European Proceedings of
Computers and Technology
EpCT

www.europeanproceedings.com

e-ISSN: 2672-8834

DOI: 10.15405/epct.23021.25

HMMOCS 2022
International Workshop "Hybrid methods of modeling and optimization in complex systems"

SELECTIVE PRESSURE STRATEGY IN MULTILEVEL COOPERATIVE COEVOLUTION FOR SOLVING LSGO PROBLEMS

Aleksei Vakhnin (a)*, Evgenii Sopov (b)
*Corresponding author

(a) Reshetnev Siberian University of Science and Technology, Krasnoyarsk, Russian Federation, alexeyvah@gmail.com
(b) Siberian federal university, Krasnoyarsk, Russian Federation, evgeniisopov@gmail.com

Abstract

Continuous large-scale global optimization (LSGO) is a challenging task for a wide range of state-of-the art metaheuristic algorithms. This is due to the curse of dimensionality because the size of the search space grows exponentially with the increasing the number of variables. Thus, metaheuristics lose efficiency in solving LSGO problems. For increasing the performance of metaheuristic algorithms in solving LSGO problems, cooperative coevolution (CC) is used. CC-based metaheuristics have two main control parameters, which are the population size and the type of variable grouping. In this paper, a novel self-adaptive multilevel cooperative coevolution algorithm is proposed. The subcomponent optimizer of the proposed CC-based algorithm is SHADE. The proposed algorithm self-adapts the number of subcomponents and the population size during the optimization process. The complete title of the proposed algorithm is CC-SHADE-ML. We have evaluated the performance of the proposed algorithm on fifteen benchmark problems chosen from the LSGO CEC'2013 benchmark set. The performance of the CC-SHADE-ML algorithm has been evaluated using well-known mutation strategies and selection operators. We can conclude that mutation and selection operators make a significant impact in the performance of DE-based metaheuristics. All numerical experiments are proven statistically.

Keywords: Large-scale global optimization, multilevel cooperation coevolution, differential evolution

https://doi.org/10.15405/epct.23021.25
Corresponding Author: Aleksei Vakhnin
Selection and peer-review under responsibility of the Organizing Committee of the conference
eISSN: 2672-8834

1. Introduction

Metaheuristics are successfully used in solving black-box optimization problems, see in Zhan et al. (2022). Potter and Jong (1994) have proposed cooperative coevolution (CC) framework, to increase the performance of the traditional genetic algorithm (GA) to solve continuous optimization problems. The core idea of CC is to divide an optimization problem into disjoint subcomponents and evolve them independently to each other. As authors noted, any optimizer (evolutionary algorithm) can be used to optimize subcomponents. This research work formed the basis for solving LSGO problems, see in Mahdavi et al. (2015). Later, it was found that CC is a good tool for solving large-scale global optimization (LSGO) problems.

In the last two decades, many CC-based approaches have been applied to increase the performance of metaheuristics for solving real-world LSGO problems (Dong et al., 2010; Maniadakis & Trahanias, 2005). There are three main CC branches depending on variable grouping type: static, random, and learning-based variable grouping.

Static grouping, this type of variable grouping was proposed in the paper of Potter and Jong (1994) for the first time. The main idea is to set the fixed number of subcomponents and to define variables in these subcomponents. It is a preferable grouping if a relationship between variables is known. However, many real-world LSGO problems are presented by a black-box model.

Random grouping, optimized variables can be placed in different subcomponents many times during the optimization process (Yang et al., 2008a). When a predefined number of fitness evaluation is reached or other termination conditions are fulfilled, an algorithm randomly mixes variables between subcomponents, also the number of subcomponents can be changed. The main idea of random grouping is to increase the probability that two or more non-separable variables will be in the same subcomponent.

Learning grouping, this grouping is based on finding the original interaction between variables using special experiments (Omidvar et al., 2014). Usually, learning-based grouping approaches increment each variable of fitness function and to track changes. Based on the changes, variables are grouped in subcomponents.

In practice, determining the true relationship between variables is a challenging task, because of unknown optimization problem properties. In static and random grouping, setting the arbitrary group size can lead to low performance of an algorithm. On the other hand, learning grouping needs quite a lot of function evaluations (FEs) to determine true connections between variables, and there is no guarantee that an EA will perform better using the discovered true connection between variables. Thereby, there is a need to develop a self-adaptive mechanism for the automatic selection of the number of subcomponents and the population size.

The rest of the paper is organized as follows: section 2 proposes CC-SHADE-ML, contains description of selection operators and mutation schemes, section 3 contains experimental setup and results of numerical experiments. Section 4 concludes the paper.

https://doi.org/10.15405/epct.23021.25
Corresponding Author: Aleksei Vakhnin
Selection and peer-review under responsibility of the Organizing Committee of the conference
eISSN: 2672-8834

2. Problem Statement

An LSGO problem can be stated as a continuous optimization problem:

$$f(\bar{x}) \to \min_{\bar{x} \in D \subset R^n}, f^* = f(x^*) \leq f(\bar{x}), x \in D \subset R^n,$$

where $f(\bar{x})$ is an fitness function to be minimized, $f: R^n \to R^1$, $\bar{x}$ is an n-dimensional vector of continuous variables, D is the search space defined by box constrains $x_i^l \leq x_i \leq x_i^u, i = \overline{1,n}$, x_i^l and x_i^u are the lower and upper borders of the i-th variable, respectively, x^* is a global optimum. It is assumed that the fitness function is continuous.

3. Research Questions

In course of the study the following questions were raised:

- What is the role of cooperative coevolution approach in solving LSGO problems?
- What is the role of selection operators in CC-based evolutionary algorithms in solving LSGO problems?
- What is the role of mutation schemes in CC-based evolutionary algorithms in solving LSGO problems?

4. Purpose of the Study

The answers to the issues raised above will help achieve the goal and contribute to the development of recommendations on parameters selection in CC-based evolutionary algorithms for solving LSGO problems.

5. Research Methods

5.1. CC-SHADE-ML

This section gives the description of the proposed algorithm in detail. The algorithm combines cooperative coevolution, the multilevel self-adaptive approach for selecting the number of subcomponents and the number of individuals, and SHADE, proposed by Tanabe and Fukunaga (2013), and it is titled as CC-SHADE-ML. An original idea of using a multilevel approach is proposed in the MLCC algorithm, proposed by Yang et al. (2008b). Before the main cycle of the optimization process, it is needed to define a set of integer values, each value is a level of the algorithm with a unique number of subcomponents. The optimization process is divided into a predefined number of cycles. In each cycle, the number of subcomponents is selected according to the performance of a decomposition in the previous cycles. In each cycle, variables are divided in subcomponents randomly. To evaluate the performance of each level, we use the following formula:

$$performance_i = \left(f_i^{before} - f_i^{after}\right)/f_i^{before},$$

https://doi.org/10.15405/epct.23021.25
Corresponding Author: Aleksei Vakhnin
Selection and peer-review under responsibility of the Organizing Committee of the conference
eISSN: 2672-8834

here f_i^{before} and f_i^{after} the best-found fitness values before and after the optimization cycle in the *i*-th level. If $performance_i$ is equal to a value which is less than 1E-4, then it is set to 1E-4. In the beginning of the optimization process, all values of $performance_i$ are set to 1.0. The selection probability of each level is calculated using the following formula:

$$p_i = \frac{e^{k*performance_i}}{\sum_{j=1}^{t} e^{k*performance_i}}, i = \{1,2, \dots, t\},$$

here, *k* is a control parameter and it was set to 7 according to our numerical experiments. Also, authors of MLCC have noted that in their case, when *k* equals to 7, MLCC shows better performance. In each new optimization cycle, it is needed to recalculate the performance of the last applied level and to select a new level based on new probabilities. In this study, the number of cycles was set to 50. SaNSDE, proposed by Yang et al. (2007), is used in MLCC as optimizers and has the fixed population size. The population size, in population-based searching algorithms, is one the most important parameter. The proposed algorithm uses the same idea in selection the number of individuals as MLCC uses in selection the number of subcomponents.

The proposed CC-SHADE-ML algorithm differs from MLCC in the following. It uses SHADE instead of SaNSDE and extends MLCC by applying a self-adaptation multilevel (ML) approach for the population size. The pseudo-code of CC-SHADE-ML is presented in Table 1.

5.2. Operators of Selection in DE

In addition to the population size and the number of subcomponents, the type of mutation and selection also significantly affects the efficiency of CC-based evolutionary algorithms. In the paper, we investigate the performance of the proposed algorithm with different mutation schemes and selection operators.

Proportional selection. Every individual can become a parent with probability which is proportional to its fitness value. In case of solving minimization problem, the probability of selecting an individual is equal to:

$$p_i = \frac{f_{max} - f_i}{\sum_{j=1}^{pop\_size}(f_{max} - f_j)},$$

here f_i is the value of the fitness function of i-th individual. f_{max} is the maximum value of fitness function in the current generation in the population. This type of selection demonstrates low performance because of fast convergence to a local optimum. This happens because the best individual after some generations will quickly dominate the population.

Linear rank selection. This is mostly used selection strategy in population-based algorithms. Individuals should be sorted according to their fitness values and get ranks using the following formula: $rank_i = pop\_size - i$, here $rank_i$ is the rank of i-th individual. The selection probability linearly depends on the value of rank.

Exponential rank selection. Another well-known selection is based on exponential distribution. The rank of fitness values is calculated using the following formula:

https://doi.org/10.15405/epct.23021.25
Corresponding Author: Aleksei Vakhnin
Selection and peer-review under responsibility of the Organizing Committee of the conference
eISSN: 2672-8834

$$rank_i = e^{\frac{-ki}{pop\_size}},$$

here *k* is a parameter of the selection.

The probability of selecting individuals using liner and exponential rank selections is equal to:

$$p_i = \frac{rank_i}{\sum_{j=1}^{pop\_size}(rank_j)}.$$

Tournament selection. It is another well-known selection in population-based algorithms. It is needed to define the size of tournament (*t*), it should be $2 \leq t \leq pop\_size$. The operator of selection randomly chooses *t* different individuals from the current population and selects the one who has the best fitness function value.

5.3. Schemes of Mutation in DE

In the study, the following mutation schemes have been used:

- DE/rand/1 – $v_j = x_{r1,j} + F\left(x_{r2,j} - x_{r3,j}\right)$
- DE/rand/2 – $v_j = x_{r1,j} + F\left(x_{r2,j} - x_{r3,j}\right) + F\left(x_{r4,j} - x_{r5,j}\right)$
- DE/best/1– $v_j = x_{b,j} + F\left(x_{r2,j} - x_{r3,j}\right)$
- DE/best/2 – $v_j = x_{b,j} + F\left(x_{r2,j} - x_{r3,j}\right) + F\left(x_{r4,j} - x_{r5,j}\right)$
- DE/current-to-rand/1 – $v_j = x_{i,j} + F\left(x_{r2,j} - x_{i,j}\right) + F\left(x_{r4,j} - x_{r5,j}\right)$
- DE/current-to-best/1 – $v_j = x_{i,j} + F\left(x_{b,j} - x_{i,j}\right) + F\left(x_{r4,j} - x_{r5,j}\right)$
- DE/current-to-pbest/1 – $v_j = x_{i,j} + F\left(x_{pb,j} - x_{i,j}\right) + F\left(x_{r4,j} - x_{r5,j}\right)$

However, random uniform distribution is almost always used to generate *r1*, *r2*, *r3*, *r4*, and *r5* indexes. r1, r2, and r4 indexes are generated from the main population, *r3* and *r5* can be taken from the main population and the archive.

Table 1. Pseudocode of CC-SHADE-ML

Line	Pseudocode
1	Generate an initial population randomly;
2	Initialize performance vectors, *CC_performance* and *pop_performance;*
3	*FEs_cycle_init = FEs_total*/ cycles_number;
4	**while** (FEs>0) **do**
5	*FEs_cycle = FEs_cycle_init;*
6	Randomly shuffle indices;
7	Randomly select *CC_size* and *pop_size* from *CC_performance* and *pop_performance*;
8	**while** (FEs_cycle>0) **do**
9	Find the best fitness value before the optimization cycle *f_best_before*;
10	**for** i=1 to *CC_size*
11	Evaluate the i-th subcomponent using the SHADE algorithm;
12	**end for**

https://doi.org/10.15405/epct.23021.25
Corresponding Author: Aleksei Vakhnin
Selection and peer-review under responsibility of the Organizing Committee of the conference
eISSN: 2672-8834

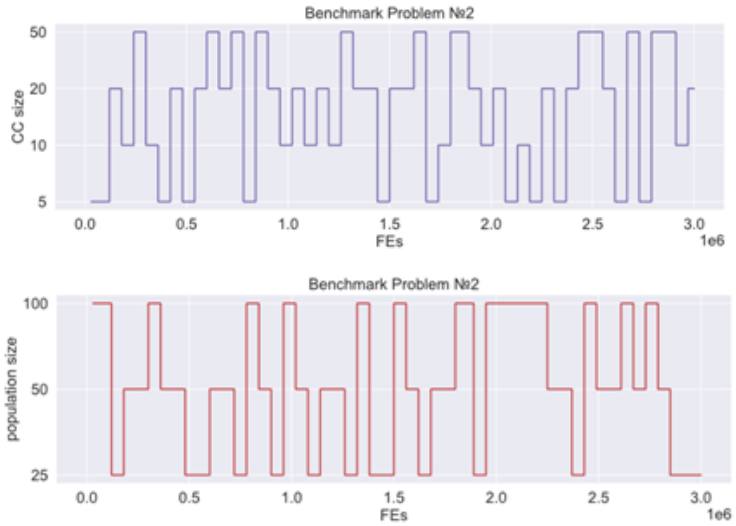
13 Find the best fitness value after the optimization cycle *f_best_after*;
14 Evaluate performance of *CC_size* and *pop_size* using eq. (2);
15 Update *CC_performance* and *pop_performance*;
16 **end while**
17 **end while**
18 Generate an initial population randomly;

6. Findings

In this section, experimental setup and results are presented. In the paper, the LSGO CEC'13 benchmark set, Li et al. (2013), has been used for evaluating the algorithm performance. Numerical experiments have been conducted according to the conditions of the LSGO CEC'2013 competition. This set consists of 15 continuous optimization problems. The number of variables of each problem is equal to 1000. The maximum number of fitness evaluations is 3.0E+06 in each independent run. The comparison of algorithms is based on the mean values, which are obtained in 25 independent runs. Numerical experiments have been performed on a computation cluster based on eight personal computers. The total number of computational threads is 128.

The performance of CC-SHADE-ML has been evaluated. All combinations of mutation schemes and selection operators have been compared between each other. The proposed CC-SHADE-ML algorithm has the following parameters. The set of subcomponents is equal to {5,10,20,50}. The set of the population size is equal to {25,50,100}. In SHADE, the size of the archive is 100. The number of cycles is set to 50. According to our numerical experiments, this value for the number of cycles performs better than other tested values. Thus, in each cycle, CC-SHADE-ML evaluates 6.0E+4 FEs.

Figure 1 presents how CC-SHADE-ML switches the number of subcomponents and the population size during the optimization process in one independent run on F_2 benchmark problem using current-to-pbest/1 mutation scheme and tournament selection. The x-axis shows FEs, the y-axis shows the levels of subcomponents and population size.

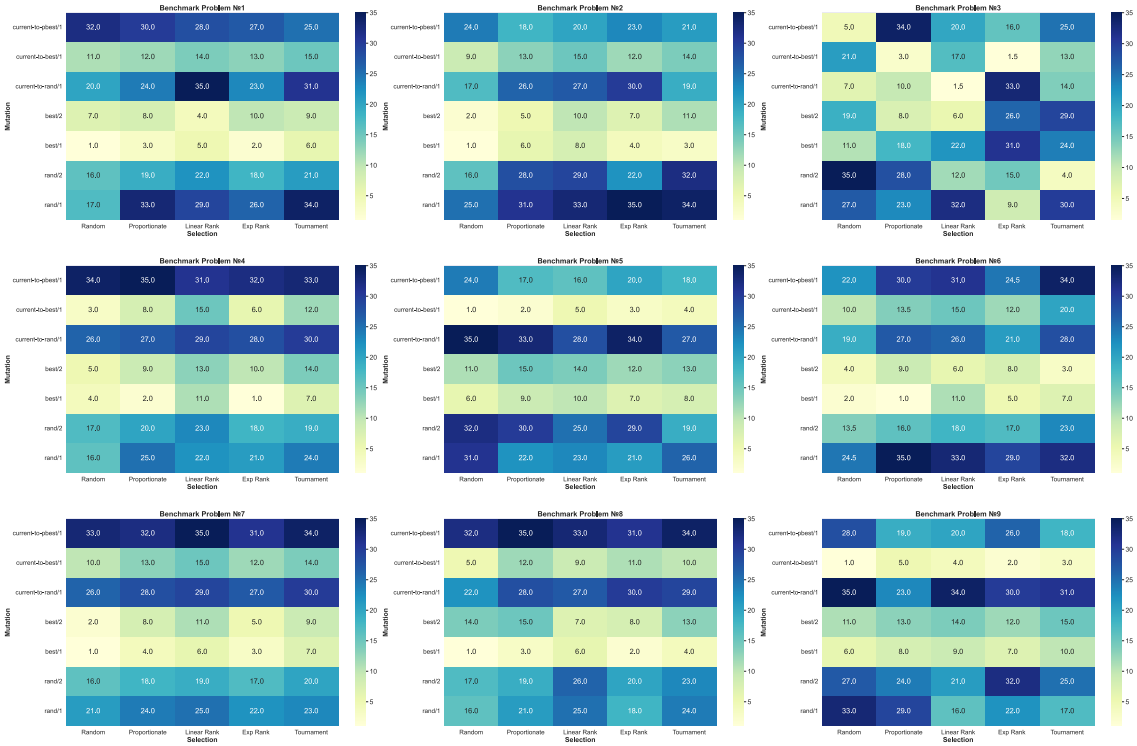


https://doi.org/10.15405/epct.23021.25
Corresponding Author: Aleksei Vakhnin
Selection and peer-review under responsibility of the Organizing Committee of the conference
eISSN: 2672-8834

Figure 1. The self-adaptation curves of the number of subcomponent (top) and the population size (bottom) using CC-SHADE-ML algorithm on F_2 benchmark problem

Figure 2 shows heat maps for each benchmark problem and each combination of mutation and selection. The x-axis denotes the selection type, the y-axis denotes the mutation scheme. The performance of each combination of mutation and selection is presented by rank. The biggest number corresponds to the best average fitness value obtained in 25 independent runs. Dark blue (dark) and white color (light) colors denote the best and worst combination, respectively. The rank distribution in heat maps depends on the optimization problem.

Figure 3 shows the rank sum for the algorithm's parameters for all benchmark problems (left) and the sum of scores based on comparison using the Wilcoxon rank-sum test (left). The x-axis denotes the selection type, the y-axis denotes the mutation scheme. The highest sum is the best achieved result. The dark color denotes the worst on average combination of parameters. The results in the left picture are based on the results from Figure 2. The results in the right picture are based on pairwise comparison corresponding combinations with other combinations on all benchmark problems. If statistical significance is found between two combinations on a problem, then add one score to the best-performed combination and remove one score from the other.



https://doi.org/10.15405/epct.23021.25
Corresponding Author: Aleksei Vakhnin
Selection and peer-review under responsibility of the Organizing Committee of the conference
eISSN: 2672-8834

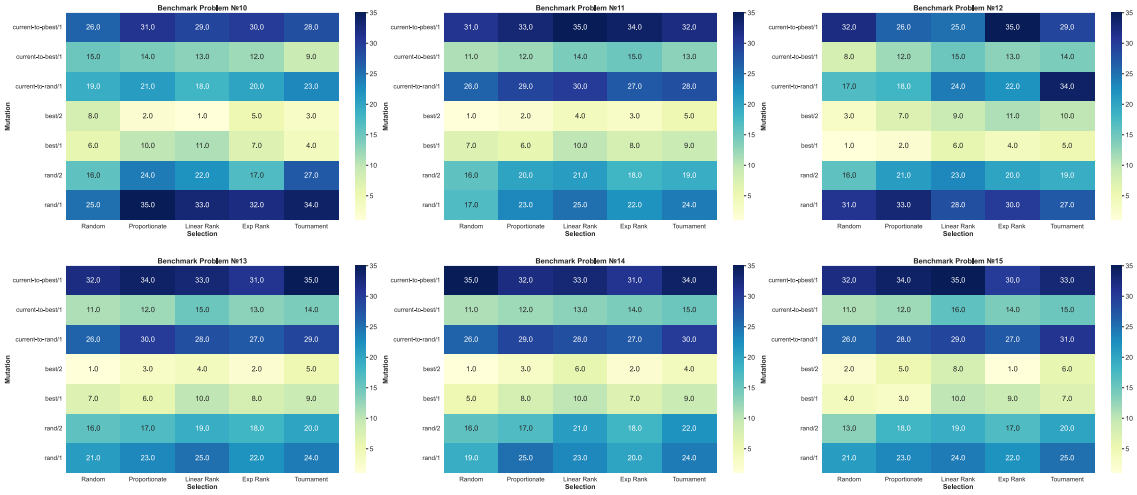


Figure 2. Ranking the CC-SHADE-ML algorithm with different mutation schemes and selection strategies on each benchmark problem from LSGO CEC'2013

7. Conclusions

The paper proposes the CC-SHADE-ML algorithm for solving large-scale global optimization problems. CC-SHADE-ML changes the number of subcomponents and the number of individuals based on their performance. We have investigated the proposed metaheuristic using the LSGO CEC'13 benchmark set with fifteen benchmark problems. Numerical experiments show that the best-performed mutation scheme, on average, is the current-to-pbest/1, the best-performed selections are tournament and proportionate selection. All experiments are proven by the Wilcoxon test. CC-SHADE-ML has a high potential for enhancement. In our further works, we will improve the performance of the CC-SHADE-ML algorithm by modifying the optimizer and tuning the parameters of selections. Also, we will test the use of hybrid memetic metaheuristics for solving continuous large-scale global optimization problems.

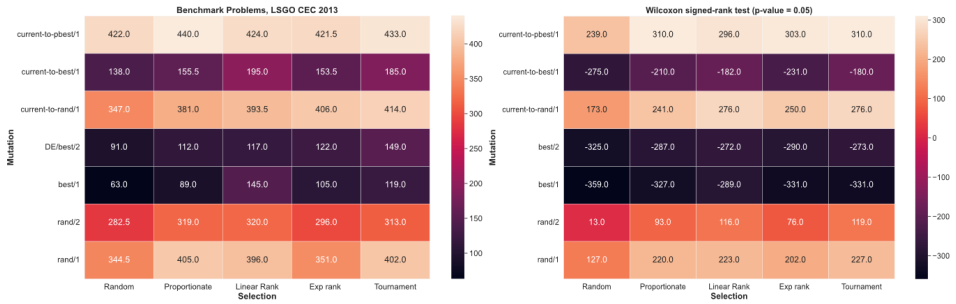


Figure 3. The ranks sum (left) and the sum of scores based of Wilcoxon test (right) of the CC-SHADE-ML algorithm with different mutation schemes and selection strategies on each benchmark problem from LSGO CEC'2013

https://doi.org/10.15405/epct.23021.25
Corresponding Author: Aleksei Vakhnin
Selection and peer-review under responsibility of the Organizing Committee of the conference
eISSN: 2672-8834

Acknowledgments

This research was funded by the Ministry of Science and Higher Education of the Russian Federation, Grant No. 075-15-2022-1121.

References

Dong, X., Yu, H., Ouyang, D., Cai, D., Ye, Y., & Zhang, Y. (2010). Cooperative coevolutionary genetic algorithms to find optimal elimination orderings for bayesian networks. *In 2010 IEEE fifth international conference on bio-inspired computing: Theories and applications (BIC-TA)* (pp. 1388-1394). IEEE. https://doi.org/10.1109/BICTA.2010.5645605

Li, X., Tang, K., Omidvar, M. N., Yang, Z., & Qin, K. (2013). Benchmark functions for the CEC'2013 special session and competition on large scale global optimization. *Technical Report,* 1-23.

Mahdavi, S., Shiri, M. E., & Rahnamayan, S. (2015). Metaheuristics in large-scale global continues optimization: A survey. *Information Sciences, 295,* 407-428. https://doi.org/10.1016/j.ins.2014.10.042

Maniadakis, M., & Trahanias, P. (2005). A hierarchical coevolutionary method to support brain-lesion modelling. Proceedings. *2005 IEEE International Joint Conference on Neural Networks* (pp. 434-439). https://doi.org/10.1109/ijcnn.2005.1555870

Omidvar, M. N., Li, X., Mei, Y., & Yao, X. (2014). Cooperative Co-Evolution with Differential Grouping for Large Scale Optimization. *IEEE Transactions on Evolutionary Computation, 18*(3), 378-393. https://doi.org/10.1109/tevc.2013.2281543

Potter, M. A., & Jong, K. A. (1994). A cooperative coevolutionary approach to function optimization. *Parallel Problem Solving from Nature — PPSN III, 666,* 249-257. https://doi.org/10.1007/3-540-58484-6_269

Tanabe, R., & Fukunaga, A. (2013). Success-history based parameter adaptation for differential evolution. In 2013 IEEE congress on evolutionary computation (pp. 71-78). IEEE. https://doi.org/10.1109/cec.2013.6557555

Yang, Z., Tang, K., & Yao, X. (2007). Differential evolution for high-dimensional function optimization. *In 2007 IEEE congress on evolutionary* computation (pp. 3523-3530). IEEE. https://doi.org/10.1109/CEC.2007.4424929

Yang, Z., Tang, K., & Yao, X. (2008a). Large scale evolutionary optimization using cooperative coevolution. *Information Sciences, 178*(15), 2985-2999. https://doi.org/10.1016/j.ins.2008.02.017

Yang, Z., Tang, K., & Yao, X. (2008b). Multilevel cooperative coevolution for large scale optimization. In *2008 IEEE congress on evolutionary computation (IEEE World Congress on Computational Intelligence)* (pp. 1663-1670). IEEE. https://doi.org/10.1109/CEC.2008.4631014

Zhan, Z.-H., Shi, L., Tan, K. C., & Zhang, J. (2022). A survey on evolutionary computation for complex continuous optimization. *Artificial Intelligence Review, 55*(1), 59-110. https://doi.org/10.1007/s10462-021-10042-y