Refining the CC-RDG3 Algorithm with Increasing Population Scheme and Persistent Covariance Matrix

Dani Irawan[1(✉)], Margarita Antoniou[2], Boris Naujoks[1], and Gregor Papa[2]

[1] Institute for Data Science Engineering and Analytics, TH Köln, Gummersbach, Germany
{dani.irawan,boris.naujoks}@th-koeln.de

[2] Jožef Stefan Institute, Ljubljana, Slovenia
{margarita.antoniou,gregor.papa}@ijs.si

Abstract. The cooperative coevolution framework has been used extensively to solve large scale global optimization problems. Recently, the framework is used in CC-RDG3 where it uses recursive differential grouping and covariance matrix adaptation evolution strategies (CMA-ES). It was shown that the algorithm performs well on the CEC2013-LSGO benchmark functions. In this study, some modifications to the CC-RDG3 algorithm are proposed to improve performance. The modifications should be applied differently depending on the modality of the problem at hand.

Keywords: Cooperative coevolution · Large scale optimization · Evolutionary algorithms

1 Introduction

The cooperative coevolution (CC) framework [15] is a popular framework for solving large scale global optimization (LSGO) problems. The framework uses a divide-and-conquer concept where the large scale problem is decomposed into smaller problems with fewer variables, that are further optimized. However, the decomposition step in using the CC framework is still a challenge despite the various decomposition methods that have been proposed before.

One of the most popular decomposition schemes is the differential grouping (DG) [13,14] and its family, such as the extended DG (XDG) [19], and the recursive DG (RDG) and RDG3 [17,18], which decomposes the problem based on variable interaction. The variable interactions are detected based on the second-order differentials. The rationale behind these schemes is that tightly-interacting variables should be in the same group while interactions among distinct subcomponents should be weak [4]. Some algorithms indeed rely on separability between the subproblems and their performance may deteriorate if the decomposition produces bad grouping [16].

B. Filipič et al. (Eds.): BIOMA 2020, LNCS 12438, pp. 69–83, 2020.
https://doi.org/10.1007/978-3-030-63710-1_6

Once the problem is decomposed, the optimal values of the subproblems should be found by an optimizer. Many evolutionary algorithms (EAs) have been used as optimizers in the context of the CC framework for LSGO. The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm proposed in [8], is an evolution strategy that relies on the adaptation of the full covariance matrix of a normal search distribution. This algorithm performs well on unimodal functions, but its performance deteriorates in multimodal functions. To tackle this problem, Auger et al. [1] suggested a CMA-ES with an increasing population (IPOP-CMA-ES), where the algorithm adopts a restart strategy with successively increasing the population size giving promising results.

The CMA-ES has been used together with RDG3 within the CC framework and named as the CC-RDG3 algorithm. In this work, we refine the CC-RDG3 algorithm that uses a standard CMA-ES optimizer, by using IPOP-CMA-ES. Furthermore, instead of a complete restart of the CMA-ES in every cycle, we use a persistent covariance matrix instead.

Another important aspect after the decomposition and during the optimization is the budget allocation. The simplest method in this context is, after the problem is decomposed, to use a round-robin method to assign the computational time equally to each subproblem, ignoring the different effects that each subproblem can have to the general problem. The contribution based budget allocation CC (CBCC) [12] and CC with a sensitivity analysis-based budget assignment method (SACC) [10] investigate the influence of each subcomponent and allocate accordingly the number of iterations for the optimization. In this study, the SACC method is also tested.

The combinations of the various modifications are tested on numerous test-functions from standard LSGO benchmark suites and compared with the base CC-RDG3 algorithm. The results of each combination vary and depend hugely on the characteristics of the test problem, especially on the modality.

The remainder of this paper is organized as follows. Section 2 contains a short description of the CC framework and the RDG3 decomposition method used. Section 3 explains the proposed refinement of the CC-RDG3 algorithm. Section 4 presents the numerical experiments and the benchmark used, the obtained results along with their comparison and analysis. Lastly, Sect. 5 concludes this paper and shows future directions.

2 Cooperative Coevolution with Recursive Differential Grouping

CC framework was first proposed by [15] in 1994. The main two steps of the general CC framework can be summarized as follows: 1. *Decomposition*: Decompose the problem into several subproblems, by dividing a given high-dimensional problem into a number of low-dimensional subcomponents and 2. *Optimisation*: Optimise each subproblem cooperatively with the use of an optimizer.

The existing decomposition methods are classified by [18] as manual or automatic (or blind and intelligent as proposed later in [20] as more appropriate terminology). The manual (or blind) decomposition method ignores the underlying structure of variable interaction, and the number and the size of the subcomponents are manually designed. Examples of such methods is uni-variable grouping [15], S_k grouping [2] and random grouping [21], and have been proved to work well in fully separable problems. In the automatic decomposition, the variable interactions are identified and the problem is decomposed accordingly.

The Recursive Differential Grouping (RDG) is one of the most effective automatic methods, capable of quickly grouping variables based on interaction. The grouping is done recursively and requires $\mathcal{O}(d \log d)$ function evaluations. There are several versions of RDG and the most recent is RDG3 [17]. Compared to previous versions, the RDG3 scheme puts emphasis on handling overlapping variables. The differential grouping schemes usually put groups with overlapping variables into a single, big group. This means that there are many variables that are not directly interacting (also termed "weak interactions") in the group.

In RDG3, when groups have overlapping variables, a size-limit-threshold is imposed. When the threshold is exceeded, no further overlapping variables are grouped together. This allows some overlapping variables to be grouped together, while also preventing the groups to grow too big. A small size-threshold will prevent variables with weak interactions from being grouped together, while a larger size-threshold will allow more weak interactions.

The RDG3 has been used in the CC framework in CC-RDG3 [17], paired with the covariance matrix adaptation-evolution strategy (CMA-ES) [8] as the solver. The algorithm shows exceptional results on the CEC2013 problems for LSGO [9], especially on overlapping problems.

3 Proposed Algorithm

The proposed modifications to the CC-RDG3 algorithm are described in this section. Each modification can be applied separately.

3.1 CMA-ES with Increasing Population

The CMA-ES algorithm explore the search space using the multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$. The search at generation $g+1$ follows the equation

$$x^{(g+1)} = \mathcal{N}(x^{(g)}, (\sigma^{(g)})^2 \mathbf{C}^{(g)}), \tag{1}$$

where $x^{(g+1)}$ is the offspring, $x^{(g)}$ is the current best point, while $\sigma^{(g)}$ and $\mathbf{C}^{(g)}$ are the step size (scaling factor) and the covariance matrix at current generation g, respectively. The CMA-ES adapts the σ and $\mathbf{C}$.

The performance of CMA-ES on multi-modal functions depends strongly on population size [7]. To address this, Auger, et al. [1] proposed a restart strategy with increasing population. When some stopping criteria is triggered, the CMA-ES is restarted and the population size is increased hence promoting exploration of the search space. In this work, the same stopping criteria as in [1] are used, except that the *equalfunvalhist* stopping criterion only check for flat fitness.

In regards with the CC framework, when any stopping criteria is triggered, the optimization on the current group is stopped and it will be restarted in the next cycle with double population size up to 8 times the original size (see Algorithm 1). The size limit is imposed to prevent the population size growing too large. When the size limit is reached, the step size σ is doubled instead. For brevity, algorithms that use the IPOP-CMA-ES strategy will be marked with "IPOP" in the name.

Algorithm 1. CC-RDG3-IPOP

```
$G \leftarrow$ Group variables using RDG3
Set initial population size $\lambda = \lambda_0$ and CMA-ES step size $\sigma = \sigma_0$
while Budget still available do
    for $i = 1 : |G|$ do
        Use CMA-ES with step size $\sigma$, other parameters at default on $f(\mathbf{x}_i)$
        Update $\mathbf{x}_i$
        Check termination code
        if CMA-ES terminated due to IPOP restart criterion then
            if $\lambda \leq \lambda^8$ then $\lambda \leftarrow 2\lambda$
            else $\sigma \leftarrow 2\sigma$
            end if
        end if
    end for
end while
```

3.2 Persistent Covariance Matrix

The CMA-ES algorithm will continuously adapt the covariance matrix, step size, and also records the evolution path through cumulation. Every time the algorithm is restarted, these information are usually lost and only the initial values of $\mathbf{x}$ are updated. With regards to the CC framework, a restart would happen after each cycle finishes.

We propose to use a persistent covariance matrix and step size. Persistent means that the covariance matrix, step size, and also the evolution path are not reset at each restart (see Algorithm 2). All values are retained and the next restart will start with these values. The function landscape may change after each cycle, but the information retained may help to kick-start the optimization in the subsequent cycles. The procedure will promote exploitation of potential areas in the search space.

Due to the conflicting nature between the persistent covariance matrix strategy against the IPOP-CMA-ES strategy, they are set to be mutually exclusive when used together (see Algorithm 3). The covariance matrix (and other values) are only retained if the stopping criteria in Sect. 3.1 are not triggered and the CMA-ES ends because it reaches maximum number of iterations. When any stopping criteria in Sect. 3.1 is triggered, the IPOP-CMA-ES will be used instead. Algorithms that use the persistent covariance strategy are marked with "KC" (keep covariance) in the name.

Algorithm 2. CC-RDG3-KC

```
$G \leftarrow$ Group variables using RDG3
Set initial CMA-ES step size $\sigma = \sigma_0$, and covariance matrix $\Lambda = \mathbb{1}$
while Budget still available do
    for $i = 1 : |G|$ do
        Use CMA-ES with $\sigma$ and $\Lambda$, other parameters at default on $f(\mathbf{x}_i)$
        Update $\mathbf{x}_i, \sigma$, and $\Lambda$
    end for
end while
```

Algorithm 3. CC-RDG3-IPOP-KC

```
$G \leftarrow$ Group variables using RDG3
Set $\lambda = \lambda_0$, $\sigma = \sigma_0$, and $\Lambda = \mathbb{1}$
while Budget still available do
    for $i = 1 : |G|$ do
        Use CMA-ES with step size $\sigma$, other parameters at default on $f(\mathbf{x}_i)$
        Update $\mathbf{x}_i$
        Check termination code
        if CMA-ES terminated due to IPOP restart criterion then
            if $\lambda \leq \lambda^8$ then $\lambda \leftarrow 2\lambda$
            else $\sigma \leftarrow 2\sigma$
            end if
        else
            Update $\sigma$, and $\Lambda$
        end if
    end for
end while
```

3.3 Sensitivity Analysis Based Budget Allocation

Equation 2 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_2 (10^4 times larger).

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

The differential analysis (DA), also known as Morris method, is a sensitivity analysis (SA) method based on the first order differential. Sensitivity analysis methods assess the extent of the variables' effect on the objective function. The DA has been used previously for LSGO problem in [10,11].

For DA, the search space is divided into p intervals in each variable. A grid jump $\Delta = N * \frac{1}{(p-1)}$, with $N \in \mathbb{Z}_{>0} < p - 1$. Elementary effect (EE) for each variable can then be calculated using Eq. 3

$$EE_j(\mathbf{x}) = \frac{f(x_1, \ldots, x_{j-1}, x_j + \Delta, \ldots) - f(\mathbf{x})}{\Delta}, j = 1, ..., d \tag{3}$$

The $\mathbf{x}$ is picked randomly within the search space such that $\mathbf{x} + \Delta$ is still within the search space. Several EE_j are sampled with sample size r. The distributions of EE_j can be obtained. Further, we compute the mean of the absolute value of EE_j, μ^*, to rank the importance of each variable following Eq. 4, with s the sample number. Higher μ^* signifies higher impact/contribution to the objective value [3]. The budget allocated to a group can then be allocated based on μ^*. In this work, the portion p_s for group s follows Eq. 5.

$$\mu_j^* = \frac{\sum_{s=1}^{r} |EE_j(\mathbf{x})|_s}{r} \tag{4}$$

$$p_s = \begin{cases} 1 + \log \sum_{i \in S} \mu_i^*, & \text{if} \sum_{i \in s} \mu_i^* > 1 \\ 1, & \text{otherwise} \end{cases} \tag{5}$$

In [8], the maximum number of iterations is set at $30 \times d$. In this study, d is the number of variables in the main problem (without decomposition). The p_s is used to scale the number of iteration for each group with respect to $30 \times d$, i.e. each group will have a budget of $30p_s \times d$ in each cycle. Algorithms that use the sensitivity analysis budget allocation strategy will be marked with "SA" in the name. Algorithms without "SA" assume μ^* for all variables are equal to 1.

4 Numerical Experiments

4.1 Setup of Experiments

To analyze the performance of the proposed algorithms, we compared the algorithms with the base CC-RDG3 algorithm. For each function, all algorithms are run 15 times and compared to the CC-RDG3 algorithm using the pairwise Wilcoxon test.

The test functions used in this study are a subset of the CEC2013 LSGO benchmark suite [9] $f_1 - f_{14}$. Problem f_{15} is omitted from this study because the algorithm implementation used in this study cannot find a feasible solution, most likely due to step size divergence. The problems use 1 000 input variables, except $f_{13} - f_{14}$ with only 905 variables. The budget is set at 3 000 000 function evaluations for each run for these functions.

Moreover, the test functions $f_{16} - f_{19}$ and $f_{21} - f_{24}$ from BBOB-largescale benchmark suite [5] are used to further assess the algorithms' performances on multimodal functions. The BBOB benchmark functions are configured to accept 160 input variables and each optimization run cannot use more than 1 600 000 function evaluations. In Table 1, the test functions and their properties are reported. Note that we keep the original numbering of each benchmark suite.

4.2 Numerical Results

Performances of the algorithms on the test problems can be observed in Table 2 and Table 3 and boxplots Fig. 1 to Fig. 4. For the boxplots, the data ranges are normalized to the range [0,1]. Due to the normalization, small differences may be exaggerated, and vice versa. Additionally, in Table 2 it can be seen that for f_3,

Table 1. Test Functions and their properties. Properties listed are modality (U = Unimodal, M = Multimodal), additive separability, number of input variables d, and special features of the functions.

CEC2013	**Modality**	**Add. Sep.**	d	**Features**
f_1: Elliptic	U	Separable	1 000	
f_2: Rastrigin	M	Separable	1 000	
f_3: Ackley	M	Separable	1 000	
f_4: 7 Elliptic	U	Partial	1 000	
f_5: 7 Rastrigin	M	Partial	1 000	
f_6: 7 Ackley	M	Partial	1 000	Deceptive
f_7: 7 Schwefel 1.2	U	Partial	1 000	
f_8: 20 Elliptic	U	Partial	1 000	
f_9: 20 Rastrigin	M	Partial	1 000	
f_{10}: 20 Ackley	M	Partial	1 000	Deceptive
f_{11}: 20 Schwefel 1.2	U	Partial	1 000	
f_{12}: Rosenbrock	M	Non-Separable	1 000	Overlapping
f_{13}: Schwefel 1.2	U	Non-separable	905	Overlapping, conforming
f_{14}: Schwefel 1.2	U	Non-separable	905	Overlapping, conflicting
BBOB	**Modality**	**Add. Sep.**	d	**Features**
f_{16}: Weierstrass	M	Non-separable	160	
f_{17}: Schaffers	M	Non-separable	160	
f_{18}: Schaffers	M	Non-separable	160	Ill conditioned
f_{19}: Griewank-Rosenbrock	M	Non-separable	160	
f_{21}: Gallagher's 101 Peaks	M	Non-separable	160	
f_{22}: Gallagher's 21 Peaks	M	Non-separable	160	
f_{23}: Katsuura	M	Non-separable	160	
f_{22}: Lunacek bi-rastrigin	M	Non-separable	160	

f_6 and f_{10} (Ackley functions), all algorithms have similar performances which are not far off from their starting points. This is because the Ackley function has a landscape similar to the needle-in-haystack problem where directed search strategies are expected to fail [1].

From Table 2 and Table 3, it can be seen when an algorithm with SA strategy performs well, the corresponding algorithm without SA strategy also shows a significant advantage over CC-RDG3. The SA strategy does not provide a significant improvement to the algorithms.

On unimodal functions, the KC strategy shows its superiority. In Fig. 1, Fig. 3, and Fig. 4, the CC-RDG3, and CC-RDG3-IPOP algorithms perform much worse on all unimodal functions. The KC strategy will consistently push the search to a local optima wherein in unimodal functions, any local optima is also a global optimum. Combined with the high grouping accuracy of RDG3, the performance of these algorithms on unimodal functions will be boosted.

However, on the highly multimodal f_2, f_5 and f_9 functions (see Fig. 2), the RDG-KC and RDG-KC-SA algorithms are not performing so well. On these

Table 2. Median of the best values obtained by the algorithms on CEC2013 test problems. Each algorithm is run 15 times on each function. Bold texts indicate the best results, ● indicates better performance than the base CC-RDG3 algorithm, while ▽ indicates worse performance.

Fn	CC-RDG3	IPOP	KC	IPOP-KC	KC-SA	IPOP-KC-SA
f_1	1.16+07	1.16E+07	**3.68+05** ●	**3.68+05** ●	**3.68+05** ●	**3.68+05** ●
f_2	6.01E+03	**1.28E+03** ●	6.21E+03	**1.28E+03** ●	6.21E+03	**1.28E+03** ●
f_3	2.05E+01	2.05E+01	2.04E+01	2.04E+01	2.04E+01	2.04E+01
f_4	1.61E+08	1.49E+08	**1.41E+06**●	3.64E+06 ●	1.42E+06 ●	3.330E+06 ●
f_5	1.66E+06	5.17E+05 ●	2.46E+06 ▽	4.90E+05 ●	2.52E+06 ▽	**4.74+05** ●
f_6	1.00E+06	1.01E+06	9.96E+05 ●	9.96E+05 ●	9.96E+05 ●	9.96E+05 ●
f_7	1.44E+04	1.84E+04	**2.06E-19** ●	2.07E-07●	4.13E-18 ●	2.62E-07 ●
f_8	1.23E+12	3.20E+12 ▽	2.03E+06 ●	3.09E+07 ●	**1.73E+06** ●	3.81E+07 ●
f_9	1.18E+08	**4.40E+07**●	1.54E+08 ▽	4.87E+07 ●	1.70E+08 ▽	4.56E+07 ●
f_{10}	9.12E+07	9.12E+07	9.05E+07 ●	9.05E+07 ●	9.05E+07 ●	9.05E+07 ●
f_{11}	1.99E+08	2.05E+08	1.56E+00 ●	1.07E+02 ●	**1.54E+00** ●	1.06E+01 ●
f_{12}	1.57E+03	1.58E+03	9.50E+02 ●	8.93E+02 ●	8.54E+02 ●	**8.42E+02** ●
f_{13}	4.57E+09	4.32E+09	1.90E+05 ●	3.33E+06 ●	**1.77E+05** ●	1.76E+06 ●
f_{14}	2.27E+09	2.56E+09	1.36E+08 ●	2.92E+08 ●	**1.09E+08** ●	2.74E+08 ●

Table 3. Median of the difference-to-optimum values obtained by the algorithms on BBOB-largescale test problems. Each algorithm is run 15 times on each function. Bold text indicates the best performance. Bold texts indicate the best results, ● indicates better performance than the base CC-RDG3 algorithm, while ▽ indicates worse performance.

Fn	CC-RDG3	IPOP	KC	IPOP-KC	KC-SA	IPOP-KC-SA
f_{16}	**7.404E-01**	8.113E-01	1.258E+00 ▽	1.347E+00 ▽	1.318E+00 ▽	1.200E+00 ▽
f_{17}	**3.707E-01**	4.062E-01	1.173E+00 ▽	8.908E-01 ▽	1.043E+00 ▽	1.175E+00 ▽
f_{18}	1.778E+00	**1.737E+00**	4.067E+00 ▽	3.798E+00 ▽	3.162E+00 ▽	2.944E+00 ▽
f_{19}	2.503737e-01	2.503737e-01	1.659E-02 ●	**1.659E-02** ●	1.659E-02 ●	2.291E-02 ●
f_{21}	2.922E-08	**1.198E-08**	6.730E+00 ▽	1.451E-08	6.740E+00 ▽	1.340E-08
f_{22}	2.596E+00	3.299E+00	4.640E+00	2.448E+00	**2.438E+00**	3.114E+00
f_{23}	1.897E-02	**1.884E-02**	2.138E-02	2.452E-02 ▽	2.392E-02	2.040E-02
f_{24}	9.999E+01	**9.316E+01**	1.511E+02 ▽	1.445E+02 ▽	1.479E+02 ▽	1.445E+02 ▽

functions, the KC strategy will likely lead to early convergence which may trap the search at local optima. This can be observed in Fig. 7 for f_9 where the algorithms with KC strategy become flat very early. In multimodal functions, algorithms with IPOP strategy show better performances. This indicates that the observation in [7] holds true in large scale settings, a larger population will improve CMA-ES performance on multimodal functions.

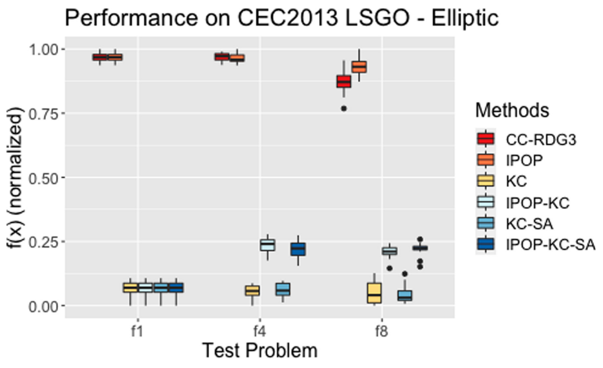

Fig. 1. Boxplot of the best values obtained on CEC2013 elliptic test problems.

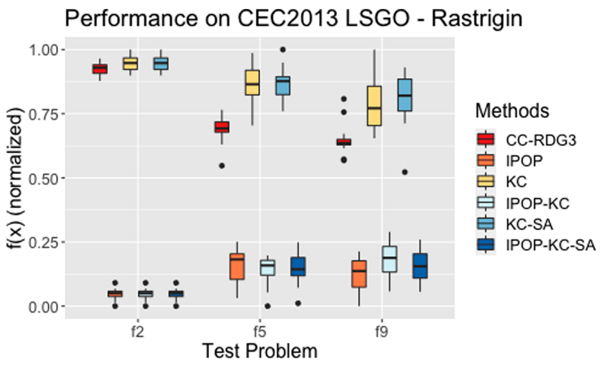

Fig. 2. Boxplot of the best values on CEC2013 test problems based on Rastrigin function.

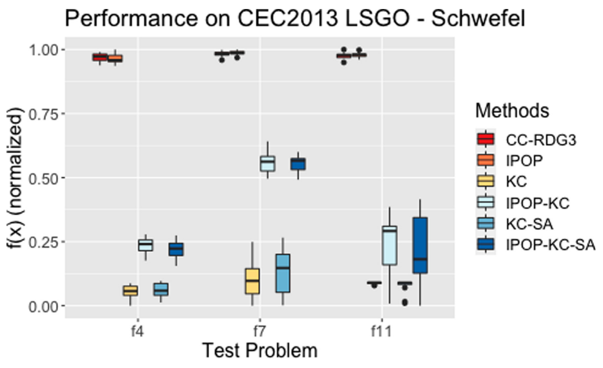

Fig. 3. Boxplot of the best values obtained on CEC2013 test problems based on Schwefel function.

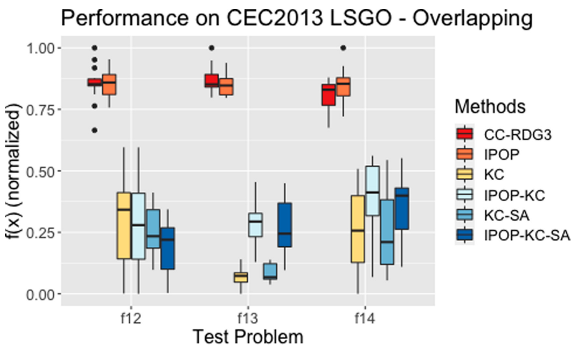

Fig. 4. Boxplot of the best values obtained on CEC2013 test problems with overlapping variables.

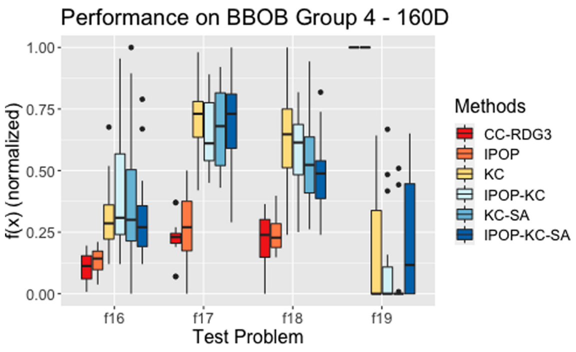

Fig. 5. Boxplot of the distance-to-optimum values obtained by the algorithms on BBOB-largescale test problems with adequate global structure.

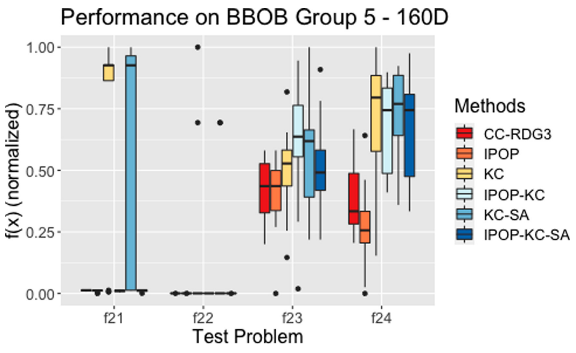

Fig. 6. Boxplot of the distance-to-optimum values obtained by the algorithms on BBOB-largescale test problems with weak global structure.

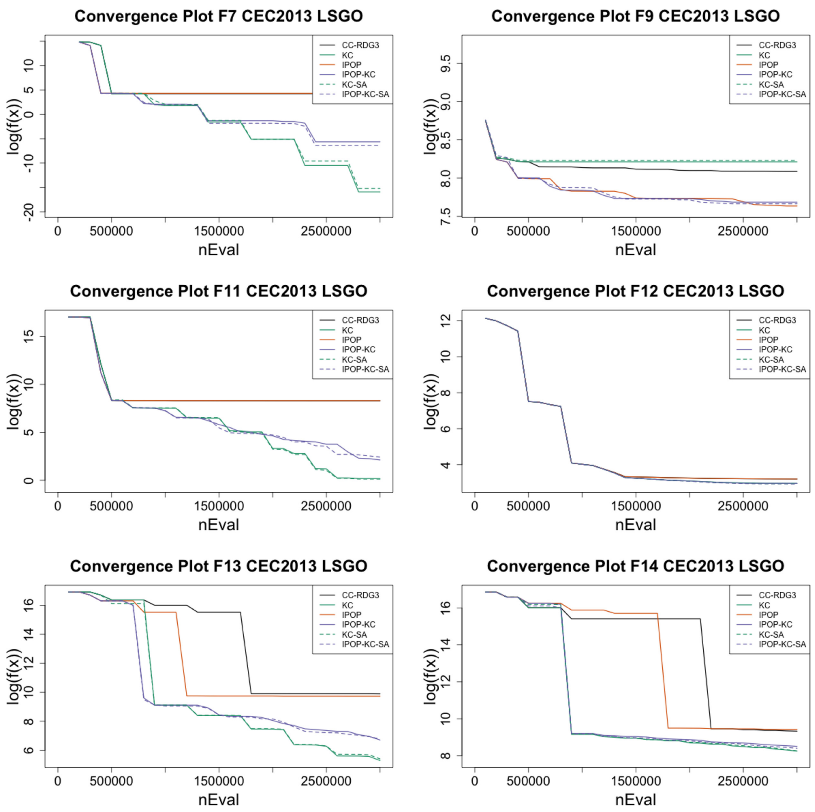

Fig. 7. Convergence plots for CEC2013 test problems. Each line is the mean achieved values for different algorithms after certain numbers of function evaluations.

The test results presented in Table 3 further confirm the dread of KC strategy and potency of IPOP strategy for multimodal functions. In most of the multimodal BBOB functions, the IPOP strategy has an advantage over the KC strategy. However, unlike on the f_3, f_5, and f_{10}, the improvement obtained from the IPOP strategy is insignificant on the BBOB problems. This may be because the restart is triggered too late and a too small budget to see the effect of increasing population. In a similar study for smaller problems, Hansen [6] used CMA-ES with a different population adaptation scheme called BIPOP-CMA-ES. The study in [6] uses more stopping criteria (hence it may stop earlier) and number of function evaluations up to $3 \times 10^5 d$.

Looking at Fig. 6, the CC-RDG3, and CC-RDG3-IPOP algorithms seem to perform terribly on f_{19}. If we look into Table 3, although they are indeed worse, the distance-to-optimum value on both algorithms are actually very low. However, we can still analyze why it performs worse than other algorithms.

By assessing the convergence history, we found that the two algorithms cannot find better solutions than the initial samples, hence the flat line in Fig. 8

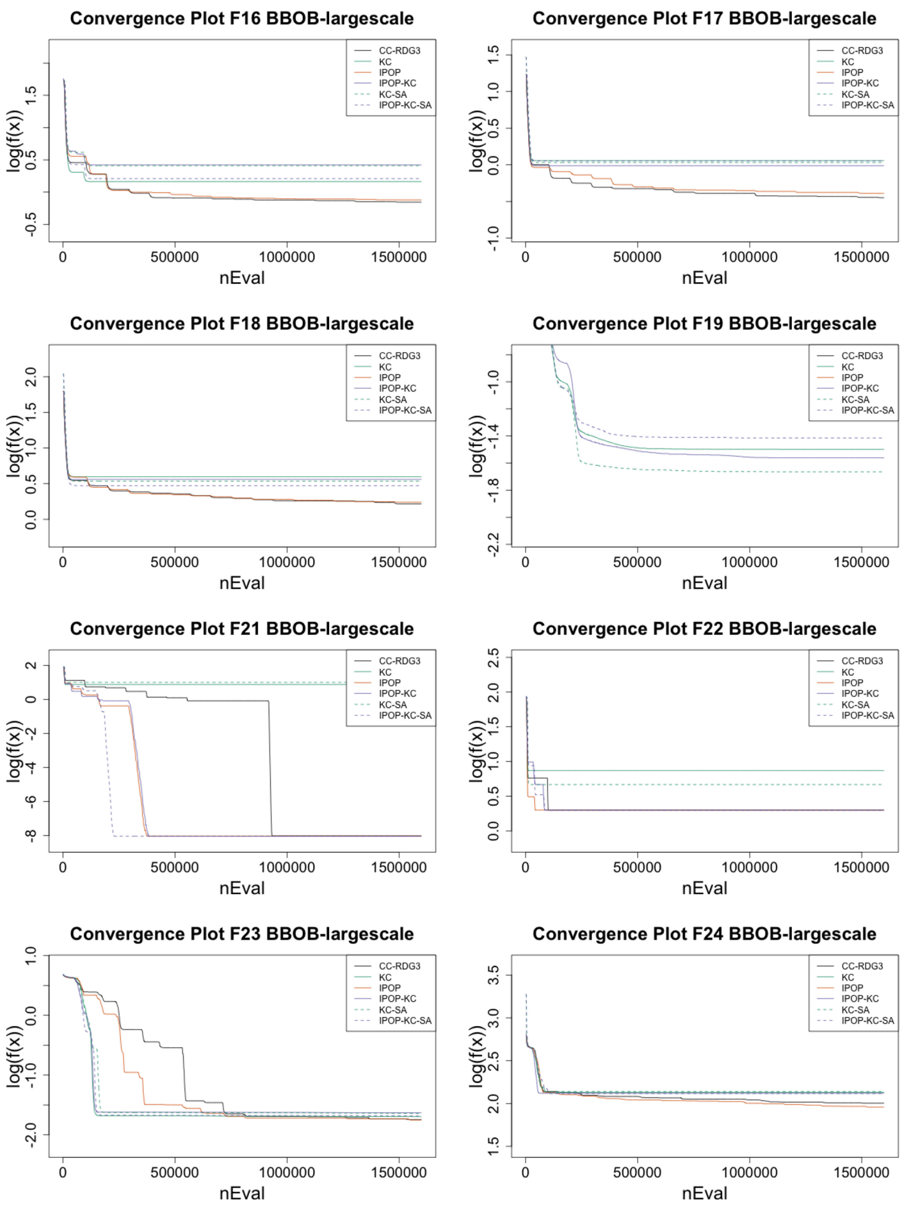

Fig. 8. Convergence plots for BBOB test problems. Each line is the mean distance-to-optimum values for different algorithms after certain numbers of function evaluations.

for f_{19}. In such a case, the search is restarted from the same initial point and the CMA-ES is also restarted with the same covariance matrix and step size as the previous cycle, repeating a failed search over and over. The problem with such restart is that the step size resets to a large value while what is needed in this case is a local search. The IPOP strategy also promotes a more global search instead of a local search hence the CC-RDG3-IPOP also does not perform well. On the other hand, with the introduction of the KC strategy, the step size will normally decrease in every cycle leading to a local search. The KC strategy clearly improves performance in such cases. However, the risk of early convergence to local optima still holds for the KC strategy.

In general, control over whether the search should be local or global is crucial in solving multimodal function. The two strategies provide a way to control it. The IPOP strategy will lead to a more global search, while the KC strategy will lead to a local search. To take full advantage of the strategies, a fitness landscape analysis can be conducted before choosing the strategies.

5 Conclusion and Future Work

In this study, three strategies to improve the CC-RDG3 algorithm are proposed and tested: persistent covariance, increasing population, and budget allocation based on sensitivity analysis. The budget allocation based on sensitivity analysis does not seem to provide significant improvement.

For unimodal functions, a persistent covariance strategy will improve performance while the IPOP strategy does not produce improvement on such functions. On multimodal functions, on the other hand, the persistent covariance could be detrimental as it leads to early convergence. On these functions, the IPOP strategy could potentially improve performance as the restart strategy prevents local entrapment. However, more tests on larger problems are needed. Furthermore, we identified a special case where the KC strategy is good for multimodal function: when a good candidate solution is found early and a local search is needed. To fully take advantage of the proposed strategies, a fitness landscape analysis should be conducted. How the landscape analysis will be integrated into the CC framework and the algorithms are left as future work.

Acknowledgment. This work is partly funded by the European Commission's H2020 program, UTOPIAE Marie Curie Innovative Training Network, H2020-MSCA-ITN-2016, under Grant Agreement No. 722734. The authors also acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0098) as well as the DAAD (German Academic Exchange Service), Project-ID: 57515062 "Multi-objective Optimization for Artificial Intelligence Systems in Industry".

References

1. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: Congress on Evolutionary Computation, vol. 2, pp. 1769–1776. IEEE (2005)
2. Van den Bergh, F., Engelbrecht, A.P.: A cooperative approach to particle swarm optimization. Trans. Evol. Comput. **8**(3), 225–239 (2004)
3. Campolongo, F., Cariboni, J., Saltelli, A., Schoutens, W.: Enhancing the Morris method. In: Sensitivity Analysis of Model Output, pp. 369–379 (2005)

4. Chen, W., Tang, K.: Impact of problem decomposition on cooperative coevolution. In: Congress on Evolutionary Computation, pp. 733–740. IEEE (2013). https://doi.org/10.1109/CEC.2013.6557641
5. Elhara, O., et al.: COCO: the large scale black-box optimization benchmarking (BBOB-largescale) test suite. arXiv preprint arXiv:1903.06396 (2019)
6. Hansen, N.: Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, GECCO 2009, pp. 2389–2396. ACM, New York (2009). https://doi.org/10.1145/1570256.1570333
7. Hansen, N., Kern, S.: Evaluating the CMA evolution strategy on multimodal test functions. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 282–291. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30217-9_29
8. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001)
9. Li, X., Tang, K., Omidvar, M.N., Yang, Z., Qin, K., China, H.: Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization. Gene **7**(33), 8 (2013)
10. Mahdavi, S., Rahnamayan, S., Shiri, M.E.: Cooperative co-evolution with sensitivity analysis-based budget assignment strategy for large-scale global optimization. Appl. Intell. **47**(3), 888–913 (2017)
11. Mahdavi, S., Rahnamayan, S., Shiri, M.E.: Multilevel framework for large-scale global optimization. Soft Comput. **21**(14), 4111–4140 (2017). https://doi.org/10.1007/s00500-016-2060-y
12. Omidvar, M.N., Kazimipour, B., Li, X., Yao, X.: CBCC3 – a contribution-based cooperative co-evolutionary algorithm with improved exploration/exploitation balance. In: Congress on Evolutionary Computation, pp. 3541–3548, July 2016. https://doi.org/10.1109/CEC.2016.7744238
13. Omidvar, M.N., Li, X., Mei, Y., Yao, X.: Cooperative co-evolution with differential grouping for large scale optimization. Trans. Evol. Comput. **18**(3), 378–393 (2014). https://doi.org/10.1109/TEVC.2013.2281543
14. Omidvar, M.N., Yang, M., Mei, Y., Li, X., Yao, X.: DG2: a faster and more accurate differential grouping for large-scale black-box optimization. Trans. Evol. Comput. **21**(6), 929–942 (2017). https://doi.org/10.1109/TEVC.2017.2694221
15. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) PPSN 1994. LNCS, vol. 866, pp. 249–257. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58484-6_269
16. Salomon, R.: Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms. Biosystems **39**(3), 263–278 (1996). https://doi.org/10.1016/0303-2647(96)01621-8
17. Sun, Y., Li, X., Ernst, A., Omidvar, M.N.: Decomposition for large-scale optimization problems with overlapping components. In: Congress on Evolutionary Computation, pp. 326–333. IEEE (2019). https://doi.org/10.1109/CEC.2019.8790204
18. Sun, Y., Kirley, M., Halgamuge, S.K.: A recursive decomposition method for large scale continuous optimization. Trans. Evol. Comput. **22**(5), 647–661 (2017)
19. Sun, Y., Kirley, M., Halgamuge, S.K.: Extended differential grouping for large scale global optimization with direct and indirect variable interactions. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO 2015, pp. 313–320. ACM, New York (2015). https://doi.org/10.1145/2739480.2754666

20. Sun, Y., Omidvar, M.N., Kirley, M., Li, X.: Adaptive threshold parameter estimation with recursive differential grouping for problem decomposition. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 889–896 (2018)
21. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Inf. Sci. **178**(15), 2985–2999 (2008)