A Knowledge Guided Multi-Population Evolutionary Algorithm for Dynamic Workflow Scheduling

Jingyuan XU*
dept. of
Computer Science and Engineering
Southern University of
Science and Technology
Shenzhen, Guangdong, China
12012913@mail.sustech.edu.cn

Jiajian YANG*
dept. of
Computer Science and Engineering
Southern University of
Science and Technology
Shenzhen, Guangdong, China
12012711@mail.sustech.edu.cn

Peiru LI*
dept. of
Computer Science and Engineering
Southern University of
Science and Technology
Shenzhen, Guangdong, China
12011415@mail.sustech.edu.cnk

[1] ***Abstract*—This work introduces a novel algorithm called the Feature-Order Guided Multi-Population (FOGMP) algorithm for dynamic workflow scheduling in cloud environments. The paper addresses the challenge of scheduling in dynamic and uncertain cloud environments, where resources may become inaccessible due to hardware or software issues. The FOGMP leverages a multi-population framework, integrating evolutionary strategies with repair mechanisms to adapt to environmental changes. Key contributions include the development of repair strategies to replace inaccessible resources and the use of solution features to guide evolutionary processes, enhancing both diversity and convergence. The algorithm's performance is validated through extensive experiments against existing dynamic scheduling approaches, demonstrating its effectiveness in maintaining solution quality despite environmental changes. The paper concludes with future work prospects in multi-objective optimization and fairness-oriented metrics in cloud-based workflow scheduling.**

***Index Terms*—workflow scheduling, cloud computing, evolutionary algorithm, multi-objective optimization**

I. Introduction

With the rapid development of cloud computing, workflow scheduling problem (WSP) is becoming increasingly important. The primary objective of WSP is to find optimal or multiple viable scheduling schemes for assigning interdependent tasks within a workflow to various virtual machines (VMs), also known as instances. Due to the complexity of the workflow topology and the dynamics and heterogeneity of cloud computing resource pools, the WSP in cloud computing is widely recognized as an NP-hard problem.

Usually, WSP is modelled as a static optimization problem, either a single objective optimization problem, which focuses on optimizing a single objective, or a multi-objective optimization problem (MOP), which attempts to provide multiple trade-off solutions among different objectives [1]. Makespan and cost are two most common but conflicting quality-of-service (QoS) requirements. Makespan represents the overall completion time of a workflow, while cost denotes the total expenses associated with executing the workflow.

In real-world scenarios, the WSP is inherently dynamic [2] [3] since the performance of resource pool fluctuates continuously and occurrences of hardware or software faults lead to frequent resource inaccessibility. This dynamic nature renders solutions obtained from static scheduling algorithms infeasible. Consequently, it is more appropriate to model the WSP in cloud computing as a dynamic optimization problem.

In contrast to the static model, in the dynamic WSP, some solutions may become infeasible after changes occurs over time. Static scheduling algorithms, which do not account for resource inaccessibility, are limited in their ability to handle such situations. Their only option is to restart the scheduling process. However, this approach not only wastes computational resources but also discards valuable information from previous solutions. While there have been efforts to develop dynamic scheduling algorithms tailored to address the challenges of changing environments in WSP, the existing algorithms for dynamic WSP still lack effectiveness.

To address the limitations of current dynamic workflow scheduling algorithms and advance the resolution of dynamic WSP, we propose the Feature-Order Guided Multi-Population Algorithm (FOGMP) for dynamic multi-objective workflow scheduling problem in cloud computing. First, we introduce a similarity repair strategy that replaces inaccessible instances with the most similar ones based on their attribute values using the Manhattan Distance metric. Our analysis and experiments demonstrate that this strategy outperforms other baseline repair strategies and achieves better convergence. However, the similarity repair strategy may lead to a loss of diversity as the repaired population becomes similar to the previous one near the steady state. To mitigate this issue, we incorporate newly generated feasible individuals from another population to maintain diversity by replacing some of the infeasible individuals. Furthermore, through observation and analysis of non-dominated solutions in the Pareto optimal

[1]

*The first three authors (Jingyuan Xu, Jiajian Yang, and Peiru Li) are equally contributed to this paper.

front (POF) for specific workflows, we discover that certain tasks are preferably executed in a specific order on the same instance, which we term "feature-order" (FO). We leverage this FO knowledge to guide the evolution of inferior solutions. The performance of our proposed FOGMP algorithm has been validated through comprehensively experimental study.

The remainder of this paper is organized as follows. Section!II describes background of dynamic workflow scheduling problems, including the formulation of WSP and the related work for dynmaic multi-objective WSP. Our proposed method, that is, FOGMP, is described in Section III. Section IV presents the experimental study to validate the performance of the proposed FOGMP algorithm. Finally, the paper is concluded in Section V.

II. Background

A. Dynamic Workflow Scheduling Problem

A workflow is composed of various tasks or nodes interconnected by dependencies or data flows, which can be depicted as a directed acyclic graph (DAG), denoted as $W = (T, D)$, where $T = \{t_1, t_2, \cdots, t_n\}$ is the set of tasks and $D = \{(t_i, t_j) | t_i, t_j \in T\}$ describes the dependency relationship between tasks. A workflow is scheduled and executed in a distributed computing platform consist with a cluster of virtual machines (VMs) with varying capabilities, denoted as $R = \{r_1, r_2, \cdots, r_m\}$. Each task has its referred execution time depends on each VM's compute unit $cu(r_j)$. Different tasks would be assigned to different VMs, thus the communication time between VMs should be concerned, referred as $TT(t_i, r_j)$, decided by the bandwidth $bw(r_j)$ of VMs.

In our study, the workflow scheduling problem aims to minimize both makespan and cost. Makespan refers to the total time it takes for a workflow to start and end, while cost refers to the charge of renting virtual machines following Amazon EC2's on-demand billing model [4]. Total cost involves the hourly price of each instance and the difference between the earliest start and latest finish times of tasks on that instance. The makespan and cost are computed as Eq.(1), Eq.(2).

$$Makespan = \max_{t_i \in T, r_j \in R} FT(t_i, r_j) \tag{1}$$

$$Cost = \sum_{r_j \in R} pr(r_j) \cdot |\max_{t_i \in T} FT(t_i, r_j) - \min_{t_k \in T} ST(t_k, r_j)| \tag{2}$$

Herein, a workflow scheduling problem is modeled as a multi-objective optimization problem that seeks a balance between conflicting goals – makespan and cost, because the better the performance of a virtual machine, the shorter the task execution time, but on the contrary, its cost is higher. Therefore, the primary challenge is minimizing both makespan and cost under the constraint of dependencies.

As for dynamic, this paper considers the dynamic nature of resource inaccessibility, as in real-world environments, VMs in distributed platforms may be inaccessible due to hardware or software factors. To measure such environmental changes, we characterize it by the changing frequency (f) and severity (s). Here, f denotes the rate of change, and s represents the percentage of instances becoming inaccessible at each change. To be specific, the instances that can be allocated in the cloud platform are in set $R_{acce} = R$, $R = \{r_1, r_2, \cdots, r_m\}$, where m is the number of instance. Once the environment changes, some instances become unavailable, noted as $R_{dis} = \{r | r \in R\}$, where $card(R_{dis}) = m \cdot s$. R_{acce} need to remove those unavailable instances.

Resource pool shrinkage impacts cloud computing workflow scheduling in two ways. Firstly, scheduling algorithms must address infeasible solutions in the current POF, with different strategies leading to varied convergence to the new Pareto-optimal Set. Secondly, changes in the POF can render previously dominated solutions as non-dominated within the POF. Repair these infeasible solutions presents a significant challenge.

B. Related Work

Several approaches for dynamic workflow scheduling in cloud environments have been proposed. Ismayilov, G. et al. [?] proposed NN-NSGA-II, combined neural networks with NSGA-II [?] to predict potential solutions in changing environments. Tingting Dong et al. [?] develop a double deep Q network framework(DDQN) based on reinforcement learning to select re-submission and replication strategy adaptively considering fault tolerance. Deb, Kalyanmoy et al. proposed DNSGA-II-B [?] while handling hydro-thermal power scheduling, which focus on preserving population diversity to enhance global search capabilities and prevent premature convergence. Similarly, by maintaining solution diversity, Carlos R. B. Azevedo et al. proposed DNSGA-II-gIDG [5] to incorporate both random and mutated immigrants into the population, ensuring diversity in each generation. Another studies focus to extract some knowledge from solutions, making algorithms find the optimal solutions faster. Rani et al. [6] proposed an ant-lion colony optimization method, using ant-lions to capture the characteristics of ants while searching. Koo et al. [7] proposed MB-NSGA-II utilizes a memory archive to store recent pareto sets and retrieves them upon environmental changes. However, there are many well-designed static approaches also can be used in dynamic environments by adding a repair strategy. Qiu et al. [8] proposed DMGA, which identify the longest common sequence in the solution set to guide solution evolution, ensuring both rapid convergence and diverse solutions. Han et al. [9] proposed CMSWC utilize search trees and heuristics to narrow the solution space. Belgacem et al. [10] combined ant colony optimization with MOHEFT [11], effectively explore extensive solution spaces. Vavak et al. [12] proposed DNSGA-II-HM by dynamically adjusting algorithm parameters to direct the search process.

III. Feature-order Guided Multi-Population (FOGMP) Approach

The FOGMP algorithm integrates a multi-population framework with distinct evolutionary and repair strategies in response to environmental changes. We established two distinct populations: a normal population utilizing the NSGA-II [?] algorithm, which incorporates a similarity repair strategy during environmental changes, and a feature population, divided into five regions. In each region, offspring generation adheres to a feature order randomly selected from the region's Pareto front. This feature population employs an individual substitution strategy in response to enviroment. The FOGMP process is detailed in Algorithm 1.

Algorithm 1: Feature-order Guided Multi-Population (FOGMP) Algorithm

```
Data: Population size $ps$; Maximum generation $g$
Result: Pareto front $PF$
1  $P_N \leftarrow$ InitPopulation($ps/2$);
2  $P_F \leftarrow$ InitPopulation($ps/2$);
3  i = 0;
4  while $i < g$ do
5      Divide $P_F$ into five regions;
6      while $|P_N| + |P_F| < 2 \cdot ps$ do
7          $p_1, p_2 \leftarrow BinaryTournamentSelection(P_N)$;
8          $p_3, p_3 \leftarrow BinaryTournamentSelection(P_F)$;
9          if Random(0, 1) $< 0.5$ then // Inter Population Reproduce
10             $(c_1, c_3) \leftarrow$ Reproduce($p_1, p_3$);
11             $(c_2, c_4) \leftarrow$ Reproduce($p_2, p_4$);
12         else // Intra Population Reproduce
13             $(c_1, c_2) \leftarrow$ Reproduce($p_1, p_2$);
14             $(c_3, c_4) \leftarrow$ FeatureOrderGuidedReproduce($p_3, p_4$);
15         end
16         $P_N \leftarrow P_N \cup \{c_1, c_2\}$;
17         $P_F \leftarrow P_F \cup \{c_3, c_4\}$;
18     end
19     $P_N \leftarrow$ Selection($P_N$);
20     $P_F \leftarrow$ Selection($P_F$);
21     if change occurs then
22         NormalRepair($P_N$);
23         SimilarityRepair($P_F$);
24     end
25     $PF \leftarrow$ ParetoFront($P_F, P_N$);
26     i++;
27 end
28 return $PF$
```

FOGMP starts by encoding chromosomes. Our focus encompasses two primary aspects: the execution order of tasks and the task-to-machine mapping, devised as two gene sequences 'order' and 'task2Ins'. In line 1 and line 2, FOGMP randomly generates initial normal population(P_N) and initial feature-order guided population(P_F), together with population size(ps), with half of the initial solutions randomly created (detailed in [?]) and the other half assigned to single machine. In line 5, feature-order guided population(P_F) is divided into five regions as illustrated in Fig. 3. In lines 4-21, FOGMP decides between intra reproduction and inter reproduction for two populations.Inter reproduction operates normal crossover and mutation to generate offsprings. Intra reproduction involves randomly selecting two individuals from the same population, and if they're from the same region of P_F, the offspring's order sequence is adjusted according to a randomly selected $FeatureOrder$, the process is shown in Fig. 4. Conversely, if the individuals are from different regions, adherence to the feature order is not required. In lines 19-20, FOGMP employs fast nondominated sorting and fast crowding distance sorting on P_N and P_F, and updated them by the best ps individuals after sorting. In lines 22-23, FOGMP execute repair strategies when environment change occurs. For normal population(P_N), we execute normal repair, which means replacing infeasible solutions with randomly generated one(individual substitution1(a)). On the contrary, for feature-order guided population(P_F), FOGMP performs similarity repair1(c). Details of different repair strategies can be seen in section III-B

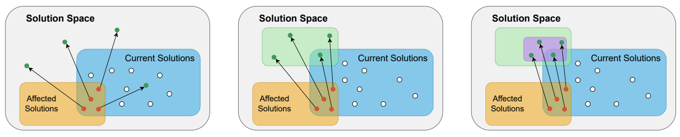
A. Region division

Before each iteration begins, we need to divide the individuals in the Feature population into five regions. First, we define five region vectors, bounded by two vectors, (0,1) and (1,0), and the angle between each vector is the same. In the partitioning, we first normalize the makespan and cost of all individuals in the Feature population to Max-Min, and then measure the straight-line distance of the individuals from the five vectors, and select the number of the vector with the smallest distance as the corresponding region to which the individuals belong. Fig. 3 shows the operation of region division.

The reason we partition is that individuals in the same region usually have similar makespan and cost properties on the Pareto front. Solutions with similar makespan and cost properties are more likely to have similar feature-order. The definition of feature-order can be found in Section III-C. Therefore, we will use the operations in Section III-C to adjust the sub-individuals generated by the solution in the same region to achieve the effect of optimizing the solution.

B. Repair Strategies

In our dynamic environment, original solutions become infeasible when environment changes, necessitating repairs for adaptation. Current approach for such repairs is random substitution, where tasks from the affected machine are redistributed randomly to another. However, in our study, we propose two repair strategies: one based on similarity and the other on replacing impacted individuals in the population. And the combination of two strategies is proved to improve the diversity and convergence of the population.



(a) Individual Substitution (b) Randomly Repair (c) Similarity Repair

Fig. 1: Different repair strategies

The similarity-based repair strategy, as shown in Fig. 1(c), tends to yield solutions closer to post-convergence, though it reduced population diversity. It evaluates machine attributes—computing unit, bandwidth, and price—using Min-Max normalization and Manhattan distance to determine similarity. Once a machine become inaccessible, the machine with the highest similarity would be chosen to substitute it. Conversely, the individual substitution strategy focuses on enhancing diversity by replacing affected individuals with randomly generated ones, since it maps the newly generated feasible solution to the entire solution space, as shown in Fig. 1(a)

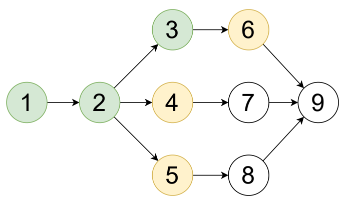


Fig. 2: Sample Workflow with nine tasks

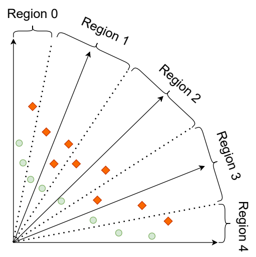


Fig. 3: Regions and individuals in feature population

C. Feature-order-based Operator

The study shows that the order of task execution is crucial, especially when tasks are processed on the same machine. Random topological sorting results in diverse task sequences, but the order often doesn't matter for tasks without inner dependencies. For example, as illustrated in Fig. 2, tasks 1, 2 and 3 represent completed tasks, and tasks 4, 5 and 6 signify pending tasks. We noticed that when tasks 4, 5, and 6 are on separate machines, their execution order is less important. However, if they are on the same machine, certain sequences like 4→5→6 are better, as they allow subsequent tasks (7 and 8) to start sooner, optimizing overall performance. This finding led to the definition of 'Feature Order' – the sequence of tasks on certain machine. Optimal performance is achieved when the length of the Feature Order is between 1 and 1/4 of the total task length. Algorithm 2 delineates the methodology for obtaining feature orders.

In the FOGMP algorithm, adjustments after crossover are necessary because the individuals after crossover may not meet the Feature-Order. This adjustment involves initial crossover

Algorithm 2: Get Feature Order

```
Data: $order$, $task2ins$, The number of VMs $\mathbf{M}$, the
      size of order $\mathbf{N}$
Result: $FO = \bigcup_{i=1}^{M}\{fo_k\}$, $fo_k = \bigcup_{i=1}^{N}\{task_i\}$
1 for $T$ in order do
2     ins = task2ins[$T$];
3     Add $T$ to $FO$[ins];
4 end
5 for $fo$ in $FO$ do
6     if $|fo| \leq 1or|fo| \geq N/4$ then
7         remove $fo$ from $FO$;
8     end
9 end
```

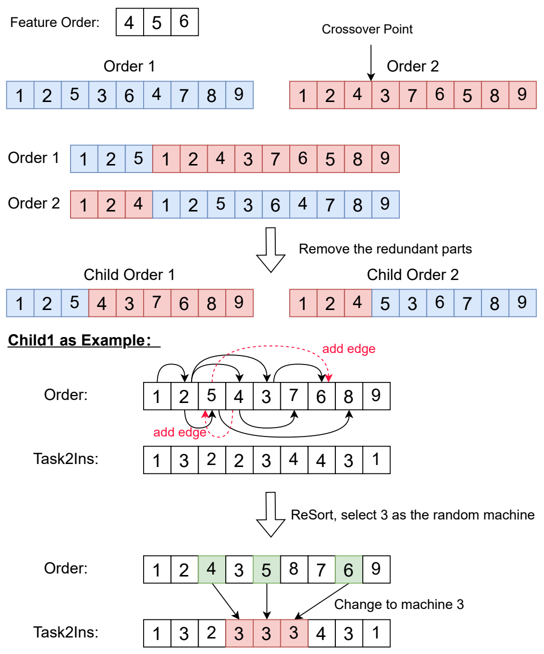


Fig. 4: Example of crossover based on a Feature Order

as NSGA-II [?], logging each task's assigned machine, adding dependencies to the DAG based on Feature Order, re-sorting the DAG topologically, and updating the child's sequence order. Subsequently, machine assignments are based on prior records. Finally, an available machine is randomly selected to accommodate any unassigned tasks in the Feature Order, as exemplified by the DAG in Fig. 2 and detailed in Fig. 4.

IV. Experiments and Comparisons

This section outlines the experimental setup, IaaS service model, comparative methodologies, and performance metrics. Subsequently, we verify the validation of repair strategies we proposed in section 3.1 and compares the FOGMP algorithm with other dynamic scheduling approaches and analyse the results.

A. *Experiment settings*

1) Workflow instances.: We examined five types of real-world scientific workflows: Montage (I/O intensive), CyberShake (data intensive), Epigenomics, Sipht (CPU intensive), and Inspiral (LIGO), as published by the Pegasus workflow project [13]. These workflows are commonly used for assessing scheduling algorithm performance and feature diverse structural elements such as pipelines, data distribution/aggregation, and redistribution. Table I details their key characteristics, including task and edge counts, average data size, and runtime per unit.

2) IaaS service model.: Our simulation platform replicates a data center with eight EC2-based instance types, each offering limited capacity (see Table II). Instance specifications are consistent with previous studies, and differentiation is based on standardized runtimes. The simulation uses a 60-minute interval on-demand billing model. To mimic a dynamic environment, we simulated environmental changes at specific intervals. For evolutionary algorithms spanning 500 generations, low-frequency environmental changes occur at the 250th generation, and high-frequency changes every 100 generations. For list-based scheduling with 100 tasks, low-frequency changes occur at the 50th task, and high-frequency changes every 20 tasks. The severity of these changes is set at 0.03, 0.08, and 0.15. This model aims for practical relevance and allows for repeatable evaluations without the constraints of actual deployment, establishing an environment with discernible performance trade-offs.

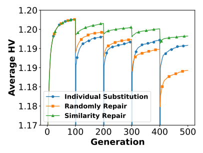
TABLE II: Instance Types and their Specifications

Type	Compute Unit (CU)	Bandwidth (BW)	Price ($)	Capacity
1	1.7/8	39321600	0.06	10
2	3.75/8	85196800	0.12	10
3	3.75/8	85196800	0.113	10
4	7.5/8	85196800	0.24	10
5	7.5/8	85196800	0.225	10
6	15/8	131072000	0.48	10
7	15/8	131072000	0.45	10
8	30/8	131072000	0.9	10

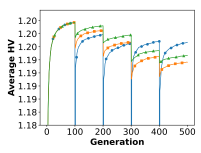
3) Performance metric.: Hypervolume (HV) metric emerges as a predominant tool for performance evaluation. It calculates the volume of the space dominated by a solution set relative to a specific reference point. A larger HV signifies enhanced convergence towards the true Pareto frontier, coupled with increased diversity along this frontier.

For this study, the objective values of the scheduling solutions are normalized by the upper bounds found so far in certain dynamic case for all experimental algorithms. The reference point for each objective is set to 1.1 [**?**]. The HV was determined for each of the 20 runs independently, with the final reported value representing the mean HV across these runs.

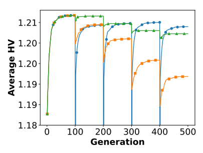
4) Comparative algorithm.: In our study, we compared the performance of FOGMP with several algorithms, namely DNSGA-II-B [**?**], DNSGA-II-gIDG [5], MB-NSGA-II [7], DMGA [8], and CMSWC [9]. The parameters for these algorithms were set in accordance with recommendations from their respective literature sources. As DNSGA-II-B, DNSGA-II-gIDG, MB-NSGA-II, DMGA, and FOGMP are all based on evolutionary algorithms, we standardized their population size at 200 and mutation rate at 0.01. Contrarily, CMSWC, a list heuristic approach, was configured with a solution number of 50 and an exploit rate of 0.3.



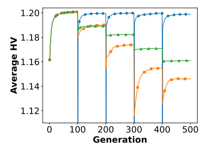
(a) Montage



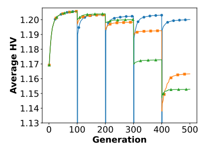
(b) CyberShake



(c) Epigenomics



(d) Sipht



(e) Inspiral

Fig. 5: Performance of 3 repair strategies on five workflow instances.

B. *Results and analysis*

1) Validation of the effectiveness of repair strategies: To verify that the repair strategies mentioned in section 3.1 does indeed have a positive impact on the algorithm's results, we designed the following experiment. We adopt NSGA-II as the base algorithm and put it under the most serious dynamic environment case (f=100,s=0.15). Here we design three repair strategies to handle environment change: (1). individual substitution, (2). randomly repair, (3) similarity repair. The experiments were conducted independently 20 times under five workflow instances. The variation of the HV value with generations are shown in Fig. 5

As can be seen, when the environment changes, the HV values of different repair strategies all decrease to varying degree, while the HV value of similarity repair decreases the least, indicating its effectiveness in maintaining the convergence of the population. The final hv value of individual substitution is the highest, indicating that it increases population diversity and jumps out of local optimal solutions after environmental changes occur.

2) Performance Evaluation: For statistical analysis, each experiment was conducted independently 20 times, with the resulting mean HV values recorded. The highest mean values are highlighted in bold for enhanced clarity. Additionally, tables for each test case include the rank values of all comparable algorithms, ascertained based on their mean HV values. Furthermore, t-tests were performed to assess the statistical significance of the results, with a particular focus on comparing FOGMP with five other peer algorithms. The comparative performance of FOGMP and various dynamic

TABLE I: Scientific Workflows

workflow	Number of Tasks	Number of Edges	Avg. edge data size	Avg. task runtime
Montage 100	100	433	3.23 MB	10.58 s
CyberShake 100	100	380	849.60 MB	31.53 s
Epigenomics 100	100	322	395.10 MB	3954.90 s
Sipht 100	100	335	6.27 MB	194.48 s
Inspiral 100	100	319	8.93 MB	206.12 s

multi-objective optimization algorithms on average HV is systematically detailed in Table III based on students' t test with 20 repetitions.

Number in the parenthese behind average HV value is the rank of that algorithm in that test instance. FOGMP ranks first in all test instances on workflow Epigenomics, Inspiral and Montage. As for CyberShake workflow, FOGMP is the best algorithm among all except for the case with severity 0.03 and frequency 250, in which FOGMP only ranks third. Nevertheless, according to the statistical analysis, all algorithms show no significant differences in this test instance and they are statistically equivalent.

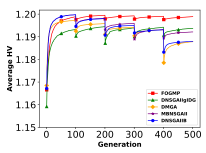
It is worth noting that in the Epigenomics test cases, our algorithm outperforms other algorithms by a wide margin, which is related to the characteristics of this workflow and FO. From observing the structure of Epigenomics workflow, it is apparent that many tasks have only one predecessor and one successor, which means all of the tasks in one single chain are supposed to be executed on the same instance to reach better $makespan$ and $cost$ because this allocation scheme can save transmission time. FO, defined as the sequentially executed tasks on the same instance, can restore the excellent characteristics of one good solution. Taking FO of superior solutions to guide the inferior solutions can improvde the convergence of our algorithm.

In the Sipht workflow, FOGMP ranks first in only two cases and second for remaining three cases, in which CMSWC ranks first. Although FOGMP and CMSWC are statistically equivalent, the reason for the suboptimal performance of our algorithm may be the presence of overlapping individuals, which does not contribute to the growth of the HV.

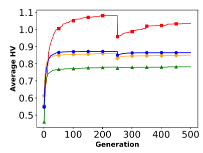
The data reveals that FOGMP consistently outperforms other algorithms in terms of average HV. Notably, FOGMP ranks highest in the number of best outcomes (#Best), outshining CMSWC and DNSGAIIB. Across all test cases, FOGMP is statistically superior to or at least not inferior to the competing algorithms.

In a general sense, it is evident that FOGMP surpasses the other five methods when evaluated using HV. By doing The verification of a feature order-based operator, a similarity-repair strategy, and a multi-population approach enables FOGMP to effectively address the challenges of dynamic workflow scheduling in cloud environments.

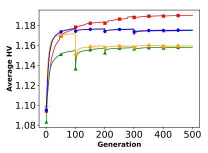
In Fig. 6 shows some experimental results of the variation of the average HV value over generations. Since CMSWC is not based on evolutionary framework, the figures do not contain CMSWC's HV variation process. As can be seen



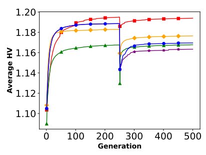
(a) CyberShake s=0.15



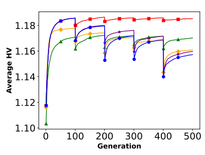
(b) Epigenomics s=0.08



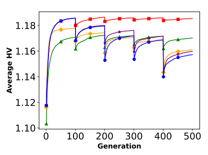
(c) Inspiral s=0.03



(d) Inspiral s=0.15



(e) Montage s=0.15



(f) Sipht s=0.15

Fig. 6: Average HV comparison of five algorithms over generations on six test instances(20 repetitions).

from the figures, after several times of resource failures, our algorithms performs the best among all the five algorithms on all the test cases. This demonstrates the strong capability of our algorithm in solving dynamic multi-objective workflow scheduling problems. However, from the observation of the curve depicting the change in HV before any failure, we can see that our algorithm does not always perform as well as or even better than other algorithms. This is one point that needs improvement in our future work.

V. Conclution and Future Work

In this study, we introduced a multi-population algorithm guided by knowledge of feature-order designed to address the dynamic workflow scheduling problem, particularly in scenarios involving resource inaccessibility due to hardware or software errors. Building upon the NSGA-II framework, our approach leverages feature-order as crucial knowledge to guide the evolutionary process of the population. Upon encountering changes, a novel similarity repair strategy is implemented to rectify infeasible solutions, thereby enhancing convergence. Furthermore, to preserve diversity, a multi-population strategy is employed. Specifically, the feature-order, indicating the tasks executed on the same instance, is taken from superior solutions, suggesting it as a key factor in their enhanced performance. Consequently, feature-order is regarded as useful knowledge to guide less successful solutions by mandating its adoption after traditional crossover and mutation processes.

Our experimental framework involved simulating five real-world scientific workflows and utilizing Amazon EC2 in-

TABLE III: Results of FOGMP and other algorithms on average HV.

Note: The number in parentheses indicates the ranking of the algorithm in that test instance. Symbols ”+”, ”-”, and ” ≈” are used to indicate statistically significant superiority, equivalence, or inferiority of FOGMP relative to each algorithm respectively, based on students’ t test(20 repetitions).

Workflow	Severity	Frequency	FOGMP	DMGA	DNSGAIIB	DNSGAIIgIDG	MBDNSGAII	CMSWC
CyberShake	0.03	100	**1.18883(1)**	1.18589(5+)	1.18804(3≈)	1.18499(6+)	1.18855(2≈)	1.18690(4≈)
		250	1.18823(3)	1.18475(5≈)	**1.18838(1≈)**	1.18456(6≈)	1.18836(2≈)	1.18674(4≈)
	0.08	100	**1.18793(1)**	1.18171(6+)	1.18605(3≈)	1.18425(4+)	1.18671(2≈)	1.18344(5+)
		250	**1.18790(1)**	1.18266(6+)	1.18723(4≈)	1.18430(5+)	1.18732(3≈)	1.18785(2≈)
	0.15	100	**1.18653(1)**	1.16653(5+)	1.16138(6+)	1.18246(2+)	1.17430(4+)	1.18160(3+)
		250	**1.19010(1)**	1.18470(6+)	1.18648(5+)	1.18664(3+)	1.18652(4≈)	1.18858(2≈)
Epigenomics	0.03	100	**0.95535(1)**	0.62355(4+)	0.64423(3+)	0.56220(5+)	0.65064(2+)	0.54604(6+)
		250	**0.98483(1)**	0.57061(4+)	0.58683(2+)	0.50186(6+)	0.58637(3+)	0.50776(5+)
	0.08	100	**1.01088(1)**	0.60628(5+)	0.66541(2+)	0.64810(4+)	0.65654(3+)	0.54776(6+)
		250	**0.96368(1)**	0.56135(4+)	0.58109(2+)	0.52083(6+)	0.57916(3+)	0.52757(5+)
	0.15	100	**1.02278(1)**	0.54668(3+)	0.53690(4+)	0.61613(2+)	0.52739(5+)	0.50090(6+)
		250	**0.97791(1)**	0.56454(4+)	0.58193(3+)	0.55396(5+)	0.58527(2+)	0.53637(6+)
Inspiral	0.03	100	**1.15999(1)**	1.10918(6≈)	1.13905(2≈)	1.11283(5+)	1.13884(3≈)	1.13255(4≈)
		250	**1.17004(1)**	1.15199(4≈)	1.15517(3≈)	1.13426(6+)	1.15519(2≈)	1.14830(5≈)
	0.08	100	**1.15932(1)**	1.03165(5+)	1.09446(4+)	1.11780(2+)	1.02591(6+)	1.10911(3+)
		250	**1.16839(1)**	1.14764(2≈)	1.12095(6≈)	1.13019(4+)	1.12492(5≈)	1.14400(3≈)
	0.15	100	**1.12922(1)**	0.96377(5+)	0.95150(6+)	1.08344(2≈)	0.98620(4+)	1.03992(3+)
		250	**1.17496(1)**	1.14678(3+)	1.12619(5+)	1.13811(4+)	1.11401(6+)	1.15152(2+)
Montage	0.03	100	**1.16080(1)**	1.15029(5+)	1.15648(3+)	1.14802(6+)	1.15806(2≈)	1.15644(4+)
		250	**1.16092(1)**	1.15337(5+)	1.15894(3≈)	1.14975(6+)	1.15908(2≈)	1.15831(4≈)
	0.08	100	**1.16157(1)**	1.14077(6+)	1.15390(2+)	1.14861(5+)	1.15009(4+)	1.15111(3+)
		250	**1.16085(1)**	1.15169(5+)	1.15768(3≈)	1.15018(6+)	1.15701(4≈)	1.15840(2≈)
	0.15	100	**1.15755(1)**	1.12599(4+)	1.10812(5+)	1.14396(2+)	1.10788(6+)	1.14353(3+)
		250	**1.16055(1)**	1.14945(6+)	1.15378(3+)	1.15049(5+)	1.15352(4+)	1.15782(2≈)
Sipht	0.03	100	**1.12287(1)**	1.09377(4≈)	1.09501(3≈)	1.08050(5≈)	1.06564(6≈)	1.12284(2≈)
		250	1.11959(2)	1.10514(3≈)	1.10430(5≈)	1.08703(6≈)	1.10444(4≈)	**1.12930(1≈)**
	0.08	100	1.12556(2)	1.10278(3≈)	1.06267(5≈)	1.09407(4≈)	1.04275(6≈)	**1.13385(1≈)**
		250	1.11931(2)	1.10888(3≈)	1.10635(4≈)	1.09108(6≈)	1.10557(5≈)	**1.13647(1≈)**
	0.15	100	**1.14478(1)**	0.99011(4+)	0.97658(5+)	1.11407(3+)	0.89822(6+)	1.14422(2≈)
		250	1.13059(2)	1.11908(3≈)	1.11069(4≈)	1.10360(6≈)	1.10921(5≈)	**1.14482(1≈)**
(#)Best			25	0	1	0	0	4
(#)+			-	21	16	23	14	14
(#)-			-	0	0	0	0	0
(#)≈			-	9	14	7	16	16

stances. The results, particularly in terms of HV, demonstrate that our Feature-Order Guided Multi-Population (FOGMP) algorithm markedly surpasses other comparative algorithms in the majority of test cases.

For future work, there are several promising directions to explore in the realm of multi-objective optimization of clustering-based scheduling for multi-workflow on clouds, with a specific focus on fairness. One avenue involves enhancing the Feature-Order Guided Multi-Population (FOGMP) algorithm to incorporate fairness-oriented metrics. This would involve designing and integrating new objectives that ensure equitable resource distribution among different workflows, thereby preventing resource monopolization by certain workflows and promoting a more balanced utilization of cloud resources.

References

[1] Z. Zhu, G. Zhang, M. Li, and X. Liu, “Evolutionary multi-objective workflow scheduling in cloud,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1344–1357, 2016. I

[2] G. Ismayilov and H. R. Topcuoglu, “Dynamic multi-objective workflow scheduling for cloud computing based on evolutionary algorithms,” in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*. IEEE, 2018, pp. 103–108. I

[3] ——, “Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing,” *Future Generation computer systems*, vol. 102, pp. 307–322, 2020. I

[4] “Amazon ec2 pricing,” https://aws.amazon.com/cn/ec2/pricing/, 2023, accessed: 2023-11-27. II-A

[5] C. R. Azevedo and A. F. Araújo, “Generalized immigration schemes for dynamic evolutionary multiobjective optimization,” in *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, 2011, pp. 2033–2040. II-B, IV-A4

[6] R. Rani and R. Garg, “Pareto based ant lion optimizer for energy efficient scheduling in cloud environment,” *Applied Soft Computing*, vol. 113, p. 107943, 2021. II-B

[7] W. T. Koo, C. K. Goh, and K. C. Tan, “A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment,” *Memetic Computing*, vol. 2, pp. 87–110, 2010. II-B, IV-A4

[8] H. Qiu, X. Xia, Y. Li, and X. Deng, “A dynamic multipopulation genetic algorithm for multiobjective workflow scheduling based on the longest common sequence,” *Swarm and Evolutionary Computation*, vol. 78, p. 101291, 2023. II-B, IV-A4

[9] P. Han, C. Du, J. Chen, F. Ling, and X. Du, “Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique,” *Journal of Systems Architecture*, vol. 112, p. 101837, 2021.

II-B, IV-A4

[10] A. Belgacem and K. Beghdad-Bey, “Multi-objective workflow scheduling in cloud computing: trade-off between makespan and cost,” *Cluster Computing*, vol. 25, no. 1, pp. 579–595, 2022. II-B

[11] J. J. Durillo, H. M. Fard, and R. Prodan, “Moheft: A multi-objective list-based method for workflow scheduling,” in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*. IEEE, 2012, pp. 185–192. II-B

[12] F. Vavak, “Adaptive combustion balancing in multiple burner boiler using a genetic algorithm with variable range of local search,” in *7th Int. Conf. on Genetic Algorithm*. Morgan Kaufmann, 1997. II-B

[13] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, “Characterizing and profiling scientific workflows,” *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682–692, 2013, special Section: Recent Developments in High Performance Computing and Security. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X12001732 IV-A1