An Empirical Study of Cooperative Frequency in Distributed Cooperative Co-evolution

Ling-Yu Li
South China University of Technology
Guangzhou, China

Wen-Jie Ou
South China University of Technology
Guangzhou, China

Xiao-Min Hu
Guangdong University of Technology
Guangzhou, China

Wei-Neng Chen
South China University of Technology
Guangzhou, China
cwnraul634@aliyun.com

An Song
South China University of Technology
Guangzhou, China

ABSTRACT

Cooperative co-evolution (CC) is an effective technique for optimizing high dimensional problems. The scalability and efficiency of CC can be further improved by combing with distributed computing, i.e., distributed CC (dCC). How to balance the cooperation and communication among sub-problems is an important issue in dCC. In this paper, we construct series of large-scale benchmark functions, and conduct experimental studies to investigate the effect of different cooperative frequency in dCC. Experiments suggest that frequency of cooperation will not affect the performance of fully separable and partially separable problems unless frequent coordination among sub-populations consumes too many limited computing resources. And overlapping problems need a moderate cooperative frequency to get better performance.

CCS CONCEPTS

• **Theory of computation → Design and analysis of algorithms**; *Mathematical optimization*; *Continuous optimization*;

KEYWORDS

Cooperative co-evolution, distributed computation, cooperative frequency

ACM Reference Format:
Ling-Yu Li, Wen-Jie Ou, Xiao-Min Hu, Wei-Neng Chen, and An Song. 2021. An Empirical Study of Cooperative Frequency in Distributed Cooperative Co-evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference 2021 (GECCO '21).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3449726.3459501

1 INTRODUCTION

Distributed cooperative co-evolution(dCC) is an efficient technique to optimize large-scale problems[1],[2]. In dCC, an optimization problem is decomposed into several sub-components and these sub-components are handled by different processors. In most real-world problems, the grouping strategy is difficult to achieve almost no interaction between the sub-problems. Thus, during the evolutionary process in dCC, processors need to exchange information in a set interval to ensure the cooperation among related sub-populations. The interval reflects the cooperative frequency among sub-populations. Researchers usually use a high cooperative frequency to ensure performance. Theoretically, frequent communication among computing nodes will cause large cost of time and computing resource. How to balance the communication and cooperation becomes an important issue in dCC[2].

This paper intends to study the influence of cooperative frequency in dCC on different types of problems. We construct three types of benchmark functions and a series of cooperative frequencies are set for all constructed benchmark functions to conduct experiments. Experimental results show that different problems need different cooperative frequency to achieve a balance between low resource consumption and good performance.

2 LARGE-SCALE GROUPING FUNCTIONS

In this work, to explore the effect of cooperative frequency on different kinds of optimization problems, we construct three kinds of benchmark functions: *fully separable*, *partially separable*[6][7], and *overlapping*. Four base functions $Elliptic$, $Rastrigin$, $Schwefel$ and *Rosenbrock* are used to construct 14 benchmark functions[4]. Each function has 5 sub-components. F_1-F_2 are fully separable functions and F_3-F_6 are partially separable functions. The constructed partially separable functions are separable globally but non-separable locally. And the constructed overlapping functions are ring-topology. F_7-F_{10} are overlapping functions with consistent overlaps and F_{11}-F_{14} are overlapping functions with conflicted overlaps. In overlapping functions, the overlapping size is set to 10.

3 EXPERIMENTS

We use the master-slave[3] paralleling model to implement a dCC algorithm in a synchronous way. In this work, CLPSO[8] is adopted as an evolutionary optimizer. We use the single best collaborator selection strategy[5] to construct the context vector $\mathbf{b} = (b_1, ..., b_i, ..., b_s)$, where b_i is the best individual from i-th sub-population and s is the number of sub-components. In our experiments, after each communication, master node needs to gather recent best individuals from all sub-populations to the context vector $\mathbf{b}$ and then send the new

GECCO '21, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07…$15.00
https://doi.org/10.1145/3449726.3459501

Table 1: Summary of Suitable Cooperative Frequency of Different Problems

Type of Function	Suitable Cooperative Frequency
Fully separable	Cooperate once in an evolution.
Partially separable	Cooperate once in an evolution.
Fully separable	The cooperative frequency should not be too low or too high.

b to slave nodes. Slave nodes need to execute EAs and calculate the fitness of individuals.

In these experiments, the maximum number of particles NP is set to 100. As aforementioned, the number of sub-populations s is 5. The maximum evolutionary generations $maxG$ of each sub-population is 6000 ($maxG = MAX\_FEs/(s * NP)$). To investigate the effect of cooperative frequency in different benchmark functions, we conduct experiments on constructed functions with a set of cooperative frequency cf [1,2,6,12,60,120,600,6000]. The cf means sub-populations cooperate cf times in an evolution. The results will be discussed as following according to different types of benchmark functions. Additionally, the data in all following figures are the results obtained by taking the average of 30 independent experiments.

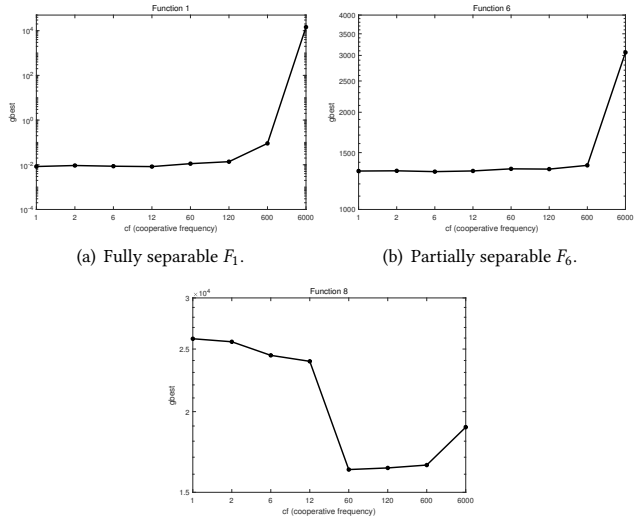
1)*Fully separable* and *partially separable*: As shown in Fig.1(a) and Fig.1(b), when the cooperative frequency is very high, performance of fully separable and partially separable problems will be bad. If the cooperative frequency is not too high, these two types of problems can be optimized to the same level in any cooperative frequency.

In fully separable problems, all variables are independent of each other. Theoretically, the cooperative frequency in dCC will not affect the performance of fully separable problems. But most evolutionary algorithms update particles by comparing fitness values of two generations. Therefore, after each cooperation, each sub-population must recalculate the fitness value of all particles to avoid interference caused by the change of **b**. So, when the cooperative is very high, a lot of evaluation times will be wasted. In partially separable, the variables in the sub-problems interact with each other, but the sub-problems are still independent of each other. Due to sub-problems will not disturb each other, partially separable do not need any cooperation in dCC, too.

2)*Overlapping*: Overlapping functions need a moderate cooperative frequency to achieve a balance between performance and resource consumption (shown in Fig.1(c)). For overlapping functions, frequent cooperation will not only cause good sub-population to be disturbed by other sub-populations, but also cause consumption of computing resource. However, the overlapping problem requires communication to ensure that the value of the overlap is conducive to the complete problem. So, when optimizing overlapping functions, the frequency cannot be set too high or too low.

4 CONCLUSION

In this paper, experiments are conducted to investigate the effect of cooperative frequency in dCC. The results are summarized in Table 1.



(a) Fully separable F_1. (b) Partially separable F_6.

(c) Overlapping F_8.

Figure 1: Results of large-scale grouping functions in different cooperative frequency.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 61772142, in part by the Pearl River S&T Nova Program of Guangzhou No. 201806010059, and in part by Natural Science Foundation of Guangdong No. 2019A15150-11270. The research team was supported by the Guangdong-Hong Kong Joint Innovative Platform of Big Data and Computational Intelligence No. 2018B050502006, and the Guangdong Natural Science Foundation Research Team No. 2018B030312003. (Corresponding Author: Xiao-Min Hu)

REFERENCES

[1] Yue-Jiao Gong, Wei-Neng Chen, Zhi-Hui Zhan, Jun Zhang, Yun Li, Qingfu Zhang, and Jing-Jing Li. 2015. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing* 34 (2015), 286–300. https://doi.org/10.1016/j.asoc.2015.04.061

[2] Y. Jia, W. Chen, T. Gu, H. Zhang, H. Yuan, S. Kwong, and J. Zhang. 2019. Distributed Cooperative Co-Evolution With Adaptive Computing Resource Allocation for Large Scale Optimization. *IEEE Transactions on Evolutionary Computation* 23, 2 (2019), 188–202. https://doi.org/10.1109/TEVC.2018.2817889

[3] X. Lei and F. Zhang. 2007. Parallel Particle Swarm Optimization for Attribute Reduction. In *Eighth Acis International Conference on Software Engineering*.

[4] Xiaodong Li, Ke Tang, Mohammmad Nabi Omidvar, Zhenyu Yang, and Kai Qin. 2013. Benchmark Functions for the CEC'2013 Special Session and Competition on Large-Scale Global Optimization. (01 2013).

[5] X. Li and X. Yao. 2012. Cooperatively Coevolving Particle Swarms for Large Scale Optimization. *IEEE Transactions on Evolutionary Computation* 16, 2 (2012), 210–224. https://doi.org/10.1109/TEVC.2011.2112662

[6] X. Ma, X. Li, Q. Zhang, K. Tang, Z. Liang, W. Xie, and Z. Zhu. 2019. A Survey on Cooperative Co-Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* 23, 3 (2019), 421–441. https://doi.org/10.1109/TEVC.2018.2868770

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

[8] Q. Qin, S. Cheng, Q. Zhang, L. Li, and Y. Shi. 2016. Particle Swarm Optimization With Interswarm Interactive Learning Strategy. *IEEE Transactions on Cybernetics* 46, 10 (2016), 2238–2251. https://doi.org/10.1109/TCYB.2015.2474153