

PAPER • OPEN ACCESS

An approach for initializing the random adaptive grouping algorithm for solving large-scale global optimization problems

To cite this article: A Vakhnin and E Sopov 2019 *IOP Conf. Ser.: Mater. Sci. Eng.* **537** 042006

View the [article online](#) for updates and enhancements.



IOP | ebooks™

Bringing you innovative digital publishing with leading voices to create your essential collection of books in STEM research.

Start exploring the collection - download the first chapter of every title for free.

An approach for initializing the random adaptive grouping algorithm for solving large-scale global optimization problems

A Vakhnin and E Sopov

Reshetnev Siberian State University of Science and Technology
31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660037, Russian Federation

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

Abstract. Many real-world optimization problems deal with high dimensionality and are known as large-scale global optimization (LSGO) problems. LSGO problems, which have many optima and are not separable, can be very challenging for many heuristic search algorithms. In this study, we have proposed a novel two-stage hybrid heuristic algorithm, which incorporates the coordinate descent algorithm with the golden-section search (CDGSS) and the random adaptive grouping for cooperative coevolution of the Self-adaptive Differential Evolution with Neighborhood Search (DECC-RAG) algorithm. At the first stage, the proposed algorithm roughly scans the search space for a better initial population for the DECC-RAG algorithm. At the second stage, the algorithm uses the DECC-RAG framework for solving the given LSGO problem. We have evaluated the proposed approach (DECC-RAG1.1) with 15 most difficult LSGO problems from the IEEE CEC'2013 benchmark set. The experimental results show that DECC-RAG1.1 outperforms the standard DECC-RAG and some the state-of-the-art LSGO algorithms.

1. Introduction

Many surveys and reviews on complex optimization problems show that real-world optimization problems have large number of variables [1-5]. The NASA has published a report [6], which demonstrates an exponential growth for dimensionality of modern real-world optimization problems since 1960s. Optimization problems with many hundreds or thousands of objective variables are called as large-scale global optimization (LSGO) problems. The LSGO is known as one of the most challenging problem for many search techniques from the field of mathematical programming and evolutionary optimization. Many well-known real-world LSGO problems are not separable and are complex for comprehensive analysis, thus, they are viewed as black-box optimization problems even objective functions are represented analytically using mathematical formula.

Without loss of generality, a large-scale global optimization (LSGO) problem can be stated as:

$$f(\bar{x}) \rightarrow \min_{\bar{x}}, a_i \leq x_i \leq b_i, i = \overline{1, n} \quad (1)$$

where $f(\bar{x}): R^n \rightarrow R$ is the objective function to be minimized over the n -variable vector \bar{x} , a_i and b_i are the lower and upper bounds of a search space interval for the i -th dimension.

The most advanced algorithms for LSGO are based on the cooperative coevolution (CC) framework [7] with problem decomposition using different grouping methods, which decomposes LSGO problems into multiple low-dimensional non-overlapping subcomponents.



In our previous studies [8, 9], we have proposed a novel grouping technique that combines the ideas of the random dynamic grouping and the learning dynamic grouping. The approach is called the random adaptive grouping (RAG). In our implementations, the RAG is combined with CC of the Self-adaptive Differential Evolution (DE) with Neighborhood Search (SaNSDE) [10] (the whole search algorithm is called DECC-RAG). The RAG starts with random subcomponents of an equal predefined size. After some generations of the DECC (so-called adaptation period), we estimate the performance of each subcomponent. A portion of the best subcomponents is saved for the next adaptation period. And we apply the random grouping again for the rest subcomponents. Such a feedback forms different groups of variables and adaptively changes them during the search. The proposed DECC-RAG outperforms some the state-of-the-art LSGO algorithms on the LSGO benchmarks proposed within the IEEE CEC 2010 and 2013.

In this study, we have proposed and have analyzed an approach for improving the performance of the DECC-RAG using a new initialization scheme.

The rest of the paper is organized as follows. Section 2 describes the proposed approach. In Section 3 the experimental setups and results of numerical experiments are discussed. In the Conclusion the results and further research are discussed.

2. Proposed framework for initializing population in the DECC-RAG

Many different types of evolutionary algorithms (EAs) demonstrate high performance when solving various complex optimization problems. Many stochastic search techniques, including EAs, start a new search process using an initialization procedure for distributing a population of candidate-solutions over the search space. Generally, the initialization uses the random uniform distribution (2):

$$x_{i,j} = rand(0,1) * (b - a) + a \quad (2)$$

where $x_{i,j}$ is the j -th variable of the i -th individual, $rand(0,1)$ is a random real number uniformly distributed in the interval from 0 to 1.

There are many reasons why we need to apply the random uniform distribution during the initialization stage. When solving the black-box (BB) optimization problems, we have no information on regions of the search space, where the best solutions are located. We also cannot provide the fundamental analysis of the given objective for identifying if the problem is multimodal, separable, nonlinear, etc. An implemented EA can only request a value of the objective function $f(\bar{x})$ for a point \bar{x} .

Modern LSGO deals with BB problems which contain more than 1000 variables. LSGO problems with huge dimensionality are extremely difficult for many search algorithms, because the search space grows exponentially and properties of the search space may vary as the number of decision variables increases. Also evaluations of the fitness function for LSGO problems are usually computationally expensive. In a case of LSGO, a random distribution of an initial population may lead to loss of performance because of the limited number of fitness evaluations. As result, the standard EA is not able to find a good solution starting with randomly generated points.

In this study, in order to save computational resources, we have proposed a framework for generating better initial points. The framework is based on a rough scanning of the search space. The scanning procedure uses the coordinate descent [11] with the golden section search [12]. Finally, we randomly generate the initial population near points produced by the scanning procedure. The proposed approach is presented using the pseudo-code in algorithm 1 and algorithm 2.

In our study, we run 50 cycles of algorithm 2 for obtaining one new starting point in a cycle. After that, we generate the initial population for the DECC-RAG using the following approach:

$$x_{i,j} = rand_norm(bestFoundCDGSS_j, (b - a) * rate) \quad (3)$$

where $rand_norm(M, \sigma^2)$ is the normal distribution with the mean (M) and the standard deviation (σ), $rate$ is a coefficient that defines deviations around the best point. In this study, we use $rate=0.15$, the value is chosen based on some prelaminar experiments.

Algorithm 1. Pseudo-code of golden-section search (GSS) for 1-D minimization problem.

Set $[a, b]$ as a search interval, $\phi = (\sqrt{5} + 1)/2$

while Stopping criteria is not met **do**

$x1 = b - (b - a)/\phi$, $x2 = a + (b - a)/\phi$;

if $f(x1)$, $f(x2)$ not available

 then compute;

end if

if $f(x1) < f(x2)$

 then $b = x2$;

else $a = x1$;

end if

end while

return $(a + b)/2$.

Algorithm 2. Pseudo-code of (CDGSS) for N minimization problem.

initialize all variables: $x_i = a_i + (b_i - a_i)/\phi$

for $i=1$ to N

 fix values of all variables except x_i ;

 optimize the i -th dimension x_i using GSS (Algorithm 1);

$i=i+1$;

end for

return vector of *best_found_solution*.

The detailed description of the DECC-RAG1.1 approach is presented in Algorithm 3:

Algorithm 3. Pseudo-code of DECC-RAG1.1(m).

Set FEV_global , T , $FEV_local = 0$, $rate$, $cycles_of_CDGSS$

Find $good_initial_solution$ using Algorithm 2 $cycles_of_CDGSS$ times;

According to Equation (3) generate population;

An n -dimensional object vector is randomly divided into m

s -dimensional subcomponents;

Randomly mix indices of variables;

while ($FEV > 0$) **do**

for $i = 1$ to M ;

 Evolve the i -th subcomponent with SaNSDE algorithm, record CBS and PBS;

$\Delta f_i += |PBS - CBS|$;

end for

if ($FEV_local \geq T$)

 then choose $m/2$ subcomponents with the worse performance ($m/2$ smallest Δf_i)

 and randomly mix indices of its subcomponents, restart parameters of SaNSDE

 in these $m/2$ subcomponents, $FEV_local = 0$, reset Δf_i values;

end if

end while

return the best solution;

*previous best solution

**current best solution

3. Experimental setups and results

3.1. Benchmark Functions of the CEC'2013 Special Session and Competition on Large-Scale Global Optimization

We have evaluated the proposed DECC-RAG1.1 algorithm with the IEEE LSGO CEC'2013 benchmark [13]. The LSGO CEC'2013 benchmark contains 15 complex LSGO problems, which include F1-F3: Fully-separable functions, F4-F11: Partially (additively) separable functions, F11-F15: Fully non-separable functions. The number of variables for all problems is 1000. In order to compare the experimental results of the proposed approach with other the state-of-the-art techniques, we apply the same experimental settings as specified in the rules of the benchmark.

3.2. Software implementation and setups

All numerical experiments were executed using the following computational system:

- OS: Ubuntu Linux 16.04 LTS.
- CPU: Ryzen 7 1700x.
- RAM: 16GB.
- IDE: Code::Blocks 17.12.
- Language: C++.
- Compiler: g++ (gcc) with -O3 optimization flag.

General settings for all numerical experiments are: the dimension is $D=1000$, the number of independent runs is 25 runs per each benchmark problem, the maximum number of fitness evaluations is $Max_FE = 3E+6$, the termination criteria is Max_FE . In order to provide fair comparison, all algorithms use the same Max_FE value. Fitness evaluations at the initialization stage of the DECC-RAG1.1 are also counted in the FEs .

The DECC-RAG1.1 have been implemented using the parallel computation framework [17]. The parallel implementation essentially decreases the runtime of the experiments and makes it possible to carry out experimental analysis of the approach with different settings in reasonable time. Table 1 presents the runtime for the benchmark.

Table 1. Runtime of 10,000 FEs (in seconds) on the benchmark functions.

Benchmark problem	F1	F2	F3	F4	F5	F6	F7	F8
Runtime	2.11	2.63	2.66	2.20	2.77	2.87	0.7	2.6
Benchmark problem	F9	F10	F11	F12	F13	F14	F15	-
Runtime	3.16	3.25	2.42	0.026	2.5	2.41	1.92	-

3.3. Experimental results

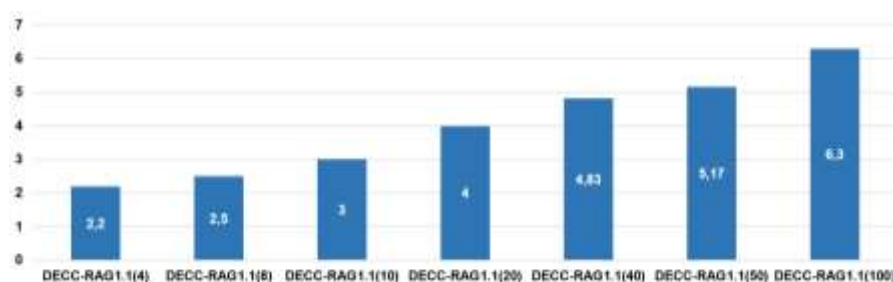
The experimental results for different settings of the grouping size in the DECC-RAG1.1 are presented in table 2. Cells highlighted with bold corresponds to the best results obtained with the given number of subcomponents. Figure 1 shows the results of ranking the investigated algorithms (ranks are averaged over the benchmark problems).

As we can see from table 2, the DECC-RAG1.1 algorithm performs better with small number of subcomponents, which contain large number of variables. This means that the approach is able to provide an efficient decomposition of LSGO problems and is able to form efficient combinations of variables in subcomponents. The same results have been obtained for the standard DECC-RAG algorithm, thus the proposed initialization scheme don't infuse on the grouping ability of the initial algorithm.

Table 3 contains some results of the Mann–Whitney U test with p-value equal to 0.05. We have counted the number of benchmark functions for which the proposed algorithm was significant better or worse than other compared algorithms.

Table 2. The experimental results of DECC-RAG1.1(m) for different number of subcomponent (m) on the CEC'2013 LSGO benchmark problems.

№	DECC-RAG1.1(m)						
	4	8	10	20	40	50	100
1	1.04E-20	1.06E-22	8.73E-23	8.73E-15	2.51E-11	2.51E-11	2.51E-11
	5.77E-21	6.73E-23	2.12E-22	4.93E-15	3.30E-27	3.30E-27	3.30E-27
2	2.36E+03	8.85E+02	6.69E+02	1.55E+03	2.41E+03	2.41E+03	2.40E+03
	3.33E+01	6.61E+01	4.28E+01	1.93E+02	1.09E+01	1.07E+01	2.42E+01
3	2.00E+01	2.00E+01	2.00E+01	2.00E+01	2.00E+01	2.00E+01	2.00E+01
	1.09E-14	1.09E-14	1.09E-14	1.09E-14	1.09E-14	1.09E-14	1.09E-14
4	6.16E+09	1.12E+10	1.32E+10	2.62E+10	4.18E+10	4.85E+10	7.75E+10
	2.44E+09	3.51E+09	3.35E+09	7.12E+09	6.35E+09	7.18E+09	8.46E+09
5	2.86E+06	3.32E+06	4.40E+06	4.84E+06	8.27E+06	8.92E+06	1.74E+07
	4.32E+05	6.73E+05	7.84E+05	1.18E+06	4.76E+06	5.33E+06	7.79E+06
6	1.06E+06	1.06E+06	1.06E+06	1.06E+06	1.06E+06	1.06E+06	1.06E+06
	1.09E+03	1.12E+03	1.20E+03	2.00E+03	2.75E+03	2.50E+03	3.75E+03
7	8.35E+05	2.40E+06	4.11E+06	3.39E+07	9.50E+07	1.35E+08	3.24E+08
	3.03E+05	1.05E+06	2.05E+06	8.72E+06	2.23E+07	3.31E+07	9.54E+07
8	1.14E+14	2.24E+14	2.94E+14	6.10E+14	9.87E+14	1.18E+15	1.91E+15
	4.41E+13	5.37E+13	8.65E+13	1.62E+14	2.00E+14	2.95E+14	3.81E+14
9	2.53E+08	3.11E+08	3.96E+08	1.14E+09	1.74E+09	1.58E+09	1.89E+09
	3.41E+07	6.48E+07	9.40E+07	3.60E+08	2.39E+08	2.43E+08	2.56E+08
10	9.45E+07	9.42E+07	9.40E+07	9.40E+07	9.40E+07	9.38E+07	9.42E+07
	2.71E+05	5.59E+05	4.38E+05	4.25E+05	5.19E+05	5.57E+05	5.38E+05
11	2.48E+08	9.03E+08	1.50E+09	5.09E+10	1.39E+11	1.94E+11	2.64E+11
	8.92E+07	3.76E+08	1.16E+09	2.48E+10	5.94E+10	5.39E+10	6.32E+10
12	9.73E+02	9.89E+02	9.97E+02	1.02E+03	9.86E+02	9.90E+02	1.28E+03
	4.70E+00	4.61E+00	5.00E+00	2.05E+01	5.66E+01	2.62E+01	7.47E+02
13	1.17E+08	4.89E+08	7.47E+08	2.25E+09	3.92E+09	4.46E+09	4.65E+09
	4.31E+07	1.45E+08	1.14E+08	3.04E+08	5.59E+08	4.47E+08	6.08E+08
14	1.59E+08	3.85E+09	1.64E+10	1.07E+11	2.30E+11	2.45E+11	2.81E+11
	9.90E+07	3.56E+09	6.54E+09	2.40E+10	3.71E+10	5.48E+10	4.84E+10
15	4.45E+06	4.23E+06	4.74E+06	8.01E+06	2.31E+07	1.94E+07	4.46E+07
	3.86E+05	5.61E+05	5.21E+05	8.01E+06	1.58E+07	1.11E+07	1.36E+07

**Figure 1.** The ranking of RECC-RAG1.1(m) on the LSGO CEC'2013.**Table 3.** The comparison of DECC-RAG1.1(4) with DECC-RAG1.1(m) and DECC-RAG1(8) on the LSGO CEC'2013 (Wilcox, p=0.05).

vs DECC-RAG1.1(4)		LSGO CEC'2013
DECC-RAG1.1(8)	+ (better)	4
	- (worse)	9
	≈ (no sig.)	2
DECC-RAG1.1(10)	+ (better)	3
	- (worse)	10

	\approx (no sig.)	2
DECC-RAG1.1(20)	+ (better)	2
	- (worse)	11
	\approx (no sig.)	2
DECC-RAG1.1(40)	+ (better)	1
	- (worse)	12
	\approx (no sig.)	2
DECC-RAG1.1(50)	+ (better)	1
	- (worse)	12
	\approx (no sig.)	2
DECC-RAG1.1(100)	+ (better)	1
	- (worse)	12
	\approx (no sig.)	2
DECC-RAG1 (8)	+ (better)	2
	- (worse)	10
	\approx (no sig.)	3

Figures 2-7 show the convergence of the DECC-RAG1.1. We can see that the approach demonstrates almost monotonic convergence starting with initial generations for all types of LSGO problems. This also proves that the proposed initialization scheme improves the DECC-RAG performance.

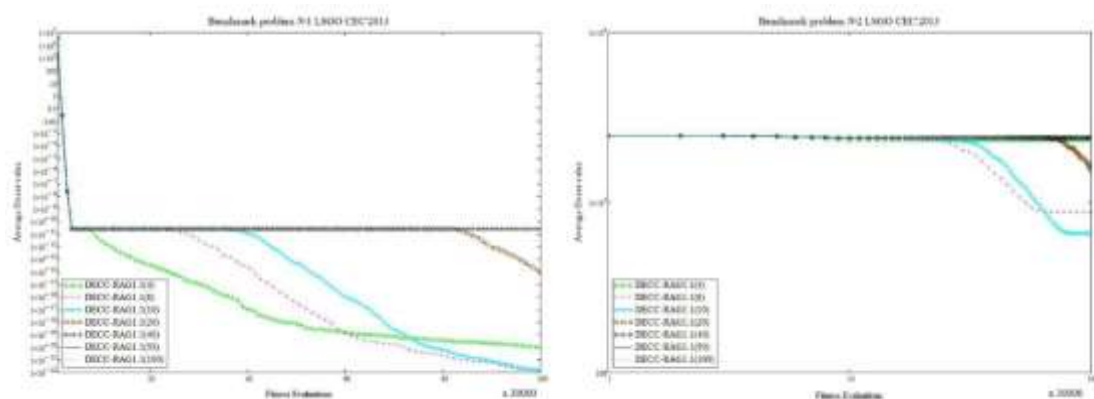


Figure 2. The convergence curve of DECC-RAG1.1(m) algorithms on F1 and F2.

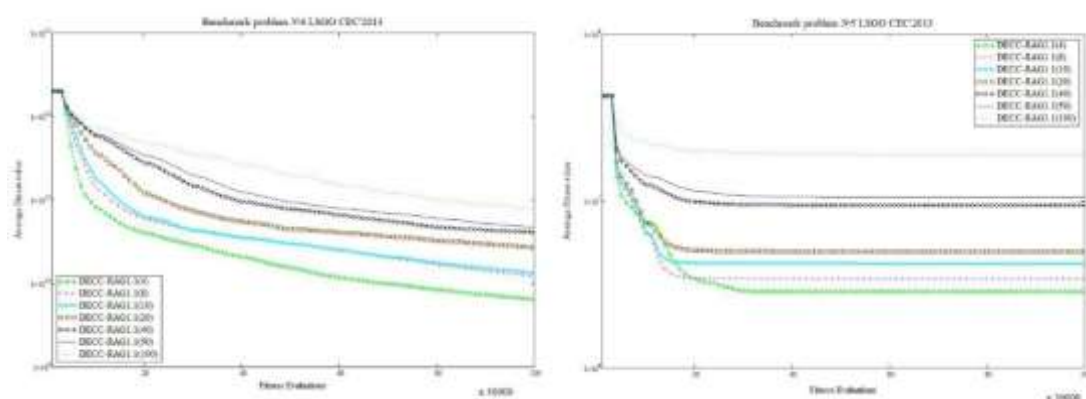


Figure 3. The convergence curve of DECC-RAG1.1(m) algorithms on F4 and F5.

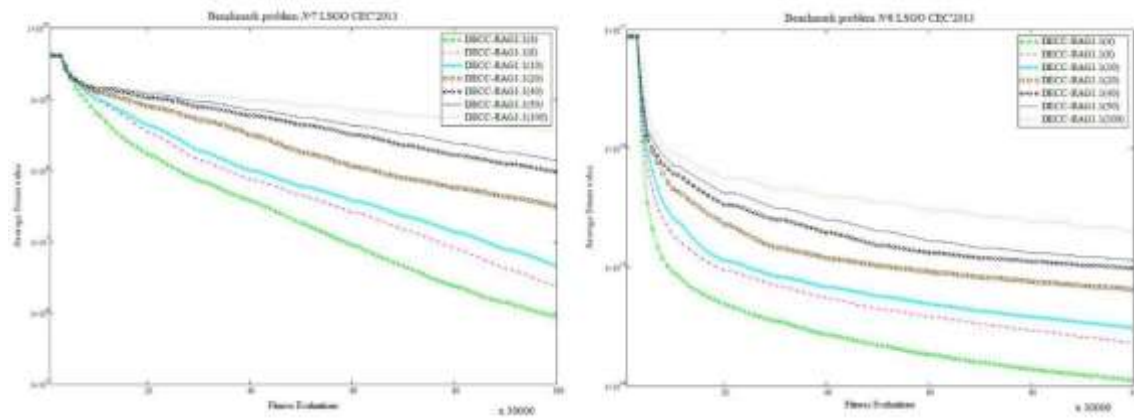


Figure 4. The convergence curve of DECC-RAG1.1(m) algorithms on F7 and F8.

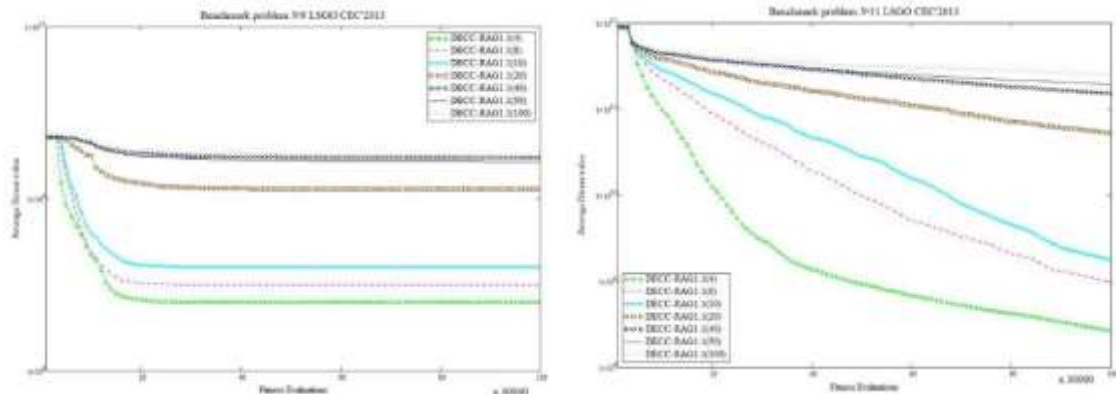


Figure 5. The convergence curve of DECC-RAG1.1(m) algorithms on F9 and F11.

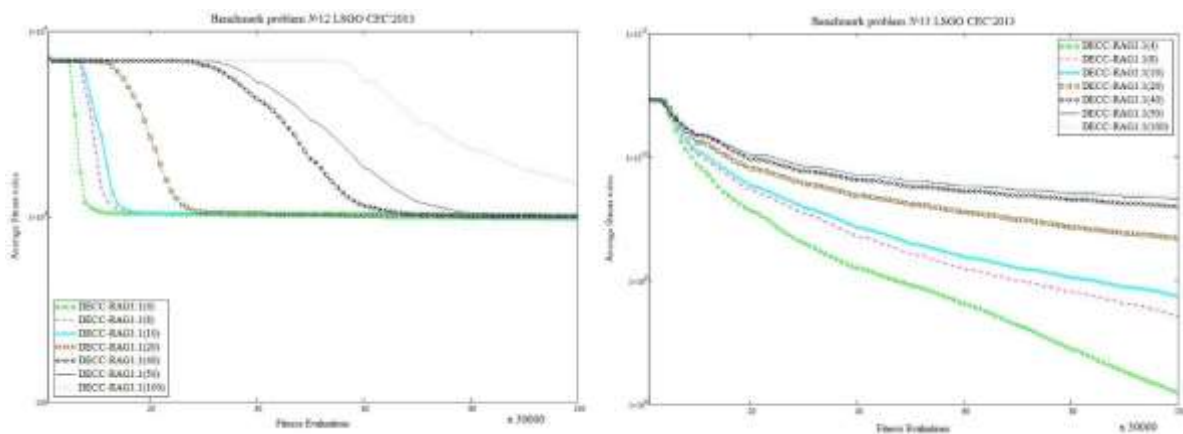


Figure 6. The convergence curve of DECC-RAG1.1(m) algorithms on F12 and F13.

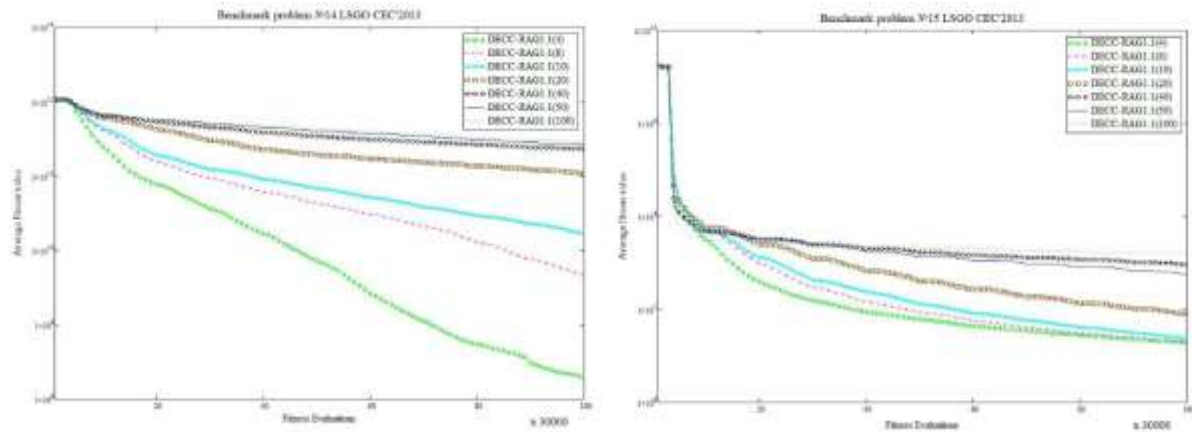


Figure 7. The convergence curve of DECC-RAG1.1(m) algorithms on F14 and F15.

We have also compared the DECC-RAG1.1 with the best settings and some the state-of-the-art LSGO approaches, such as DECC_RDG [14], CRO-SL with LS [15], DECC-CG [15], RVGDE [16], DECC-G [16]. The results are presented in table 4 and figure 8. The cells highlighted with bold corresponds to the best algorithm.

Table 4. The results of comparison of DECC-RAG1.1(4) with other state-of-the-art algorithms.

No	DECC-RAG1.1(4)	DECC-RAG1(8)	RVGDE	DECC-G	DECC_RDG	CRO-SL with LS	DECC-CG
1	1.04E-20	9.96E-16	0.00E+00	2.00E-13	5.32E-01	1.28E+06	2.00E-13
2	2.36E+03	2.35E+03	1.15E+03	1.03E+03	1.29E+04	1.01E+03	1.03E+03
3	2.00E+01	2.03E+01	2.70E-13	2.85E-10	2.13E+01	2.01E+01	2.85E-10
4	6.16E+09	8.89E+09	2.79E+10	2.12E+10	4.01E+10	1.32E+10	2.12E+10
5	2.86E+06	3.76E+06	1.02E+07	7.28E+14	5.09E+06	2.54E+07	7.28E+14
6	1.06E+06	1.06E+06	1.38E+05	6.08E+04	1.06E+06	1.06E+06	6.08E+04
7	8.35E+05	1.71E+08	3.13E+08	4.27E+08	5.41E+07	1.70E+08	4.27E+08
8	1.14E+14	4.46E+14	1.57E+15	3.88E+14	4.74E+15	3.37E+14	3.88E+14
9	2.53E+08	2.34E+08	4.04E+08	4.17E+08	4.85E+08	5.29E+08	4.17E+08
10	9.45E+07	9.44E+07	1.09E+07	1.19E+07	9.44E+07	9.44E+07	1.19E+07
11	2.48E+08	2.51E+09	1.88E+10	1.60E+11	5.31E+08	2.35E+10	1.60E+11
12	9.73E+02	1.88E+03	2.71E+03	1.03E+03	3.77E+03	2.32E+03	1.03E+03
13	1.17E+08	3.36E+09	4.78E+09	3.36E+10	3.16E+09	5.18E+09	3.36E+10
14	1.59E+08	1.07E+10	7.96E+09	6.27E+11	2.50E+09	5.39E+10	6.27E+11
15	4.45E+06	1.27E+07	6.22E+06	6.01E+07	9.67E+06	1.99E+07	6.01E+07

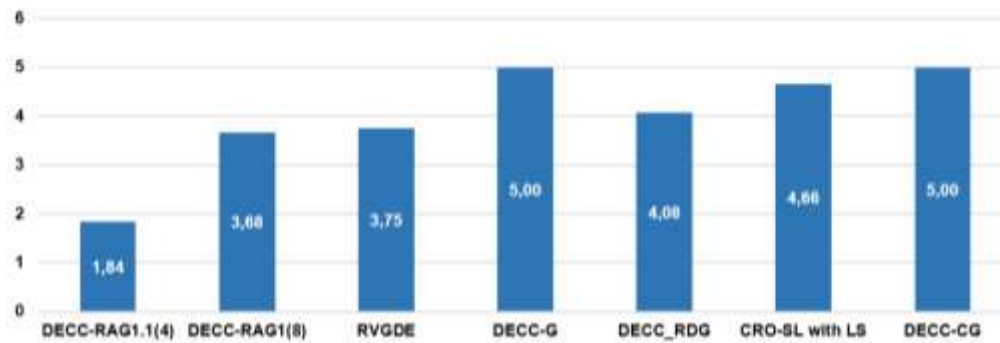


Figure 8. The ranking of RECC-RAG1.1(4) and the state-of-the-art algorithms on the CEC'2013 LSGO benchmark.

Finally, we have compared the proposed approach with the leading LSGO algorithms from the Special Session and Competition on Large-Scale Global Optimization of the IEEE WCCI 2018 conference. The results are presented in the table 5. As we can see, the BICA and MLSHADE-SPA perform better, but the difference is not too large. For proving that, we have scaled and visualized the best-found results of compared algorithms. We have also included the DECC-G algorithm, because it is used as the baseline for the CEC LSGO competitions, and it have taken the 2-nd position at the LSGO'13 and the 4-th position at the LSGO'15. Figure 9 shows the relative positions of approaches. Zero-positions correspond to the best found solution, and one-positions correspond to the worst found solution.

Table 5. The comparison of DECC-RAG1.1(4) with BICA and MLSHADE-SPA.

Nº	DECC-RAG1.1(4)	BICA	MLSHADE-SPA
1	1.04E-20	1.85E-12	6.73E-23
2	2.36E+03	8.46E-07	9.45E+02
3	2.00E+01	7.27E-01	2.00E+01
4	6.16E+09	8.85E+08	2.34E+08
5	2.86E+06	2.58E+06	1.27E+06
6	1.06E+06	1.46E+05	1.04E+06
7	8.35E+05	1.82E+05	1.37E+02
8	1.14E+14	3.78E+12	7.45E+10
9	2.53E+08	2.18E+08	1.65E+08
10	9.45E+07	1.24E+06	9.09E+07
11	2.48E+08	2.85E+07	3.81E+05
12	9.73E+02	1.40E+03	2.99E+02
13	1.17E+08	1.09E+07	2.03E+05
14	1.59E+08	4.27E+07	5.75E+06
15	4.45E+06	3.16E+06	4.35E+05

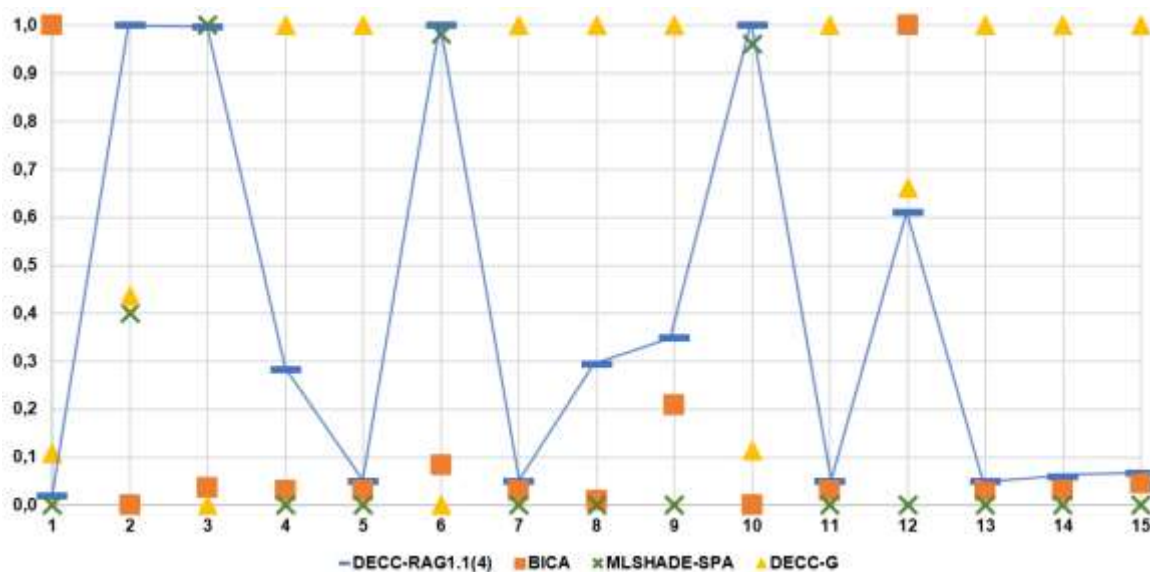


Figure 9. The relative positions of the RECC-RAG1.1(4) and top LSGO algorithms.

4. Conclusions

In this study, we have presented and have investigated a modification of the DECC-RAG algorithm with the initialization scheme based on the coordinate descent with the golden-section search. The proposed DECC-RAG1.1 demonstrated better results when using subcomponents of larger size as its parent algorithm do. At the same time, the DECC-RAG1.1 provides better convergence starting with the first generations. The experimental results have shown that the proposed approach outperforms many state-of-the-art LSGO algorithms, but yields to the leading approaches. Nevertheless, the gap between performances of algorithms is not big, and for some LSGO problems is not essential.

The DECC-RAG still has great potential for improvement, thus in our further work, we will try to implement a modification for dynamic sizing of groups on the decomposition stage.

Acknowledgments

This research is supported by the Ministry of Education and Science of Russian Federation within State Assignment № 2.1676.2017/BP.

References

- [1] Park H S and Adeli H 1997 Distributed neural dynamics algorithms for optimization of large steel structures. *J. of Structural Engineering* **123**(7) 880-8
- [2] Sarma KC and Adeli H 1996 Sparse matrix algorithm for minimum weight design of large structures. *Engineering Optimization* **27**(1) 65-85
- [3] Soegiarso R and Adeli H 1996 Optimization of large steel truss structures using standard cross sections. *Engineering J. American Institute of Steel Construction* **33** 83-94
- [4] Xu R and Wunsch D 2005 Survey of clustering algorithms. *IEEE Transactions on Neural Networks* **16**(3) 645-78
- [5] Kavrakli L E, Svestka P, Latombe J C and Overmars M H 1996 Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* **12**(4) 566-80
- [6] Vanderplaats G N 2002 *Very large scale optimization*. National Aeronautics and Space Administration (NASA: Langley Research Center)
- [7] Potter M A and D. Jong K 1994 A cooperative coevolutionary approach to function optimization in *PPSN III: Proceedings of the International Conference on Evolutionary Computation*. The Third Conference on Parallel Problem Solving from Nature (London UK:

- Springer-Verlag) pp 249-57
- [8] Vakhnin A and Sopov E 2018 Novel Method for Grouping Variables in Cooperative Coevolution for Large-scale Global Optimization Problems. *In Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics* **1** ICINCO 261-8 DOI: 10.5220/0006903102710278
 - [9] Sopov E and Vakhnin A 2018 An Investigation of Parameter Tuning in the Random Adaptive Grouping Algorithm for LSGO Problems *In Proceedings of the 10th International Joint Conference on Computational Intelligence* **1** IJCCI 255-63 DOI: 10.5220/0006959802550263
 - [10] Yang Z, Tang K and Yao X 2008 Self-adaptive differential evolution with neighborhood search, *in 2008 IEEE Congress on Evolutionary Computation* CEC 2008 1110-6
 - [11] Nesterov Y 2012 Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems *SIAM J Optim*
 - [12] Tsai C H, Kolibal J and Li M 2010 The golden section search algorithm for finding a good shape parameter for meshless collocation methods. *Eng. Anal. Bound. Elem.*
 - [13] Li X, Tang K, Omidvar M, Yang Z and Qin K 2013 Benchmark Functions for the CEC'2013 *Special Session and Competition on Large-Scale Global Optimization*
 - [14] Liu H, Wang Y, Liu L and Li X 2018 A two phase hybrid algorithm with a new decomposition method for large scale optimization *Integrated Computer-Aided Engineering Integrated Computer-Aided Engineering* **25** 349-67 DOI:10.3233/ica-170571
 - [15] Salcedo-Sanz S, Camacho-Gormez C, Molina D and Herrera F 2016 A coral reefs optimization algorithm with substrate layers and local search for large scale global optimization *Conference: 2016 IEEE Congress on Evolutionary Computation (CEC)* 3574-81
 - [16] Fei W, Yuping W and Tingting Z 2014 Variable grouping based differential evolution using an auxiliary function for large scale global optimization. *in Evolutionary Computation (CEC), 2014 IEEE Congress on OpenMP Application Programming Interface, 2015* Available from <https://www.openmp.org/specifications>