Knowledge-Driven Multi-Objective Evolutionary Scheduling Algorithm for Cloud Workflows

YA ZHOU1 AND XIAOBO JIAO2

1College of Electrical and Mechanical Engineering, Xuchang University, Xuchang 461000, China
2Internet Department, State Grid Xuchang Power Supply Company, Xuchang 461000, China

Corresponding author: Ya Zhou (zhouzhouyang2011@126.com)

This work was supported by the Key Research Project of Xuchang University: Design of Smart IoT Management and Control System at Substation Side Based on BIM Model and Edge Computing under Project 2021zd004.

ABSTRACT Cloud workflow scheduling often encounters two conflicting optimization objectives of makespan and monetary cost, and is a representative multi-objective optimization problem (MOP). Its challenges mainly come from three aspects: 1) a large number of tasks in a workflow cause large-scale decision variables; 2) the two optimization objectives are of quite different scales; 3) and cloud resources are heterogeneous and elastic. So far, many studies focus on adopting multi-objective evolutionary algorithms (MOEAs) to solve the cloud workflow scheduling problem without mining the domain knowledge. To make a good trade-off between the makespan and monetary cost, this paper puts forward a knowledge-driven multi-objective evolutionary workflow scheduling algorithm, abbreviated as KMEWSA, including two novel features. On the one hand, the structural knowledge of workflow is mined to simplify the large-scale decision variables into a series of small-scale components, such accelerating the convergence speed of MOEAs. On the other hand, the knowledge on the Pareto front range is mined to estimate the ideal and nadir points for the objective space normalization during the search process, which helps maintain population diversity for MOEAs. At last, based on twenty real-world workflows and parameters of Amazon EC2, extensive experiments are performed to compare the KMEWSA with three baseline algorithms. The results demonstrate the effectiveness of the KMEWSA in balancing makespan and monetary cost for deploying workflows into cloud computing.

INDEX TERMS Cloud computing, workflow scheduling, multi-objective optimization, evolutionary algorithm.

I. INTRODUCTION

With the advantages of virtualization, rapid elasticity, on-demand access, the economy of scale, and so forth, cloud computing has proliferated rapidly over the past decade [1]. Infrastructure as a Service (IaaS) is the most basic service paradigm that enables an on-demand supply of pre-configured computing resources for tenants by a pay-as-you-go way. In this paradigm, tenants are freed from the heavy burden of purchasing, maintaining, and updating their large-scale hardware and software facilities, and can pay more attention to their core business [1]. Consequently, cloud computing successfully attracts an increasing number of applications from various fields, such as astronomy, big data processing, internet of things, and physics [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Turgay Celik.

Workflow, formed by a set of tasks and data dependencies among these tasks, has become the general model for various applications executed in cloud platforms [1]. Workflow scheduling is one of the main bottlenecks in the management of cloud platforms. Then, powerful scheduling approaches not only satisfy the user-specified quality of service (QoS), but also substantially improve the performance of cloud platforms. Since workflow scheduling in clouds is an NP-complete problem, and the optimal solutions cannot be found within an acceptable time. Until now, there exist many works on heuristics and meta-heuristics for scheduling workflows in clouds [3]. Scheduling algorithms based on list [4] and partial critical path [5] are two representative heuristics. In addition, meta-heuristics, such as particle swarm optimization [6], [7], ant colony optimization [8], krill herd algorithm [9]–[11], and differential evolution [12], are adopted to optimize the makespan or monetary cost, while
satisfying other QoS constraints. Noteworthy, the makespan and monetary cost of executing workflows on clouds are two conflicting optimization objectives. For instance, shortening the makespan of a workflow requires more powerful and expensive resources, which often leads to a higher execution cost.

Since evolutionary algorithms are capable of obtaining a set of compromise solutions [13], [14], over the past decade, increasing efforts have been devoted to designing multi-objective evolutionary algorithms to simultaneously optimize the makespan and monetary cost of workflow execution [15]. For instance, to make a good trade-off between makespan and monetary cost of workflow execution, Wu et al. [16] improved the nondominated sorting genetic algorithm II (NSGA-II) [17] with the classic list-based scheduling approach and multiple genetic operators. Chen et al. [18] proposed a multiple population based ant colony system algorithm (MOACS), in which two ant colonies were employed to respectively handle the two objectives. In MOACS, a new pheromone update mechanism, a co-evolutionary heuristic strategy, and an elite learning rule were designed to enhance the ant colony optimization [18]. Li et al. [19] proposed a new multi-objective workflow scheduling algorithm, called SDHN, to balance the makespan and cost of workflow execution in cloud platforms. The SDHN included a scoring metric to sort the solutions belonging to the same Pareto front and an adjustment strategy to choose suitable genetic operations adaptively [19]. Farid et al. [20] employed particle swarm optimization (PSO) to balance cost and makespan while meeting reliability constraints. Ismayilov et al. [21] studied the dynamism in resource failures and the number of objectives, and formulated the workflow scheduling problem as a dynamic multi-objective optimization problem. Afterward, the artificial neural network was embedded into the NSGA-II to solve the problem [21]. Verma et al. [22] integrated budget and deadline aware heuristic strategies into multi-objective PSO to balance makespan and cost of workflows while meeting their deadline and budget constraints. The genetic algorithm, artificial bee colony optimization, and decoding heuristic are integrated to schedule workflow tasks to cloud resources to simultaneously minimize the makespan and cost [23]. Konjaang et al. [24] designed a MaxVM selection and a MinVM selection for task allocation to reduce the monetary cost and makespan of workflows while guaranteeing their deadlines. Adhikari et al. [25] aggregated four competing objectives of workflow scheduling into a fitness function, and employed the Firefly algorithm to search the near-optimal solutions. However, these existing works often regard these problems as black-box problems and neglect the domain knowledge, resulting in the low efficiency of these algorithms.

Obviously, Multi-objective workflow scheduling remains a challenging issue. The reasons are as follows. First of all, one workflow often contains hundreds or thousands of tasks [26], which means that the corresponding multi-objective scheduling problem has a large number of decision variables. When increasing the decision variables, the solution space of the problem explodes exponentially, which is a challenging issue in the multi-objective evolutionary optimization community [27]. Then, the heterogeneity and elasticity of cloud resources substantially expand the solution space, which further challenges the MOEAs. In addition, the optimization objectives of makespan and monetary cost in workflow scheduling are of entirely different scales. When employing MOEAs to solve this kind of MOPs with quite different objective ranges, accurately estimating the ideal and nadir points is essential to normalize the objective space, which helps maintain the population diversity. But, the ideal and nadir points are often not accurately estimated by using the obtained population during the evolutionary process [28].

The divide-and-conquer methods are helpful to accelerate the convergence of MOEAs in solving MOPs with large-scale decision variables [29], [30]. In divide-and-conquer methods, the large-scale decision variables are divided into many groups according to their interactions [29], and decision variables in each group are optimized alternately. However, the data dependencies among workflow tasks make the decision variables of a multi-objective workflow scheduling problem cannot be separated into multiple groups according to their interactions. To handle this challenge, we mine the knowledge of workflow structure to divide parallel tasks into the same group, simplifying the large-scale workflow scheduling problem into a series of small-scale ones. Moreover, to handle the challenge posed by differently-scaled objectives, we explore the knowledge of workflow tasks, resource capacity, and resource price to estimate the ideal and nadir points before the optimization process.

Until now, there exist some works on adopting MOEAs to solve multi-objective workflow scheduling problems in clouds. But most of the existing works regard the problems as black-box ones. Unlike them, this paper strives to mine the knowledge of workflow structure, tasks, and cloud resources to design a knowledge-driven evolutionary algorithm for simultaneously optimizing the makespan and monetary cost of workflow execution in clouds. The new contributions are as follows. 1) Based on the knowledge of workflow structure, we tailor a new decision variable grouping strategy to accelerate the convergence speed of MOEAs in solving multi-objective workflow scheduling problems; 2) We mine the knowledge of workflow tasks and cloud resources to estimate the ideal and nadir points for objective space normalization, to maintain population diversity for MOEAs; 3) Extensive comparison experiments are conducted on real-world workflows to analyze the performance of the proposal.

This paper is organized as follows. The problem formulation is described in Section II, followed by the algorithm description in Section III. The experimental results are provided in Section IV. At last, the conclusions and two interesting future directions are provided in Section V.
II. MODEL OF MULTI-OBJECTIVE WORKFLOW SCHEDULING
This section first introduces the models of workflows and cloud resources, and then formulate the model of multi-objective workflow scheduling in cloud computing.

A. MODEL OF WORKFLOW
A workflow is generally modeled by a directed acyclic graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ represents the set of tasks, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ indicates the set of edges among tasks, respectively [16].

The existence of an edge $e_{i,j} \in \mathcal{E}$ means that the start of task $v_j$ needs to wait for the data output by task $v_i$. Then, task $v_i$ is known as a direct predecessor of task $v_j$, and task $v_j$ is called as a direct successor of task $v_i$. For a task $v_i$, all its direct predecessors are denoted as $\mathcal{P}(v_i)$, while all its direct successors are indicated as $\mathcal{S}(v_i)$.

Additionally, the weight $wt(v_i)$ of task $v_i$ denotes its computational workload, and the weight $wt(e_{i,j})$ of an edge $e_{i,j}$ indicates that amount of data transferred from task $v_i$ to task $v_j$.

B. MODEL OF CLOUD RESOURCES
In cloud computing environments, tenants can on-demand configure cloud resources, e.g., computing, network, and storage, via the Internet at any time, and only pay for their actual usage in a pay-as-you-go way [31]. The problems studied in this paper mainly involve three characteristics of cloud resources, i.e., heterogeneity, elasticity, and billing mode.

For heterogeneity, cloud platforms often offer various types of resources. We model all resource types as $\mathcal{T} = \{1, 2, \cdots, k\}$, where $k$ means the total number of types, and $t \in \mathcal{T}$ means the $t$-th type [5]. Different types of resources mainly differ in their performance configurations and rental prices. For type $t$, its price is presented as $p(t)$.

Elasticity is an outstanding feature of cloud computing. It means that when the load increases, more cloud resources can be expanded rapidly, and there is no limit to the number of each type of resource. When the load peak disappears, redundant resources can be released automatically to reduce costs. That is, the number of each type of resource can be increased or decreased as needed.

Consulting the charging models of well-known cloud providers (e.g., Amazon EC2 and Google Cloud Platform), they charge resource usages in a pay-as-you-go mode. In this paper, we follow this mode. Another feature of charging for cloud computing resources is that any partial usage of the resource hour is charged by a full-time period. For instance, a charging period of Amazon EC2 is 60 minutes, and if the resource is rented for 60.1 minutes, the renter needs to pay for 120 minutes [18].

C. MODEL OF MULTI-OBJECTIVE WORKFLOW SCHEDULING
The elasticity of cloud computing enables tenants to use unlimited resources. This feature leads to infinite search spaces for optimization problems, which evolutionary algorithms cannot address. It is worth noting that the number of tasks in a workflow is bounded, and the number of resources required is not infinite. Then, we can construct a candidate resource set with a fixed number.

In one workflow, some tasks have no data dependencies and can be executed in parallel. The symbol $T_{\text{max}}$ is employed to represent the maximum number of tasks that can be executed in parallel when running the corresponding workflow.

Considering the extreme situation, $T_{\text{max}}$ parallel tasks are executed by $T_{\text{max}}$ cloud resources with the same type [18]. To simultaneously reflect the elasticity of clouds and compress the search space, we suppose that the maximum number of resources of each type is $T_{\text{max}}$. Based on these, the resource set can be defined as $\mathcal{R} = \{r_1, r_2, \cdots, r_m\}$, where $m = k \times T_{\text{max}}$. For a task $v_i$, its price is presented as $p(v_i)$, while all its direct predecessors are indicated as $\mathcal{P}(v_i)$.

Another optimization objective of executing a workflow in the cloud is to minimize the makespan, which corresponds to the maximum finish time of all tasks. Then, the optimization objective of minimizing the makespan can be formulated as (2).

Another optimization objective of executing a workflow in the cloud is to minimize the makespan, which corresponds to the maximum finish time of all tasks. Then, the optimization objective of minimizing the makespan can be formulated as (2).

$$\text{Minimize } f_1(x) = \max_{v \in \mathcal{V}} f(v_{p}, r_{s_{p}}).$$

Another optimization objective of executing a workflow in the cloud is to minimize the monetary cost. We formulate this objective in (3).

$$\text{Minimize } f_2(x) = \sum_{r \in \mathcal{R}} p(r) \times \left\lfloor \frac{t_r - b_r}{L} \right\rfloor,$$

where $t_r$ and $b_r$ represent the turn-off and boot time of resource $r$, and $L$ represents the length of a charging period.
Note that the turn-off and boot time of resources without mapped tasks are assumed to be 0.

Based on the above analysis, the model of multi-objective workflow scheduling can be summarized as (4).

\[
\begin{align*}
\text{Minimize} & \quad f(x) = [f_1(x), f_2(x)], \\
\text{S.t.} & \quad (1), \\
& \quad x \in \{1, 2, \ldots, m\}^n.
\end{align*}
\]

The optimization objectives \(f_1(x)\) and \(f_2(x)\) are formulated in (2) and (3), respectively. The constraint condition (1) denotes the data dependencies among workflow tasks.

To solve the problem in (4), in the next section, two knowledge-based strategies are designed to improve the performance of the classical multi-objective evolutionary algorithm NSGA-II [17].

III. ALGORITHM DESIGN

In this subsection, we briefly introduce the concepts on multi-objective optimization, especially the dominance relationships and classical multi-objective evolutionary algorithms. In its environmental selection operator, the solutions in the combined population are split into many non-dominated fronts based on their Pareto-dominance relationships. The solutions in the last accepted front are sorted by crowding distance. For more details on NSGA-II, please refer to work [17]. When adopting NSGA-II to solve MOPs from a specific field, designing specific operators is required to enhance convergence and diversity.

B. DESCRIPTIONS OF PROPOSED KMEWSA

The proposed KMEWSA follows the framework of NSGA-II, and its overall procedure is given in Algorithm 1.

Algorithm 1 Overall Procedure of KMEWSA

Input: The multi-objective workflow scheduling problem in equation (4); a workflow \(\mathcal{G} = (\mathcal{V}, \mathcal{E})\); a set of candidate resource \(\mathcal{R}\); population size \(N\);

Output: A set of non-dominated solutions \(\mathbb{P}\);

1. Sort tasks in \(\mathcal{V}\) by an ascending order of the upward rank;
2. \(\mathcal{G} \leftarrow \text{DecisionVariableGrouping}(\mathcal{G}, x)\);
3. \([z_{\text{min}}, z_{\text{max}}] \leftarrow \text{ObtainIdealNadirPoint}(\mathcal{G})\);
4. \(\mathbb{P} \leftarrow \text{Randomly generate a population};
5. \(i \leftarrow 0;
6. \text{while Termination Condition is not Reached do}
7. \quad \text{if rand} < 0.5 \text{ then}
8. \quad \quad \text{index} \leftarrow \text{mod}(i, |\mathcal{G}|) + 1;
9. \quad \quad \mathcal{G}_s \leftarrow \mathcal{G}([\text{index}]);
10. \quad \quad i + +;
11. \quad \quad \mathbb{Q} \leftarrow \text{Evolve the decision variables in } \mathcal{G}_s \text{ to obtain an offspring population};
12. \quad \text{else}
13. \quad \quad \mathbb{Q} \leftarrow \text{Evolve all the decision variables to obtain an offspring population};
14. \quad \quad \mathbb{P} \leftarrow \text{SelectionOperator}(\mathbb{P} \cup \mathbb{Q}, N, z_{\text{min}}, z_{\text{max}});

The inputs of the proposed KMEWSA are: the optimization problem, a workflow, a set of candidate resources, and the population size. After the KMEWSA reaches the stop condition, it outputs a population \(\mathbb{P}\).

First of all, to ensure the data dependencies among workflow tasks, all the workflow tasks are sorted by their upward ranks [4] in ascending order. The upward rank \(\text{rank}(v_i)\) of a workflow task \(v_i\) can be recursively
calculated using (10).

\[
\text{rank}(v_i) = \begin{cases} 
\text{mrt}(v_i), & \text{if } S(v_i) = \emptyset, \\
\text{mrt}(v_i) + \max_{v_j \in S(v_i)} \left( \text{rank}(v_j) + \frac{\text{wt}(e_{i,j})}{\text{bw}} \right), & \text{Otherwise,}
\end{cases}
\]

(10)

where \(\text{mrt}(v_i)\) denotes the minimum runtime of task \(v_i\), \(\text{bw}\) represents the average bandwidth.

Then, function \(\text{DecisionVariableGrouping()}\) is called to divide the large-scale decision variables into a series of groups via mining the knowledge of workflow structure, which is detailed in Algorithm 2. Next, function \(\text{ObtainIdealNadirPoint()}\) is called to estimate the ideal and nadir points via mining the knowledge of workflow tasks and cloud resources, which is detailed in Algorithm 3. After that, a population is initialized, and a counter is used to record the evolution generations.

Based on the above operations, before the KMEWSA reaches the stop condition, it iterates the following two processes: 1) generation of a new population; 2) environmental selection. When generating a new population, the algorithm either evolves decision variables in each group sequentially or evolves all decision variables simultaneously. After a new population is generated, the environment selection operator will be triggered to select the offspring population from the combined population.

As the environmental selection is not the focus of this paper, in line 14 of Algorithm 1, the proposed KMEWSA employs the selection operator in NSGA-II [17] to select the best \(N\) solutions in the combined population \(P \cup Q\).

Clustering large-scale decision variables into many small groups and optimizing them one by one is an effective way to improve the convergence of evolutionary algorithm in solving large-scale multi-objective problems. Inspired by this, function \(\text{DecisionVariableGrouping()}\) is to divide the decision variables corresponding to a set of parallel workflow tasks into a group, as shown in Algorithm 2.

In Function \(\text{DecisionVariableGrouping()}\), the set \(G\) is used to record the decision variables in each group, and \(T_u\) is used to store the unselected workflow tasks. In set \(G\), each element stores a set of decision variables in the same group. After the two sets are initialized, this function runs as follows. The subgroup of decision variables \(G_x\) and the set of selected tasks \(T_x\) are initialized as empty. Afterward, each unselected task is checked. If a task has no predecessors or all its predecessors are not in set \(T_u\), this task will be selected in this iteration. Note that the corresponding decision variable of task \(v_i\) is \(x_i\). When all the unselected tasks are checked, the sets \(G\) and \(T_u\) are updated.

To visualize Function \(\text{DecisionVariableGrouping()}\), an example is provided in Figure 1.

As illustrated in Figure 1(a), the example workflow is composed of five tasks, i.e., \([v_1, v_2, \cdots, v_5]\). At first, the set \(T_u\) stores all the workflow tasks, i.e., \(T_u = \{v_1, v_2, \cdots, v_5\}\).

Since the task \(v_1\) has no predecessor, it is selected in the first iteration, and its corresponding decision variable \(x_1\) is divided into a group. The set \(T_u\) becomes \(T_u = \{v_2, \cdots, v_5\}\) during the second iteration, since the predecessor of tasks \(v_2, v_3, v_4\) are not in set \(T_u\), they are selected and their corresponding decision variables \(x_2, x_3, x_4\) are put into the same group. Similarly, the corresponding decision variable of task \(v_5\) is put into a group. The final grouping result is shown in Figure 1(b), and each rectangular box represents a group.

The Function \(\text{ObtainIdealNadirPoint()}\) is to estimate the ideal and nadir points for the multi-objective workflow scheduling problem, which is described in Algorithm 3.

The parameter \(M_{\text{min}}\) denotes the minimum makespan of the workflow, and it is estimated as follows. All the workflow tasks are scheduled to one most powerful type of resources, and the data transfer time among tasks is ignored. Then, the makespan of this scheduling scheme is regarded as \(M_{\text{min}}\).

The parameter \(C_{\text{min}}\) denotes the minimum cost of the workflow execution, and it is estimated as follows. All the workflow tasks are executed on the resources with the cheapest price, and the data transfer time among tasks is ignored. Then, the total execution time of this workflow is used to estimate the \(C_{\text{min}}\).

After the parameters \(M_{\text{min}}\) and \(C_{\text{min}}\) are estimated, the ideal point \(z_{\text{min}}\) can be obtained.
Algorithm 3 ObtainIdealNadirPoint()

Input: The workflow $G = (V, E)$;
Output: The ideal point $z_{\text{min}}$, and nadir point $z_{\text{max}}$;

1. $M_{\text{min}} \leftarrow \sum_{G_s \in G} \max_{v_i \in G_s} \{e_{\text{min}}(v_i)\}$;
2. $C_{\text{min}} \leftarrow \min_{t \in T} p(t) \times \lceil \sum_{v_i \in V} \max_{e_{\text{max}}(v_i)} / P \rceil$;
3. $z_{\text{min}} \leftarrow [M_{\text{min}}, C_{\text{min}}]$;
4. $M_{\text{max}} \leftarrow \sum_{G_s \in G} \max_{v_i \in G_s} \{e_{\text{max}}(v_i) + \sum_{v_j \in G_s} \max_{e_{\text{max}}(v_j)} \}$;
5. $C_{\text{max}} \leftarrow \max_{t \in T} p(t) \times \lceil \sum_{v_i \in V} \max_{e_{\text{max}}(v_i)} / P \rceil$;
6. $z_{\text{max}} \leftarrow [M_{\text{max}}, C_{\text{max}}]$;

The parameter $M_{\text{max}}$ denotes the maximum makespan of the workflow, and it is estimated as follows. The maximum execution time of all the workflow tasks and data transfer time among tasks are considered. Then, each workflow task is executed by one resource, and the makespan of this scheduling scheme is regarded as $M_{\text{max}}$.

The parameter $C_{\text{max}}$ denotes the maximum cost of the workflow execution, and it is estimated as follows. The only use the resource with the most expensive price, and schedule each workflow task to a resource, and the cost of this scheduling scheme is regarded as $C_{\text{max}}$.

Then, based on the maximum makespan and cost, the nadir point $z_{\text{max}}$ can be obtained.

C. COMPLEXITY ANALYSIS

The time complexity of sorting workflow tasks is $O(n \cdot e)$, where $n$ and $e$ denote the number of tasks and edges, respectively. In algorithm 2, it takes $O(n^2)$ to group all the workflow tasks. The time complexity of estimating the ideal point is $O(n \cdot k)$, where $k$ denotes the number of resource types. To estimate the nadir point, the time complexity is $O(n^2 \cdot k)$.

The time complexity of generating a new population is $O(n \cdot N)$, where $N$ denotes the population size. Similar to NSGA-II [17], the environmental selection procedure costs $O(N \log(N))$. Thus, the complexity of main optimization is $O(g \cdot n \cdot N + g \cdot N \log(N))$, with $g$ generations.

In sum, the computational complexity of the KMEWSA is $O(n \cdot e + n^2 \cdot k + g \cdot n \cdot N + g \cdot N \log(N))$.

IV. EXPERIMENTAL VERIFICATION

In this section, based on twenty real-world workflows and parameters of resource instances in Amazon EC2, we carry out numerical experiments to verify the competitive performance of the proposal.

A. EXPERIMENTAL SETUP

The Pegasus repository\footnote{https://confluence.pegasus.isi.edu/display/pegasus} has released a workflow suite from five different fields, i.e., Montage from Astronomy, Inspiral...
from Gravitational physics, CyberShake from Earthquake, Sipht from Bioinformatics, and EpiGenomics from Biology. The topological structures of these workflows with small-scale tasks are symbolically represented in Figure 2. These workflows pose quite different features, and have been widely used in the performance verification of workflow scheduling approaches. We choose four instances for each workflow with around 30, 50, 100, and 1000 tasks. That is, a total of 20 workflow instances are used for the experiment. Some main characteristics of these workflows are summarized in Table 1.

For cloud resources, we adopt five types of instances and the pricing model from Amazon EC2. The relevant parameters of these resource types are summarized in Table 2. The billing period of resources is set to 60 min and all partial usages are rounded up to one billing period. The bandwidth among resources is set to 1 Gbps.

### Table 2. Parameters of five resource types.

| Type       | Price ($/h) | vCPU | Memory (GB) |
|------------|-------------|------|-------------|
| a1.medium  | 0.0255      | 1    | 2.0         |
| a1.large   | 0.051       | 2    | 4.0         |
| a1.xlarge  | 0.102       | 4    | 8.0         |
| a1.2xlarge | 0.204       | 8    | 16.0        |
| a1.4xlarge | 0.408       | 16   | 32.0        |

The proposed KMEWSA is compared with three baseline algorithms: MOELS [16], MOHEFT [35], and KMEWSA-N. The existing MOELS follows the framework of classical MOEA NSGA-II, and contains a list-based scheduling strategy. The MOHEFT is also an existing multi-objective cloud workflow scheduling algorithm based on heuristics. Compared with MOELS and MOHEFT, the purpose is to examine the advantages of the proposed knowledge-driven approach over the state-of-the-art approaches. The KMEWSA-N is a variant of the proposed KMEWSA. In KMEWSA-N, the ideal and nadir points are estimated using the current population. The purpose of comparing KMEWSA with KMEWSA-N is to distinguish the contribution of the estimated ideal and nadir points in KMEWSA to its overall performance.

Hyppervolume (HV) [36], referring to the volume between a reference point and a set of non-dominated solutions, is a widely used indicator to measure the quality of the output population of MOEA. In general, the larger HV means that the output population is closer to the Pareto front and has better distribution, thus is preferred.

We also employ the pure diversity (PD) [37] to measure the diversity of population $\mathbb{P}$, which is defined in (11).

$$PD(\mathbb{P}) = \max_{x \in \mathbb{P}} PD(\mathbb{P} - x) + dis(x, \mathbb{P} - x),$$

where $dis(x, \mathbb{P} - x)$ represents the dissimilarity of solution $x$ to the population $\mathbb{P}$, as calculated in (12).

$$dis(x, \mathbb{P} - x) = \min_{x' \in \mathbb{P}}(dissimilarity(x, x')),$$

To compare the dominance relationship between populations output by different algorithms, we employ the metric on the coverage of two populations [18], which is defined in (13).

$$C(\mathbb{P}, \mathbb{Q}) = \frac{|\{x \in \mathbb{Q}, \exists x' \in \mathbb{P}, x' \text{ dominates or equal to } x\}|}{|\mathbb{Q}|},$$

where $\mathbb{P}$ and $\mathbb{Q}$ are two populations. As shown in (13), $C(\mathbb{P}, \mathbb{Q})$ denotes the ratio of solutions in $\mathbb{Q}$ that are dominated by solutions in $\mathbb{P}$.

Similar to MOELS [16], the population size of the four algorithms is set to 100. The maximum number of iterations is $100 \times n$, where $n$ represents the number of tasks in the workflow. We set the distribution indexes of simulated binary crossover and the polynomial mutation are set to be 10 and 20. Besides, the probabilities for crossover and mutation operators are set as 1 and $1/n$, respectively.

For each set of experimental settings, each multi-objective workflow scheduling algorithm is repeated 51 times to reduce or avoid random effects. All the experiments are performed on a PC with one Core i7, 3.4GHz CPU, 16GB RAM, Windows 10, JDK-8u251, and Eclipse IDE.

#### B. Comparison Experiments Based on Synthetic Workflows

In terms of metrics HV and PD, the comparison results of the four algorithms are summarized in Tables 3 and 4. The Wilcoxon rank-sum test with the significance level of 0.05 is performed to identify the significance between different algorithms for the results on each workflow instance. The signs $+$, $-$, and $\approx$ indicate that the corresponding algorithm is significantly better than, worse than, and similar to the proposed KMEWSA. Besides, on each workflow instance, the best results among the four algorithms are highlighted in bold.

From Table 3, we can summarize that the proposal significantly outperforms all the four baseline algorithms on 13 out of the 20 workflow instances. Specifically, in terms of metric HV, the proposed KMEWSA significantly performs better MOELS, MOHEFT, and KMEWSA-N on 13, 20, and 18 out of the 20 workflow instances, respectively. These comparison results illustrate that the proposal poses better overall performance.

Compared with KMEWSA-N, the performance promotion of KMEWSA can be explained by that the estimated ideal and nadir points based on the knowledge of workflow tasks and resources are helpful for maintaining the population diversity. The comparison results between KMEWSA and KMEWSA-N demonstrate that the proposed knowledge-based ideal and nadir points estimation approach is effective. Except for CyberShake workflows with different tasks, the KMEWSA and its variant KMEWSA-N perform no worse than algorithm MOELS. The fundamental reason is that the number of tasks in different parallel task sets varies greatly, as shown in Figure 2(c), which leads to significant differences.
in the number of decision variables in different groups. But, the decision variables in different groups get the same evolution opportunities, resulting in insufficient evolution for the groups with more decision variables.

The biggest difference between the proposal and the existing algorithm MOELS is that the former employs a clustering strategy to divide large-scale decision variables into a series of small-scale decision variables. Then, the advantage of algorithm KMEWSA and KMEWSA-N over compared algorithm MOELS implies the effectiveness of the knowledge-driven clustering strategy in accelerating population convergence.
Besides, the standard variance of MOHEFT in all the workflow instances is much lower than that of other algorithms. This phenomenon is because the MOHEFT is based on heuristic rules, with few random factors, bringing little change in the results of repeated experiments.

Table 4 compares the KMEWSA with MOELS, MOHEFT, and KMEWSA-N on metric PD. From Table 4, we can observe that the population diversity of KMEWSA is far better than the compared algorithms. Among the three baseline algorithms, the MOHEFT performs the best. We can also see that the proposed KMEWSA generates higher PD values than MOHEFT on most workflow instances except Montage100 and CyberShake1000. At the same time, the advantage of KMEWSA is obvious on other workflow instances.

Table 5 reports the comparison results in terms of coverage ratio. Compared with MOELS, MOHEFT, and KMEWSA-N, the values of $C(KMEWSA, \Diamond)$ and $C(\Diamond, KMEWSA)$ in most cases are much higher. Some are respectively 100% and 0%, which implies that the output population of KMEWSA completely dominates that of MOELS, MOHEFT, and KMEWSA-N.

To intuitively compare the performance of the four algorithms, Figure 3 shows their output populations with the largest HV values among 51 repeated experiments on nine workflow instances. The symbol Montage 50 represents the Montage workflow with 50 tasks.

As can be observed in Figure 3, the populations obtained by KMEWSA are superior to those of the three baseline algorithms on problems derived from workflows Montage, Inspiral, Sipht, and Epigenomics. More specifically, on these problems, the output solutions of KMEWSA are well-distributed and dominate the solutions obtained by MOELS, MOHEFT, and KMEWSA-N. These observations are consistent with the larger HV values generated by the algorithm KMEWSA in Table 3.
TABLE 5. Comparisons of KMEWSA (denoted as $\ast$) with three baseline algorithms (denoted as $\sim$) in terms of $C(A, B)$ values.

| Workflows  | Task Number | MOELS $C(\ast, \sim)$ | MOHELPT $C(\ast, \sim)$ | KMEWSA-N $C(\ast, \sim)$ |
|------------|-------------|------------------------|-------------------------|-------------------------|
| Montage    | 25          | 100%                   | 0%                      | 96%                     | 99%                     | 3%                      |
|            | 50          | 86.8%                  | 0%                      | 86%                     | 67%                     | 0%                      |
|            | 100         | 71.7%                  | 0%                      | 76%                     | 48%                     | 0%                      |
|            | 1000        | 91%                    | 0%                      | 60%                     | 67%                     | 0%                      |
| Inspiral   | 30          | 100%                   | 0%                      | 100%                    | 97%                     | 1%                      |
|            | 50          | 100%                   | 0%                      | 100%                    | 95%                     | 3%                      |
|            | 100         | 98.9%                  | 0%                      | 100%                    | 91%                     | 0%                      |
|            | 1000        | 100%                   | 0%                      | 100%                    | 93%                     | 12%                     |
| CyberShake | 30          | 20%                    | 32%                     | 100%                    | 100%                    | 0%                      |
|            | 50          | 12.8%                  | 40.3%                   | 100%                    | 87%                     | 0%                      |
|            | 100         | 9.5%                   | 84%                     | 100%                    | 73%                     | 0%                      |
|            | 1000        | 15%                    | 69%                     | 68%                     | 64%                     | 0%                      |
| Sipt       | 30          | 100%                   | 0%                      | 100%                    | 99%                     | 6%                      |
|            | 60          | 98.7%                  | 0%                      | 100%                    | 100%                    | 1%                      |
|            | 100         | 100%                   | 0%                      | 100%                    | 98%                     | 1%                      |
|            | 1000        | 100%                   | 0%                      | 100%                    | 95%                     | 0%                      |
| Epigenomics| 24          | 100%                   | 0%                      | 100%                    | 100%                    | 0%                      |
|            | 46          | 97%                    | 0%                      | 100%                    | 98%                     | 7%                      |
|            | 100         | 100%                   | 0%                      | 98%                     | 100%                    | 4%                      |
|            | 997         | 92%                    | 3%                      | 96%                     | 2%                      | 93%                     | 6%                      |

For workflow CyberShake (with 50 and 100 tasks), the HV values of KMEWSA are inferior to that of algorithm MOELS. The inherent reason is that tasks in these workflows can only be divided into fewer groups, and most groups contain only one task. The groups with fewer tasks consume the same computing resources as the groups with more tasks. This is not conducive to the overall performance of the algorithm KMEWSA.

V. CONCLUSION AND FUTURE WORK

This article strives to mine the knowledge of workflow structure, tasks, and cloud resources to make a sound balance of makespan and cost for deploying workflows in cloud platforms. Specifically, the knowledge of workflow structure is mined to group large-scale decision variables into small components, which are evolved alternately to accelerate the convergence speed of MOEAs in solving multi-objective workflow scheduling problems. In addition, the knowledge of workflow tasks and cloud resources is mined to estimate the ideal and nadir points for objective space normalization to maintain population diversity for MOEAs. Furthermore, we perform extensive simulation experiments based on real-world workflows and cloud platforms to compare the proposal with three baseline algorithms. The numerical results illustrate that the proposal significantly outperforms all the three algorithms in terms of HV (13 out of 20) and PD (15 out of 20), and its output solutions dominate that of the baseline algorithms in most cases. These results reveal that the proposal is competitive in solving multi-objective workflow scheduling problems.

In many cloud application scenarios, the optimization time allowed by the platform is very limited, so how to compress the search time of multi-objective evolutionary algorithms is one interesting direction. In addition, applying powerful machine learning techniques [38], [39] to assist new solutions generation is also a promising direction.

REFERENCES

[1] B. P. Rimal and M. Maier, “Workflow scheduling in multi-tenant cloud computing environments,” IEEE Trans. Parallel Distrib. Syst., vol. 28, no. 1, pp. 290–304, Jan. 2017.
[2] I. Sadooghi, J. H. Martin, T. Li, K. Brandstatter, K. Maheshwari, T. P. P. de Lacerda Ruivo, G. Garzoglio, S. Tinnm, Y. Zhao, and I. Raicu, “Understanding the performance and potential of cloud computing for scientific applications,” IEEE Trans. Cloud Comput., vol. 5, no. 2, pp. 358–371, Apr. 2017.
[3] M. H. Hilman, M. A. Rodriguez, and R. Buyya, “Multiple workflows scheduling in multi-tenant distributed systems: A taxonomy and future directions,” ACM Comput. Surv., vol. 53, no. 1, pp. 1–39, 2020.
[4] H. Topcuoglu, S. Hariri, and M.-Y. Wu, “Performance-effective and low-complexity task scheduling for heterogeneous computing,” IEEE Trans. Parallel Distrib. Syst., vol. 13, no. 3, pp. 260–274, Mar. 2002.
[5] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, “Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds,” Future Gener. Comput. Syst., vol. 29, no. 1, pp. 158–169, Jan. 2013.
[6] M. A. Rodriguez and R. Buyya, “Deadline based resource provisionin gand scheduling algorithm for scientific workflows on clouds,” IEEE Trans. Cloud Comput., vol. 2, no. 2, pp. 222–235, Apr./Jun. 2014.
[7] R. A. Hasan, S. N. Shahab, and M. A. Ahmed, “Correlation with the fundamental PSO and PSO modifications to be hybrid swarm optimization,” Iraqi J. Comput. Sci. Math., vol. 2, no. 2, pp. 25–32, 2021.
[8] Y.-H. Jia, W.-N. Chen, H. Yuan, T. Gu, H. Zhang, Y. Gao, and J. Zhang, “An intelligent cloud workflow scheduling system with time estimation and adaptive ant colony optimization,” IEEE Trans. Syst. Man, Cybern., Syst., vol. 51, no. 1, pp. 634–649, Jan. 2021.
[22] A. Verma and S. Kaushal, “A hybrid multi-objective particle swarm optimization for scientific workflow scheduling,” Parallel Comput., vol. 62, pp. 1–19, Feb. 2017.

[23] Y. Gao, S. Zheng, and Y. Zhou, “A hybrid algorithm for multi-objective scientific workflow scheduling in IaaS cloud,” IEEE Access, vol. 7, pp. 125783–125795, 2019.

[24] J. K. Konjaang and L. Xu, “Multi-objective workflow optimization strategy (MOWOS) for cloud computing,” J. Cloud Comput., vol. 10, no. 1, pp. 1–19, Dec. 2021.

[25] M. Adhikari, T. Amgoth, and S. N. Sriramra, “Multi-objective scheduling strategy for scientific workflows in cloud environment: A firefly-based approach.” Appl. Soft Comput., vol. 93, Aug. 2020, Art. no. 106411.

[26] H. Chen, X. Zou, G. Liu, and W. Pedrycz, “Uncertainty-aware online scheduling for real-time workflows in cloud service environment,” IEEE Trans. Services Comput., vol. 14, no. 4, pp. 1167–1178, Jul. 2021.

[27] W.-J. Hong, P. Yang, and K. Tang, “Evolutionary computation for large-scale multi-objective optimization: A decade of progresses,” Int. J. Automat. Comput., vol. 18, pp. 155–169, Jan. 2021.

[28] L. He, H. Ishibuchi, A. Trivedi, H. Wang, Y. Nan, and D. Srinivasan, “A survey of normalization methods in multiobjective evolutionary algorithms,” IEEE Trans. Evol. Comput., vol. 25, no. 6, pp. 1029–1048, Dec. 2021.

[29] M. N. Omidvar, M. Yang, Y. Mei, X. Li, and X. Yao, “DG2: A faster and more accurate differential grouping for large-scale black-box optimization,” IEEE Trans. Evol. Comput., vol. 21, no. 6, pp. 929–942, Dec. 2017.

[30] H. Chen, R. Cheng, J. Wen, H. Li, and J. Weng, “Solving large-scale many-objective optimization problems by covariance matrix adaptation evolution strategy with scalable small subpopulations,” Inf. Sci., vol. 509, pp. 457–469, Jan. 2020.