Using the iCC framework for solving unconstrained LSGO problems

A Vakhnin and E Sopov
Reshetnev Siberian State University of Science and Technology, 31, Krasnoyarskiy Rabochy Av., Krasnoyarsk, 660037, Russian Federation

E-mail: alexeyvah@gmail.com, evgenysopov@gmail.com

Abstract. Evolutionary computation is a branch of computational intelligence that is inspired by nature. This algorithm family uses fundamental rules of natural selection to find a better solution when solving optimization problems. The productivity of these methods decreases when the number of parameters to be optimized is extra-large. This area of research is known as large-scale global optimization. Cooperative coevolution is a structure for evolutionary algorithms which tries to solve the curse of dimensionality problem. The cooperative coevolution efficiency of application largely depends on the two settings: the size of each subcomponent(subproblem) and the grouping of variables. The actual work proposes improved cooperative coevolution denoted as iCC. The iCC approach dynamically resizes the groups. iCC starts with a predefined set of subproblems and reduces them gradually during the optimization process. A novel metaheuristic has been developed which is called iCC-SHADE for black-box optimization problems with a large number of variables. The proposed method has been tested on fifteen optimization tasks from the LSGO CEC’2013 competition benchmark. The experimental results have demonstrated that iCC-SHADE has statistically better performance than CC-SHADE with a static number of subproblems. Also, the effectiveness of iCC-SHADE has been tested in comparison with other modern metaheuristics. The Wilcoxon rank-sum test was used to compare the effectiveness of investigated metaheuristics.

1. Introduction
Evolutionary Algorithms (EAs) are a high performed and modern tool in the area of black-box parameter optimization. However, their performance decreases when solving high dimensional problems, i.e. with a large number of variables. Large-scale global optimization (LSGO) problems consist of more than 100 variables to be optimized. Today, many scientists in the sphere of EA apply cooperative coevolution (CC) approaches to overcome the negative effect of the large search space. CC was first proposed in [1]. An original CC idea consists of splitting the original problem into several subproblems with a small number of parameters and optimize them independently of each other using some EA. According to research of metaheuristics in LSGO [2], CC is a big branch of modern LSGO techniques that rises the effectiveness of EA-based algorithms. Furthermore, CC techniques can be split into three following grouping-based methods: static [1], random dynamic [3], and learning dynamic [4]. The last two methods demonstrate better efficiency on average than the static-based method. In the study, the effect of increasing the number of variables in CC subproblems on the algorithm efficiency has been studied. The CC framework for solving LSGO problems has been tested. A novel framework was called iCC. Previously, the performance of iCC was tested on constrained LSGO (cLSGO) problems in [5, 6]. iCC
has shown better average efficiency than the static grouping-based method on cLSGO problems. The attention of this study is directed towards a comparison of the performance between CC and iCC frameworks on unconstrained LSGO problems. The novel metaheuristic is based on the SHADE and iCC approach, and it was called iCC-SHADE. iCC does not use any variable grouping methods.

In general, an LSGO task can be stated as the standard form of a continuous optimization problem (1), where \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is the object (fitness) function to be minimized over the \( n \)-variable real-valued vector \( \bar{x} \). In (2), \( a_i \) and \( b_i \) are the left and right borders of the search interval for the \( i \)-th variable, respectively.

\[
\min_{\bar{x}} f(\bar{x}) \quad (1)
\]
\[
a_i \leq x_i \leq b_i, \ i = 1, \ldots, n \quad (2)
\]

The rest of the paper is organized in the following way. Section 2 outlines the basic CC framework and self-adaptive differential evolution called SHADE. The SHADE algorithm is a core of the CC and the iCC frameworks. In the third Section, the iCC approach is proposed. The results of the experiments and parameter settings are discussed in Section 4. Section 5 consists of conclusions and contains directions for possible future work.

2. Related work

2.1. Cooperative Coevolution

The idea of the CC framework is presented below. The CC framework divides the decision vector into a predefined number of subproblems and optimizes them independently of each other with a help of a genetic algorithm (GA) [7]. In the classic algorithm, the number of subproblems equals the number of variables. CCGA-1 and CCGA-2 [1] use the following scheme, each decision variable stands into an individual subproblem.

Any classic CC requires \( m \cdot \text{pop\_size} \) fitness evaluations to evaluate one generation of decisions. \( m \) is the number of subproblems by which the decision vector is divided. \( \text{pop\_size} \) is the number of individuals (potential solutions) into one subproblem. Thus, if we set \( m \) equal to 1, CCEA does not use the CC framework and optimizes the whole decision vector. The main steps in detail are presented in figures 1 and 2. Figure 2 shows an example of three subproblems. Usually, the termination condition is reaching a given number of fitness function evaluations.

![Figure 1. Pseudo-code of CC.](image1)

![Figure 2. The CC general scheme.](image2)

2.2. Differential Evolution with Success-History Based Adaptation of Parameters

Differential Evolution (DE) is an effective and easy to implement optimization algorithm. DE optimizes a solution vector by iteratively trying to improve candidates of a population. It consists of four steps: initialization, mutation, recombination, and selection. The canonical DE algorithm was originally
proposed in [8, 9]. DE has two main control parameters: \( F \) (scale factor) and \( CR \) (crossover rate). \( F \) is responsible for a scale difference between selected individuals at the stage of mutation. \( CR \) is responsible for the substitution probability of variables from the mutant vector. Usually, control parameters are inside the range: \( F, CR \in [0; 1] \). There are many DE-based EAs that were proposed in the last two decades [10]. One of the effective DE varieties is the parameter adaptation based on the success-history differential evolution (SHADE) algorithm [11]. The distinct feature of SHADE is an original self-adaptation technique of DE control parameters. The self-adaptation of SHADE works with historical memory. \( H \) is a predefined number of pairs \( F \) and \( CR \). For every generation, the \( r_i \) index is randomly selected from the range \([1; H]\). \( r_i \) denotes the pair number of \( F \) and \( CR \). SHADE records success values of \( F, CR \), and then update the historical memory into \( r_i \) position. The replaced individuals are preserved in an external archive to maintain diversity [12]. If the external archive is full, randomly selected individuals are replaced by new individuals. The SHADE mutation operator may use individuals which were taken from the archive. Usually, the size of the archive equals twice the number of individuals in the population.

3. Proposed iCC Framework

This section describes our proposed iCC framework when solving unconstrained LSGO problems. As we mentioned earlier, the efficiency of metaheuristics decreases when problems with a huge number of variables are being solved. As it is well known, the subproblems number is one of the main CC framework parameters. There is no “good” predefined number of subproblems when the black-box optimization problems are being solved. iCC uses gradual increasing the number of variables in subproblems. In other words, iCC reduces the total number of subproblems during the optimization process. The iCC framework allocates an equal number of fitness evaluations (FEVs) and for each set of subproblems. During the first generations, iCCEA optimizes the search space in parts using the “divide and conquer” principle. For each subproblem, EA optimizes its sub-search space.

Figure 3 shows the pseudo-code of iCCEA. In this case, EA can be any other metaheuristics. In the study, iCC has the following set of subproblems, \( S = \{10, 8, 4, 2, 1\} \). Allowed FEVs for each stage of iCC is calculated as follows. \( iCCFEVs = \frac{MAX\_FEVs}{|S|} \), where \( MAX\_FEVs \) is the budget of fitness evaluations in one run, \( |S| \) is the cardinality of the \( S \) set (the total number of stages of iCC). We have tried the opposite strategy to decrease the number of variables for each subproblem, but its performance was worse. The pseudo-code of iCCEA is shown in figure 3.

```
Algorithm 2 iCCEA
1: \( \text{Set population size, } S = \{10, 8, 4, 2, 1\}; iCCFEVs = \frac{MAX\_FEVs}{|S|}, FEV\_counter \ = \ 0, iCC\_index \ = \ 1 \)
2: \( \text{for } i \text{ from } iCC\_index \text{ to } S \text{ do} \)
3: \( \text{Randomly initialize population for } i \)-th subcomponent; \)
4: \( \text{Evaluate fitness function for each individual in } i \)-th subcomponent; \)
5: \( \text{FEV\_counter} \ += \ 1; \) // counting the number of fitness function calculations \)
6: \( \text{end for} \)
7: \( \text{while } \text{Termination condition} = \text{false} \text{ do} \)
8: \( \text{for } i \text{ from } iCC\_index \text{ to } S \text{ do} \)
9: \( \text{Apply EA operators to } i \)-th subcomponent; \)
10: \( \text{Evaluate fitness of each individual into } i \)-th subcomponent; \)
11: \( \text{FEV\_counter} \ += \ 1; \)
12: \( \text{end for} \)
13: \( \text{if} \text{FEV\_counter} > iCCFEVs \text{ then} \)
14: \( iCC\_index \ += \ 1; \)
15: \( \text{FEV\_counter} \ = \ 0; \)
16: \( \text{Evaluate fitness function for each individual in } i \)-th subcomponent; \)
17: \( \text{end if} \)
18: \( \text{end while} \)
19: \( \text{return the best-found solution} \)
4. The results of the experiments and parameter settings

In this study, all computational numerical experiments have been executed using a computing cluster. Four Ryzen CPUs with sixty-four threads were used in calculations. We have used the C++ programming language to implement both the CC-SHADE and the iCC-SHADE algorithms. The computational cluster has been used because the LSGO problems are high computational complexity. We have investigated five CC-SHADE algorithms with a static grouping of subproblems and iCC-SHADE on the LSGO CEC’2013 benchmark [13]. Also, the performance of the iCC-SHADE algorithm was compared to other state-of-the-art metaheuristics.

4.1. Large-scale Global Optimization CEC 2013 benchmark

The IEEE Congress on Evolutionary Computation covers all topics in the sphere of evolutionary computation techniques for optimization, LSGO is no exception. Over the past years, several LSGO benchmark sets have been developed for evaluating various metaheuristics: CEC’08 [14], CEC’10 [15], and CEC’13 [13]. The latest and relevant LSGO CEC’2013 benchmark set consists of 15 problems. These functions are divided into four categories: Fully-separable (F1 – F3), Partially Additively Separable (F4 – F11), Overlapping (F12 – F14), and Non-separable (F15). The number of variables for all problems was set to $D = 1000$.

The following settings have been used for the iCC-SHADE and the CC-SHADE algorithms:

- $\text{pop\_size} = \{25, 50, 100, 150, 200\}$ (for each subproblem),
- $m = \{1, 2, 4, 8, 10\}$,
- $s = \{10, 8, 4, 2, 1\}$,
- $\text{external\_archive\_size} = 2 \cdot \text{pop\_size}$, the maximum number of fitness evaluations is $3 \cdot 10^6$ FEVs for each independent run, the 25 independent runs for each problem in this benchmark were used.

Algorithms were compared in accordance with the rules of the Competition on Large-Scale Global Optimization.

4.2. The experimental results

This subsection presents the results received during the numerical experiments. We use the following notation in this study: CC-SHADE(m), where $m$ denotes the number of subproblems. The experimental results of the CC-SHADE(m) and the iCC-SHADE algorithms are shown in figures 4-8. The ranking results of the CC-SHADE and the iCC-SHADE are demonstrated with population size equals to 25, 50, 100, 150, and 200, respectively, in figures 4-8.

The smallest rank was assigned to the best EA and the ranking procedure was based on the mean best-found fitness values averaged over fifteen problems from the CEC 2013 benchmark set. As figures 4-8 show, the iCC approach almost always has the lowest rank, an exception is iCC with 50 individuals (figure 5). The percentage difference in average ranking between CC-SHADE(10) and iCC-SHADE with 50 individuals is 7.33%. In all other cases, iCC-SHADE demonstrated a lower average rank than other variants of CC-SHADE with a different fixed number of subproblems.

![Figure 4](image-url) CC(m) and iCC ranking, pop_size is 25.
Figure 5. CC(m) and iCC ranking, pop_size is 50.

Figure 6. CC(m) and iCC ranking, pop_size is 100.

Figure 7. CC(m) and iCC ranking, pop_size is 150.

Figure 8. CC(m) and iCC ranking, pop_size is 200.

Table 1 proves that the difference in the results of the efficiency estimation of the iCC-SHADE and the CC-SHADE(m) algorithms using the Wilcoxon rank-sum test (p-value = 0.01) is significant. The
first column and the first row contain algorithms. Each cell holds the comparison results of two algorithms on all benchmark problems and with all variants of population size. The data in each cell has the following format: (win/loss/equal). We have compared two algorithms between each other, the 1st algorithm is from the row, the 2nd is from the column. If the 1st algorithm is better, worse, or equal than the 2nd algorithm, we added points into (win/loss/equal), respectively. The last column of table 1 shows that the iCC-SHADE algorithm is always better than any CC-SHADE(m) with a fixed number of subproblems (there are more victory scores than loss points).

Table 1. Comparison of EAs on the LSGO CEC’2013 benchmark using Mann–Whitney U test.

| Number | Algorithm       | Win        | Loss       | Equal  | Average Rank |
|--------|-----------------|------------|------------|--------|--------------|
| (1)    | CC-SHADE(1)     | 45/11/19   | 51/18/6    | 48/22/5| 43/24/8      |
| (2)    | CC-SHADE(2)     | 42/13/20   | 39/21/15   | 36/26/13| 45/15/15    |
| (3)    | CC-SHADE(4)     | 31/19/25   | 31/28/16   | 38/23/14|             |
| (4)    | CC-SHADE(8)     | -          | -          | -      | 19/20/36    |
| (5)    | CC-SHADE(10)    | -          | -          | -      | 20/16/39    |
| (6)    | iCC-SHADE      | -          | -          | -      | -            |

Results that are based on table 1 and the average mean rank of investigated algorithms are presented in table 2. The 1st column contains the order number of algorithms from table 1. The algorithm names are in the 2nd column. The 3rd, 4th, and 5th columns contain the number of times when the algorithm was significantly better, worse, or equal versus another algorithm, respectively. The last column holds the sum of the algorithms’ average rank with all tested population sizes. In table 2, algorithms have been sorted according to their average rank. Based on the results from table 2, it could be concluded that the performance of the iCC-SHADE algorithm is on average better than the performance of the CC-SHADE with the fixed number of subproblems on the LSGO CEC’2013 problems. iCC-SHADE has more victory points and has the lowest average rank on all comparisons.

Table 2. The sum of Mann–Whitney statistical test results of EAs

| Number | Algorithm       | Win | Loss | Equal | Average Rank |
|--------|-----------------|-----|------|-------|--------------|
| (6)    | iCC-SHADE      | 176 | 92   | 107   | 12.53        |
| (4)    | CC-SHADE(8)    | 157 | 105  | 113   | 14.93        |
| (3)    | CC-SHADE(4)    | 169 | 131  | 75    | 15.27        |
| (5)    | CC-SHADE(10)   | 145 | 118  | 112   | 15.93        |
| (2)    | CC-SHADE(2)    | 120 | 173  | 82    | 18.93        |
| (1)    | CC-SHADE(1)    | 94  | 236  | 45    | 21.73        |

Figure 9 demonstrates the average ranking results for iCC-SHADE variations with different numbers of the population size. The rank was calculated in the same way as for figures 1-4, the smallest rank is better. The iCC-SHADE algorithm with pop_size = 150 demonstrated the best performance.

Figure 9. The ranking of iCC with different number of individuals.
We have compared tuned iCC-SHADE with other state-of-the-art metaheuristics. Tuned iCC-SHADE has the population size equals to 150. The numerical results of competition participants were taken from the Special Session and Competition on Large-Scale Global Optimization 2019. The following metaheuristics were selected for the comparison: MPS [16], SGCC [17], CC-RDG3 [18], and DGSC [19]. These algorithms were specially created for solving complex LSGO problems. Table 3 demonstrates the numerical results of the algorithms’ performance. The first column contains the number of LSGO problem. The next columns contain EA names. Each cell contains the mean performance of the algorithm which is obtained on 25 independent runs. The last row of table 3 contains the average rank.

| #  | iCC-SHADE | MPS    | SGCC  | CC-RDG3 | DGSC  |
|----|-----------|--------|-------|---------|-------|
| 1  | 1.01E-17  | 6.68E+08 | 1.85E+03 | 1.14E-18 | 2.60E-04 |
| 2  | 9.37E+02  | 4.20E+03 | 8.94E+03 | 2.31E+03 | 7.15E+02 |
| 3  | 2.07E+01  | 1.94E+00 | 2.15E+01 | 2.04E+01 | 2.07E+01 |
| 4  | 9.71E+08  | 1.07E+11 | 1.23E+09 | 4.29E+04 | 3.77E+08 |
| 5  | 2.47E+06  | 1.20E+06 | 5.02E+06 | 2.04E+06 | 3.27E+06 |
| 6  | 1.96E+06  | 6.01E+03 | 1.06E+06 | 1.00E+06 | 1.06E+06 |
| 7  | 2.70E+05  | 7.19E+07 | 1.83E+05 | 1.71E-21 | 4.83E+05 |
| 8  | 2.84E+12  | 2.04E+14 | 1.54E+11 | 7.11E+03 | 1.85E+13 |
| 9  | 1.66E+08  | 1.66E+08 | 4.06E+08 | 1.57E+08 | 1.79E+08 |
| 10 | 9.32E+07  | 3.53E+06 | 9.38E+07 | 9.16E+07 | 9.38E+07 |
| 11 | 1.94E+07  | 2.20E+09 | 2.55E+07 | 2.18E-13 | 6.92E+09 |
| 12 | 1.14E+03  | 1.75E+04 | 3.56E+03 | 7.00E+02 | 2.93E+03 |
| 13 | 1.41E+07  | 9.87E+08 | 1.21E+07 | 6.43E+04 | 8.63E+08 |
| 14 | 3.38E+07  | 1.03E+09 | 2.07E+07 | 1.65E+09 | 1.32E+08 |
| 15 | 2.63E+05  | 2.76E+07 | 1.30E+06 | 2.30E+06 | 2.67E+07 |
| Av. Rank | 2.47 | 3.53 | 3.40 | 1.80 | 3.40 |

Metaheuristics (from table 3) are ranked by their mean best-found value for each benchmark problem using the “Formula 1” scoring system. The points scoring system is as follows: “25-18-15-12-10-8-6-4-2-1”. The best EA with the highest performance gets 25 points, the second-placed EA gets 18, and so on. Figure 10 presents the results in bar charts. Each bar with four colored parts corresponds to the EA. In its turn, each colored part represents the number of scored points in a subclass of problems. The detailed information about scored points is presented in table 4. The first column contains the name of the problem category. The next columns contain performance points of iCC-SHADE and other metaheuristics. Table 4 cells contain scored points for each EA. As we can see from figure 10 and table 4, the proposed iCC-SHADE approach is placed in second place.

**Figure 10.** State-of-the-art metaheuristics versus iCC on the LSGO CEC’2013.
Table 4. The score of iCC-SHADE vs other state-of-the-art metaheuristics.

| Category                  | iCC-SHADE | MPS  | SGCC | CC-RDG3 | DGSC |
|---------------------------|-----------|------|------|---------|------|
| Fully-separable           | 51        | 47   | 32   | 58      | 55   |
| Partially Additively Separable | 126     | 135  | 110  | 179     | 103  |
| Overlapping               | 51        | 32   | 55   | 60      | 42   |
| Non-separable             | 25        | 10   | 18   | 15      | 12   |
| Total points              | 253       | 224  | 215  | 312     | 212  |

5. Conclusions and future work
In this study, we have proposed the iCC framework for solving unconstrained large-scale global optimization problems. Numerical experiments have confirmed that proposed the performance of iCC-SHADE algorithm is statistically better than the performance of CC-SHADE with a fixed number of subproblems. Mann–Whitney U test has been used to prove the results of numerical experiments. The tuned iCC-SHADE algorithm has the population size of 150 individuals in each subproblem. iCC-SHADE has been compared with other modern metaheuristics on LSGO CEC’2013. iCC-SHADE won second place. The population size is one control parameter in the iCC-SHADE algorithm. The iCC framework has great potential for improvement. In our further studies, we will try to modify iCC for adaptive variable grouping.

Acknowledgments
This work was supported by the Ministry of Science and Higher Education of the Russian Federation within limits of state contract № FEFE-2020-0013.

References
[1] Potter M A and Jong K A 1994 Lecture Notes in Computer Science pp 249-57
[2] Mahdavi S, Shiri M E and Rahnamayan S 2015 Information Sciences 295 407-28
[3] Yang Z, Tang K and Yao X 2008 Information Sciences 178(15) 2985-99
[4] Ray T and Yao X 2009 2009 IEEE Congress on Evolutionary Computation 983-89
[5] Vakhnin A and Sopov E 2020 IOP Conf. Ser.: Mater. Sci. Eng. 734 012099
[6] Vakhnin A and Sopov E 2020 Algorithms 13(5) 108
[7] Potter M A and Jong K A 2000 Evolutionary Computation 8(1) 1-29
[8] Storn R 1996 Proceedings of North American Fuzzy Information Processing 519-23
[9] Storn R and Price K 1997 Journal of Global Optimization 11(4) 341-59
[10] Das S, Mullick S S and Suganthan P N 2016 Swarm and Evolutionary Computation 27 1-30
[11] Tanabe R and Fukunaga A 2013 2013 IEEE Congress on Evolutionary Computation 71-8
[12] Zhang J and Sanderson A C 2013 2013 IEEE Transactions on Evolutionary Computation 13(5) 945-58
[13] Li X, Tang K, Omidvar M N, Yang Z and Qin K 2013 2013 IEEE Congress on Evolutionary Computation Competition on : Large Scale Global Optimization 23
[14] Tang K, Yao X, Suganthan P N, MacNish C, Chen Y-P, Chen C-M and Yang Z 2007 Nature Inspired Computation and Applications Laboratory
[15] Tang K, Li X, Suganthan P N, Yang Z and Weise T 2010 Nature Inspired Computation and Applications Laboratory 23
[16] Bolufé-Röhler A, Fiol-González S and Chen S 2015 IEEE Congress on Evolutionary Computation 1958-65
[17] Liu W, Zhou Y, Li B and Tang K 2019 2019 IEEE Congress on Evolutionary Computation 318-25
[18] Sun Y, Li X, Ernst A and Omidvar M N 2019 2019 IEEE Congress on Evolutionary Computation 326-33
[19] Li L, Fang W, Wang Q and Sun J 2019 2019 IEEE Congress on Evolutionary Computation 334-41

8