


# Cooperative-Coevolution-CMA-ES with Two-Stage Grouping

Dani Irawan\*, Boris Naujoks†

*Institute for Data Science, Engineering, and Analytics*  
TH Köln, Germany

\*✉ dani.irawan@th-koeln.de  0000-0002-4213-941X

†boris.naujoks@th-koeln.de

Michael Emmerich

*Leiden Institute of Advanced Computer Science*

Universiteit Leiden, The Netherlands

m.t.m.emmerich@liacs.leidenuniv.nl

**Abstract**—The Cooperative Coevolution (CC) framework is the state of the art for solving large scale global optimization (LSGO) problems. A particular challenge in using CC lies in the decomposition of variables and resource allocation. In this work, the decomposition phase of the framework is performed in two stages to address both variable interaction and efficient resource allocation. The algorithm starts with differential analysis followed by differential grouping. The differential analysis allows efficient resource allocation while the differential grouping will detect variable interactions. The differential grouping will act on a small number of variables and will not consume as much computational budget as a single-stage grouping. While not all variable interactions will be detected, separable variables will be recognized hence specialized solvers for separable problems can be employed on these subproblems. In this work, the two-stage grouping CC (TSCC) is paired with a hybrid algorithm where sep-CMA-ES is used to solve the separable subproblem and CMA-ES is used to solve the non-separable subproblems, the algorithm is referred as TSCC-CMAES. A comparison between TSCC and CC with groups based on either differential analysis or differential grouping is carried out. In general, TSCC could outperform the two single-stage grouping methods. Additionally, the TSCC-CMAES shows a competitive advantage on a number of more complex problems against state-of-the-art algorithms and it is shown that the effect of the population size and group size is crucial in achieving these results.

**Index Terms**—cooperative coevolution, large scale optimization

## I. INTRODUCTION

Evolutionary algorithms (EA) have been successfully used in solving large scale global optimization (LSGO) problems in the Cooperative Coevolution (CC) framework [1]. The framework employs a divide-and-conquer approach on the large scale problem, solving it as several subproblems. Each subproblem considers a subset of the optimization variables that are grouped by using one or more decomposition schemes. Throughout this paper, “grouping” refers to these decomposition schemes. Algorithms based on the CC framework rely heavily on the way the groups are generated [2].

Many studies state that the decomposition phase should tie tightly-interacting variables into the same subproblem while keeping the interactions weak among distinct subcomponents

[3]. Bad grouping may deteriorate some algorithms’ performance [4]. There are many alternative grouping schemes which are designed to detect and group interacting variables. Some methods directly evaluate the objective function to check interactions such as differential grouping (DG) [5] and its successor: DG2 [6], recursive DG (RDG) [2], and the latest version of RDG: RDG3 [7]. Some other methods learn the correlations between the variables instead of interactions, e.g. model complexity control (MCC) [8], maximum variance decomposition (MaVD), and minimum variance decomposition (MiVD) [9].

Recent studies [10]–[12] focuses on resource allocation. The sensitivity analysis budget-allocation CC (SACC) and contribution based CC (CBCC) algorithms can start with any grouping method. The groups contributions are used to determine how much computational budget are allocated to optimize the group. In CBCC [10], the group to be optimized is chosen based on how much improvement the group provided at previous iterations. In [11], on the other hand, the variables’ contributions are assessed only at the start of the optimization. Alternatively, the contributions of the variables can directly be used for grouping. In the multilevel optimization framework based on variables effect (MOFBVE) [12] variables with similar contributions are grouped together by clustering methods; then the budget for the groups are set based on their overall contributions.

In this work, a two-stage grouping similar to multi-level CC (MLCC) [13] is used. In MLCC, the variables are grouped randomly and a second stage grouping is conducted afterwards. Here, the variables are first grouped based on their contributions and further grouped based on their interactions within the subgroups. This will allow us to detect all separable variables. By knowing the separable variables, the hybrid solver approach introduced in [14] can be used. In [14], the self-adaptive neighborhood-search differential evolution (SaNSDE) [15] and artificial bee colony [16] algorithms are used for the non-separable and separable subproblems, respectively. In this work, the covariance matrix adaptation evolution strategy (CMA-ES) [17] and separable CMA-ES (sep-CMA-ES) [18] are used instead.

The remainder of this article is organized as follows: Section II explains the CC framework and the base grouping methods

This work is funded by the European Commission’s H2020 programme, UTOPIAE Marie Curie Innovative Training Network, H2020-MSCA-ITN-2016, under Grant Agreement No. 722734

that we will combine. Sections III and IV explain how the grouping methods are combined and how it affects the decomposition and optimization. Section V contains the details of what tests are conducted and the parameters used in the tests. Finally, Sections VI and VII interpret and analyze the results from the experiments.

## II. COOPERATIVE COEVOLUTION

The CC framework has two major steps. The first step is the decomposition of the problem into several subproblems and the other is the optimization of the subproblems themselves.

The first step is intended as a scalability agent for optimization algorithms. Often, algorithms struggle to solve LSGO problems due to the curse of dimensionality, i.e., by dividing the LSGO problems into a set of smaller subproblems  $S$ , it is expected that the algorithms, which are previously unusable, can become practical again. The decomposition can be performed automatically. Such automatic decomposition methods are described in subsections II-A and II-B.

The second step, after grouping, is optimization. Cooperative coevolution partitions the components of the optimization variable vector  $\mathbf{x}$  into  $|S|$  subvectors and each subvector  $\mathbf{x}_s$  defines a subproblem ( $s$  is one of the subproblem in  $S$ ). Each subvector will be optimized separately (coevolution). However, each subvector only has a subset of the variables needed to compute the objective value. It is impossible to evaluate the objective function with only  $\mathbf{x}_s$ , the whole vector  $\mathbf{x}$  must be filled. To address this, a context vector (CV) is introduced. The CV contains all the variables needed to evaluate the objective function. The optimization subproblems are then constructed as optimizing  $\mathbf{x}_s$  while all other variables  $\mathbf{x}_{-s}$  are filled with the CV (cooperation) and kept constant. After  $\mathbf{x}_s$  is optimized with respect to the current CV, the CV is updated with the optimized  $\mathbf{x}_s$  (see Fig. 1). When all variables in the context vector have been updated, a *cycle* is complete. The CC framework usually runs over several cycles.

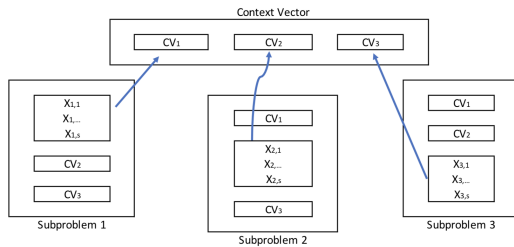


Fig. 1. Optimizing the subproblems and updating the context vector.

### A. Differential Grouping

Differential grouping [5] and its variants, e.g., DG2 [6], RDG [2], RDG3 [7], and global DG (GDG) [19] are devised to capture variable interactions to exploit additive separability in the objective function. A function is partially separable iff:

$$\arg \min_{\mathbf{x}} f(\mathbf{x}) = (\arg \min_{x_1} f(x_1, \dots), \dots, \arg \min_{x_d} f(\dots, x_d)) \quad (1)$$

Further, a function is additively separable iff:

$$f(\mathbf{x}) = \sum_{s=1}^{|S|} f_s(x_s), |S| > 1 \quad (2)$$

Where  $f(x)$  is the original objective function, and each  $f_s(\cdot)$  is one of  $|S|$  non-separable subfunctions. The variables for each  $f_s(\cdot)$  are then grouped together. Each variable group is solved as a separate subproblem. The variable groups form a partition of the set of all variables.

If the non-separable subfunctions have overlapped variables, DG would not be able to detect it at all, while in DG2, RDG, and GDG, the subfunctions would be considered as non-separable and all variables will be grouped together [6]. In this case, grouping variables for all overlapping subfunctions together is considered as the correct grouping (also known as ideal grouping); however, this could mean that no decomposition is conducted at all. The recent addition to the family, RDG3, ensures decomposition by limiting the group sizes.

Another drawback of using DG is that it requires a lot of function evaluations, e.g., the number of function evaluations required by the DG2 algorithm follows Eq. 3.

$$FE_{DG2} = \frac{d^2 + d + 2}{2} \quad (3)$$

This corresponds to 500501 function evaluations for problems with 1000 variables. The recursive variants of the DG method are more efficient and able to reduce the number to  $\mathcal{O}(d \log(d))$  [7].

### B. Differential Analysis

In solving LSGO problems, often the number of function evaluations is limited. The optimization process may have to be terminated before it converges to the global optimum. Therefore, to achieve a result as close to the global optimum as possible, more effort should be allocated to optimize variables with larger impacts on the objective value. To know the extent of the variables' influence, sensitivity analysis (SA) is performed. Several well known SA methods are available, e.g., Sobol variance decomposition [20], regression analysis (e.g., [21]), and differential analysis (DA, also known as Morris method) [22], [23]. Among these methods, DA is an efficient method to learn the variables' influences on the objective value as used in [11] and [12]. Sobol method requires many more function evaluations (in the order of  $500d$ ,  $d$  is the number of variables) [23] while regression analysis often requires its model to follow a structure (parametric) or expensive to build a reliable large scale model. In this work, the DA will be used for SA. Furthermore, SA will be used for both grouping and budget allocation.

For DA, each variable in the search space is divided into  $p$  intervals. A grid jump  $\Delta > 0$  is then chosen from a multiple of  $1/(p-1)$ . Elementary effect (EE) for each variable can then be calculated using Eq. 4

$$EE_j(\mathbf{x}) = \frac{f(x_1, \dots, x_{j-1}, x_j + \Delta, \dots) - f(\mathbf{x})}{\Delta}, j = 1, \dots, d \quad (4)$$

where  $\mathbf{x}$  is a random point in the search space such that  $\mathbf{x} + \Delta$  is still within the search space. By taking  $r$  samples of  $\mathbf{x}$ , several  $EE_j$  can be sampled and its distribution can be obtained. Campolongo, et al. [23] proposed to use the mean of the absolute value of  $EE_j$ , the  $\mu^*$ , for ranking the importance of each variable and it is used in this work. The variables are then sorted the highest rank (largest  $\mu^*$ ) to the lowest (smallest  $\mu^*$ ) and divided into levels based on the rank. For example, the variables can be grouped into two levels by taking the top-ranked half as the first group, and the lower-ranked half as another group. The function evaluation required to do DA scales linearly with  $d$  following Eq. 5.

$$FE_{DA} = r * (d + 1) \quad (5)$$

The benefit of using sensitivity analysis for grouping is in resource allocation. The variable importance measures are crucial in solving LSGO problems with a limited budget. More budget should be allocated to optimize the variables with higher impacts on the objective function. As the importance values are already calculated, the computational resources can be allocated proportional to the importance values.

Eq. 6 is an example where the variables have imbalanced effects. A small perturbation on  $x_1$  has much larger effects on  $f(\mathbf{x})$  compared to a perturbation on  $x_3$  ( $10^4$  times larger).

$$f(\mathbf{x}) = 10^6 x_1 + 10^4 x_2 + 10^2 x_3 \quad (6)$$

### III. TWO-STAGE, CONTRIBUTION-FIRST GROUPING

The two grouping schemes in Sec. II-A and II-B have their strengths and weaknesses. Considering them, a two-stage grouping scheme is proposed in this work and, applied in the CC framework, we call it two-stage CC (TSCC) and the procedure is shown in Algorithm 1.

The DA should be the primary grouping scheme because it is cheaper than the DG and its computational cost grows much slower. The DA allows us to recognize the more important variables and focus the optimization effort on higher-ranked variables. This also will ascertain that the main problem will be decomposed into specified group sizes.

---

**Algorithm 1**  $G = \text{TSCC}(func, nLevel, nVar)$

---

```

1:  $G = \{\}$ 
2:  $\mu^* = \text{Morris}(func)$  // execute differential analysis
3:  $R = \text{order the variables based on } \mu^*$ 
4:  $k = nVar/nLevel$  // group size
5: for  $i$  in  $1:nLevel$  do
6:    $A = 1:k$ 
7:    $C = R_A$  // fill group with current highest-ranked variables
8:    $R = R_{-A}$  // remove variables from  $R$ 
9:    $(seps_i, nonseps_i) = \text{DG2}(func, C_i)$  // for each level, use DG2
10:  add  $seps_i$  and  $nonseps_i$  to  $G_i$ 
11: end for

```

---

To run the TSCC algorithm, it requires several inputs: which function to be solved  $func$ ; how many levels should the Morris method produce  $nLevel$ ; and how many variables are to be optimized  $nVar$ . The inputs  $func$  and  $nVar$  are defined in the LSGO problem, while  $nLevel$  is a parameter that can be tuned. The TSCC will give the grouping list  $G$  as an output. In  $G$ , each level  $G_i$  has one separable variable group and possibly several non-separable groups ( $seps$  and  $nonseps$ , respectively). Only one of  $seps$  or  $nonseps$  can be empty.

After the first stage grouping, the groups are further grouped into subgroups based on their interactions by DG. In this work, DG2 is used for the secondary grouping. As the number of variables in each group can be controlled, the computational cost for the DG can also be controlled. For example, a problem with 1 000 variables can be first divided into 10 groups of 100 variables. By using  $r = 20$ , this process took 20 020 function evaluations (see Eq. 5). Based on Eq. 3, using DG2 on each of these groups consumes 5 051 function evaluation, so in total 70 530 function evaluations are needed. This is approximately 7 times more efficient than using DG2 on the main problem.

Despite the increased grouping efficiency for the second stage, the first stage grouping may put non-separable variables in different groups. Consequently, variable interaction between variables across different groups will not be detected. Separable variables, on the other hand, are still properly detected. The grouping accuracies for the CEC2013 LSGO test problems are shown in Tab. I. Ackley's function on  $f_3$  and  $f_6$  are not additively separable, hence DG2 unable to properly group it [6]. It is also interesting to observe the grouping on  $f_{12}$ . The function has weak overlaps and the grouping with DA breaks most of these overlaps so it ends up with many separable variable and very small non-separable groups (on average 2.4 variables per group).

### IV. GROUPING EFFECT ON COVARIANCE MATRIX

The CMA-ES algorithm draws its population from a multivariate normal distribution. This distribution is described by a  $d$ -dimensional mean vector and a  $d \times d$  covariance matrix. The CMA-ES algorithm adapts the covariance matrix to raise the probability of obtaining successful offspring [17].

In solving LSGO problems, the covariance matrix has a large degree of freedom and learning it can dominate the search cost. Internal computational complexity also scales quadratically with  $d$  [18]. To make CMA-ES viable for LSGO problems, Tong, et al. [24] uses model complexity control (MCC) framework in MCC-CMA-ES. In MCC-CMA-ES, the variables are grouped based on their correlation and then drops the correlation between variables of different groups to zero. A special case of MCC-CMA-ES is the sep-CMA-ES [18] which restricts the covariance matrix to be diagonal.

This work is similar to MCC-CMA-ES in that it reduces the degree of freedom by ignoring inter-group dependence; however, the CC framework is used instead of the MCC framework. Instead of grouping based on correlation as in MCC-CMA-ES, this work uses a differential-based grouping scheme. In other words, it is assumed that the variables

TABLE I

GROUPING ACCURACY OF TSCC. FIRST ENTRY (BEFORE "/") OF EACH CELL IS THE OBTAINED GROUPING; THE SECOND ENTRY IS THE CORRECT VALUE.

Function	Separable	Non-separable	Non-sep. group	Function	Separable	Non-separable	Non-sep. group
$f_1$	1000/1000	0/0	0/0	$f_9$	2/0	998/1000	37/20
$f_2$	1000/1000	0/0	0/0	$f_{10}$	1/0	999/1000	28/20
$f_3$	0/1000	1000/0	4/0	$f_{11}$	1/0	999/1000	39/20
$f_4$	700/700	300/300	9/7	$f_{12}$	554/0	446/1000	189/1
$f_5$	705/700	295/300	11/7	$f_{13}$	1/0	904/905	23/1
$f_6$	1/700	999/300	10/7	$f_{14}$	2/0	903/905	24/1
$f_7$	700/700	300/300	9/7	$f_{15}$	0/0	1000/1000	4/1
$f_8$	189/0	811/1000	27/20				

are independent if they do not interact and/or have different levels of contribution. If only interaction is considered, the covariance matrix will have the same form as the design structure matrix (DSM). A DSM is a binary matrix that indicates if variables interact (1s) or do not interact (0s) [6]. An example of DSM is shown in Fig. 2, while Fig. 3 and 4 show the interaction matrix learned by the SACC and TSCC.

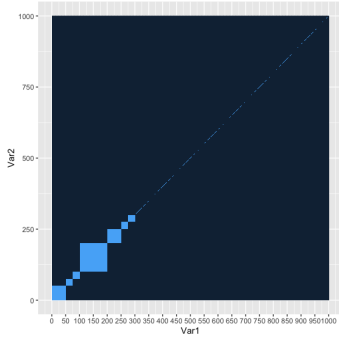


Fig. 2. Design structure matrix for  $f_4$  from CEC2013 benchmark problems, ordered by group. Light color indicates interaction.

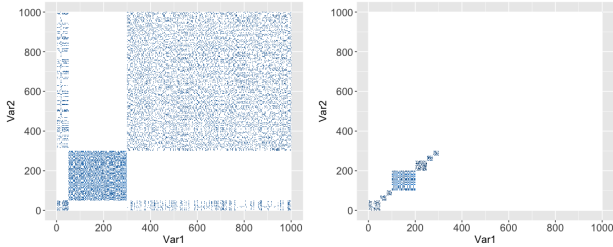


Fig. 3. Group matrix for imbalanced CEC2013  $f_4$  problem learned by SACC (left) and TSCC (right). The variables are ordered by group. Dark color indicates the pair of variables are grouped together.

The base CMA-ES algorithm will learn the dependencies (covariances) between all pairs of variables. Grouping will reduce the model complexity because the algorithm only considers dependencies between variables in the same group. Covariances between variables in different groups are not computed/learned. The less complex interaction models are indicated by sparse points in Fig. 3 and 4. When further grouped using DG, the model is further simplified and will resemble the DSM as shown by the right-plot of both figures.

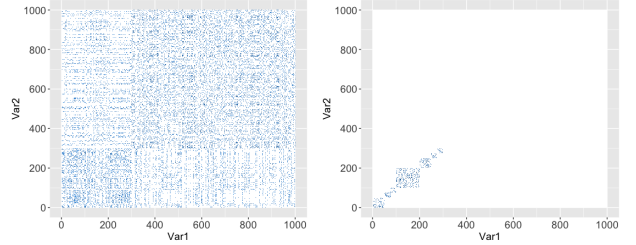


Fig. 4. Group matrix for balanced (equal weights) CEC2013  $f_4$  problem learned by SACC (left) and TSCC (right). The variables are ordered by group. Dark color indicates the pair of variables are grouped together.

The TSCC grouping may produce the ideal grouping when the problem has imbalanced subproblems as shown in Fig. 3.

## V. NUMERICAL EXPERIMENT

To assess the performance of TSCC, the grouping scheme is tested on the CEC2013 benchmark problems [25]. In this work, the following algorithms are tested:

- 1) CC-DG2-CMAES: CC decomposed by DG2. CMA-ES used on non-separable subproblems and sep-CMA-ES for separable subproblems.
- 2) SACC-CMAES: SA-based CC with CMA-ES solver.
- 3) TSCC-CMAES: Two-stage grouping CC. CMA-ES used on non-separable subproblems and sep-CMA-ES for separable subproblems.
- 4) sep-CMA-ES: sep-CMA-ES without decomposition.
- 5) CCDG2: CC with DG2 solved by SaNSDE.
- 6) SACC-DE: SACC with SaNSDE solver.
- 7) TSCC-DE: Two-stage grouping CC solved by SaNSDE.

The algorithms are chosen to allow comparison between single-stage grouping with either SA or DG against the two-stage grouping. The grouping methods are then paired with CMA-ES and SaNSDE. The two solvers are chosen because of their prominent use in the field. Additionally, while in algorithm 4 no decomposition is performed, it has the simplest covariance model a CMA-ES can have.

For each test problem, 25 repetitions with the same set of random number seeds are used. This ensures that the result from DA will always be the same for the same seed. The groups obtained in SACC-CMAES will be the same as in SACC-DE. These groups are also the ones that are further processed by DG2 for TSCC. In this work, the number of

variable levels produced by DA is set at 4 levels for all tests, similar to the number of levels used in [12]. However, it should be noted that the grouping used here is slightly different to the grouping in [12].

In this work, the number of iteration is scaled with respect to  $\sum_{i \in S} \mu_i^*$ , with  $S$  being a variable group. The following formula is used to determine the portion of the computational budget being assigned to a group:

$$Iter_s = \begin{cases} 1 + \log \sum_{i \in S} \mu_i^*, & \text{if } \sum_{i \in S} \mu_i^* > 1 \\ 1, & \text{otherwise} \end{cases} \quad (7)$$

A similar resource allocation scheme is used for CMA-ES based algorithms. For CMA-ES, the portion calculated by Eq. 7 is normalized by the total portion and multiplied by a fixed number  $n$  to allow cumulation learning in CMA-ES. In this work,  $n = 2000$  is used so that the cumulation is quite long while also ensuring several CC cycles will be conducted. The solvers are initially configured following the default setting as suggested in [17] and [18]. The population size and offspring size are set equal to the number of variables in the subproblems. For each subproblem, the covariance matrix, step-size, and evolution path are persistent throughout each experiment.

The computation budget is set at 3 000 000 function evaluations and data is logged every 100 000 function evaluations. All codes are written in R and ran on R version 3.6.1.

## VI. PERFORMANCE COMPARISON

### A. TSCC compared to single-stage grouping

To assess the performance of the proposed grouping scheme, we start by looking at the convergence plots of several functions. Fig. 5 shows flat performances at the early stage for both CC-DG2-CMAES and CC-DG2-DE on all plots. The flat line is due to the grouping taking approximately 500 000 function evaluations. Afterwards, the CC-DG2-CMAES algorithm starts with large step-size and most offspring are not feasible as the algorithm adapts to the problem thus the flat region is longer. On  $f_{13}$  and  $f_{14}$  the CC-DG2-CMAES does not improve at all due to step-size divergence (see Section VI-B). SACC and TSCC, on the other hand, use far fewer function evaluations for grouping and the objective values improve relatively early. For a more detailed look on the performances, Tab. II shows comparison of the performances between the algorithms at 3 000 000 function evaluations.

For DE-based algorithms, in general, TSCC-DE does not outperform SACC-DE. On fully separable ( $f_1$ - $f_3$ ) and fully non-separable functions ( $f_{15}$ ), TSCC-DE will produce the same grouping as SACC. This means the additional computational cost for the secondary grouping is wasted. Considering 4 primary groups are used, each with 250 members, the TSCC-DE is lagging by approximately 125 500 function evaluations and its performances are similar to or worse than SACC-DE on those functions.

One particularly interesting result is seen on  $f_{12}$ : TSCC-DE performs much worse than SACC-DE. The reason for the large

performance drop is because the function has a single variable-overlap between subproblems (weak link) and the sensitivity-analysis based grouping disconnects the overlaps. When DG2 is applied, the final groups consist of many small subgroups where most subgroups only contain two variables. Liu, et al. [26] have previously shown that SaNSDE is not efficient in solving subproblems with very few variables and it explains the large performance drop in TSCC-DE.

Contrary to the performance drop in DE-based algorithms, TSCC-CMAES outperforms SACC-CMAES in most cases. The performance improvement is not from exploitation of the variable interactions because CMA-ES is rotationally invariant. The performance improvement can be attributed to the smaller search space after the secondary grouping. Mei, et al. [27] has shown that CMA-ES generally performs better and converges faster on smaller groups of variables. The SACC and TSCC also address the suggestion in [27] to decompose the problem even when the ideal grouping would require the problem to be solved without decomposition, e.g.  $f_{12}$ - $f_{15}$ .

The SACC-CMAES outperforms TSCC-CMAES in  $f_2$  and  $f_5$ , but not  $f_9$ , all of which are constructed by Rastrigin's functions. According to [28], [29], the performance of CMA-ES on Rastrigin's function increases as its population size grows. Because on TSCC-CMAES the population size is scaled to the group size, the population sizes on TSCC-CMAES will be smaller. On  $f_5$ , one of the subproblem has a group size of 700, thus a large population size would be required to solve it. Compared to  $f_5$ , the largest group size in  $f_9$  is much smaller: 100. Tests on TSCC-CMAES with  $2d$  population size and  $n = 1000$  shows an improvement for  $f_5$ . With the modified setting, TSCC-CMAES can reach objective values at the order of  $10^5$  on  $f_5$ .

On  $f_2$ , the reason for the performance drop is different. The DG2 did not change the grouping but switch the solver from CMA-ES to sep-CMA-ES. As the performance of sep-CMA-ES is not largely different from SACC-CMA-ES, the performance gain from the switch is not significant. This means that the function evaluations for DG2 are wasted.

### B. TSCC-CMAES compared to sep-CMA-ES

The sep-CMA-ES algorithm is different to the original CMA-ES as it ignores the off-diagonal component of the covariance matrix. This means the mutation step in sep-CMA-ES considers each variable separately with no inter-dependency with other variables. This is the simplest covariance matrix model for CMA-ES. The algorithm has been shown to have good performance on separable problems and problems with low degree of non-separability [18].

Surprisingly, sep-CMA-ES is not the best performing algorithm on  $f_1$ . The sep-CMA-ES is suitable for the problem (fully separable) and it did not use any function evaluation for grouping. However, it is greatly outperformed by SACC- and TSCC-based algorithms. The advantage of the algorithms that uses SACC and TSCC over sep-CMA-ES is the resource allocation. While the function did not include any weighting

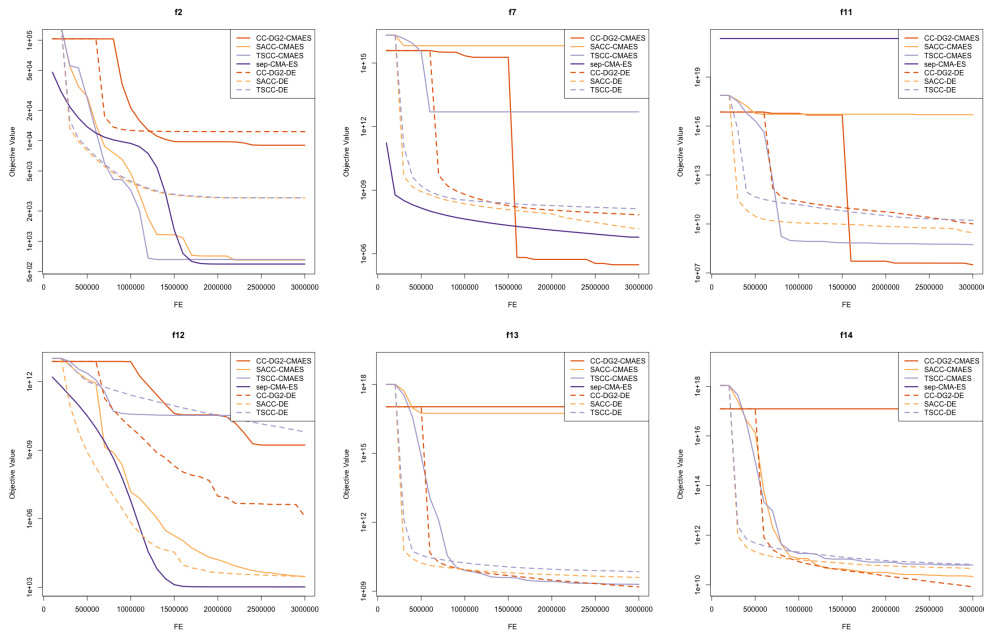


Fig. 5. Convergence plots for CEC2013 test problems. Convergence plots for some algorithms are not visible due to them having much worse performance.

factor, it is actually an imbalanced problem due to the power terms so resource allocation is crucial.

The sep-CMA-ES algorithm actually has the best performance on  $f_{12}$ , probably due to the weak links between variables, i.e. low degree of non-separability. Interestingly, the TSCC-CMAES is almost fully separable (large separable group and small non separable groups), so similar performance to sep-CMA-ES is expected; however, TSCC-CMAES performance is actually the worst for  $f_{12}$ . One might presume the bad performance is due to low grouping accuracy; however, it should be noted that CC-DG2-CMAES which has ideal grouping also has poor performance on the problem, while a completely wrong grouping (sep-CMA-ES considers all variables to be separable) gives significantly better results than the ideal grouping. This implies that the problem lies elsewhere. The performance drop is actually because the population size is directly tied to the group size so the non-separable subproblems are solved with CMA-ES with very small population. While CMA-ES is designed to work even with small population, larger population has been shown to significantly improve performance [28], [30], [31]. With  $2d$  population size and  $n = 1000$  TSCC-CMAES can reach objective values at the order of  $10^3$  on  $f_{12}$ .

The results on  $f_{13}$ - $f_{15}$  show the shortcoming of sep-CMA-ES (and CMA-ES in general) on large scale problems. The sep-CMA-ES as well as CC-DG2-CMAES fail on  $f_{13}$ - $f_{15}$  because the CMA-ES step-size diverges on all these functions. The step-size divergence is a common problem for CMA-ES as also reported in other numerical experiments, e.g. in [32], [33]. On some problems, the step-size keeps increasing to infinity and the algorithm cannot find a feasible solution except the initial samples. Varelas, et al. [33] suggest changing

the default cumulative step-size adaptation to the two-point step-size adaptation introduced in [34]. However, we tested another alternative: as the problem normally occurs on large scale problem, creating small groups of variables may also be a solution. By increasing the number of levels generated by the DA from 4 to 10, the performance of TSCC-CMAES improves by 10 orders of magnitude on  $f_{15}$ .

### C. Comparison in TACO

We compare the performance of SACC-CMAES and TSCC-CMAES with benchmark results available online in Toolkit for Automatic Comparison of Optimizers (TACO) available online at [tacolab.org](http://tacolab.org). The mean ranking of the algorithms for each test functions are presented in Tab. III.

The table shows that TSCC-CMAES is in general better than SACC-CMAES and, overall, the TSCC-CMAES performs well on problems based on Rastrigin's function. However, it is ranked low on the fully separable Ackley's function. Ackley's function is nearly flat and looks like a needle-in-a-haystack problem [35]. A guided search in such problem is likely to fail [36].

## VII. CONCLUSION

In this paper, a two-stage cooperative cooperation method is proposed. The variables are first grouped based on the result from a DA which requires  $\mathcal{O}(d)$  function evaluations and further grouped using DG2. DA inherently allows for efficient computation resource allocation. The two-stage grouping method is especially powerful when the problem is imbalanced; however, interactions between variables are not always detected. Separable variables, on the other hand, are correctly identified so solvers for separable problems can be used. In this

TABLE II  
PERFORMANCE COMPARISON OF THE ALGORITHMS ON CEC2013 TEST PROBLEMS

		CC-DG2-CMAES	SACC-CMAES	TSCC-CMAES	sep-CMA-ES	CC-DG2-DE	SACC-DE	TSCC-DE
f1	median	2.76E+07	4.02E-07	<b>7.36E-18</b>	1.63E-01	8.69E+01	4.09E-12	5.77E-12
	mean	2.88E+07	5.18E-07	7.48E-18	1.39E+00	1.01E+03	6.56E-12	1.11E-11
	std	6.89E+06	3.93E-07	2.73E-19	5.60E+00	3.69E+03	7.13E-12	1.31E-11
f2	median	9.06E+03	6.48E+02	6.62E+02	<b>5.92E+02</b>	1.23E+04	2.68E+03	2.66E+03
	mean	9.03E+03	6.50E+02	6.58E+02	5.93E+02	1.23E+04	2.70E+03	2.71E+03
	std	2.85E+02	2.41E+01	2.72E+01	2.64E+01	6.21E+02	1.86E+02	2.26E+02
f3	median	2.16E+01	2.15E+01	2.15E+01	2.16E+01	2.14E+01	2.12E+01	2.12E+01
	mean	2.16E+01	2.15E+01	2.15E+01	2.16E+01	2.14E+01	2.12E+01	2.12E+01
	std	5.46E-03	1.13E-02	1.13E-02	6.88E-03	1.40E-02	8.91E-03	1.19E-02
f4	median	1.63E+07	2.63E+08	<b>1.65E+06</b>	1.25E+10	5.09E+10	9.15E+09	3.83E+08
	mean	2.04E+07	2.70E+08	2.46E+06	1.31E+10	5.48E+10	9.39E+09	4.79E+08
	std	1.05E+07	4.28E+07	2.45E+06	2.71E+09	1.94E+10	3.40E+09	3.89E+08
f5	median	9.34E+05	6.43E+05	1.08E+06	<b>4.97E+05</b>	5.22E+06	5.66E+06	5.41E+06
	mean	9.03E+05	6.59E+05	1.12E+06	5.01E+05	5.26E+06	5.58E+06	5.35E+06
	std	1.15E+05	7.47E+04	1.89E+05	6.64E+04	4.66E+05	6.59E+05	2.83E+05
f6	median	1.06E+06	9.99E+05	<b>9.96E+05</b>	1.06E+06	1.06E+06	1.06E+06	1.06E+06
	mean	1.06E+06	1.02E+06	1.06E+06	1.06E+06	1.06E+06	1.06E+06	1.06E+06
	std	1.07E+03	3.07E+04	2.16E+04	1.46E+03	1.65E+03	1.20E+03	1.18E+03
f7	median	<b>3.14E+05</b>	6.55E+15	1.94E+07	6.01E+06	6.92E+07	1.09E+07	1.24E+08
	mean	2.99E+05	6.60E+15	4.95E+12	6.03E+06	6.83E+07	1.47E+07	1.32E+08
	std	7.96E+04	5.15E+15	2.48E+13	7.45E+05	2.49E+07	1.02E+07	9.49E+07
f8	median	4.06E+11	4.21E+12	<b>1.10E+09</b>	2.64E+14	4.48E+15	3.79E+13	6.69E+13
	mean	5.14E+11	4.31E+12	1.24E+09	2.67E+14	4.67E+15	3.84E+13	5.15E+14
	std	3.71E+11	8.34E+11	6.24E+08	7.74E+13	1.50E+15	2.02E+13	2.20E+15
f9	median	5.83E+07	8.85E+07	6.40E+07	<b>3.39E+07</b>	5.00E+08	2.92E+08	2.85E+08
	mean	5.84E+07	9.03E+07	6.75E+07	3.23E+07	5.01E+08	2.90E+08	2.71E+08
	std	1.24E+07	7.91E+06	1.19E+07	3.83E+06	2.49E+07	3.32E+07	6.35E+07
f10	median	9.44E+07	9.38E+07	<b>9.05E+07</b>	9.41E+07	9.46E+07	9.38E+07	9.43E+07
	mean	9.37E+07	9.29E+07	9.06E+07	9.41E+07	9.46E+07	9.38E+07	9.43E+07
	std	1.43E+06	1.60E+06	6.10E+04	1.12E+05	2.63E+05	2.20E+05	1.76E+05
f11	median	<b>3.15E+07</b>	3.07E+08	4.03E+08	2.06E+21	8.33E+09	3.47E+08	1.39E+09
	mean	3.17E+07	4.98E+16	5.51E+08	2.31E+21	1.01E+10	2.96E+09	1.67E+10
	std	1.30E+07	1.25E+17	4.32E+08	1.02E+21	8.79E+09	1.16E+10	2.56E+10
f12	median	1.37E+09	3.00E+03	2.45E+10	<b>1.04E+03</b>	8.00E+03	2.91E+03	5.05E+09
	mean	1.68E+09	2.97E+03	3.10E+10	1.04E+03	1.25E+06	2.95E+03	6.32E+09
	std	8.56E+08	3.06E+02	2.04E+10	1.22E+01	6.11E+06	2.21E+02	4.34E+09
f13	median	1.06245E+17	<b>5.95E+08</b>	1.72E+09	8.94E+16	1.52E+09	7.21E+08	7.40E+09
	mean	1.06245E+17	5.50E+16	1.99E+09	8.97E+16	1.54E+09	3.96E+09	7.06E+09
	std	0	1.92E+17	1.38E+09	1.82E+15	3.87E+08	6.61E+09	3.88E+09
f14	median	1.22809E+17	1.18E+10	3.56E+10	4.40E+18	6.76E+09	<b>8.75E+08</b>	4.00E+10
	mean	1.22809E+17	2.18E+10	6.29E+10	4.42E+18	8.28E+09	4.60E+10	6.64E+10
	std	0	2.94E+10	6.27E+10	1.31E+17	4.45E+09	1.03E+11	7.36E+10
f15	median	1.70625E+17	1.42E+18	1.42E+18	- <sup>a</sup>	1.70625E+17	<b>1.21E+17</b>	1.32E+17
	mean	1.70625E+17	1.74E+18	1.74E+18	- <sup>a</sup>	1.70625E+17	1.42E+17	1.51E+17
	std	0	1.18E+18	1.18E+18	- <sup>a</sup>	0	7.74E+16	1.03E+17

<sup>a</sup>The algorithm never finish successfully.

TABLE III  
FINAL-RESULT MEAN-RANK COMPARISON USING THE TACO PLATFORM

	Algorithm-Rank						
	CC-RDG3	DGSC	MLSHADE-SPA	MOS	SACC-CMA-ES	SHADEILS	TSCC-CMA-ES
F01	3	7	3	3	6	3	3
F02	7	4	1	5	2	6	3
F03	4	5	1	2	7	3	6
F04	1	6	7	4	5	3	2
F05	5	6	4	7	1	3	2
F06	3	7	1	2	6	5	4
F07	1	5	4	3	7	2	6
F08	1	7	6	5	4	3	2
F09	3	6	4	7	2	5	1
F10	4	7	1	2	6	5	3
F11	1	6	3	4	7	2	5
F12	4	5	2	3	6	1	7
F13	1	5	4	3	7	2	6
F14	5	4	2	3	6	1	7
F15	2	4	5	3	7	1	6
Mean	3.000	5.600	3.200	3.733	5.267	3.000	4.200

work, DG2 was used, but faster secondary grouping methods such as the RDG should be considered if the DSM is not needed.

The two-stage grouping method can outperform SACC when paired with CMA-ES. CMA-ES is rotationally invariant so variable interaction will not be exploited by the two-stage grouping; instead, TSCC benefit from smaller group sizes which help CMA-ES to converge more rapidly. Additionally, because separable variables are recognized, sep-CMA-ES can be used to solve the separable subproblem. A small test has proven that population-size and group-size tuning can greatly improve performance. An adaptive parameters tuning is likely to improve TSCC-CMAES performance. Lastly, the step-size divergence that is frequently observed on large-scale CMA-ES can be prevented by setting small group sizes.

Future work should also consider multiobjective optimization problems, where large scale optimization can be defined not only on the number of decision variables, but also based on the number of objective functions and disciplines involved. First works on the topic has been carried out in [37], [38].

#### REFERENCES

- [1] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature*, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds. Springer Berlin, 1994, pp. 249–257.
- [2] Y. Sun, M. Kirley, and S. K. Halgamuge, "A recursive decomposition method for large scale continuous optimization," *Transactions on Evolutionary Computation*, vol. 22, no. 5, pp. 647–661, 2018.
- [3] W. Chen and K. Tang, "Impact of problem decomposition on cooperative coevolution," in *Congress on Evolutionary Computation*. IEEE, 2013, pp. 733–740.
- [4] R. Salomon, "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms," *Biosystems*, vol. 39, no. 3, pp. 263 – 278, 1996.
- [5] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 378–393, 2014.
- [6] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, "Dg2: A faster and more accurate differential grouping for large-scale black-box optimization," *Transactions on Evolutionary Computation*, vol. 21, no. 6, pp. 929–942, 2017.
- [7] Y. Sun, X. Li, A. Ernst, and M. N. Omidvar, "Decomposition for large-scale optimization problems with overlapping components," in *Congress on Evolutionary Computation*. IEEE, 2019, pp. 326–333.
- [8] W. Dong, T. Chen, P. Tiño, and X. Yao, "Scaling up estimation of distribution algorithms for continuous optimization," *Transactions on Evolutionary Computation*, vol. 17, no. 6, pp. 797–822, 2013.
- [9] J. Liu and K. Tang, "Scaling up covariance matrix adaptation evolution strategy using cooperative coevolution," in *Intelligent Data Engineering and Automated Learning*, H. Yin, K. Tang, Y. Gao, F. Klawonn, M. Lee, T. Weise, B. Li, and X. Yao, Eds. Springer Berlin, 2013, pp. 350–357.
- [10] M. N. Omidvar, B. Kazimpour, X. Li, and X. Yao, "Cbcc3 — a contribution-based cooperative co-evolutionary algorithm with improved exploration/exploitation balance," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, July 2016, pp. 3541–3548.
- [11] S. Mahdavi, S. Rahnamayan, and M. E. Shiri, "Cooperative co-evolution with sensitivity analysis-based budget assignment strategy for large-scale global optimization," *Appl. Intell.*, vol. 47, no. 3, pp. 888–913, 2017.
- [12] —, "Multilevel framework for large-scale global optimization," *Soft Comput.*, vol. 21, no. 14, pp. 4111–4140, 2017.
- [13] Zhenyu Yang, Ke Tang, and Xin Yao, "Multilevel cooperative coevolution for large scale optimization," in *World Congress on Computational Intelligence*. IEEE, 2008, pp. 1663–1670.
- [14] M. El-Abd, "Hybrid cooperative co-evolution for large scale optimization," in *Symposium on Swarm Intelligence*. IEEE, 2014, pp. 1–6.
- [15] Zhenyu Yang, Ke Tang, and Xin Yao, "Self-adaptive differential evolution with neighborhood search," in *World Congress on Computational Intelligence*. IEEE, 2008, pp. 1110–1116.
- [16] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes University, Engineering Faculty, Tech. Rep., 2005.
- [17] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.
- [18] R. Ros and N. Hansen, "A simple modification in cma-es achieving linear time and space complexity," in *Parallel Problem Solving from Nature*, G. Rudolph, T. Jansen, N. Beume, S. Lucas, and C. Poloni, Eds. Springer Berlin, 2008, pp. 296–305.
- [19] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *Trans. Math. Softw.*, vol. 42, no. 2, pp. 13:1–13:24, 2016.
- [20] I. Sobol, "Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates," *Math. Comput. Simulat.*, vol. 55, no. 1, pp. 271 – 280, 2001.
- [21] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. R. Stat. Soc. B*, vol. 58, no. 1, pp. 267–288, 1996.
- [22] M. D. Morris, "Factorial sampling plans for preliminary computational experiments," *Technometrics*, vol. 33, no. 2, pp. 161–174, 1991.
- [23] F. Campolongo, J. Cariboni, A. Saltelli, and W. Schoutens, "Enhancing the morris method," in *Sensitivity Analysis of Model Output*, 2005, pp. 369–379.
- [24] X. Tong, B. Yuan, and B. Li, "Model complex control cma-es," *Swarm Evol. Comput.*, vol. 50, p. 100558, 2019.
- [25] X. Li, K. Tang, M. N. Omidvar, Z. Yang, and K. Qin, "Benchmark functions for the cec'2013 special session and competition on large-scale global optimization," 2013.
- [26] H. Liu, Y. Wang, X. Liu, and S. Guan, "Empirical study of effect of grouping strategies for large scale optimization," in *International Joint Conference on Neural Networks*, 2016, pp. 3433–3439.
- [27] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *Trans. Math. Softw.*, vol. 42, no. 2, pp. 13:1–24, 2016.
- [28] N. Hansen and S. Kern, "Evaluating the cma evolution strategy on multimodal test functions," in *Parallel Problem Solving from Nature*, X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiño, A. Kabán, and H.-P. Schwefel, Eds. Springer Berlin, 2004, pp. 282–291.
- [29] A. Ahrari and M. Shariat-Panahi, "An improved evolution strategy with adaptive population size," *Optimization*, vol. 64, no. 12, pp. 2567–2586, 2015.
- [30] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evol. Comput.*, vol. 11, no. 1, p. 1–18, 2003.
- [31] Y. Jin, M. Olhofer, and B. Sendhoff, "On evolutionary optimization of large problems using small populations," in *Advances in Natural Computation*, L. Wang, K. Chen, and Y. S. Ong, Eds. Springer Berlin, 2005, pp. 1145–1154.
- [32] I. Loshchilov, "A computationally efficient limited memory cma-es for large scale optimization," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 397–404.
- [33] K. Varelas, A. Auger, D. Brockhoff, N. Hansen, O. A. ElHara, Y. Semet, R. Kassab, and F. Barbaresco, "A comparative study of large-scale variants of cma-es," in *Parallel Problem Solving from Nature*, A. Auger, C. M. Fonseca, N. Lourenço, P. Machado, L. Paquete, and D. Whitley, Eds. Springer Cham, 2018, pp. 3–15.
- [34] N. Hansen, "Cma-es with two-point step-size adaptation," 2008.
- [35] A. Auger and N. Hansen, "A restart cma evolution strategy with increasing population size," in *Congress on Evolutionary Computation*, vol. 2. IEEE, 2005, pp. 1769–1776 Vol. 2.
- [36] F. Rothlauf, *Optimization Problems*. Springer Berlin, 2011, pp. 7–44.
- [37] L. M. Antonio and C. A. C. Coello, "Use of cooperative coevolution for solving large scale multiobjective optimization problems," in *Congress on Evolutionary Computation*. IEEE, 2013, pp. 2758–2765.
- [38] H. Zille, H. Ishibuchi, S. Mostaghim, and Y. Nojima, "A framework for large-scale multiobjective optimization based on problem transformation," *Trans. on Evol. Comput.*, vol. 22, no. 2, pp. 260–275, 2017.