Archive-based Cooperative Coevolution Genetic Programming for Workflow Scheduling

Yuanzi Hong[1], Wei-Li Liu[1]*, Jinghui Zhong[2]*, Peng Liang[1], Jianhua Guo[1], Chunying Li[3,1]
[1]Guangdong Polytechnic Normal University, Guangzhou, China
Email: hhnun2588@163.com, liuweili@gpnu.edu.cn, liangpeng@gpnu.edu.cn, guojianhua@gpnu.edu.cn, gscy.li@qq.com
[2]South China University of Technology, Guangzhou, China
Email: jinghuizhong@gmail.com
[3]Guangdong Provincial Key Laboratory of Intellectual Property & Big Data, Guangzhou, China

***Abstract*—Workflow Scheduling Problem (WSP) is a well-known combinatorial optimization problem, which requires allocating tasks to available computing resources to maximize system efficiency, performance, or to meet specific requirements of service quality. The cooperative coevolution hyper-heuristic method based on genetic programming is a promising approach for addressing the WSP, attracting growing attention from researchers. However, this approach still faces the challenge of individual selection bias in the fitness evaluation when coevoluting two sub-populations. To address the above issue, this paper proposes an Archive-based Cooperative Coevolution GP (A-CCGP), which leverages an archive population to improve the quality of fitness evaluation. In addition, an adaptive mechanism is proposed to dynamically adjust the training set during the evolution to reduce the computational cost of fitness evaluation. Experimental results have validated the effectiveness of the proposed A-CCGP algorithm, in comparison with several state-of-the-art algorithms.**

***Index Terms*—Workflow scheduling, genetic programming, cooperative coevolution**

I. INTRODUCTION

The Workflow Scheduling Problem (WSP) is crucial in both scientific and industrial communities, and it aims to optimize the task execution sequence and resource allocation to enhance work efficiency under several constraints. So far, existing methods for solving WSP can be roughly classified into the guided random search methods [1], [2] and the heuristic-based methods [3-7]. However, these methods require significant computational cost or rely heavily on human empirical knowledge, hindering practical applications. To automatically design high-level heuristics, Xiao et al. [8] proposed a Cooperative Coevolution Genetic Programming (CCGP) algorithm and achieved encouraging results. Nonetheless, the efficiency of its fitness evaluation method is still not high enough for practical use because of individual selection bias. Whenever a candidate sub-solution of a sub-population needs to be evaluated, it should be firstly combined with the best individual of another sub-population to form a complete solution. However, the complete solution formed in this way is not necessarily the best combination and may ignore the diversity of the overall population, making the evolutionary algorithm more likely to fall into local optima.

To solve these issues, this paper proposes an Archive-based Cooperative Coevolution Genetic Programming (A-CCGP) by using the archive strategy [9], [10], which is utilized for storing historically superior solutions. The proposed A-CCGP enhances the fitness evaluation in two following aspects.

1) An archive population is introduced to overcome the individual selection bias when combining individuals of two sub-populations in CCGP. Besides, the archive population is dynamically maintained to make the pairing more accurate.
2) The training set is adaptively adjusted to reduce the computational cost. Specifically, the evaluation process is divided into early, middle, and late stages, and the training set is adjusted to tiny, half, and complete, respectively.

In the experiments, the proposed A-CCGP is tested and compared with the CCGP [8] and several state-of-the-art algorithms such as the Heterogeneous Earliest Finish Time (HEFT) [3], Lookahead [4], the Predict Earliest Finish Time (PEFT) [5], the Performance Effective Task Scheduling (PETS) [6] and the Minimizing Schedule Length (MSL) [7] on randomly generated workflows and four real-world workflows.

The rest of the paper is organized as follows: Section II presents a review of the related works. Section III, formulates the workflow scheduling model, Section IV provides a detailed description of the proposed algorithm, while Section V presents the experimental study. Conclusion is drawn in Section VI.

II. RELATED WORK

This section will overview two types of methods commonly used to solve WSP and introduce the CCGP algorithm [8]. One type is the guided random search methods [1], [2], which enhance the search efficiency by combining random exploration of the search space with guided strategies. These methods effectively overcome local optima stagnation in the search space and improve performance in solving WSP. However,

This work was supported in part by the Guangdong Basic and Applied Basic Research Foundation under Grants 2023A1515012291 and 2021A1515110072; in part by the National Natural Science Foundation of China under Grant 62072123; in part by the Scientific Research Platforms and Projects of Guangdong Provincial Education Department under Grants 2022ZDZX1012 and 2020ZDZX305; and in part by the Research Startup Funds of Guangdong Polytechnic Normal University under Grant 2021SDKYA130. (*Corresponding authors: Wei-Li Liu; Jinghui Zhong.)

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00047

they may require high computational cost when dealing with large search space. Additionally, in case of workflow changes, these methods need re-execution to find a new solution, limiting their practical applicability.

The other type is the heuristic-based methods, which involve designing heuristic rules to guide the search process. List-based heuristics [3-7] are the most famous and popular category within this type, usually divided into task priority sorting and resource selection phase. Tasks are sorted based on priority, with high-priority task will be scheduled first and then the selected task will be assigned to the highest-priority resource. Overall, existing heuristic-based methods for WSP are often developed in an ad-hoc manner, with rules derived from a deep understanding of the specific problem domain.

For the CCGP algorithm, the cooperative coevolution architecture is integrated into genetic programming to learn coadapted high-quality sub-heuristics efficiently. Its heuristics are learned automatically and can be applied to different situations without re-execution, making it more flexible and convenient for practical applications. However, the fitness evaluation efficiency of the CCGP algorithm is not high enough due to the individual selection bias in combining two sub-populations. Therefore, there is a need to improve the fitness evaluation strategy. To address the above limitations of existing methods, this paper proposes the A-CCGP algorithm to solve WSP.

III. PROBLEM DEFINITION

A Direct Acyclic Graph (DAG) is a widely used tool for representing workflow models. A DAG, denoted as *(V, E)*, consists of tasks $V = \{v_1, v_2, ..., v_n\}$ and dependencies between the tasks *E*. In the DAG structure, a task without any predecessors is denoted by an entry task (t_{entry}), while a task without any successors is denoted by an exit task (t_{exit}).

The makespan of a workflow, used to evaluate the performance of the proposed algorithm, is defined as the maximum completion time of all tasks in the workflow. Let AFT(t_{exit}) denote the actual finish time of task t_{exit}. Then, the makespan of the workflow can be defined by

$$makespan = max\{AFT(t_{exit})\} \tag{1}$$

Since the makespan of different instances may vary significantly, it is necessary to normalize the makespan concerning a lower bound. Therefore, this paper adopts the Schedule Length Ratio (SLR) index [5] to provide a more intuitive measure of the algorithm performance, as defined by

$$SLR = \frac{makespan}{\sum_{t_i \epsilon CP_{MIN}} min_{p_j \epsilon P}\{\omega_{i,j}\}} \tag{2}$$

where $\omega_{i,j}$ represents the execution time of task t_i on resource p_j, and CP_{MIN} represents the set of critical path tasks. Equation (2) states that a lower SLR value indicates a better solution.

IV. PROPOSED ALGORITHM

Our proposed A-CCGP algorithm is based on the recently published CCGP algorithm [8] and incorporates a popular archive policy. In the CCGP algorithm, the high-level heuristic consists of two interacting and co-adaptive sub-heuristics, TSR and RSR. The TSR is used to select a ready task for scheduling, while the RSR is used to allocate resources to perform the selected task. Correspondingly, two sub-populations, P_{TSR} and P_{RSR}, work cooperatively to evolve TSR and RSR, respectively. P_{TSR} is responsible for evolving TSR, while P_{RSR} focuses on evolving RSR. To evaluate a candidate sub-solution of one sub-population, it will be paired with the best individual of the other sub-population to form a complete solution. In this way, the fitness value of a sub-solution is determined by the fitness value of the corresponding complete solution. Meanwhile, the fitness value of the complete solution (a pair of Γ_{TSR} and Γ_{RSR}) is defined as the average SLR over a set of training instances *I*, as expressed by

$$f(\Gamma_{TSR}, \Gamma_{RSR}) = \frac{\sum_{j=1}^{|I|} SLR_{I_j}}{|I|} \tag{3}$$

However, pairing only with the best individual may result in a lack of accuracy in the combination and therefore decrease the efficiency of fitness evaluation. To address this issue, this paper employs the archive strategy, which involves creating an archive population to store previously outstanding individuals of the two sub-populations. The TSR and RSR individuals in the archive population represent sub-solutions. Additionally, an adaptive scheme is adopted for adjusting the training set *I* in (3), so as to reduce computational cost.

A. Overall framework

Fig. 1 illustrates the overall framework of A-CCGP. Initially, generate two sub-populations (P_{TSR} and P_{RSR}), followed by randomly selecting individuals from these two sub-populations to initialize the archive population. Each individual represents a scheduling heuristic. Then the fitness values of individuals in the two sub-populations are traditionally evaluated.

In each generation, the training set is adaptively adjusted, followed by applying the proposed archive-based fitness evaluation to evaluate individuals of each sub-population. Specifically, the process begins by checking if the conditions for combining with the archive population are met. If so, fitness evaluation is performed in conjunction with the archive population, followed by updating the archive population. Otherwise, fitness evaluation is directly carried out between the two sub-populations. More details of the archive-based fitness evaluation will be provided in Section IV-B.

Next, the two sub-populations undergo evolution through mutation, crossover, and selection operations. These genetic operations correspond to the search mechanism based on differential evolution in Self-Learning Gene Expression Programming (SL-GEP) [11]. The process of fitness evaluation, mutation, crossover, and selection operations is repeated until the termination conditions are satisfied.

B. Archive-based fitness evaluation

The pseudocode of archive-based fitness evaluation is shown in Algorithm 1, composed of three main steps.

Step 1 - Combination with the archive population: This step begins with a condition check. If the best fitness value is updated or maintained for a specified number of iterations (e.g., three), a combination with the archive population to evaluate the fitness will be triggered. This involves evaluating the RSR and TSR sub-solutions of the newly added archive population with the best individuals of P_{TSR} and P_{RSR}, respectively. If there is an update to the best fitness value after evaluation, the archived sub-solution will replace the best individual of the corresponding sub-population. For example, if combining the best individual of P_{TSR} and a newly added RSR sub-solution in the archive population results in the best fitness value update, the newly added RSR sub-solution will be set as the best individual of P_{RSR}.

Step 2 - Archive Population Update: If the condition of Step 1 is met, the archive population will be dynamically updated. Specifically, the combination of individuals with better fitness in the current iteration is selected and subsequently mutated. The mutated combination is then regarded as the newly added sub-solutions of the archive population.

Step 3 - Fitness evaluation: The fitness value of the complete solution, which includes the current sub-population and the best individual from the other sub-population, is calculated according to (3).

It should be noted that adaptive adjustment of the training set has been employed before every archive-based fitness evaluation to reduce computational cost.

Algorithm 1: Archive-based fitness evaluation

Input: two sub-populations P_{TSR}, P_{RSR} and archive population; training set I; iteration number: T
Output: the best sub-solutions TSR and RSR

```
Adaptively adjust numbers of the training set I;
if (best fitness value updates || maintain the same best fitness
  value for a long time) then
    Triggers a fitness evaluation in combination with the
      archive population;
    if best fitness value updates then
        Replace the best individual in the corresponding
          sub-population;
    end
    Update archive population;
end
Evaluate fitness of individuals in each sub-population;
Update the best individual indexes;
P^r_TSR, P^r_RSR ← best individual of P_TSR, P_RSR ;
return P^r_TSR, P^r_RSR
```

V. EXPERIMENTAL STUDIES

A. Experimental Settings

In experiments, two types of workflows have been tested, including randomly generated workflows and four real-world workflows. The random workflows are generated using a random DAG generator, while the four real-world workflows

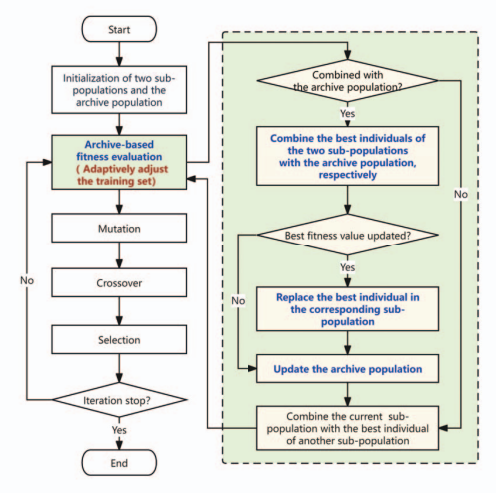

Fig. 1. The flowchart of the proposed A-CCGP algorithm.

are Gaussian Elimination, Fast Fourier Transform (FFT) [3], [4], Montage [12], and Epigenomics [13].

On both types of workflows, five well-known manually-designed heuristics, i.e., HEFT [3], Lookahead [4], PEFT [5], PETS [6], MSL [7], and the recently published CCGP [8] are chosen to compare with our proposed A-CCGP. For CCGP and A-CCGP, there are four hyper parameters to consider. These include the sub-population size (*NP*), the number of Automatically Defined Functions (ADFs) (*K*), the head length of the main program (*h*), and the head length of the ADFs (*h'*). The ADFs are sub-functions that provide sub-solutions to specific sub-problems. In the context of WSP, the four hyper parameters are empirically set as *NP* = 30, *K* = 2, *h* = 20, and *h'* = 3, while the other parameter settings follow those used in CCGP [8].

All compared algorithms are coded in C++ and executed on a computer with Intel(R) Core(TM) i9-13900 CPU @ 3.0GHz, 128 GB memory, and Microsoft Windows 11 64-bit system. The metric for comparison is mainly the average SLR over the whole training set as calculated according to (3) and the best fitness values attained by the algorithms. Note that for the sake of comparison fairness, the results of CCGP and A-CCGP have been averaged over 20 independent runs.

B. Result Analysis

Fig. 2 depicts the search convergence trend regarding the best fitness values of all compared algorithms on randomly generated and four real-world workflows. It is important to note that each algorithm terminates when the predetermined time limit is reached. Manually-designed heuristics yield consistent fitness values under fixed inputs, wherein their decisions are based on predetermined rules. It can be observed that the proposed A-CCGP algorithm achieved a much faster convergence rate than the other algorithms. Additionally, the A-CCGP and CCGP algorithms can learn high-level heuristics better than manually designed ones.

TABLE I
THE SCHEDULING LENGTH RATIO OF THE HIGH-LEVEL HEURISTICS ALGORITHM LEARNED ON RANDOMLY GENERATED WORKFLOWS AND FOUR REAL-WORLD WORKFLOWS COMPARED WITH OTHER ALGORITHMS

		Randomly Generated Workflows					Gaussian Elimination					FFT				
		PEFT	Lookahead	HEFT	MSL	PETS	PEFT	Lookahead	HEFT	MSL	PETS	PEFT	Lookahead	HEFT	MSL	PETS
A-CCGP	Better	55%	79%	73%	94%	90%	62%	76%	83%	95%	96%	66%	53%	68%	85%	89%
	Equal	12%	4%	4%	2%	2%	16%	4%	4%	2%	2%	9%	8%	6%	4%	3%
	Worse	33%	17%	23%	4%	8%	22%	20%	13%	3%	2%	25%	39%	26%	11%	8%

		Montage					Epigenomics				
		PEFT	Lookahead	HEFT	MSL	PETS	PEFT	Lookahead	HEFT	MSL	PETS
A-CCGP	Better	75%	58%	77%	87%	84%	54%	59%	77%	85%	86%
	Equal	6%	9%	4%	3%	2%	18%	8%	4%	2%	3%
	Worse	19%	33%	19%	10%	14%	28%	33%	19%	13%	11%

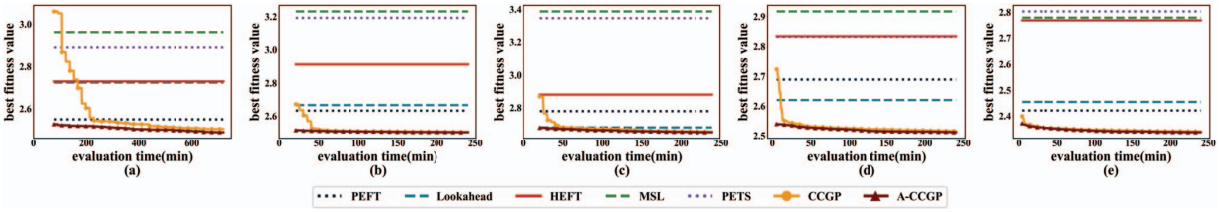

Fig. 2. Evolution of the best fitness values derived from all compared algorithms on (a) randomly generated workflows, (b) Gaussian Elimination, (c) FFT, (d) Montage, and (e) Epigenomics.

To investigate our proposed A-CCGP's effectiveness, its learned high-level heuristics are compared with five manually designed heuristics on randomly generated workflows and four real-world workflows. Table 1 shows the percentage of schedules generated by high-level heuristics of the A-CCGP that are better than, equal to, or worse than those produced by other heuristics. It can be seen that the SLR values of the A-CCGP's best heuristics are more than 50% better than all other heuristics. Moreover, the A-CCGP performs better or equal to the second-best algorithm PEFT on five test workflows, with rates of 67%, 78%, 75%, 81%, and 72% respectively.

VI. CONCLUSION

This paper has proposed an A-CCGP algorithm to automatically generate high-level heuristics for WSP. The proposed A-CCGP algorithm utilizes an archive strategy to improve pairing accuracy, enhance fitness evaluation quality and accelerate search efficiency. In addition, we also propose an adaptive training set adjustment strategy to effectively reduce the computational cost. Experiments on randomly generated workflows and four real-world workflows have validated the effectiveness of the proposed A-CCGP. In future, we would extend our method by considering more optimization objectives, such as maximize utilization and minimize cost.

REFERENCES

[1] Yi Gu and Chandu Budati. Energy-aware workflow scheduling and optimization in clouds using bat algorithm. Future Generation Computer Systems, 113:106–112, December 2020.

[2] Navpreet Kaur Walia, Navdeep Kaur, Majed Alowaidi, Kamaljeet Singh Bhatia, Shailendra Mishra, Naveen Kumar Sharma, Sunil Kumar Sharma, and Harsimrat Kaur. An Energy-Efficient Hybrid Scheduling Algorithm for Task Scheduling in the Cloud Computing Environments. IEEE Access, 9:117325–117337, 2021.

[3] H. Topcuoglu, S. Hariri, and Min-You Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Transactions on Parallel and Distributed Systems, 13(3):260–274, March 2002.

[4] Luiz F. Bittencourt, Rizos Sakellariou, and Edmundo R. M. Madeira. DAG Scheduling Using a Lookahead Variant of the Heterogeneous Earliest Finish Time Algorithm. In 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, pages 27–34, February 2010. ISSN: 2377-5750.

[5] Hamid Arabnejad and Jorge G. Barbosa. List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table. IEEE Transactions on Parallel and Distributed Systems, 25(3):682–694, March 2014.

[6] E. Ilavarasan and P. Thambidura. Low Complexity Performance Effective Task Scheduling Algorithm for Heterogeneous Computing Environments. Journal of Computer Science, 3(2):94–103, February 2007.

[7] D. Sirisha and G.Vijaya Kumari. A new heuristic for minimizing schedule length in heterogeneous computing systems. In 2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), pages 1–7, March 2015.

[8] Qin-zhe Xiao, Jinghui Zhong, Liang Feng, Linbo Luo, and Jianming Lv. A Cooperative Coevolution Hyper-Heuristic Framework for Workflow Scheduling Problem. IEEE Transactions on Services Computing, 15(1):150–163, January 2022.

[9] Shaolin Wang, Yi Mei, Mengjie Zhang, and Xin Yao. Genetic Programming With Niching for Uncertain Capacitated Arc Routing Problem. IEEE Transactions on Evolutionary Computation, 26(1):73–87, February 2022.

[10] Qingqing Liu, Xianpeng Wang, Yao Wang, and Xiangman Song. Evolutionary convolutional neural network for image classification based on multi-objective genetic programming with leader–follower mechanism. Complex & Intelligent Systems, 9(3):3211–3228, June 2023.

[11] Jinghui Zhong, Yew-Soon Ong, and Wentong Cai. Self-Learning Gene Expression Programming. IEEE Transactions on Evolutionary Computation, 20(1):65–80, February 2016.

[12] Zulfiqar Ahmad, Ali Imran Jehangiri, Nader Mohamed, Mohamed Othman, and Arif Iqbal Umar. Fault Tolerant and Data Oriented Scientific Workflows Management and Scheduling System in Cloud Computing. IEEE Access, 10:77614–77632, 2022.

[13] Xiaoyong Tang, Wenbiao Cao, Huiya Tang, Tan Deng, Jing Mei, Yi Liu, Cheng Shi, Meng Xia, and Zeng Zeng. Cost-Efficient Workflow Scheduling Algorithm for Applications With Deadline Constraint on Heterogeneous Clouds. IEEE Transactions on Parallel and Distributed Systems, 33(9):2079–2092, September 2022.