CC framework on these functions with the dynamic grouping strategy and the static grouping strategy.

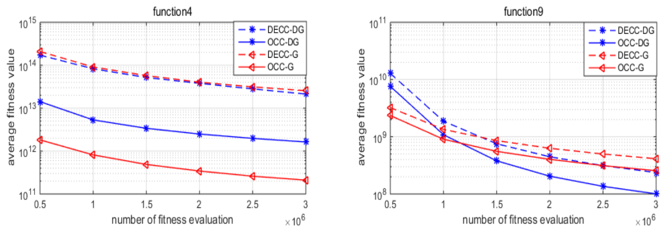

Fig.1. the converging curves on f_4 and f_9 for compared algorithms

V. Conclusions

In this paper, we have proposed the overlapped cooperative co-evolution (OCC) framework for large scale optimization problems. In OCC framework, the decision variables that can make more contributions to optimizations are detected and more computation resources are allocated to them to improve the optimizing capability of algorithms. The OCC framework is general and thus different grouping strategies can be embedded. The experimental results on benchmark CEC2010 verify that the proposed OCC framework is promising.

This work has shown the promising performance of the overlapped grouping mechanism, but the overlapped size for each group is predefined. When the group sizes are various, the fixed group size might be inappropriate. Thus the quality of solution could be further improved with the adaptive control of the overlapped size in future work.

References

[1] S. Wu, Y. Liu, T. Hsieh, Y. Lin, C. Chen, C. Chuang, and C. Lin, "Fuzzy Integral with Particle Swarm Optimization for a Motor-Imagery-based Brain-Computer Interface," *IEEE Transactions on Fuzzy Systems,* vol. 25, no. 1, pp. 21-28, 2016.

[2] W. Chen, J. Zhang, Y. Lin, N. Chen, Z. Zhan, H. S. Chung, Y. Li, and Y. Shi, "Particle swarm optimization with an aging leader and challengers," *IEEE Trans. Evol. Comput.,* vol. 17, no. 2, pp. 241-258, 2013.

[3] R. Shang, K. Dai, L. Jiao, and R. Stolkin, "Improved Memetic Algorithm Based on Route Distance Grouping for Multiobjective Large Scale Capacitated Arc Routing Problems," *IEEE Trans. Cybern.,* vol. 46, no. 4, pp. 1000-1013, 2015.

[4] Q. Yang, W. Chen, T. Gu, H. Zhang, J. D. Deng, Y. Li, and J. Zhang, "Segment-based predominant learning swarm optimizer for large-scale optimization," *IEEE Trans. Cybern.,* vol. PP, no. 99, 2017.

[5] X. Hu, F. He, W. Chen, and J. Zhang, "Cooperation coevolution with fast interdependency identification for large scale optimization," *Information Sciences,* vol. 381, pp. 142-160, 2017.

[6] W. Shi, W. Chen, Q. Yang, and J. Zhang, "A multi-optimizer cooperative coevolution method for large scale optimization," in *IEEE SSCI*, 2016, pp. 1-7.

[7] H. Xie, Q. Yang, X. Hu, and W. Chen, "Cross-generation Elites Guided Particle Swarm Optimization for large scale optimization," in *IEEE SSCI*, 2016, pp. 1-8.

[8] Q. Yang, H. Xie, W. Chen, and J. Zhang, "Multiple parents guided differential evolution for large scale optimization," in *IEEE CEC*, 2016, pp. 3549-3556.

[9] L. M. Antonio and C. Coello, "Use of cooperative coevolution for solving large scale multiobjective optimization problems," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, 2013, pp. 2758-2765.

[10] J. Fan, J. Wang and M. Han, "Cooperative coevolution for large-scale optimization based on kernel fuzzy clustering and variable trust region methods," *IEEE Transactions on Fuzzy Systems,* vol. 22, no. 4, pp. 829-839, 2014.

[11] Y. Sun, M. Kirley and S. K. Halgamuge, "Extended Differential Grouping for Large Scale Global Optimization with Direct and Indirect Variable Interactions," in *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, 2015, pp. 313-320.

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

[13] Z. Yang, K. Tang and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences,* vol. 178, no. 15, pp. 2985-2999, 2008.

[14] G. Dai, X. Chen, L. Chen, M. Wang, and L. Peng, "Cooperative coevolution with dependency identification grouping for large scale global optimization," in *2016 IEEE Congress on Evolutionary Computation*, 2016, pp. 5201-5208.

[15]K. Tang, X. Yao and P. N. Suganthan, " Benchmark functions for the CEC' 2010 special session and competition on large scale global optimization," *Rapport technique, USTC, Nature Inspired Computation and Applications Laboratory,* pp. 1-23, 2010.

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

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

[18] M. N. Omidvar, Y. Mei and X. Li, "Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms," in *IEEE CEC*, 2014, pp. 1305-1312.

[19] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM TOMS,* vol. 42, no. 2, p. 13, 2016.

[20] S. Strasser, J. Sheppard, N. Fortier, and R. Goodman, "Factored evolutionary algorithms," *IEEE Trans. Evol. Comput.,* vol. 21, no. 2, pp. 281-293, 2017.