Adaptive Cloud Bundle Provisioning and Multi-Workflow Scheduling via Coalition Reinforcement Learning

Xiaogang Wang, *Member, IEEE*, Jian Cao, *Senior Member, IEEE*, and Rajkumar Buyya, *Fellow, IEEE*

Abstract—The efficient cloud resource provisioning for the execution of complex workflow applications has always been one of the important research issues. Most of the existing approaches focus on the resource provisioning of single-type virtual machine (VM) instances for the single or multiple workflows, while few consider the situation of provisioning multi-type VM instances simultaneously. As a result, the executing performance of complex workflows degrades. Different from the existing work, this paper proposes an adaptive cloud bundle provisioning and multi-workflow scheduling model to dynamically perform both the horizontal and vertical cloud resource scaling on multi-type VM instances for the execution of complex workflows. Among the model, a depth-first-search coalition reinforcement learning (DFSCRL) provisioning policy is presented to realize the resource scaling, which integrates the physical machine (PM) coalition formation with the Q-learning algorithm, then dynamically generates an optimal multi-type VM instance bundle from the PM coalition, and finally provisions these instances to the concurrent execution of multiple workflows. The theoretical proofs and various experiments with the multifaceted metrics demonstrate that the performance of the proposed algorithms is superior to that of the state-of-the-art relevant policies.

Index Terms—Cloud computing, coalition formation, depth-first-search, multi -type VM instance bundle provisioning, multi-workflow scheduling, reinforcement learning

1 Introduction

Lots of complex applications (e.g., various domains of scientific workflows) are deployed to cloud computing systems with abundant resources, especially for public clouds such as Amazon AWS, Windows Azure and Aliyun. These cloud service providers build their solid infrastructure services based on high-speed interconnected physical machine (PM) computing nodes similar to server farms in a data center that contains a large amount of meta-computing resources (e.g., CPU cores, Memory sizes, Storage sizes, etc.). These cloud resources in PMs are virtualized into many virtual machine (VM) instances with multiple types (combinations) of resources to the execution of workflow tasks on the pay-per-use mode over the Internet [1], [2], in the meantime, guaranteeing the signed service level agreements (SLA) [3] and minimizing the execution time of workflows and the resource usage costs. Different from the general business systems with independent tasks each other, complex workflow applications consist of many logic-dependent tasks, such that it is necessary to analyze the task branches and the processing relationships between tasks when scheduling the workflow tasks to the VM instances.

In the workflow scheduling and execution, there are some important issues that need to be addressed, e.g., minimizing the total execution time and the makespan of workflows, maximizing the cloud resource execution efficiency, and minimizing the renting cost of cloud VMs. At the same time, we should consider both the applications with multiple types of workflows and the self-scaling capability of multi-type VM resources before scheduling the workflow tasks. Even though some studies [4], [5] have been conducted on the cloud resource provisioning and workflow task scheduling, they are generally based on single workflow applications and only adjust the number of VM instances of one type at a time. This makes it difficult to deal with complex multi-workflow applications, and the cloud resource provisioning efficiency is not high, which greatly impacts on the performance of the workflow scheduling.

The workflow scheduling is often confronted with the problem of uncertain task execution time, and the types of workflows are also diverse, which make the provision of

- *Xiaogang Wang was with the CLOUDS Laboratory, School of Computing and Information Systems, The University of Melbourne, Melbourne, VIC 3010, Australia. He is now with School of Electronic and Information, Shanghai Dianji University, Shanghai 201306, China. E-mail: wangxg@sdju.edu.cn.*
- *Jian Cao is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: cao-jian@cs.sjtu.edu.cn.*
- *Rajkumar Buyya is with the CLOUDS Laboratory in the School of Computing and Information Systems, The University of Melbourne, Melbourne, VIC 3010, Australia. E-mail: rbuyya@unimelb.edu.au.*

Manuscript received 27 September 2021; revised 12 May 2022; accepted 10 July 2022. Date of publication 18 July 2022; date of current version 13 March 2023.
This work was supported in part by the National Natural Science Foundation of China under Grants 61702320 and 62072301, in part by the Natural Science Foundation of Shanghai Science and Technology Innovation Action Plan under Grant 22ZR1425300, in part by the Teacher Professional Development Project of Shanghai Municipal Education Commission in 2019, and in part by the ARC Discovery Project from the CLOUDS Laboratory at the University of Melbourne. (Corresponding authors: Xiaogang Wang and Jian Cao.)
Recommended for acceptance by Jian Cao.
This article has supplementary downloadable material available at https://doi.org/10.1109/TC.2022.3191733, provided by the authors.
Digital Object Identifier no. 10.1109/TC.2022.3191733

cloud resources become dynamic and complex. Some recent work, such as the stochastic dynamic level scheduling (SDLS) algorithm [6], the proactive and reactive scheduling methods [7], and the dynamic cloud resource provisioning and scheduling algorithm by using the sum of task execution time expectation and standard deviation [8], can solve the problem of the task scheduling efficiently, mitigate the impact of uncertainty on the task scheduling, and meet the workflow deadline, respectively. However, these approaches still focus on the task scheduling of the single workflow by using single type of VM instances. The latest methods in [9], [10] separately put forward an uncertainty-aware online scheduling algorithm for dynamic and multiple workflows with deadlines, and a cloud-based workflow scheduling policy for compute-intensive workflow applications. Although these two methods address the multi-workflow scheduling problem within a given deadline, the workflow scheduling performance suffers some degree of degradation because multi-type VM instances cannot be provisioned during each task scheduling process.

In the adaptive cloud resource provisioning for the execution of multi-workflow tasks, the approaches [11], [12], [13], [14], [15], [16] based on reinforcement learning (RL) have been proposed to automatically calculate out the appropriate number of VM resources by taking the best action that corresponds to the maximum long-term reward per decision round. However, the strategy in [11] fails to consider logic-dependent workflow tasks when allocating VM instances, and different types of VM instances are only generated from a PM each time. The method in [12] does also not take interdependent workflow tasks into account, and adjusts the number of only one type of VM instances at a time. The approach in [13] uses the deep RL model to calculate the number of different types of VM instances for workflow tasks, but these multi-type VM instances are also from only one server (PM) when provisioning the cloud resources. The experimental system in [14] was designed by using RL for the online QoS aware adaptive task allocation schemes, but the tasks are independent, and have not the dependencies between them. In [15], a hybrid anomaly-aware deep RL-based cloud resource scaling (ADRL) approach was presented only in the single-workflow cloud setting. Similar to our multi-workflow scenario, a RL-based scheduler in [16] was proposed to provide heterogeneous software and hardware resources in the multi-tenant cloud computing, however, this resource provisioning mode is also single-type, and cannot more effectively organize the multi-type VM instances at a time to the workflows.

In view of the limitations of the above approaches, this paper concentrates on the dynamic multi-type VM instance provisioning that supports the single- and multi-workflow scheduling, especially for multi-type scientific workflows, and proposes an adaptive cloud bundle provisioning and multi-workflow scheduling model. This model mainly involves a depth-first-search (DFS) coalition reinforcement learning (CRL) provisioning policy (abbreviated as the DFSCRL) that combines the PM coalition formation with the Q-learning algorithm. The DFSCRL policy changes the various types of input workflows (i.e., composed of some complex dependent subtasks) into some DFS queues, then forms an optimal PM coalition in the existing adjacent PMs of a data center through multiple rounds of the CRL execution, and further generates an optimal bundle (group) of multi-type VM instances from the PM coalition. The transformed DFS workflow task branches are concurrently scheduled into the optimal VM instance bundle. The proposed theorems and abundant experiments indicate that the DFSCRL policy has the overall advantage than other policies in the multifaceted metrics.

The main contributions of this paper are summarized as follows:

- An adaptive cloud bundle provisioning and multi-workflow scheduling model is proposed to dynamically perform both the horizontal and vertical cloud resource scaling for the execution of multiple workflows.
- A depth-first-search coalition reinforcement learning (DFSCRL) provisioning policy is put forward, which integrates the PM coalition formation with the Q-learning algorithm.
- An optimal multi-type VM instance bundle is generated from the PM coalition formed by using the DFSCRL policy, and further scheduled to the concurrent execution of multiple workflows.
- Theoretical and experimental results demonstrate that the proposed algorithms outperform the existing relevant approaches in the different performance evaluation metrics.

The rest of the paper is organized as follows. Section 2 outlines the related work. In Section 3, the overall structure of the proposed adaptive cloud bundle provisioning and multi-workflow scheduling model and its formulation are elaborated. Section 4 describes CRL-based cloud resource provisioning mechanism, and the DFSCRL and multi-workflow task scheduling algorithms in detail. In Section 5, the proposed policy and algorithms are evaluated through the different performance metrics. Finally, we draw the conclusions and give future work in Section 6.

2 Related Work

There has been much research work on the cloud resource provisioning and workflow scheduling problem in recent years. We summarize the existing methods into three categories: (a) *single-type cloud resource provisioning for the single-workflow scheduling*; (b) *multi-type cloud resource provisioning for the single-workflow scheduling*; and (c) *single-type cloud resource provisioning for the multi-workflow scheduling*.

2.1 Single-Type Cloud Resource Provisioning for the Single-Workflow Scheduling

Some researches use intelligent optimization algorithms to realize the resource allocation and task scheduling with multi-objective constraints. Several multi-objective scheduling approaches based on evolutionary algorithms [4], [17] were presented to convert multiple QoS constraints into a single objective optimization constraint. Some types of particle swarm optimization (PSO) scheduling algorithms [18] were proposed to minimize the overall execution cost of workflows when meeting the deadline constraints, while the other ones such as NSPSO [19] and ε-Fuzzy PSO [20] used the PSO algorithm to generate the Pareto optimal tradeoffs between the makespan and cost. The two classical genetic algorithms-based methods such as NSGA-II and SPEA2 were explained in [21], which solve the workflow execution planning problem for different workflow structures and constraint levels of users.

The above methods can optimize one or more objective well under the given constraints, however, most of them are the single-type VM resource provisioning for the single-workflow scheduling, and were devised in the context of grid computing environment. Further, the multi-objective scheduling methods are time consuming and not very suitable for large workflow applications, also it is difficult to cope with the dynamic provisioning of multiple types of resources for multi-workflow scheduling.

2.2 Multi-Type Cloud Resource Provisioning for the Single-Workflow Scheduling

Many studies use the heuristics to solve the efficient cloud resource provisioning and workflow scheduling problem with certain constrains of the cloud resources and budget in grid and cloud systems. Heterogeneous earliest-finish-time (HEFT) [22] policy is a performance-effective workflow scheduling method that schedules workflow tasks to the highest rank of processors on the total task computational and communication cost of task nodes from a workflow. Budget-constrained earliest-finish-time (BHEFT) [23] policy extends the HEFT, which finds a feasible plan for the execution of the workflow within a certain deadline and budget. A novel greedy resource provisioning and modified HEFT (GRP-HEFT) [24] approach was proposed to minimize the makespan of a given workflow within a budget constraint for the IaaS clouds. In terms of focusing on the execution time of workflow tasks, latest work [25] presented a novel two-stage machine learning approach to predict the execution time of workflow tasks for varying input data of tasks in the cloud, and another new method [26] proposed a new task scheduling algorithm with a weight-based mechanism to preassign energy consumption for unassigned tasks, which can minimize the schedule length (the execution time interval of tasks) for energy consumption.

Although the above approaches can be used for heterogeneous processors or multi-type cloud resources, however, their resource provisioning are mainly for the single-workflow scheduling. Moreover, the matching of multi-type cloud resources with multi-workflow scale is not considered, which will lead to the increase of multi-workflow scheduling costs.

2.3 Single-Type Cloud Resource Provisioning for the Multi-Workflow Scheduling

There is little work on the resource provisioning and task scheduling for multi-workflow or multi-tenant applications. Kwok *et al.* [27] addressed the calculations of resource requirements for multi-tenants with applied constraints, and proposed the optimal placement of tenants and instances with maximum cost savings while not violating service level agreements for all tenants in a set of servers. Tsai *et al.* [28] put forward a two-tier SaaS scaling and scheduling architecture to save resources, further proposed the duplication strategies including the lazy duplication and pro-active duplication to achieve higher system performance in a clustered cloud environment. To create a cost-effective scalable environment, the formal measurements were constructed in [29] for under- and over-provisioning of cloud resources by considering the multi-tenancy. The recent work in [10] proposed the novel cloud-based workflow scheduling (CWSA) policy to minimize the different execution performance of workflows for compute-intensive workflow applications in multi-tenant cloud environments. The other new work in [9] devised the uncertainty-aware online scheduling algorithm (ROSA) to schedule dynamic and multiple workflows with deadlines. However, these methods do not take into account the dynamic provisioning of multi-type VM resources according to the characteristics of multi-workflow scheduling.

In addition, for the provisioning of multi-type VM instances, several cloud coalition or federation formation mechanisms were proposed in the recent work [30], [31], [32], which use the coalition or federation game model to generate their appropriate group of multi-type VM instances that are allocated to the workflow subtasks. The cloud federation formation mechanism was devised in [30], which can dynamically form the cloud federation for maximizing the providers' profit. Further, the sub-task scheduling mechanism was proposed in a generated winner coalition [31]. In [32], Marinescu *et al.* proposed a two-stage protocol to manage cloud resources, which first dynamically forms rack-level coalitions of servers to execute a workflow component, second creates a package of these coalitions to support all the components of the complete workflow. However, these mechanisms do not consider multi-workflow scheduling and the complex dependencies between workflow subtasks, and their VM instance coalitions are only static, which cannot be dynamically adjusted according to the actual running workflows.

Different from all the above methods, this paper proposes the adaptive *multi-type VM instance bundle provisioning for the multi-workflow scheduling*, which simultaneously considers the adaptability of resource bundle provisioning, the multi-workflow task branches and the processing relationships between tasks, which can effectively improve various performance metrics of the workflow execution.

3 Adaptive Cloud Bundle Provisioning Model

In a cloud computing environment, the task workloads are often submitted in the form of a number of jobs, each of which contains some user tasks with dependencies. Since user jobs consist of the process flows of tasks, this paper refers to the user jobs as multiple workflows, and a job is just a workflow. This section presents the adaptive cloud bundle provisioning model for the multi-workflow scheduling, and gives the formulation of the proposed problem.

3.1 System Model

There are some related researches about cloud workflow management systems, but many of them mainly focus on the workflow scheduling algorithms and the limited cloud resource (e.g., one type of VM instances) provisioning adjustment. To enhance the flexibility of cloud resource allocation

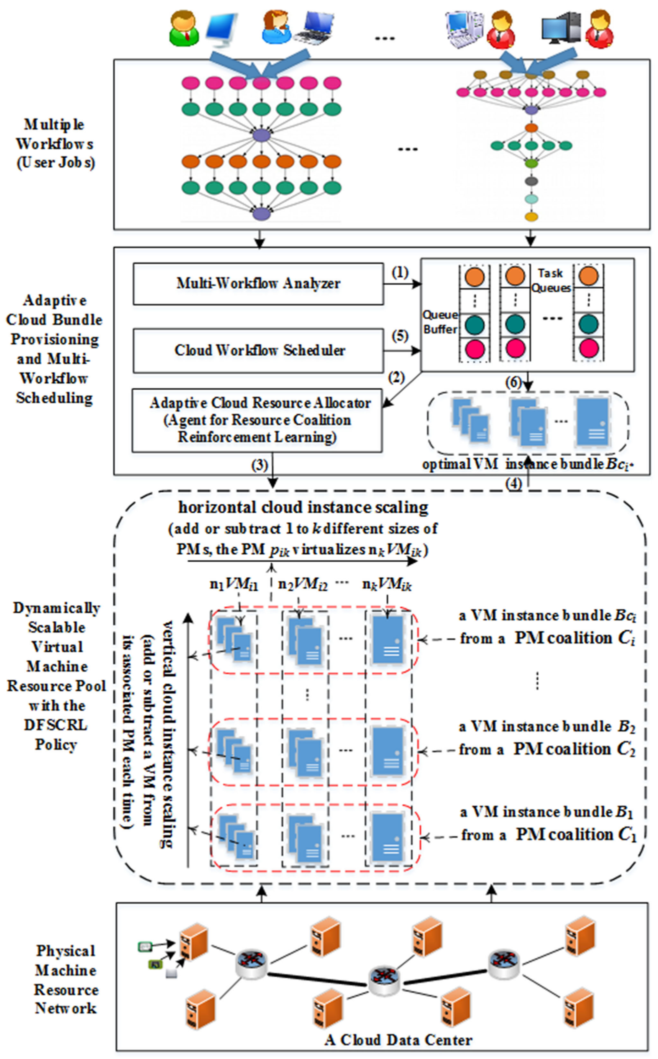
Multiple Workflows (User Jobs)
Adaptive Cloud Bundle Provisioning and Multi-Workflow Scheduling
Multi-Workflow Analyzer
(1)
Task Queues
Queue Buffer
Cloud Workflow Scheduler
(5)
(2)
Adaptive Cloud Resource Allocator (Agent for Resource Coalition Reinforcement Learning)
(6)
(3)
optimal VM instance bundle $B_{C_i^*}$
(4)
horizontal cloud instance scaling (add or subtract 1 to k different sizes of PMs, the PM p_{ik} virtualizes $n_k VM_{ik}$)
n_1VM_{i1} n_2VM_{i2} ... n_kVM_{ik}
Dynamically Scalable Virtual Machine Resource Pool with the DFSCRL Policy
vertical cloud instance scaling (add or subtract a VM from its associated PM each time)
a VM instance bundle B_{C_i} from a PM coalition C_i
a VM instance bundle B_2 from a PM coalition C_2
a VM instance bundle B_1 from a PM coalition C_1
Physical Machine Resource Network
A Cloud Data Center

Fig. 1. Adaptive cloud bundle provisioning and multi-workflow scheduling model.

and adjustment, we propose the adaptive cloud bundle provisioning model that can efficiently perform the scaling of both horizontal and vertical cloud resources for the multi-workflow scheduling with the coalition reinforcement learning (CRL). Fig. 1 depicts a general framework of the proposed system model that comprises four layers, i.e., the Multiple Workflows (User Jobs) layer, the Adaptive Cloud Bundle Provisioning and Multi-Workflow Scheduling layer, the Virtual Machine Resource Pool layer, and the Physical Machine Resource Network layer.

The Multiple Workflows (User Jobs) layer has a lot of different types of workflows initiated by users from diversified domains. These workflows or jobs consist of many tasks with some sequential dependency, and are usually added some constraints such as the deadline of running jobs, the size of files, and the number of cloud resource plan. Further, one or more workflows may form a complex user application, e.g., a common business application or a scientific computing application.

The Adaptive Cloud Bundle Provisioning and Multi-Workflow Scheduling layer is the core of the proposed model, which provides adaptive horizontal and vertical cloud resource provisioning mechanism to support the multi-workflow scheduling. It includes four modules which are respectively Multi-Workflow Analyzer, Adaptive Cloud Resource Allocator, Cloud Workflow Scheduler and Queue Buffer. The execution flow of these modules is as follows: (1) Multi-Workflow Analyzer receives the different execution requests of workflows (jobs) from users, analyzes the demand constraints and the task process relationships in the multiple workflows, and dispatches the multiple workflows to different task queue buffers. As the key module of performing our DFSCRL policy, Adaptive Cloud Resource Allocator (2) receives the depth-first-search task branch information of each workflow, and (3) conducts cloud resource coalition reinforcement learning and realizes the dynamically scaling of both horizontal and vertical cloud resources (i.e., different sizes of PMs and their respective type of VMs generated) for the multi-workflow scheduling. (4) An agent that serves as the allocator figures out an optimal VM instance bundle $\mathcal{B}_{c_i^*}$ from a best PM coalition $\mathcal{C}_i^*$ that will be provisioned to one or more workflows, and the VM instances come from the Virtual Machine Resource Pool layer. (5) Cloud Workflow Scheduler schedules the tasks in the Queue Buffer to the optimal VM instance bundle provisioned by the Adaptive Cloud Resource Allocator. (6) The task branches of each workflow are sequentially assigned to different types of VMs in an optimal VM Bundle for their execution.

The Virtual Machine Resource Pool layer contains some VM instance bundles $\mathcal{B}_{c_i}$ separately provided by multiple groups of PMs (named as PM coalitions) $\mathcal{C}_i$ with different sizes from the Physical Machine Resource Network layer in a cloud data center. Each VM instance bundle, created by different sizes of PMs in advance, is composed of some different types of VMs such as Small, Medium, Large and Xlarge VM instances which are respectively a combination of different sizes of CPU, Memory (RAM) and Storage (Disk) resources. This layer provides the VM repository for the cloud instance scaling initiated by the upper-layer allocator.

The Physical Machine Resource Network layer consists of a great number of physical machines which connect with each other through using some routing devices. In this PM resource network of a cloud data center, more than one local areas of PMs, called server farms, link with each other by their areas of routing devices. We assume that the routing devices are connected by two-way high speed fibre channels for guaranteeing the availability of physical machine resources. In this paper, we name a local area of PMs as an initial cloud PM coalition that can provide some initial VM instance bundles. Based on this idea, we reconstruct each PM coalition using the proposed DFSCRL learning algorithm so as to provision an optimal VM instance bundle for the execution of multiple workflows.

3.2 Problem Formulation

3.2.1 Multi-Workflow Model

The different types of user workloads are generally composed of multiple workflows (jobs) with some constraints, and each of workflows has many tasks with sequential dependencies. Multiple workflows can be represented as the set of multiple disjoint directed acyclic graphs $G = \{G^m \mid 1 \leq m \leq |G|\}$, and a workflow is expressed as a graph $G^m = (V^m, E^m)$, where $V^m = \{v_i^m \mid 1 \leq i \leq |V^m|\}$ means the set of tasks in the mth workflow, and $E^m = \{e^m_{ij} \mid v^m_i, v^m_j \in V^m, (i,j) \neq null, i \neq j\}$ denotes the set of data transmitting edges (or data flows) between two tasks with the sequential dependency. Let $Pre(i)$ be the set of predecessors of a task v^m_i and $Suc(i)$ be the set of successors of the task. Assume that when all of instances of predecessors $Pre(i)$ of a task v^m_i finish their execution, the task can be executed, and after the execution of all instances of the task are completed, its successors $Suc(i)$ can be performed.

The multiple workflows initiated by a user generally contain some constraints, each of which comprises a constraint tuple $CST^m = (Size^m, DDL^m, P^m_{cpu}, P^m_{mem}, P^m_{stg})$, where $Size^m = \sum_{i=1}^{|V^m|} inst^m_i mi^m_i$ denotes the size of mth workflow in which $inst^m_i$ and mi^m_i are the number of task instances and millions of instructions of each task v^m_i in the mth workflow, respectively, DDL^m is the deadline of the running time of mth workflow, and P^m_{cpu}, P^m_{mem} and P^m_{stg} are separately the planning quantity of cloud resources CPU, Memory (RAM) and Storage (Disk) for mth workflow.

3.2.2 Cloud Resource Allocation and Provisioning Model

Due to the uncertain arrival of multi-workflow execution requests from cloud users, the requirements on cloud resources constantly change. Thus, we construct an adaptive cloud resource allocation and provisioning model for the execution of cloud workflows while satisfying the requirement constraints of users. All of workflows initiated by a user need to be allocated some suitable cloud resources that are usually specified as some different types of VMs. To accelerate the execution of cloud workflows and reduce the using cost of VM resources, the system can derive an optimal VM instance bundle provided by a suitable PM coalition, and allocate the VM instance bundle to the workflows before running them.

Assume that there are n_c PM coalitions $\mathcal{C}_i$, $1 \leq i \leq n_c$, each of which can provide a VM bundle $\mathcal{B}_{c_i}$ that consists of n_v different types of VM instance combination $\mathcal{B}_{c_i} = (y^1_{c_i}, y^2_{c_i}, \ldots, y^k_{c_i}, \ldots, y^{n_v}_{c_i})$, where $y^k_{c_i}$ denotes the number of the kth type of VM instances (Small, Medium, Large or Xlarge type, etc.) in this VM bundle. In general, each type of VM instance has a fixed resource configuration represented by a vector $(r^k_{cpu}, r^k_{mem}, r^k_{stg})$, and $1 \leq k \leq n_v$. We let a workflow $G^m = (V^m, E^m)$ be scheduled to a VM bundle $\mathcal{B}_{c_i}$, and it means that each task v^m_i in the workflow must obtain one type of VM instances. Thus, we give Eq. (1) as the indicator function of the cloud VM resource allocation

$$I(v^m_i y^k_{c_i}) = \begin{cases} 1 & \text{if } k\text{th type of VM instances } y^k_{c_i} \\ & \text{are allocated to task } v^m_i \text{ in the} \\ & \text{workflow } G^m; \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The required amount of different types of VM instances, which are used to execute a workflow G^m on the tth decision-making time point, are represented as $\mathcal{R}_m(t) = (x^1_m(t), x^2_m(t), \ldots, x^k_m(t), \ldots, x^{n_v}_m(t))$, where $x^k_m(t)$ denotes the required number of the kth type of VM instances for the workflow G^m in a VM bundle provided. The decision parameter vector $\mathcal{R}_m(t)$ realizes the scaling of both horizontal and vertical cloud resources for the execution of a workflow G^m on the tth time point. Note that each task v^m_i in the workflow G^m will be allocated its actual VM instances from the required number $x^k_m(t)$ of the kth type of VM instances in the task scheduling phase (see Section 4.3). To ensure the sufficient cloud resources, the model needs to consider the constraints of the cloud resource usage amount and the execution time of workflows, respectively. Moreover, the indicator function in Eq. (1) needs to be added when VM instances are allocated, so $x^k_m(t)$ can be calculated by Eq. (2)

$$x^k_m(t) = \sum_{i=1}^{|V^m|} [s^{m,k}_i(t) \cdot I(v^m_i y^k_{c_i})], \quad (2)$$

where $s^{m,k}_i(t)$ denotes the number of the kth type of VM instances that are actually allocated to a task v^m_i of the workflow G^m on the tth time point. Assume that the required amount of CPU, Memory and Storage in each type of VM instances for a workflow G^m is expressed as a vector $(x^k_m(t)r^k_{cpu}, x^k_m(t)r^k_{mem}, x^k_m(t)r^k_{stg})$ where r^k_{cpu}, r^k_{mem} and r^k_{stg} are the fixed sizes of CPU, Memory and Storage in the kth type of VM instances advertised by a public cloud provider's website. The provisioned amount of CPU, Memory and Storage in a PM coalition $\mathcal{C}_i$ is represented as a three tuple $(R^{c_i}_{cpu}, R^{c_i}_{mem}, R^{c_i}_{stg})$. Thus, the model is subject to the three limitations before the cloud resources are allocated to tasks in a workflow such as $\sum_{k=1}^{n_v} x^k_m(t) r^k_{cpu} \leq R^{c_i}_{cpu} \leq P^m_{cpu}$, $\sum_{k=1}^{n_v} x^k_m(t) r^k_{mem} \leq R^{c_i}_{mem} \leq P^m_{mem}$, and $\sum_{k=1}^{n_v} x^k_m(t) r^k_{stg} \leq R^{c_i}_{stg} \leq P^m_{stg}$.

Response time. The whole running time of a workflow G^m is called its response time $respT^m$ that includes the execution time $execT^m$ running tasks and the data transmission time between dependent tasks of the workflow after being allocated VM instances. Because the PM network in a cloud data center generally reaches above gigabit bandwidth, the data transmission time between VM instances from the same PM can be negligible, and this time is also very little between different PMs that connect to the same or nearby routing devices in the same data center. Thus, the response time running a workflow mainly depends on the execution time on the path of dependent tasks, and the data transmission time on this path can be not considered in the same cloud data center. In addition, because the model provisions a bundle of different types of VM instances to tasks of a workflow, each task in the workflow is scheduled to a type of VM instance such that all of tasks run in parallel in the VM instance bundle. As a result, the execution time $execT^m$ depends on the maximum completion time of tasks in the VM instance bundle. Based on the above reasons, we use Eq. (3) to express the response time $respT^m$ or the execution time $execT^m$ of the workflow G^m

$$respT^m \approx execT^m = \max_{k=1}^{n_v} \left\{ \sum_{i=1}^{h_{n_v}} (inst^m_i \cdot execT^{m,k}_i) \right\}, \quad (3)$$

where n_v is the number of VM instance type, h_{n_v} is the number of tasks from the workflow G^m in the n_vth type of VM instances provisioned, and $inst^m_i$ denotes the number of task instances of the task v^m_i in the workflow G^m.

In light of the process of the resource provisioning and multi-workflow scheduling above, we suppose all of tasks in a workflow G^m have been scheduled to a provisioned VM instance bundle on the tth time point, and the tasks of next workflow G^{m+1} are left in the Queue Buffer and wait for being scheduled. When a task v_i^m of the workflow G^m on the tth time point is allocated to a VM instance, the task starts to execute. According to the definition of constraint element $Size^m$ of the workflow G^m in Section 3.2.1, the execution time $execT_i^m$ of the task v_i^m can be calculated by Eq. (4)

$$execT_i^{m,k} = \frac{mi_i^m}{mips^{m,k}}, \tag{4}$$

where mi_i^m denotes the number of millions of instructions owned by the task v_i^m, and $mips^{m,k}$ is the number of millions of instructions per second executed by the kth-type VM instance in the workflow G^m. From Eq. (4), we can derive that given m_k VM instances of the kth type for executing a task branch of the workflow G^m, when increasing n_k VM instances of this type, the quantitative relationship between the new execution time $execT_{br,new}^m$ of the task branch and the previous execution time $execT_{br,old}^m$ is $execT_{br,new}^m = \frac{m_k}{m_k+n_k} execT_{br,old}^m$. Conversely, when decreasing n_k VM instances of the kth type, the above relationship is $execT_{br,new}^m = \frac{m_k}{m_k-n_k} execT_{br,old}^m$.

Cloud resource utilization rate. Since a PM coalition $\mathcal{C}_i$ provides a bundle of VM instances to a workflow G^m in our model, the utilization rates of the CPU, Memory and Storage are represented as $U_{cpu}^{c_i}, U_{mem}^{c_i}$ and $U_{stg}^{c_i}$, which are the percentages of the three cloud resources in the VM instances from the PM coalition $\mathcal{C}_i$ used by the workflow G^m. We describe the integrated cloud resource utilization rate U^{c_i} of a PM coalition $\mathcal{C}_i$ by using Eq. (5)

$$\begin{aligned} U^{c_i} &= \alpha \cdot U_{cpu}^{c_i} + \beta \cdot U_{mem}^{c_i} + \gamma \cdot U_{stg}^{c_i} \\ &= \alpha \frac{\sum_{k=1}^{n_v} x_m^k(t) r_{cpu}^k}{R_{cpu}^{c_i}} + \beta \frac{\sum_{k=1}^{n_v} x_m^k(t) r_{mem}^k}{R_{mem}^{c_i}} \\ &\quad + \gamma \frac{\sum_{k=1}^{n_v} x_m^k(t) r_{stg}^k}{R_{stg}^{c_i}}, \end{aligned} \tag{5}$$

where α, β and γ are the weights of CPU, Memory and Storage in the U^{c_i} of the PM coalition $\mathcal{C}_i$, respectively, and $\alpha + \beta + \gamma = 1$. For compute-intensive workflow applications, we may set the values of α, β and γ as 0.4, 0.4 and 0.2, respectively. For data-intensive ones, the three corresponding weight values can be set as 0.2, 0.4 and 0.4.

3.2.3 *Reward Model for Cloud Resource Provisioning*

Due to the widespread use of pay-as-you-go billing mode on VM instances, for saving usage cost of cloud resources, the amount of VM instances provisioned to each workflow needs to be adjusted according to the using case of the previous workflows. In the process of cloud resource allocation and provisioning, the immediate reward will be obtained when the cloud system adopts a new action policy to adjust the amount of VM instances. To achieve the optimal provisioning scheme on a VM instance bundle, we construct the reward function for the resource reinforcement learning and the formation of cloud resource coalition in Section 4.

In light of the requirement constraint CST^m of a workflow G^m and Eqs. (1), (2), (3), (4), and (5), we define the immediate reward function $RW_{t+\tau^{c_i}}^{c_i}$ in the time interval $[t, t+\tau^{c_i}]$ by using Eq. (6) in which a PM coalition $\mathcal{C}_i$ will be formed before provisioning a VM instance bundle

$$\begin{cases} RW_{t+\tau^{c_i}}^{c_i} = \delta^{c_i} \cdot [\tau^{c_i} \sum_{k=1}^{n_v} pr_k x_m^k(t)] \\ \tau^{c_i} = respT^m \\ \delta^{c_i} = (1 - \frac{respT^m}{DDL^m}) \cdot U^{c_i} \end{cases} \tag{6}$$

where pr_k denotes the price of the kth type of VM instance, $x_m^k(t)$ represents the actual number of the kth type of VM instances in a VM bundle provisioned to the workflow G^m on the tth time point, τ^{c_i} is the actual running time of the workflow G^m of renting the VM instance bundle from a PM coalition $\mathcal{C}_i$, and δ^{c_i} denotes the discount rate of the reward. δ^{c_i} indicates the running cost of provisioned VM instances, which is influenced by two factors, the response time $respT^m$ of running the workflow G^m and the cloud resource utilization rate U^{c_i} of the PM coalition $\mathcal{C}_i$. Note that only if $respT^m < DDL^m$(i.e., meeting the deadline of running the mth workflow), $RW_{t+\tau^{c_i}}^{c_i} > 0$, otherwise, $RW_{t+\tau^{c_i}}^{c_i} = 0$. From Eq. (6), we can see that the less response time $respT^m$ and the higher cloud resource utilization rate U^{c_i}, the more immediate reward.

4 CRL-Based Cloud Resource Provisioning Mechanism and Algorithms

In this section, we propose the key cloud resource provisioning mechanism base on coalition reinforcement learning (CRL). The mechanism first gives the concept and formation method of cloud resource coalitions, then presents adaptive cloud resource provisioning mechanism using the CRL, and finally proposes multi-workflow task scheduling scheme to the best-matched VM bundle provided by an optimal PM coalition.

4.1 Cloud Resource Coalition Formation for the Best-Matched VM Bundle

The cloud resource coalition refers to a group of physical machines (PMs) from the physical machine resource network in a data center. Due to the limitation of a physical machine's computing capacity, we need to combine suitable number of machines to provide a variety of computing resources (e.g., a bundle of VM instances) for speeding up the execution of user tasks while maximizing the cloud resource utilization.

Suppose that an initial grand PM coalition $\mathcal{GPC} = \{p_1, p_2, \ldots, p_i, \ldots, p_N\}$ that consists of N different PMs. To search an optimal cloud resource set, we generally need to partition the $\mathcal{GPC}$ to some independent and disjoint subsets. Thus, a PM coalition structure is defined as $\mathcal{PC} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_i, \ldots, \mathcal{C}_{n_c}\}$ which is a set of n_c PM coalitions, where $\sum_{i=1}^{n_c} |\mathcal{C}_i| = N$, and for all $i \neq j \in [1, n_c], \mathcal{C}_i \cap \mathcal{C}_j = \varnothing$. In general, searching the optimal PM coalition structure is an exhaustive job through all PM coalition structures, and it is not very feasible for the cloud resource provisioning. To obtain an optimal PM coalition, we propose a PM coalition formation game based on the hedonic game [30] with the players' preference selection between different coalitions as follows.

PM Coalition Formation Game is defined as a tuple $(\mathcal{PC}, \succeq)$, where $\succeq_i$ is a reflexive, antisymmetric and transitive preference selection relation on the disjoint coalitions $\mathcal{C}_i$ and $\mathcal{C}_j$, and $\mathcal{C}_i, \mathcal{C}_j \in \mathcal{PC} \subseteq \mathcal{GPC}$. Specifically, $\succeq_{i+j}$ means that if the PM coalition $\mathcal{C}_i \cup \mathcal{C}_j$ can obtain higher reward, the PM coalition $\mathcal{C}_i$

prefers to merge with $\mathcal{C}_j$, i.e., $v(\mathcal{C}_i \cup \mathcal{C}_j) \geq v(\mathcal{C}_i)$ and $v(\mathcal{C}_i \cup \mathcal{C}_j) \geq v(\mathcal{C}_j)$, where $v(\cdot)$ denotes the PM coalition reward. Conversely, $\succeq_{i|j}$ indicates that if the reward of the merged PM coalition decreases, the PM coalition $\mathcal{C}_i \cup \mathcal{C}_j$ prefers to be split into two PM coalitions $\mathcal{C}_i$ and $\mathcal{C}_j$, i.e., $v(\mathcal{C}_i) \geq v(\mathcal{C}_i \cup \mathcal{C}_j)$ or $v(\mathcal{C}_j) \geq v(\mathcal{C}_i \cup \mathcal{C}_j)$.

In the process of the PM coalition formation game, the merging or splitting action depends on which action maximizes the immediate reward $RW^{c_i}_{t+\tau^{c_i}}$ of the merged or split PM coalition on the time t. Assume that the actions of the current coalition formation are taken during the execution time of the previous workflow prior to that of the current workflow. Thus, on the basis of Eq. (6), the best coalition action policy on the time t is formulated by Eq. (7)

$$a^*_t = \arg \max_{a \in \{\succeq_{i+j}, \succeq_{i|j}\}} RW^{c_i}_{t+\tau^{c_i}}(a), \tag{7}$$

where i in the set of the action selection denotes the coalition $\mathcal{C}_i$. When a^*_t is the merging action, j represents the coalition $\mathcal{C}_j$ that does not intersect $\mathcal{C}_i$. When a^*_t is the splitting action, j means a PM in the coalition $\mathcal{C}_i$, and the PM p_j is split out of $\mathcal{C}_i$, i.e., the PM p_j is removed from $\mathcal{C}_i$.

The Principles of the PM Coalition Formation. To simplify the complexity of the problem, we assume that a PM can generate multiple VMs of just one type (e.g., Small, Medium, Large or Xlarge type) that fit the size of the PM's resources, and when a PM is joined into the resource provisioning coalition, all of VMs generated by the PM must be allocated to the execution of multiple workflows, on the contrary, only if all of VMs from the PM are in the unoccupied idle states, the PM can be removed from the provisioning coalition. Furthermore, suppose that in the beginning phase before any action is taken, each of the PM coalitions is a single PM different with each other, e.g., $\mathcal{C}_k = \{p_k\}$. Each of PMs has a certain amount of CPU, Memory and Storage, and the respective current utilization rates of these resources. After performing the merging or splitting actions, the adjusted PM coalition will contain a group of different PMs that come preferentially from adjacent physical machines connected to the same router, and then from physical machines on adjacent routers. The reason for this setup is to take into account the cost of communication between machines for tasks to be performed. According to this order, the initial PM coalition structure $\mathcal{PC} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_i, \ldots, \mathcal{C}_{n_c}\}$ is formed where a PM coalition is a PM, i.e., $\mathcal{C}_i = \{p_i\}$. Through our coalition reinforcement learning, a maximum long-term reward $v(\mathcal{C}^*_i)$ will be gained as described in Section 4.2, and an optimal PM coalition $\mathcal{C}^*_i = \mathcal{C}_1 \cup \cdots \mathcal{C}_{i-1} \cup \mathcal{C}_i = \{p_1, p_2, \ldots, p_{i-1}, p_i\}$ will be also obtained to provide the best-matched VM bundle $\mathcal{B}_{c^*_i}$ for the execution of multiple workflows. The PM coalition reward $v(\mathcal{C}^*_j)$ is equal to $v(\mathcal{C}_1 \cup \cdots \mathcal{C}_i \cup \mathcal{C}_j)$ for some merging actions when these coalitions do not intersect each other, or the PM coalition reward $v(\mathcal{C}^*_{i'})$ is equal to $v(\mathcal{C}_i) - \cdots - v(p_j)$ for the splitting actions on some PMs when the split (removed) PMs are members of the PM coalition $\mathcal{C}_i$. As the changes of multiple workflows and their VM requirements can take place, the optimal PM coalition needs to be adjusted dynamically.

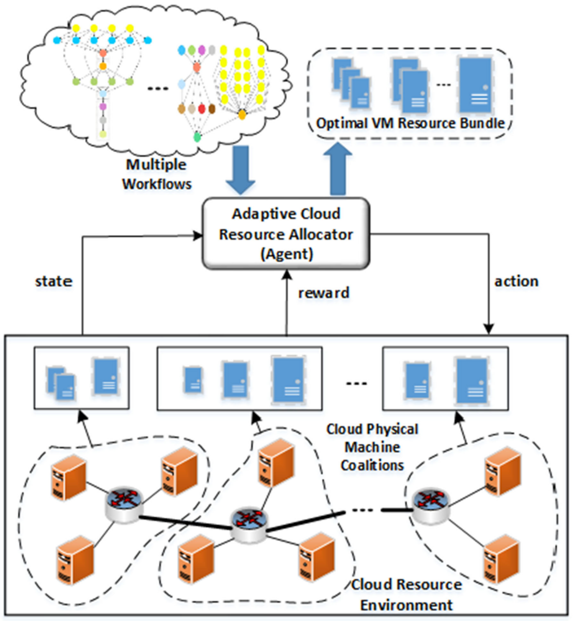
Optimal VM Resource Bundle
Multiple Workflows
Adaptive Cloud Resource Allocator (Agent)
state
reward
action
Cloud Physical Machine Coalitions
Cloud Resource Environment

Fig. 2. The coalition reinforcement learning (CRL) model.

4.2 Coalition Reinforcement Learning Model and Algorithm for the Adaptive Resource Provisioning

In this section, we propose a coalition reinforcement learning (CRL) model that adds the dynamic coalition formation into the reinforcement learning process for realizing the adaptive cloud resource provisioning. The whole CRL model is shown in Fig. 2.

The Adaptive Cloud Resource Allocator (Agent) learns the optimal VM resource provisioning scheme (the optimal VM instance bundle) for the execution of multiple workflows by using a coalition reinforcement learning algorithm. The Agent first receives the resource request from a cloud workflow (a Job of users) in a certain period of time, and this workflow has previously been decomposed into several different task queues by the Multi-Workflow Analyzer. Then, the Agent observes the states of the cloud resource environment that contains some physical machines. Next, the Agent takes an action to form an initial PM coalition for performing the tasks of the cloud workflow according to the states observed, and finally it gains an immediate expected reward from the environment. The learning process continues until the optimal PM coalition is formed, i.e., the optimal VM instance bundle is obtained, which contains some different types and number of best-matched VMs generated from the PM coalition. In the coalition reinforcement learning algorithm, we integrate the PM coalition formation from Section 4.1 into the Q-learning algorithm.

In the CRL algorithm, the action space is represented as $\mathcal{A} = \{\succeq^1_{i+j}, \succeq^1_{i|j}, add^1_{V_{ik}}, subtract^1_{V_{ik}}, \ldots, \succeq^t_{i+j}, \succeq^t_{i|j}, add^t_{V_{ik}}, subtract^t_{V_{ik}}, \ldots\}$, where $\succeq^t_{i+j}$ or $\succeq^t_{i|j}$ is the action of merging or splitting the PM coalition, $add^t_{V_{ik}}$ or $subtract^t_{V_{ik}}$ is the action of adding or subtracting a VM belonging to kth PM in the ith PM coalition at the time step t, and the action $a_t = \{\succeq^t_{i+j}, \succeq^t_{i|j}, add^t_{V_{ik}}, subtract^t_{V_{ik}}\}$. The state space is indicated as $\mathcal{S} = \{s_1, a_1, s_2, a_2, \ldots, s_t, a_t, \ldots\}$ where s_t denotes the state of cloud environment that involves the current workflow's task queues that will be executed, the available resources of the previous PM coalition $\mathcal{C}_i$, and the rest of PMs in the PM network. In order to reduce thread conflicts for different workflows' tasks in the VMs, we assume that the PM coalition for the execution of a new workflow's tasks will be adjusted (or formed) when all tasks of a workflow have been completed.

The goal of the learning Agent is to maximize the long-term reward $v(\mathcal{C}_i^*)$ through taking a sequence of actions of merging or splitting PMs that form the PM coalition $\mathcal{C}_i$ as well as actions of adding or subtracting the kth VM in the ith PM coalition at the time step t. After taking an action a_t at the current state s_t, the learning system will transfer to a new state s_{t+1} and then gain an immediate reward $RW^{c_i}_{t+\tau^{c_i}}$ in the time interval $[t, t+\tau^{c_i}]$ by using Eq. (7). Note that t is the learning time step, and τ^{c_i} is the time interval of renting the VM instance bundle provided by the PM coalition $\mathcal{C}_i$ starting from the time point t. We use Q-learning process to realize the proposed CRL algorithm which considers the required actions to form an optimal PM coalition when choosing a random action or an optimal action maximizing Q-value at each learning time step. According to Eq. (7) and the PM coalition formation game in Section 4.1, we get the judgment formula Eq. (8) to take the best coalition forming action policy at the time step t

$$a_t^* = \begin{cases} \{\succeq^t_{i+j}\} & \text{if } RW^{c_i \cup c_j}_{t+\tau^{c_i \cup c_j}} \geq RW^{c_i}_{t+\tau^{c_i}} \\ \{\succeq^t_{i|j}\} & \text{if } RW^{c_i - p_j}_{t+\tau^{c_i - p_j}} \geq RW^{c_i}_{t+\tau^{c_i}} \end{cases} \tag{8}$$

At each time step of learning episodes, the coalition forming actions taken at all of time steps in Eq. (8) determine how many different types of PMs are formed in the optimal coalition, and a PM has only one type of multiple VMs as described in Section 4.1. The coalition forming actions realize the adaptive horizontal cloud resource scaling up or down by adding (i.e., $\succeq^t_{i+j}$) or subtracting (i.e., $\succeq^t_{i|j}$) a certain number of PMs according to their current states.

Specifically speaking, the learning Agent updates the Q-value table by employing the Bellman equation as Eq. (9) in which each Q-value score is just the maximum expected future reward $maxQ_{t+1}(s_{t+1}, a_{t+1})$ that the learning Agent obtains when it takes the best action policy at the state s_{t+1}. During the whole updating process of the Q-value table, all of maximum expected future rewards are obtained through adding (i.e., $\succeq^t_{i+j}$) or subtracting (i.e., $\succeq^t_{i|j}$) a certain number of PMs at their current states, and the two actions achieve the adaptive horizontal cloud resource scaling up or down

$$\begin{aligned} &Q_t(s_t, a_t) \leftarrow Q_t(s_t, a_t) \\ &+ \alpha_r(RW^{c_i^*}_{t+\tau^{c_i^*}}(s_t, a_t) + \gamma_r \max Q_{t+1}(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)), \end{aligned} \tag{9}$$

where $Q_t(s_t, a_t)$ is the current Q-value at the current state s_t and action a_t, $RW^{c_i^*}_{t+\tau^{c_i^*}}(s_t, a_t) = \delta^{c_i^*}\tau^{c_i^*}(\sum_{k=1}^{n_{|c_i^*|}} pr_k x^k_m(t))$ denotes the immediate reward after taking the action $\succeq^t_{i+j}$ or $\succeq^t_{i|j}$ that forms the PM coalition $\mathcal{C}_i^*$ provisioned to a workflow G^m of users at the current state s_t, $\max Q_{t+1}(s_{t+1}, a_{t+1})$ means the maximum expected Q-value given all of new possible actions a_{t+1} at the new state s_{t+1}, α_r is the learning rate and γ_r is the discount rate. After updating the Q-value table at the time step t, the learning system takes the action that adds (i.e., $add^t_{V_{ik}}$) or subtracts (i.e., $subtract^t_{V_{ik}}$) a VM of each PM p_k in the PM coalition $\mathcal{C}_i^*$ provisioned to each depth-first task branch of the workflow G^m at the current state s_t, and the two actions achieve the adaptive vertical cloud resource scaling up or down.

Algorithm 1. Depth-First-Search Coalition Reinforcement Learning Algorithm

```
Input: The current cloud resource environment and
       multi-workflow task queues initiated by users
Output: The optimal cloud resource provisioning action set
        $\mathcal{A}^*$, the optimal PM coalition $\mathcal{C}_i^*$ and its VM instance
        bundle $\mathcal{B}_{c_i^*}$
1  Initialize $Q(s, a)$;
2  $\mathcal{A}^* \leftarrow null, \mathcal{C}_i^* \leftarrow null, \mathcal{B}_{c_i^*} \leftarrow null$;
3  foreach $episode = 1, 2, \ldots, E$ do
4    $L \leftarrow$ the number of the depth-first-search task branches of
     a workflow $G^m$ to be run;
5    Initialize state space $\mathcal{S}$;
6    Initialize a PM coalition $\mathcal{C}_1 \leftarrow \{p_1, p_2, \ldots, p_L\}$, $\mathcal{C}_i^* \leftarrow \mathcal{C}_1$, and
     each of PMs in the coalition generates a VM instance;
7    foreach $t = 1, 2, \ldots, T$, and $i, j \in [1, L], i \neq j$ do
8      Choose a random action $a_t = \succeq^t_{i+j}$ or $\succeq^t_{i|j}$ with probability
       $\epsilon$, otherwise $a_t = argmax_a Q(s_t, a)$;
9      if $a_t = \succeq^t_{i+j}, \mathcal{C}_j = \{p_j\}, RW^{c_i \cup c_j}_{t+\tau^{c_i \cup c_j}} \geq RW^{c_i}_{t+\tau^{c_i}}$ then
10       $\mathcal{C}_i^* \leftarrow \mathcal{C}_i \cup \mathcal{C}_j$, take the action $a_t = \succeq^t_{i+j}$ ,
         $\mathcal{A}^* \leftarrow \mathcal{A}^* \cup \{\succeq^t_{i+j}\}, k \leftarrow j$;
11       Take $n_k$ actions $a_t = add^t_{V_{ik}}$ on the changed $p_k$ for $n_k$
         tasks of the $k$th depth-first-search task branch matching
         with the sizes of the tasks;
12       $\mathcal{A}^* \leftarrow \mathcal{A}^* \cup n_k a_t, \mathcal{B}_{c_i^*} \leftarrow \mathcal{B}_{c_i^*} \cup n_k VM_{ik}$;
13     end
14     if $a_t = \succeq^t_{i|j}, p_j \in \mathcal{C}_i, RW^{c_i - p_j}_{t+\tau^{c_i - p_j}} \geq RW^{c_i}_{t+\tau^{c_i}}$ then
15       $\mathcal{C}_i^* \leftarrow \mathcal{C}_i - p_j$, take the action $a_t = \succeq^t_{i|j}$ , $\mathcal{A}^* \leftarrow \mathcal{A}^* \cup \{\succeq^t_{i|j}\}$,
         $k \leftarrow j$;
16       Take $n_k$ actions $a_t = subtract^t_{V_{ik}}$ one by one;
17       $\mathcal{A}^* \leftarrow \mathcal{A}^* \cup n_k a_t, \mathcal{B}_{c_i^*} \leftarrow \mathcal{B}_{c_i^*} - n_k VM_{ik}$;
18     end
19     Observe the next state $s_{t+1}$ and reward
       $RW^{c_i^*}_{t+\tau^{c_i^*}}(s_t, a_t) = \delta^{c_i^*}\tau^{c_i^*}(\sum_{k=1}^{n_{|c_i^*|}} pr_k x^k_m(t))$;
20     $Q_t \leftarrow Q_t + \alpha_r(RW^{c_i^*}_{t+\tau^{c_i^*}} + \gamma_r \max Q_{t+1} - Q_t)$;
21     $s_t \leftarrow s_{t+1}$;
22     if $|\mathcal{C}_i^*| >$ L or beyond the limitation of cloud resources then
23       break;
24     end
25   end
26 end
27 return $\mathcal{A}^*, \mathcal{C}_i^*, \mathcal{B}_{c_i^*}$;
```

In light of the CRL model, we devise the corresponding CRL algorithm for the adaptive cloud resource provisioning as shown in Algorithm 1. Lines 1 and 2 initialize the Q-value table and the optimal result sets $\mathcal{A}^*$, $\mathcal{C}_i^*$ and $\mathcal{B}_{c_i^*}$. Line 3 gives the rounds of the Q-training in the learning system. Next, Line 4 obtains the number of the depth-first-search task branches of a workflow G^m to be run. Lines 5 and 6 initialize the state space and a PM coalition. The following statements from Lines 7 to 25 are the key parts of the proposed adaptive cloud resource provisioning procedures using the CRL method. Line 8 selects an action through a random or maximal Q-value way. In Lines 9 to 10, the algorithm realizes the PM coalition merging (i.e., the horizontal cloud PM scaling up) for different task branches of the workflow G^m through employing the PM coalition formation principles in Section 4.1 and Eq. (8). Lines 11 to 12 implement the vertical cloud VM scaling up on each PM for each depth-first task branch of the G^m, and update the $\mathcal{A}^*$ and $\mathcal{B}_{c_i^*}$. On the contrary, Lines 14 to 15 perform the PM coalition splitting (i.e., the horizontal cloud PM scaling down), and Lines 16 to 17 conduct the vertical cloud VM scaling down. After the formation of the PM coalition and the generation of corresponding VMs, Lines 19 to 21 update the Q-value table and current state by Eq. (9). Lines 22 to 24 give the ending condition of T steps in an episode of Q-training. Finally, the algorithm outputs the optimal result set $\mathcal{A}^*$, $\mathcal{C}_i^*$ and $\mathcal{B}_{c_i^*}$.

The time complexity of Algorithm 1 is $O(|\mathcal{S}| \cdot |\mathcal{A}|) = O(E\sum_{k=1}^{L} n_k)$, where $|\mathcal{S}| = E$ denotes the maximum number of states (i.e., the training episodes), $|\mathcal{A}| = \sum_{k=1}^{L} n_k$ is the maximum number of actions taken (i.e., merging or splitting the PM coalition, and adding or subtracting a VM). The space complexity is $O(\sum_{k=1}^{L} n_k)$ for storing the actions that form the PM coalition.

To illustrate the validity of the CRL algorithm, we present the first theorem and its proof as follows.

Theorem 1. *The proposed depth-first-search coalition reinforcement learning (DFSCRL) algorithm can converge to the optimal cloud resource provisioning scheme (i.e., the optimal PM coalition $\mathcal{C}_i^*$ and VM bundle $\mathcal{B}_{c_i^*}$) for the execution of multiple workflows through learning the current cloud resource environment.*

Proof. The entire proof of the theorem is given in the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TC.2022.3191733. □

4.3 Multi-Workflow Scheduling Scheme to the Optimal VM Bundle and Its Algorithm

The Multi-Workflow Analyzer shown in Fig. 1 decomposes different workflows into many task sets by the way of the breadth-first-search (BFS) graph traversal, and dispatches them into different task queues according to the order of BFS. Based on the maximum task queue length among all the task queues of a workflow, the DFSCRL algorithm in Section 4.2 first calculates the matched PM coalition $\mathcal{C}_i^*$, then generates the different types and number of suitable VMs in PMs of the coalition, and finally obtains the optimal VM instance bundle for the execution of a workflow. After obtaining the VM provisioning scheme, all of the workflow tasks in the queues will form some task branch groups using the depth-first-search (DFS) graph traversal way, and the Cloud Workflow Scheduler then schedules the tasks of the same workflow into the matched PMs of the PM coalition $\mathcal{C}_i^*$ through dequeuing one by one from the head of each task queue.

To show the above process more clearly, we give a case that tasks of a workflow are scheduled to the optimal VM instance bundle as depicted in Fig. 3. From this figure, we can see that all the tasks of a workflow in the left side are dispatched into five task queues in the breadth-first traversal mode, and they are subsequently fetched by the way of depth-first traversal. These depth-first task branches are respectively scheduled into the PMs p_1 and p_2 of the optimal PM coalition $\mathcal{C}_i^*$ with the principle of a maximum task branch to a maximum size of PM, where a type of PM is virtualized into a group of same-type VM instances. After that, another new workflow's tasks move forward to the queue spaces occupied by the previous workflow's tasks. Before the tasks are scheduled, the optimal PM coalition $\mathcal{C}_i^*$ has been generated by the DFSCRL algorithm, and the PMs p_1 and p_2 in the coalition have been virtualized into n_1 VM_{i1} and n_2 VM_{i2}, respectively. Finally, all the above tasks are divided into two groups of the most relevant tasks, and executed in parallel. Through the above-mentioned processes, the proposed method can maximize the VM resource utilization, reduce the task execution time, and save the system operation cost.

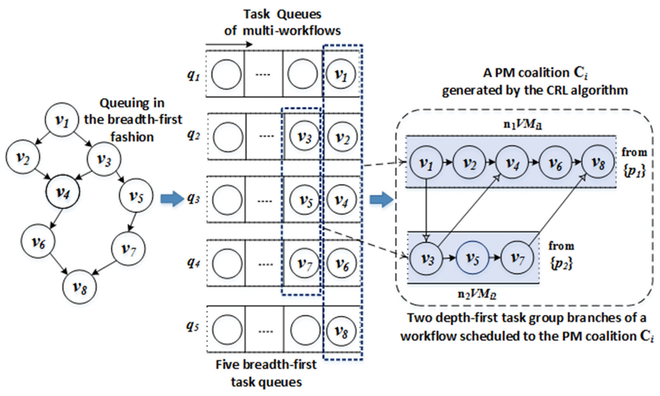

Fig. 3. A case that tasks of a workflow are scheduled to the optimal VM instance bundle $\mathcal{B}_{c_i^*} = \{n_1 VM_{i1}, n_2 VM_{i2}\}$.

According to the task scheduling scheme, we present Algorithm 2, a scheduling and executing algorithm of the workflow tasks allocated into the optimal VM instance bundle. Because of the limitation of space, Algorithm 2 and its explanation are given in the supplementary material, available online.

To further demonstrate the effectiveness of the DFSCRL method for the multi-workflow task scheduling, we give the second theorem and its proof as follows.

Theorem 2. *Using the optimal VM instance bundle $\mathcal{B}_{c_i^*}$ to execute multi-workflow tasks can achieve less execution time and higher cloud resource utilization rate of the optimal PM coalition $\mathcal{C}_i^*$. This enables the system to obtain more total rewards in the proposed algorithms.*

Proof. The entire proof of the theorem is given in the supplementary material, available online. □

5 Performance Evaluation

In this section, we conducted eleven groups of simulation experiments based on the real-world scientific workflow application dataset to evaluate our algorithms. Six groups of these experiments run in the cloud workflow simulator - WorkflowSim,[1] and the other five groups used the real container cloud - Kubernetes.[2] The Kubernetes is deployed on a local physical server with the configuration of six cores of Intel Xeon CPU and 16 GB RAM, and runs in a Docker

1. https://github.com/WorkflowSim/WorkflowSim-1.0
2. https://kubernetes.io/releases/download/

TABLE 1
The Experimental Dataset of Montage and SIPHT From Pegasus Workflow Applications

Workflow Type	Workflow Name	Number of Tasks	Average All Tasks' Total Runtime (MS)
Small	Montage	25, 50	2733, 6054
	SIPHT	30, 60	71894, 169051
Medium	Montage	100	12748
	SIPHT	100	243283
Large	Montage	1000	133898
	SIPHT	1000	2336151

container connected to an external Aliyuncs storage mirror. Through comparing with the existing six relevant methods about the cloud resource provisioning and multi-workflow scheduling, our solution shows the better performance.

5.1 Experimental Settings

5.1.1 Experimental Environment and Dataset

Our simulation experiments used and extended the WorkflowSim [33] which is implemented by Java, and based on the discrete event cloud simulator - CloudSim [34]. We have added some new Java classes which mainly involve the proposed DFSCRL and scheduling algorithms, the PM coalition, and the compared methods' core algorithms. The experimental dataset adopts the Pegasus Workflows[3] which include many different fields of real-world scientific workflow applications generated from different DAX (Directed Acyclic Graph of XML format) files. Each DAX file contains a large number of tasks described in XML format which have dependencies on each other with different average runtime time and input/output data sizes. Because the complexity of various scientific workflows is generally large and their calculation processes are similar, we randomly selected two types of workflows in the Pegasus Workflows, Montage and SIPHT [35]. The former is used to generate custom spatial mosaics of the sky, and calculates the geometry of the output. The latter, from the bioinformatics project at Harvard University, is employed to automatically search the sRNA encoding-genes for all of the bacterial replicons in the National Center for Biotechnology Information (NCBI) database. Each of these workflows consists of three different sizes of tasks, such as Small, Medium and Large types, which are respectively shown in Table 1.

5.1.2 The Brief Description of the Compared Methods

In this subsection, we briefly describe all of the compared methods as follows.

DFSCRL, namely the proposed policy, is the adaptive cloud bundle provisioning method for multi-workflow scheduling.

RLC [16] adopts a reinforcement learning-based controller to schedule the computational workflow for multi-tenant cloud computing. This method only uses the single type of VM provisioning for the workflow scheduling, while our approach employs the multi-type VM instance bundle provisioning through the PM coalition for multi-workflow scheduling.

ROSA [9] is an uncertainty-aware online scheduling algorithm that is designed to schedule dynamic and multiple workflows with deadlines.

CWSA [10] denotes a cloud-based workflow scheduling algorithm that makes use of the gaps between scheduled workflow tasks for scheduling other tasks to reduce the overall makespan of workflows.

MCT [36] is a minimum completion time scheduling policy in which each task of workflows is assigned in an arbitrary order to the cloud resource such that the task has the minimum expected completion time.

HEFT [22] refers to a heterogeneous earliest finish time policy that belongs to a heuristic algorithm.

FCFS [37] denotes a first-come-first-serve algorithm in which the first arrival workflow task is first fulfilled otherwise the task waits until the resources become idle.

5.1.3 Performance Evaluation Metrics and Experimental Parameters

In terms of the performance evaluation metrics, we assume that all of PMs are deployed in a cloud data center. As mentioned in Eq. (3), due to the gigabit bandwidth of the PM network within the same data center, the data transmission time between VM instances can be negligible. Thus, we focus on the analysis of execution time and cost of multiple workflows in the given scenario, and utilize six evaluation metrics as follows.

VM Instance Number Provisioned. Before executing multiple workflows, the DFSCRL algorithm in the proposed policy can provision the optimal VM instance bundle which contains a group of different types of VM instances from the formed PM coalition, and each type of VMs has a certain number. The other compared methods, however, provide only a single type of number of VM instances when the workflows are scheduled.

Execution Time. It denotes the total execution time of all tasks forming multiple workflows after they are scheduled to execute.

Makespan. This metric is used to calculate the maximum completion time of multiple workflows. On the basis of Eq. (3), the makespan of any executed multiple workflows G can be expressed as follows:

$$Makespan_G = \max_{k=1}^{n_b} \left\{ \sum_{m=1}^{|G|} \sum_{i=1}^{h_{n_b}} (inst_i^m \cdot execT_i^{m,k}) \right\}, \tag{10}$$

where n_b is the total number of depth-first-search task branches in the multiple workflows G, h_{n_b} is the number of tasks in the n_bth depth-first-search task branch of a workflow G^m, and $inst_i^m$ denotes the number of task instances of the task v_i^m in the workflow G^m.

Makespan Standard Deviation (MSD). It is a metric that measures the sensitivity and stability of different scheduling algorithms. This metric can be represented as follows:

$$MSD = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(X_i - \overline{X})^2}, \tag{11}$$

3. https://confluence.pegasus.isi.edu/display/pegasus/WorkflowHub

TABLE 2
On-Demand Instance Types and Prices From Amazon EC2, US East (Ohio)

Instance Type	CPU Core	Memory (GB)	Storage (GB)	Processing Speed (MIPS)	Price ($/H)
t2.small	1	2	16	1000	0.023
t2.medium	2	4	32	2000	0.0464
t2.large	2	8	64	2000	0.0928
t2.xlarge	4	16	128	4000	0.1856

where X_i, N and $\overline{X}$ refer to the ith sample of the makespan, the total number of samples, and the average value of the makespan, respectively. The smaller MSD value is, the more stable the scheduling algorithm is.

Resource Execution Efficiency (REE). This metric defines the overall efficiency that the used VM instances perform the tasks of multiple workflows G in different scheduling algorithms. We can express it as follows:

$$REE = 1 - \frac{realET_G}{\min(predictedET_G, requiredET_G)}, \quad (12)$$

where $realET_G$, $predictedET_G$, and $requiredET_G$ denote multiple workflows G's real execution time, predicted (average) execution time, and required execution time (deadline), respectively.

Total Renting Cost (TRC). It measures the total renting cost of different types of VM instances used for performing multiple workflows, which is calculated as follows:

$$TRC = \sum_{k=1}^{n_v} \sum_{m=1}^{|G|} x_m^k \cdot Price^k \cdot \sum_{i=1}^{h_{n_v}} (inst_i^m \cdot execT_i^{m,k}), \quad (13)$$

where n_v is the number of VM instance type, h_{n_v} means the number of tasks from the workflow G^m in the n_vth type of VM instances provisioned, $inst_i^m$ is the number of task instances of the task v_i^m in the workflow G^m, x_m^k denotes the number of the kth type of VM instances used for the workflow G^m, and $Price^k$ is the advertised price of the kth type of VM instance. When calculating the renting cost of VM instances, we adopted the hourly billing way, and less than one hour is counted as one hour.

Our experimental simulation platform is assumed to be deployed in a real data center that contains the selected four types of the Amazon EC2 On-Demand Pricing Instances [38] as listed in Table 2.

On the basis of the different types of resource instances in Table 2, we further set the running parameters of the proposed DFSCRL algorithm as shown in Table 3. Since each VM running a task, in the used simulator-WorkflowSim, is marked with the occupied status (i.e., *VM_STATUS_BUSY*) while VMs of unassigned tasks take the unoccupied status (i.e., *VM_STATUS_IDLE*), we assume that the resources of each VM assigned to a task are fully utilized. Thus, the cloud resource utilization rate U^{c_i} of a PM coalition $\mathcal{C}_i$ is set as 1 in the simulation experiments. In addition, all of the experiments on the seven compared algorithms are implemented 10 rounds at random, and the final experimental results take the approximate mean value on the 10 rounds of results when not violating the parameter constraints in Table 3.

TABLE 3
Experimental Parameters for the Proposed DFSCRL Algorithm

Parameter	Value or Value Range
ϵ (greedy factor)	0.8
α_r (learning rate)	0.2
γ_r (discount rate)	0.8
$MAX-EPISODES$ (training number)	100
$Size^G$ (Million Instructions, MI)	1×10^6
DDL^G (Millisecond, MS)	$1 \times 10^4 - 5 \times 10^5$
P^G_{cpu}	100
P^G_{mem} (GB)	200
P^G_{stg} (GB)	5000

5.2 Experimental Results

5.2.1 The Experiments Only Running in the Cloud Workflow Simulator

In this section, six groups of experiments with the different metrics are implemented in the cloud workflow simulator - WorkflowSim, and compared with the six existing methods mentioned in Section 5.1.2.

(1) VM Instance Number Provisioned

In this part, we conducted the experimental comparisons on the cloud VM instance provisioning quantity. The proposed DFSCRL provisioning policy calculated out the optimal VM instance bundle that consists of different types and number of VM instances from the optimal PM coalition generated. From Fig. 4, we can see that when inputting the Montage workflow dataset from Small size (25) to Large one (1000) of tasks, three groups of different types of VM instances, corresponding to three PMs (PM1, PM2 and PM3) in the optimal PM coalition, have been formed. The number of VM instances of each type increases with the growth of task sizes. From another dimension of this cubic graph, we can also observe that the number of VM instances goes up from the first types of VM instances in the PM1 to the third ones in the PM3. Notice that, in this paper, the first type of VMs in PM1, the second ones in PM2, and the third ones in PM3 correspond to t2.xlarge, t2.large and t2. medium Instance types, respectively. The t2.small Instance type was not chosen because it did not match with any current task sizes in the experiments.

We also did experiments of the other six compared algorithms through using the same dataset and required parameters. We compared the total number of VM instances provisioned. Fig. 5 depicts that the total number of VM instances in these methods are almost the same in the cases of 25 to 100 task sizes of Montage workflow application, while that of VM instances in the proposed DFSCRL policy is slightly higher than other methods in the case of the Large size (1000 tasks) of the Montage workflow. This is because that the large size of workflows are parsed into more depth-first-search (DFS) task branches such that many different types of VM instances are generated when the DFSCRL algorithm calculates the optimal PM coalition and corresponding VM bundle. We can also see that the VM instance number of the RLC method is most when there are 1000 tasks, since it only uses the single type of VM provisioning and scheduling, thus this policy must rent more VM instances to achieve the effect of multi-type VM provisioning and scheduling capabilities.

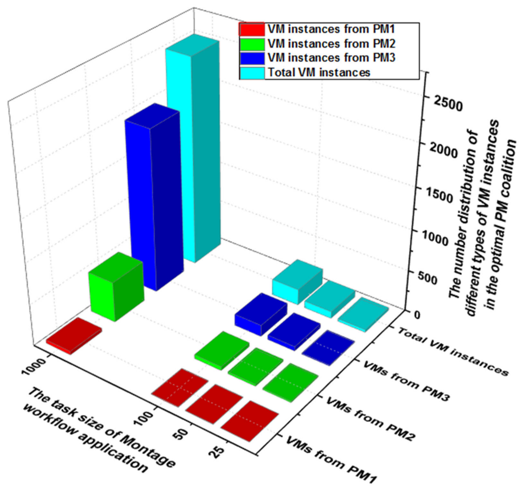

Fig. 4. The different types of cloud VM instances provisioned by the proposed DFSCRL policy.

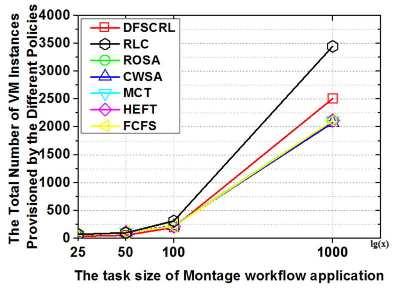

Fig. 5. The total number of cloud VM instances generated by the different cloud provisioning and scheduling policies.

(2) Execution Time

Total execution time of all tasks of multiple workflows measures the whole performance of running the workflow scheduling algorithms. To verify the scheduling performance of the different algorithms on different types and sizes of workflows, we carried out two groups of experiments by using single type of Montage and two types of Montage and SIPHT multi-workflow applications.

Fig. 6 shows that the execution time of all of policies on the Montage workflow gradually increases with the growth of task sizes. This is a reasonable rising trend as a whole. In the cases of Small (25 and 50) and Medium (100) task sizes, the execution time is close and lower than 2500 MS. Whereas, the time quantity becomes larger when there is Large (1000) task size. The execution time of FCFS algorithm arrives at 30000 MS, that of other algorithms are all less than 30000 MS, and this value of the proposed DFSCRL algorithm is always least in most cases. This is due to the fact that our method computes a group of different types and quantity of VMs, i.e., the optimal VM instance bundle matching with the sizes of tasks, to concurrently perform the different DFS task branches of workflows, such that the execution time is greatly reduced. However, the other six methods only use one type of VMs or mainly support the single-workflow scheduling by the static or dynamic selection to execute all tasks in sequence, thus, it will cause these algorithms to consume more time. Interestingly, the RLC strategy consumes the least amount of execution time, this may be due to the better performance of scheduling by

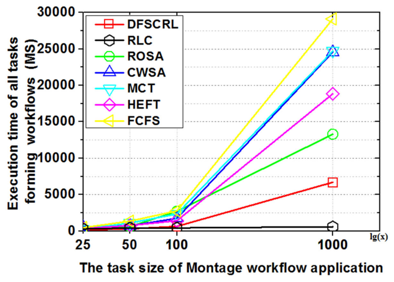

Fig. 6. Execution time of all tasks forming single type of workflows when performing the different scheduling algorithms in the cases of Small (25, 50), Medium (100) and Large (1000) task sizes of Montage workflow applications.

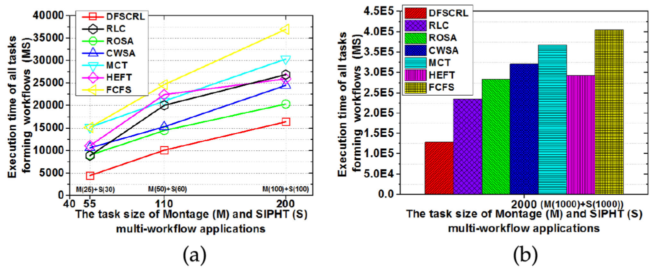

Fig. 7. Execution time of all tasks forming multiple types of workflows when performing the different scheduling algorithms in the cases of Small (55, 110), Medium (200) and Large (2000) task sizes of Montage and SIPHT multi-workflow applications.

reinforcement learning in the scenario of combing the single workflow and maximum number of VM instances.

There is the similar situation in Figs. 7a and 7b. The difference is that using two types of Montage and SIPHT workflows makes the scale and complexity of the calculation bigger in Fig. 7. Therefore, the execution time of all compared algorithms becomes longer correspondingly, i.e., the time is from rough 5000 MS (55 tasks) to 37500 MS (200 tasks), and it is from rough 1.25×10^5 MS to 4×10^5 (2000 tasks). However, our DFSCRL algorithm still takes the least time in all cases. This also indicates that our algorithm can adapt well to the input workloads of multiple complex types of workflows.

(3) Makespan

This experiment compares the makespan of executing multiple workflows by using Eq. (10), and still adopts the single type of Montage and two types of Montage and SIPHT multi-workflow applications as the input dataset. Fig. 8 reveals that with an increase of task sizes from the single type of workflows, the makespan obtained by the proposed DFSCRL policy is lowest in all of ones, i.e., our algorithm achieves best key performance improvement. The makespan of the RLC and ROSA policies is the closest to that of ours, and the FCFS algorithm has the worst result. The reason for the results is similar to the analysis of the execution time in the previous subsection.

In the scenario of multi-type input workflows as shown in Fig. 9, even though the task sizes have doubled in comparison with the single type of workflows, the total trends of makespan in all of policies are increasing with the growth of task sizes. However, we can also see that there is a respective lowest point of the makespan in the six polices except the DFSCRL under the Medium (200) task sizes from two types of workflows. This is because that all the compared six algorithms select one type of VM instances or mainly favor the

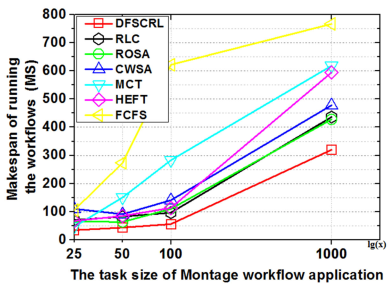

Fig. 8. Makespan of running single type of workflows in the cases of Small (25, 50), Medium (100) and Large (1000) task sizes of Montage workflow applications.

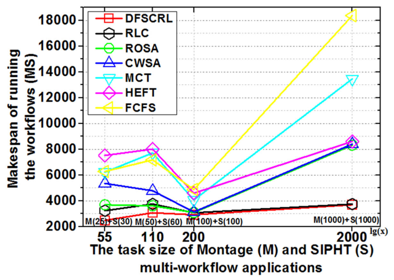

Fig. 9. Makespan of running multiple types of workflows in the cases of Small (55, 110), Medium (200) and Large (2000) task sizes of Montage and SIPHT multi-workflow applications.

single-workflow scheduling such that the task scheduling and execution in the complex multi-type workflow scenario cannot be handled very stably. The proposed DFSCRL policy has always a stable minimum makespan value due to the concurrent scheduling mode of task branches allocated the respective suitable types and number of VM instances. Thereinto, the result of the RLC policy is very close to ours in the case of Large (2000) task sizes of the multi-workflow applications, this is because both approaches use the reinforcement learning method to adaptively provision VM instances for the workflow scheduling, but as mentioned in the previous sections, the RLC policy organizes VM instances for the single type of VM provisioning, our DFSCRL policy can calculate an optimal PM coalition to form VM instance bundles for a more efficient way to organize cloud resources.

(4) Makespan Standard Deviation

Since the limitation of space, we describe this section in the supplementary material, available online.

(5) Resource Execution Efficiency

Due to the limited space, this section is depicted in the supplementary material, available online.

(6) Total Renting Cost

As the limitation of space, this section is also expounded in the supplementary material, available online.

5.2.2 The Experiments Using the Real Container Cloud

In order to carry out the simulation experiments in the real cloud environment, we adopted the popular container cloud - Kubernetes as the cloud resource provisioning pool for the workflow scheduling. The smallest unit of creation, scheduling and management in Kubernetes is called Pod, and a Pod can contain one or more containers. In the Kubernetes environment, different types of Pods resembling VM instances can be generated to perform multi-workflow scheduling. Since the results of resource provisioning for the scheduling of single workflow are similar to those in the case of the cloud simulator alone, in this section, we only conducted five groups of comparative experiments in the same multi-workflow scenarios, and our DFSCRL policy shows better overall performance when using real Pod scaling instances. Because of the limitation of space, the entire experiments of this section are given in the supplementary material, available online.

6 Conclusions and Future Work

The dynamic provisioning of multi-type resources is a challenge for the execution of multiple workflows, and there is not much existing work in this aspect. Toward this end, in this paper, we put forward an adaptive cloud bundle provisioning and multi-workflow scheduling model to dynamically perform both the horizontal and vertical cloud resource scaling for the execution of multiple workflows. The DFSCRL provisioning policy is proposed to realize the resource scaling according to the input workflows, which generates an optimal multi-type VM instance bundle from the PM coalition formed in advance. The VM instance bundle is afterwards provisioned to the concurrent execution of multiple workflows. Theoretical proofs and various experiments show that our policy and algorithms have better performance than the existing related methods. The future work will consider the time cost of data transmission between complex workflow tasks among multiple data centers, as well as the addition of distributed agent reinforcement learning to improve the efficiency of cloud resource provisioning and to further reduce the workflow scheduling cost.

References

[1] R. Buyya, C. S. Yeo, and S. Venugopal, “Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities,” in *Proc. 20th Int. Conf. High Perform. Comput. Commun.*, 2008, pp. 5–13.

[2] H. M. Fard, R. Prodan, and T. Fahringer, “A truthful dynamic workflow scheduling mechanism for commercial multicloud environments,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1203–1212, Jun. 2013.

[3] L. Wu, S. K. Garg, S. Versteeg, and R. Buyya, “SLA-based resource provisioning for hosted software-as-a-service applications in cloud computing environments,” *IEEE Trans. Serv. Comput.*, vol. 7, no. 3, pp. 465–485, Jul.–Sep. 2014.

[4] Z. Zhu, G. Zhang, M. Li, and X. Liu, “Evolutionary multi-objective workflow scheduling in cloud,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1344–1357, May 2016.

[5] T. Sun, C. Xiao, and X. Xu, “A scheduling algorithm using sub-deadline for workflow applications under budget and deadline constrained,” *Cluster Comput.*, vol. 22, no. 1, pp. 5987–5996, Jan. 2018.

[6] K. Li, X. Tang, B. Veeravalli, and K. Li, “Scheduling precedence constrained stochastic tasks on heterogeneous cluster systems,” *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 191–204, Jan. 2015.

[7] H. Chen, X. Zhu, H. Guo, J. Zhu, X. Qin, and J. Wu, “Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment,” *J. Syst. Softw.*, vol. 99, pp. 20–35, Jan. 2015.

[8] Z. Cai, X. Li, R. Ruiz, and Q. Li, “A delay-based dynamic scheduling algorithm for bag-of-task workflows with stochastic task execution times in clouds,” *Future Gen. Comput. Syst.*, vol. 71, pp. 57–72, Jun. 2017.

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

[10] 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.

[11] D. Cui, Z. Peng, X. Jianbin, B. Xu, and W. Lin, "A reinforcement learning-based mixed job scheduler scheme for grid or IaaS cloud," *IEEE Trans. Cloud Comput.*, vol. 8, no. 4, pp. 1030–1039, Oct.–Dec. 2020.

[12] A. Alsarhan, A. Itradat, A. Y. Al-Dubai, A. Y. Zomaya, and G. Min, "Adaptive resource allocation and provisioning in multi-service cloud environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 1, pp. 31–42, Jan. 2018.

[13] M. Cheng, J. Li, and S. Nazarian, "DRL-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in *Proc. 23rd Asia South Pacific Des. Automat. Conf.*, 2018, pp. 129–134.

[14] L. Wang and E. Gelenbe, "Adaptive dispatching of tasks in the cloud," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 33–45, Jan.–Mar. 2018.

[15] S. Kardani-Moghaddam, R. Buyya, and K. Ramamohanarao, "ADRL: A hybrid anomaly-aware deep reinforcement learning-based resource scaling in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 3, pp. 514–526, Mar. 2021.

[16] D. S. Kumar and R. J. Kannan, "Reinforcement learning-based controller for adaptive workflow scheduling in multi-tenant cloud computing," *Int. J. Elect. Eng. Educ.*, 2020, doi: 10.1177/0020720919894199.

[17] G. Ismayilov and H. R. Topcuoglu, "Dynamic multi-objective workflow scheduling for cloud computing based on evolutionary algorithms," in *Proc. IEEE/ACM Int. Conf. Utility Cloud Comput. Companion*, 2019, pp. 103–108.

[18] M. A. Rodriguez and R. Buyya, "Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 222–235, Apr.–Jun. 2014.

[19] R. Garg and A. K. Singh, "Multi-objective workflow grid scheduling based on discrete particle swarm optimization," in *Proc. Int. Conf. Swarm, Evol., Memetic Comput.*, 2011, pp. 183–190.

[20] G. Ritu and A. Singh, "Multi-objective workflow grid scheduling using ε-fuzzy dominance sort based discrete particle swarm optimization," *J. Supercomput.*, vol. 68, no. 2, pp. 709–732, 2014.

[21] J. Yu, M. Kirley, and R. Buyya, "Multi-objective planning for workflow execution on grids," in *Proc. 8th IEEE/ACM Int. Conf. Grid Comput.*, 2007, pp. 10–17.

[22] 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.

[23] W. Zheng and R. Sakellariou, "Budget-deadline constrained workflow planning for admission control," *J. Grid Comput.*, vol. 11, no. 4, pp. 633–651, Dec. 2013.

[24] H. R. Faragardi, M. R. S. Sedghpour, S. Fazliahmadi, T. Fahringer, and N. Rasouli, "GRP-HEFT: A budget-constrained resource provisioning scheme for workflow scheduling in iaas clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1239–1254, Jun. 2020.

[25] T. P. Pham, J. J. Durillo, and T. Fahringer, "Predicting workflow task execution time in the cloud using a two-stage machine learning approach," *IEEE Trans. Cloud Comput.*, vol. 8, no. 1, pp. 256–268, Jan.–Mar. 2020.

[26] Z. Quan, Z. J. Wang, T. Ye, and S. Guo, "Task scheduling for energy consumption constrained parallel applications on heterogeneous computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 5, pp. 1165–1182, May 2020.

[27] T. Kwok and A. Mohindra, "Resource calculations with constraints, and placement of tenants and instances for multi-tenant saas applications," in *Proc. Int. Conf. Serv.-Oriented Comput.*, 2008, pp. 633–648.

[28] W. T. Tsai, X. Sun, Q. Shao, and G. Qi, "Two-tier multi-tenancy scaling and load balancing," in *Proc. 7th IEEE Int. Conf. e- Bus. Eng.*, 2010, pp. 484–489.

[29] J. Espadas, A. Molina, G. Jimenez, M. Molina, R. Ramirez, and D. Concha, "A tenant-based resource allocation model for scaling software-as-a-service applications over cloud computing infrastructures," *Future Gen. Comput. Syst.*, vol. 29, no. 1, pp. 273–286, Jan. 2013.

[30] L. Mashayekhy, M. M. Nejad, and D. Grosu, "Cloud federations in the sky: Formation game and mechanism," *IEEE Trans. Cloud Comput.*, vol. 3, no. 1, pp. 14–27, Jan.–Mar. 2015.

[31] J. Zhu, H. Song, Y. Jiang, and B. Li, "On complex tasks scheduling scheme in cloud market based on coalition formation," *Comput. Elect. Eng.*, vol. 58, pp. 465–476, Feb. 2017.

[32] D. C. Marinescu, A. Paya, and J. P. Morrison, "A cloud reservation system for Big Data applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 606–618, Mar. 2017.

[33] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *Proc. IEEE 8th Int. Conf. e- Sci.*, 2012, pp. 1–8.

[34] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Experience*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[35] H. Chen, J. Wen, W. Pedrycz, and G. Wu, "Big data processing workflows oriented real-time scheduling algorithm using task-duplication in geo-distributed clouds," *IEEE Trans. Big Data*, vol. 6, no. 1, pp. 131–144, Mar. 2020.

[36] F. Dong and S. G. Akl, "Scheduling algorithms for grid computing: State of the art and open problems," Tech. Rep. Open Issues Grid Scheduling Workshop, School of Computing, Queen's University, Kingston, Ontario, CA, Tech. Rep. 2006–504, 2006.

[37] W. Li and H. Shi, "Dynamic load balancing algorithm based on FCFS," in *Proc. IEEE 4th Int. Conf. Innov. Comput., Inf. Control*, 2009, pp. 1528–1531.

[38] Amazon. Amazon ec2 on-demand pricing. Jul. 2020. [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand/

Xiaogang Wang (Member, IEEE) received the PhD degree in computer science and technology from Shanghai Jiao Tong University, China, in 2018. He is currently an associate professor with the School of Electronics and Information, Shanghai Dianji University, Shanghai, China. He was a visiting research scholar with the CLOUDS Laboratory in the School of Computing and Information Systems, University of Melbourne, Australia, from September 2019 to September 2020. He has published more than 20 papers in some journals and conferences such as the *IEEE Transactions on Services Computing*, the *Journal of Systems and Software*, *Future Generation Computer Systems*, *Applied Intelligence*, WI-IAT, APSCC, CSCWD, and ICSAI. His main research interests include cloud computing, edge computing, service computing and multi-agent systems. He is a member of the China Computer Federation.

Jian Cao (Senior Member) received the PhD degree from the Nanjing University of Science and Technology, in 2000. He is currently a professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His main research interests include service computing, cloud computing, cooperative information systems and software engineering. He has published more than 150 papers in prestigious journals, such as *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Mobile Computing*, *Journal of Systems and Software*, *Future Generation Computer Systems*. He is a distinguished member of the China Computer Federation.

Rajkumar Buyya (Fellow, IEEE) received the PhD degree in computer science and software engineering from Monash University, Melbourne, Australia, in 2002. He is a Redmond Barry distinguished professor and director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, the University of Melbourne, Australia. He served as a Future fellow of the Australian Research Council during 2012-2016. He has authored more than 625 publications and seven textbooks. He is one of the highly cited authors in computer science and software engineering worldwide (h-index=155, g-index=334, 126,300+ citations). He served as the founding editor-in-chief of the *IEEE Transactions on Cloud Computing*. He is currently serving as co-editor-in-chief of *Journal of Software: Practice and Experience*.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.