Dynamic Scheduling Stochastic Multiworkflows With Deadline Constraints in Clouds

Lingjuan Ye, Yuanqing Xia, *Senior Member, IEEE*, Liwen Yang, and Yufeng Zhan

***Abstract*— Nowadays, more and more workflows with different computing requirements are migrated to clouds and executed with cloud resources. In this work, we study the problem of stochastic multi-workflows scheduling in clouds and formalize this problem as an optimization problem that is NP-hard. To solve this problem, an efficient stochastic multi-workflows dynamic scheduling algorithm called SMWDSA is designed to schedule multi-workflows with deadline constraints for optimizing multi-workflows scheduling cost. The proposed SMWDSA consists of three stages including multi-workflows preprocessing, multi-workflow scheduling and scheduling feedback. In SMWDSA, a novel task sub-deadlines assignment stretagy is design to assign the task sub-deadlines to each task of multi-workflows for meeting workflow deadline constraints. Then, we propose a task scheduling method based on the minimal time slot availability to execution task for minimizing workflow scheduling cost while meetingt workflow deadlines. Finally, a scheduling feedback strategy is adopted to update the priorities and sub-deadlines of unscheduled tasks, for further minimizing workflow scheduling cost. We conduct the experiments using both synthetic data and real-world data to evaluate SMWDSA. The results demonstrate the superiority of SMWDSA as compared with the state-of-the-art algorithms.**

***Note to Practitioners*—Workflow scheduling in clouds is significantly challenging due to not only the large scale of workflows but also the elasticity and heterogeneity of cloud resources. Moreover, minimizing workflow scheduling cost and satisfying workflow deadlines are two critical issues in scheduling with cloud resources, especially the uncertainty of workflow arrive time and task execution time are considered. To meet workflow deadlines, it is an effective strategy to decompose workflow deadline constraints into task sub-deadline constraints. To minimize the workflow scheduling cost, each task in a workflow needs to be assigned to their most suitable VMs for execution. This article presents a novel workflow scheduling algorithm to schedule stochastic multi-workflows in clouds for optimizing multi-workflows scheduling cost and meeting workflows deadlines. This algorithm obtains the task sub-deadline constraints based on the characteristics of workflows for meeting the worklfow deadline constraint. Under the premise of meeting task deadlines, it schedules tasks to a VM with minimum the slot time, for minimizing the cost. Case studies based on well-known real-world workflows data sets suggest that it outperforms traditional ones in terms of success and cost of multi-workflows scheduling. It can thus aid the design and optimization of multi-workflows scheduling in a cloud environment. It can help practitioners better manage the scheduling cost and performance of real-world applications built upon cloud services.**

***Index Terms*— Dynamic scheduling, multi-workflows, deadline constraints, cloud computing.**

Manuscript received 31 July 2022; accepted 25 August 2022. This article was recommended for publication by Associate Editor L. Li and Editor K. Saitou upon evaluation of the reviewers' comments. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1003700 and in part by the National Natural Science Foundation of China under Grant 61836001 and Grant 62173035. *(Corresponding authors: Yuanqing Xia; Yufeng Zhan.)*

Lingjuan Ye is with the School of Mathematics and Statistics, Beijing Institute of Technology, Beijing 100081, China (e-mail: lingjuanye@126.com).

Yuanqing Xia, Liwen Yang, and Yufeng Zhan are with the School of Automation, Beijing Institute of Technology, Beijing 100081, China (e-mail: xia_yuanqing@bit.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASE.2022.3204313.

Digital Object Identifier 10.1109/TASE.2022.3204313

I. Introduction

CLOUD computing is a new way to provide a powerful computing infrastructure with virtualized technologies and is widely used in the industrial and commercial fields [1], [2]. The cloud model consists of a great number of servers which are equipped with adequate cloud resources, such as CPU cores and memory. In clouds, multiple virtual machines (VMs) instances are running simultaneously on these servers. The main advantages of such computing model are virtualization, pay-as-you-use, rapid elasticity, on-demand access, and so on [3]. With these characteristics, cloud computing is very appropriate to hand the diversity requirements and various computing applications [4]. As a result, many workflows applications, such as earthquake science workflow Cybershake, bioinformatics workflow Epigenomics, Gaussian elimination workflow, and fast Fourier transform workflow in mathematics, have been successfully migrated to clouds and executed with cloud resources [5].

In clouds, workflows scheduling is one of the crucial issues which is to choose the most appropriate cloud resources to workflow tasks for satisfying QoS constraints [6]. Workflows are a type model which is used to describe applications executed with cloud resources and generally comprised of a large number of precedence constrained tasks which are connected by controlflow dependencies and data-flow [7]. In general, these workflows require lots of distributed computing resources and need to be processed within deadlines. Moreover, workflows usually show high heterogeneity in deadline constraints, suggesting the potential performance boost if the deadline constraint can be effectively processed. Accordingly, how to satisfy workflow deadline constraints has become an challenge for scheduling in the cloud environment.

In addition, due to the diversity of workflows, a large number of VMs with different computing capacity and configurations are required for distributed collaborative computing. With the heterogeneity and the elasticity of cloud resources, VMs can be dynamically acquired from the infinite cloud resource pool for processing various workflows and faster

VMs are usually more expensive than slower ones. Moreover, VMs generally charged per time frame or billing period because of the different pricing model of cloud resources, which would lead to unexpectedly high charges if the workflow scheduling does not take the cloud pricing model into consideration. Consequently, it is another challenge to minimize the workflow scheduling cost for scheduling in the cloud environment.

To tackle these challenges, various scheduling methods including heuristic scheduling [8], meta-heuristic scheduling [9], hybrid scheduling [10] and learning scheduling [11] have been investigated. Among these methods, heuristic scheduling and meta-heuristic scheduling are probably the most attractive ways to schedule workflows for minimizing workflow scheduling cost and satisfying workflow deadlines. Zhu *et al.* [12] introduced a heuristic algorithm called DyDL based the list-scheduling framework to schedule workflows with deadline constraints for optimizing workflow execution cost. This algorithm prioritizes tasks by their latest start times and appoints tasks the placements which can meet their latest start times and incur the minimal cost increases. Chen *et al.* [9] modeled the workflow scheduling as a multi-objective optimization problem and adopted two colonies to design a multiple objectives framework for minimizing workflow execution time and execution cost.

However, most of these studies assume that the task execution time with a specific VM is certain and can be accurately estimated in advance [13], [14]. In fact, the performance of VMs that is running on the servers of real commercial IaaS clouds can not be kept stable [15], which results that the actual task execution time may fluctuate frequently and the execution of tasks is delayed. Moreover, successor tasks also are delayed with the data dependencies and the continuous cumulative delay may lead to the workflow deadline violated [16]. In addition, the uncertainty of task execution time seriously affects the effectiveness of scheduling strategy. If we use the task execution time is greater than the actual task execution time in scheduling, more VM instances will be used to execution tasks which results the idle time slots between assumed execution time of tasks and their actual execution time will be wasted in the VM instances. Hence, it is very significant to design an effective scheduling algorithm for multi-workflow with uncertain task execution time in clouds.

In this paper, we consider the stochastic multi-workflows dynamic scheduling problem in clouds. This work aims to design an efficient stochastic multi-workflows dynamic scheduling algorithm called SMWDSA which is used to schedule multi-workflows with deadline constraints for optimizing multi-workflows scheduling cost. The proposed SMWDSA consists of three stages including multi-workflows preprocessing, multi-workflow scheduling and scheduling feedback. In SMWDSA, a novel task sub-deadlines assignment stretagy is design to assign the task sub-deadlines to each task of multi-workflows for meeting workflow deadline constraints. Then, we propose a task scheduling method based on the minimal time slot availability to execution task for minimizing workflow scheduling cost while meetingt workflow deadlines. Finally, a scheduling feedback strategy is adopted to update the priorities and sub-deadlines of unscheduled tasks, for further minimizing workflow scheduling cost. The contributions of this paper are as follows:

(i) We study the problem of scheduling multi-workflows with uncertain arrive time and task execution time in clouds and model this problem as a constrained optimization problem that optimizes workflow scheduling cost and satisfies workflow deadline constraints.

(ii) A novel stochastic multi-workflows dynamic scheduling algorithm (SMWDSA) is developed to minimize workflows scheduling cost and satisfy workflows deadlines. The basic idea is to prioritize the tasks of multi-workflows and assign the sub-deadlines for these tasks to meet workflow deadline constraints and satisfy tasks dependency requirements. Furthermore, we update dynamically the priorities and sub-deadlines of unscheduled tasks by a scheduling feedback strategy to optimize workflow scheduling cost while ensuring workflow deadlines.

(iii) We evaluate the proposed SMWDSA with extensive simulations. The experimental results show that SMWDSA can significantly reduce the workflows scheduling cost compared with benchmark algorithms.

The remainder of this article is organized as follows. Section II discusses the related work. We describe the system models and the problem formulation in Section III. Section IV presents a stochastic multi-workflows dynamic scheduling algorithm (SMWDSA). We evaluate the performance of SMWDSA in Section V. We conclude this paper in Section VI.

II. Related Work

In the past few years, many researchers have conducted extensive explorations, among which workflows scheduling in clouds is a comparatively hot issue [17], [18], [19]. In this section, we introduce some related works on workflow scheduling in clouds.

The problem of workflows scheduling with deadline-constrained and cost optimization in a cloud computing system was investigated in some recent works. In [20], two workflow scheduling algorithms which are the IaaS cloud partial critical paths (IC-PCP) algorithm and the IaaS cloud partial critical paths with deadline distribution (IC-PCPD2) were designed to reduce workflow scheduling cost in clouds. The IC-PCP and IC-PCPD2 find the critical path of the workflow and distribute the workflow deadline among the critical nodes with the PCP to minimize the workflow execution while meeting the deadline constraint. In [21], Singh *et al.* proposed a dynamic algorithm based on k-means clustering to schedule deadline constraint workflow for minimizinf workflow execution cost. In [22], Sahni *et al.* developed a heuristic algorithm to scheduling science workflows with deadline constraints for reducing workflow execution cost in the cloud. Wu *et al.* [5] proposed a workflow scheduling algorithm which is based on list ant colony optimization

(L-ACO) to minimize workflow execution cost for satisfying the deadline constraint in clouds. Arabnejad *et al.* [8] focused on the scientific workflows scheduling problem which aims to dynamically schedule scientific workflows for reducing the execution cost while meeting deadline constraints.

In addition, some researches focus on multi-objective optimization problem (MOP) for cost and makespan, which minimizes cost and makespan in clouds simultaneously. In [23], an evolutionary optimization algorithm based on genetic algorithm (GA) was designed by Zhu *et al.* for solving the multi-objective workflow scheduling problem in clouds. Moreover, a novel multi-objective workflow scheduling algorithm based on ant colony optimization was proposed by Chen *et al.* [9] to optimal wokflow finish time and workflow execution cost. Han *et al.* [24] introduced an efficient two-phase list scheduling algorithm to scheduling workflows in clouds for reducing the execution cost and the workflow makespan. Durillo *et al.* [25] designed a multi-objective HEFT (MOHEFT) to optimize workflow execution time and the execution cost. Wu *et al.* [26] proposed an evolutionary list scheduling algorithm which adopts the list scheduling and multiobjective evolutionary algorithm for optimizing makespan and cost.

Recently, the stochastic workflows scheduling problems are considered by some scholars in cloud environments. In [27], a job scheduling problem was studied, which aims to dynamically schedule jobs with requesting different cloud resources. In [28], Cai *et al.* considered a effective method to provision cloud resources and dynamic schedule Bag-of-Tasks workflows which are with uncertain task execution times for decreasing the workflow scheduling cost. Chen *et al.* [29] proposed a dynamic scheduling algorithm for solving the uncertainty-aware workflow scheduling problems. Furthermore, an uncertainty-aware online scheduling algorithm (ROSA) was designed by Chen *et al.* [30] to schedule online workflows with stochastic task execution time for minimizing workflow execution cost and satisfying workflow deadlines. Furthermore, Liu *et al.* [31] extended the ROSA and introduced an new workflow scheduling framework (NOSF) to schedule online multi-workflows which are with the stochastic task execution time and workflow arrival rates. Arabnejad *et al.* [32] presented a new algorithm, dynamic workload scheduler (DWS) that handles the dynamics of multiple deadline constrained workflows arriving randomly and scheduling these workflows with reducing cost. Gu *et al.* [33] considered the problem of scheduling microservice workfows with hybrid resource provisioning and proposed an adaptive-learning based scheduling algorithmic framework to intelligently sequence, allocate and online adjust tasks as well as monitor spot instance. Ma *et al.* [34] proposed a real-time multiple-workflow scheduling (RMWS) scheme to schedule workflows dynamically with minimum cost under different deadline constraints. Dong *et al.* [35] studied the workflow scheduling problem considering the performance variation of cloud resources and proposed a dynamic workflow scheduling approach based on deep reinforcement learning (RLWS) to minimize the makespan. In addition, some learning-based algorithms [36], [37] are used to solve the workflow scheduling problem in cloud computing.

TABLE I
SYMBOL AND NOTATION

Symbol	Notation
$\mathcal{W}$	Workflow set
W^s	sth workflow
N	The number of total workflows
N_{succ}	The number of success workflows
CT^s	Complete time of sth workflow
A^s	Arriving time of W^s
D^s	Deadline of W^s
G^s	The dependence relation graph of W^s
N^s	The number of tasks in W^s
t_j^s	The jth task of W^s
$ST(t_j^s)$	Start time of t_j^s
$FT(t_j^s)$	Finish time of t_j^s
$Aet(t_j^s)$	Basic approximate execution time of t_j^s
$TT(e_{jk}^s)$	Transmission time between t_j^s and t_k^s in W^s
$IT(t_j^s(k))$	Idle time of t_j^s in W^s with the VM instance I_k
$Est(t_j^s)$	Earliest start time of t_j^s
$Ect(t_j^s)$	Earliest finish time of t_j^s
$Lct(t_j^s)$	Last finish time of t_j^s
$ET(t_j^s)$	Execution time of t_j^s
$pre(t_j^s)$	Parent tasks set of t_j^s
$succ(t_j^s)$	Children tasks set of t_j^s
$\mathcal{I}$	VM instances set
$\|I\|$	The number of rental VM instances for scheduling
I_k	k-th VM instance
$P(I_k)$	The price of the k-th VM

III. SYSTEM MODEL

In this section, we firstly introduce cloud resource model and application model. Then, we formulate workflows scheduling as a constrained optimization problem that minimizing total cost while satisfying workflow deadlines. The important definitions and notations are shown in Table I.

A. Cloud Resource Model

A cloud service platform provides different cloud resources, such as CPU, RAM, disk storage and bandwidth to deal with different applications based on their requirements. These cloud resources are elastically used to process computing applications with different types of VMs. These VMs providing services to end users are commonly measured in terms of central processing unit (CPU) capacity, memory, and storage; this allows VMs to be classified based on rankings and costs [3]. A VM with a higher ranking provides a higher task execution speed, which is more expensive to rent [26]. In this paper, the available VM types is set as *M* and each VM type is equipped with a price and configuration of cloud resources.

VMs are pay-per-use on-demand in clouds and the VMs set for executing workflows is denoted by $\mathcal{I} = \{I_k|, k = 1, 2, \ldots\}$ which consists of M types of VMs, where I_k represents the k-th VM. For a VM I_k, we use $P(I_k)$ to represent the VM unit price with a billing period. The parameter r_k denotes the VM weight for task execution time with VM I_k, which is defined as the ratio between the task execution time on I_k and shortest task execution time. Specifically, $Rank(I_k) = 1$ if task execution time in minimal with VM I_k. In addition, by the pricing model of cloud resources, VMs are leased with the whole hour by users at any time [38], that is the rental time of a VM is generally billed in multiples of a unit of time.

B. Application Model

We consider a stochastic workflow set $\mathcal{W} = \{W^1, \ldots, W^N\}$ with N independent sporadic workflows in this work. Let N denote the total number of the workflow set $\mathcal{W}$. A workflow W^s has the following features:

$$W^s = \{A^s, D^s, G^s, T^s, E^s\}, \tag{1}$$

where the arrival time is denoted as A^s and the deadline of W^s is D^s. Moreover, the directed acyclic graph (DAG) of workflow W^s [39] is denoted as G^s. A DAG G^s consists of a task set and a edge set which are represented as $T^s = \{t_1^s, \ldots, t_{N^s}^s\}$ and $E^s = \{e_{i,j}^s|i, j = 1, \ldots, N^s\}$, respectively. The jth task of W^s is denoted as t_j^s which has a stochastic execution time $ET(t_j^s)$ with the highest rank of VMs for processing task t_j^s, where $ET(t_j^s)$ is a random variable with normal distribution. The edge e_{ji}^s is the data dependency between task t_i^s and t_j^s and indicates that task t_i^s is a immediate predecessor of task t_j^s and task t_j^s is a immediate successor of task t_i^s. We use $pre(t_j^s)$ and $succ(t_j^s)$ to represent the immediate predecessors set and the successors set of task t_j^s, respectively. Then, for each task the t_j^s in $pre(t_l^s)$, task t_l^s is a immediate successor of task t_j^s. Specially, the workflow entry task t_{entry}^s has no the immediate predecessor a and the exit task t_{exit}^s has no immediate successors.

In addition, the existing workflow scheduling approaches adopted the normal distributions to address the fluctuation of the task execution time which is along with the variation of the VM computing capability. In our work, the independent and normal distributions are used to describe the task execution time on a VM. Similar to [40], we use an approximate value $Aet(t_j^s)$ to compute the task execution time $ET(t_j^s)$, where $ET(t_j^s)$ denotes the task execution time with the highest rank VM. Let $ET(t_j^s)$ follow a normal distributions $N(e(t_j^s), v(t_j^s))$ with the expected value $e(t_j^s)$ and the variance $v(t_j^s)$. Then, the $Aet(t_j^s)$ is calculated as

$$Aet(t_j^s) = e(t_j^s) + v(t_j^s). \tag{2}$$

C. Problem Formulation

In this subsection, we model this problem as a constrained optimization problem. Let $\mathcal{W}$ denote the mutil-workflows set which contains N independent sporadic workflows. In this paper, we find a feasible schedule solution to minimize the workflows scheduling cost $WSC(\mathcal{W})$ for the workflow set $\mathcal{W}$. We use $P(I_k)$ to be the price of I_k and the leased time of VM I_k is denoted as LT_k. The finish time of workflow W^s is denoted as FT^s and is defined as

$$FT^s = \max_{t_j^s \in T^s} FT(t_j^s), \tag{3}$$

where $FT(t_j^s)$ denotes finish time of task t_j^s and T^s is the task set for workflow W^s. The data transmission time between task t_j^s and t_l^s is calculated as

$$TT(e_{jl}^s) = \frac{d_{jl}^s}{bw}, \tag{4}$$

the data transferred from t_j^s to t_l^s is represented as d_{jl}^s and bw represents the average network bandwidth between different VMs. Then, the problem formulations are as

$$\min WSC(\mathcal{W}) = \sum_{I_k \in \mathcal{I}} P(I_k) LT_k, \tag{5}$$

subject to:

$$CT^s \leq D^s, \tag{6}$$

$$FT(t_j^s) + TT(e_{jl}^s) \leq ST(t_l^s), \tag{7}$$

$$\sum_{k=1}^{|I|} x_{j,k}^s \leq 1, \tag{8}$$

where the binary variable $x_{j,k}^s$ is defined as

$$x_{j,k}^s = \begin{cases} 1, & \text{if } t_j^s \text{ is scheduled on } I_k; \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Constraint (6) explains that each workflow deadline is satisfied. Constraint (7) indicates the data dependencies between tasks are satisfied. Constraint (8) and (9) guarantee that every task should be scheduled once only.

IV. Scheduling Algorithms

In clouds, multi-workflows are executed with many heterogeneous VMs which have different computing capacity and billing modes. For the heterogeneity of VMs, this paper uses the VM weight and the VM service price to represent the execution capacity and rental cost of VMs for the multi-workflows scheduling. In addition, the elasticity is another feature of cloud resources. In the process of multi-workflow dynamic scheduling, VMs can be used flexibly, which greatly improves the work efficiency. The cloud computing system can provide unlimited VMs for workflow scheduling and VMs can be scaled up at any time. In this section, we first introduce the lower bound of multi-workflows scheduling cost with heterogeneous VMs in clouds. After that, we propose an efficient multi-workflows scheduling algorithm with elastic VMs and analyze its computational complexity.

A. Lower Bound of Multi-Workflows Scheduling Cost

To assist the workflow scheduling algorithm design to optimize multi-workflows scheduling cost in Section IV.B, we first give the lower bound of workflow scheduling cost for multi-workflows in this subsection.

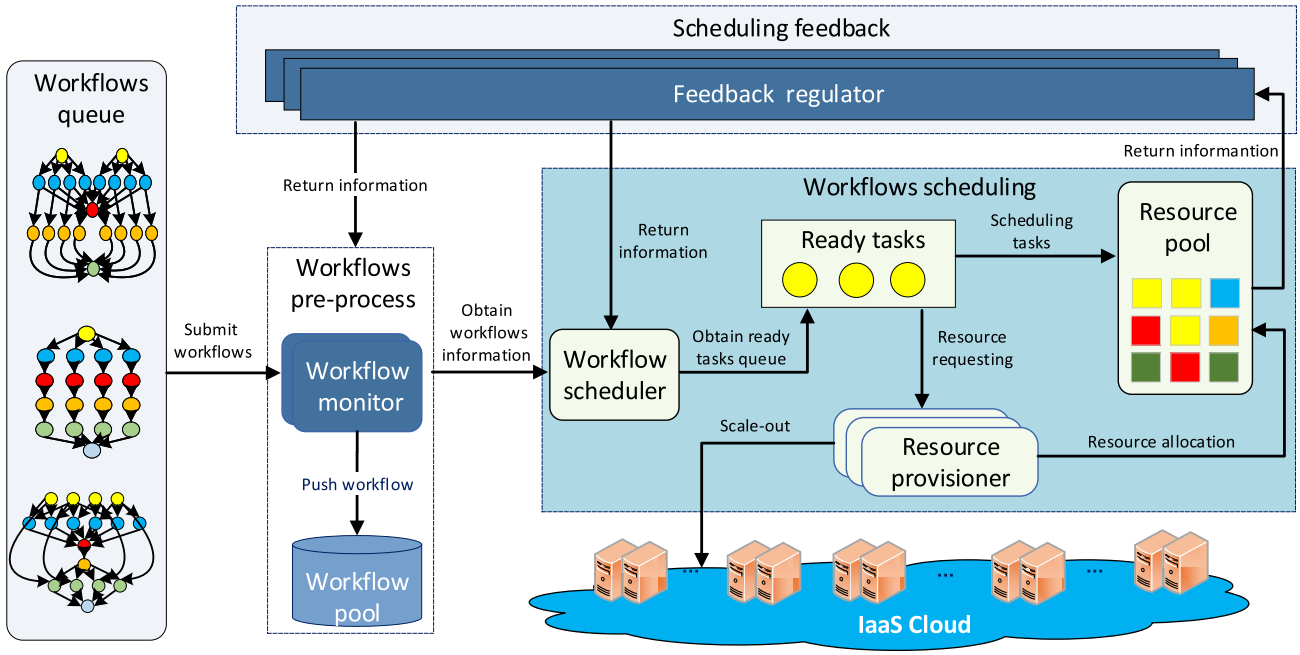


Fig. 1. Stochastic Multi-Workflows Dynamic Scheduling Process.

Lemma 1: Let $\mathcal{W} = \{W^1, \ldots, W^N\}$ be a workflow set and $m \in \mathcal{M} = \{1, 2, \ldots, M\}$ be the available types of VMs, then the lower bound of the workflows scheduling cost for workflows $\mathcal{W}$ can be obtained as

$$WSC(\mathcal{W}) \geq \sum_{s=1}^{N}\sum_{j=1}^{N^s} \min_{k\in\mathcal{M}}\{P(I_k) * ET_k(t_j^s)\} + \sum_{k=1}^{|I|}(P(I_k) * \Delta_k), \quad (10)$$

where $P(I_k)$ represents the price of VM I_k, $ET_k(t_j^s)$ is the actual execution time of task t_j^s with VM I_k and Δ_k corresponds to the time slot between last task real finish time and the released time of VM I_k.

Proof: For the workflow set $\mathcal{W} = \{W^1, \ldots, W^N\}$, the workflows scheduling cost is calculated as

$$\begin{aligned} WSC(\mathcal{W}) &= \sum_{k=1}^{|I|} P(I_k) * LT_k \\ &= \sum_{k=1}^{|I|} P(I_k) * (\sum_{l=1}^{n_k}(ET_l^k) + \Delta_k) \\ &= \sum_{k=1}^{|I|} P(I_k) * \sum_{l=1}^{n_k}(ET_l^k) + \sum_{I=1}^{|I|} P(I_k) * \Delta_k \\ &= \sum_{s=1}^{N}\sum_{j=1}^{N^s} P(I_k) * ET_k(t_j^s) + \sum_{k=1}^{|I|} P(I_k) * \Delta_k \\ &\geq \sum_{s=1}^{N}\sum_{j=1}^{N^s} \min_{k\in\mathcal{M}}\{P(I_k) * ET_k(t_j^s)\} \\ &\quad + \sum_{k=1}^{|I|}(P(I_k) * \Delta_k), \end{aligned} \quad (11)$$

where n_k represents the number of tasks scheduled on VM I_k and ET_l^k is the execution time of $l-th$ task on VM r_k. Hence, the lower bound of the workflows scheduling cost is as:

$$\sum_{s=1}^{N}\sum_{j=1}^{N^s} \min_{I_k\in\mathcal{M}}\{P(I_k) * ET_k(t_j^s)\} + \sum_{k=1}^{|I|}(P(I_k) * \Delta_k). \quad (12)$$

By inspiring with the Lemma 1, we optimize the workflow scheduling cost via the following two ways: 1) reducing the VM idle time slots for scheduling workflows $\sum_{k=1}^{|I|}(P(I_k) * \Delta_k)$; and 2) minimizing the executing cost for each task $\min_{k\in\mathcal{M}}\{P(I_k) * ET_k(t_j^s)\}$. With this two ways, a heuristic algorithm is designed to schedule multi-worklfows in clouds in Section IV. B.

B. Stochastic Multi-Workflows Dynamic Scheduling Algorithm

In this part, we present the details of our proposed algorithm, which has three phases. First, workflow monitor estimates the task execution time for each workflow and obtains the task sub-deadlines constraints without violating their respective completion time constraints in the workflows preprocess stage. Second, Workflow scheduler constructs a ready task priority queue for multiple workflows without violating their respective deadlines and schedules tasks to VMs to reduce multi-workflows scheduling cost in the workflow scheduling stage. Finally, feedback regulator updates the original priorities and sub-deadlines of unscheduled tasks to reduce the multi-workflows scheduling cost in the scheduling feedback stage. The multi-workflows scheduling process is shown in Fig. 1. The detailed description of the proposed algorithm is given in Algorithm 1 and Algorithm 2.

1) Mutil-Workflows Preprocessing: When a workflow set arriving, we firstly compute the approximation task execution

Algorithm 1 Stochastic Multi-Workflows Dynamic Scheduling Algorithm (SMWDSA)

Input: The current workflows set $\mathcal{W} = (W^1, \ldots, W^N)$.
Output: A schedule solution S for workflows set $\mathcal{W}$.

$\mathcal{W}^L \leftarrow empty$, $\mathcal{W}^S \leftarrow empty$;
while termination conditions not met **do**
  for each workflow W^k in $\mathcal{W}$ **do**
    Calculate the execution time of each task in $\mathcal{W}$;
    Calculate the Earliest Start Time and the Earliest Finish Time of each task in W^k.
    Calculate the Least Start Time and the Least Finish Time of each task in W^k.
    Calculate the workflow minimum finish time Mft^k for W^k;
    if $Mft^k > BU$ **then**
      add W^k to $\mathcal{W}^L$
    else
      add W^k to $\mathcal{W}^S$
  while a new workflow does not arrive **do**
    for each workflow W in $\mathcal{W}^L$ **do**
      for each task t in W **do**
        Calculate the sub-deadline of each task in W with latest complete time by Eq. (17).
    for each workflow W in $\mathcal{W}^S$ **do**
      for each task t in W **do**
        if t is an entry task or the parents of t are assigned **then**
          the task cluster number $CN(t) = 1$;
        if t is not an entry task and the parents of t are assigned **then**
          $CN(t)$=max$_{t^* \in par(t)} CN(t^*) + 1$;
        Calculate the sub-deadline of each task in W by Eq. (19).
    Add each task of $\mathcal{W}$ into $Taskpool$
    if $Taskpool$ is not empty **then**
      for each task t in $Taskpool$ **do**
        if each parent tasks of t is scheduled and t is not scheduled || t is an entry task **then**
          Add task t into the ready task set $\mathcal{RTS}$;
      if $\mathcal{RTS}$ is not empty **then**
        Call *Tasks scheduling*($\mathcal{RTS}$);
  end while
  if a new workflow W^* arrives **then**
    $\mathcal{W} \leftarrow \mathcal{W} + W^*$;
return A schedule S

time by Eq. (2). Then, we calculate the earliest start time and the earliest completion time by VM weight $r_k = 1$. The earliest start time $Est(t_i^s)$ for task t_i^s is calculated as

$$Est(t_i^s) = \begin{cases} A^s, & \text{if } t_i^s = t_{entry}^s; \\ \max_{t_p^s \in pre(t_i^s)}\{Est(t_p^s) + & \\ Aet(t_p^s) + TT(e_{pi}^s)\}, & \text{otherwise.} \end{cases} \quad (13)$$

where $Aet(t_p^s)$ is the approximation execution time and can be calculated by Eq. (2). The transmission time between t_p^s and t_i^s is denoted as $TT(e_{pi}^s)$ and can be calculated by Eq. (4). Accordingly, we calculate the earliest completion time $Ect(t_i^s)$ as

$$Ect(t_i^s) = Est(t_i^s) + Aet(t_i^s). \quad (14)$$

Then, we calculate the latest completion time $Lct(t_i^s)$ as

$$Lct(t_i^s) = \begin{cases} D^s & \text{if } t_i^s = t_{exit}^s, \\ \max_{t_q^s \in succ(t_i^s)}\{Lct(t_q^s) - & \\ Aet(t_q^s) - TT(e_{iq}^s)\} & \text{otherwise.} \end{cases} \quad (15)$$

For a workflow W^s, the minimum finish time (MFT) is defined as

$$Mft^s = \max_{t_i^s \in W^s} Ect(t_i^s). \quad (16)$$

In addition, to reduce the the idle time slots cost of all rented VM instance $\sum_{k=1}^{|I|}(P(I_k) * \Delta_k)$, the workflow set is divided into the long term workflow set and the short term workflow set based on their MFT for reducing the number of rented VMs I. The workflow is a long term workflow with $Mft(W^s) > BU$ and the short term workflow with

Algorithm 2 Tasks Scheduling

Input: A ready task set $\mathcal{RTS}$ and the active VMs set $\mathcal{I}$.
Output: A VM I^* for scheduling each task t in R.

$\mathcal{I}_* \leftarrow empty$, $Finshset \leftarrow empty$, $idleMax \leftarrow \infty$, $I^* = null$;
Sort $\mathcal{RTS}$ by non-descending sub-deadlines of tasks;
for each task t in R **do**
  for each I_k in active VMs set $\mathcal{I}$ **do**
    the finish time $FT_k(t)$ and cost $C_k(t)$ of the task t4 with I_k;
  for each I_k in active VMs set $\mathcal{I}$ **do**
    if $FT_k(t) \le D_{sub}(t)$ and $C_k(t) = min_{i \in \mathcal{I}} C_i(t)$; **then**
      Add I_k into $\mathcal{I}_*$, and $C(t) \leftarrow C_k(t)$;
  if $\mathcal{I}_*$ is not null **then**
    for each VM instance I_k in $\mathcal{I}_*$ **do**
      Calculate the idle time $idle_k(t)$ on I_k;
      if $idle_k(t) \le idleMax$ **then**
        $I^* = I_k$;
  else
    Start a new VM instance I^+ with the minimum execution cost while finish time $FT_+(t) \le D_{sub}(t)$;
    $I^* = I^+$;
  Scheduling task t on the VM I^*;
  Add task I^* into $\mathcal{I}$;
  Add task t into $Finshset$;
  $R \leftarrow R - t$;
for each task t in $FinishedTaskSet$ **do**
  Calculate the actual finish time of task t;
  for each successor task $succ(t)$ of t **do**
    if $succ(t)$ is a ready task **then**
      Update $Est(succ(t))$ of $succ(t)$ by Eq. (28);
      Update $Ect(succ(t))$ of $succ(t)$ by Eq. (29);
      Update $D_{sub}(succ(t))$ of $succ(t)$ by Eq. (30);
return results;

$Mft(W^s) \leq BU$, where BU is the time of billing unit. In order to deal with the difference of two workflow type, we adopt two different workflow deadline constraint decomposition strategies to set task sub-deadline constraints for long term workflows and short term workflows, respectively. For the long term workflow, we make full use of the margin between deadline and minimum completion time and set the latest completion time of the task as the task sub-deadline to reduce the workflow execution cost. Then, the task sub-deadline of the lager workflow is denoted as $D_L(t_j^s)$ and calculated as

$$D_L(t_j^s) = Lct(t_j^s). \tag{17}$$

Furthermore, we design a cluster-based deadline distribution strategy for the short term workflows. The all tasks of a short term workflow are divided into some clusters by satisfying the dependency constraints between tasks. The the cluster number of task t_i^s is defined as

$$CN(t_i^s) = \begin{cases} 1, & \text{if } t_i^s = t_{entry}^s, \\ \max_{t_p^s \in pre(t_i^s)\{CN(t_p^s)+1\}}, & \text{otherwise.} \end{cases} \tag{18}$$

In this way, the sub-deadline $D_S(t_i^s)$ for task t_i^s can be described as

$$D_S(t_i^s) = AEst(t_i^s) + \frac{Ect(t_i^s) - Est(t_{i_{min}}^s)}{Ect(t_{i_{max}}^s) - Est(t_{i_{min}}^s)} \times (Lct(t_{i_{max}}^s) - Est(t_{i_{min}}^s)). \tag{19}$$

where the minimal earliest start time $Est(t_{i_{min}}^s)$ with the same cluster number of t_i^s is as follow:

$$Est(t_{i_{min}}^s) = \min_{CN(t_j^s)=CN(t_i^s)} Est(t_j^s), \tag{20}$$

and the maximal earliest complete time $Ect(t_{i_{max}}^s)$ with the same cluster number of t_i^s is defined as

$$Ect(t_{i_{max}}^s) = \max_{CN(t_j^s)=CN(t_i^s)} Ect(t_j^s). \tag{21}$$

Similarly, the maximal last complete time $lct(t_{i_{max}}^s)$ with the same cluster number of t_i^s is denoted as

$$Lct(t_{i_{max}}^s) = \max_{CN(t_j^s)=CN(t_i^s)} Lct(t_j^s). \tag{22}$$

Then, we calculate the sub-deadline $D_{sub}(t_j^s)$ for task t_j^s of multi-workflows as:

$$D_{sub}(t_j^s) = \begin{cases} D_L(t_j^s), & \text{if } Mft(W^s) > BU\ , \\ D_S(t_j^s), & \text{otherwise.} \end{cases} \tag{23}$$

2) Multi-Workflows Scheduling: The basic idea of the workflows scheduling phase is to assign one VM with the minimal task execution cost for each task while satisfying the task sub-deadline. We first obtain a ready task set from workflows with the same arrival time. Moreover, we create a task scheduling priority queue for the ready task set by increasing earliest start time of tasks. Finally, each ready task is assign to a VM according to the task priority. The detailed steps are shown in following.

Ready Task Queue: We put the workflows with the same arrival time into the workflow pool. Then, the ready tasks are obtained from the workflow pool. A ready task has no parent task or its parent tasks are scheduled. Furthermore, we obtain the task sub-deadline for each task based on the workflow deadline distribution methods and construct a task priority queue for these ready tasks by increasing order of task sub-deadlines. In this way, we prioritize one task with smaller sub-deadline i.e., a higher priority to be scheduled is assigned for one task with more stringent sub-deadline. Therefore, a ready task priority queue is obtained with the task sub-deadlines.

VM Instance Selection: We aim to identify VMs for executing the current ready tasks. We choose task t_j^s with highest task priority from the ready task priority queue and compute the ready time $RT_k(t_j^s)$ of task t_j^s on VM I_k as

$$RT_k(t_j^s) = \max\{\max_{t_i^s \in pre(t_j^s)} AFT(t_i^s), AFT(I_k)\}. \tag{24}$$

where $AFT(t_i^s)$ and $AFT(I_k)$ denote the actual finish time of task t_i^s and VM I_k, respectively. Then, we compute task finish time $FT_k(t_j^s)$ on VM I_k as

$$FT_k(t_j^s) = RT_k(t_j^s) + ET_k(t_j^s). \tag{25}$$

where $ET_k(t_j^s)$ represents task execution time on VM I_k and is calculated as

$$ET_k(t_j^s) = Aet(t_j^s) * r_k. \tag{26}$$

where $Aet(t_j^s)$ denotes the approximate task execution time of task t_j^s with the highest rank VM according to Eq. (2) and r_k denotes the rank parameter of VM I_k. To minimize the cost for executing each task $\min_{I_k \in I}\{P(I_k) * ET_k(t_j^s)\}$, the VM set I^* that satisfy the task sub-deadline and are with minimal execution cost are selected from the active VM set. To further minimize the VM idle time slots $\sum_{k=1}^{I}(P(r_k) * \Delta_k)$, the VM with minimal idle time is chosen from I^* to execute tasks, where idle time $IT(t_{j,k}^s)$ is calculated as:

$$IT(t_{j,k}^s) = RT(t_j^s) - RT(I_k). \tag{27}$$

where $RT(t_j^s)$ denotes the task ready time and $RT(I_k)$ denotes the VM ready time. If all active VMs can not satisfy the task sub-deadline, we will start a new VM with minimal execution cost and satisfying task sub-deadline. Specially, if multi-workflows are sparse tasks, we can construct a workflow priority queue for these sparse tasks by increasing order of workflow deadlines. Then, we choose the sparse task with highest task priority from the workflow priority queue and compute the ready time for it. Finally, these sparse tasks are assigned to a most suitable VM for execution by Algorithm 2.

3) Scheduling Feedback: To ensure that all workflows are finished within their deadlines, we adjust the original priorities and sub-deadlines of the next ready tasks from Short term workflows according to the actual finish time of scheduled task in the scheduling feedback stage. For a ready task of the small workflow, we calculate the new earliest start time and earliest complete time as:

$$Est^*(t_j^s) = \max_{t_p^s \in pre(t_j^s)} \{Aft(t_p^s) + TT(e_{p,j}^s)\}, \tag{28}$$

and

$$Ect^*(t_j^s) = Est^*(t_j^s) + Aet(t_j^s), \tag{29}$$

where $Aft(t_p^s)$ denotes the actual finish time of task t_p^s and $TT(e_{p,j}^s)$ is the data transmission time from task t_p^s to task t_j^s with Eq.(4). Then, we update the sub-deadline for task t_j^s as:

$$D_{sub}^*(t_j^s) = \begin{cases} D_L(t_j^s), & \text{if } Mft(W^s) > BU, \\ \min\{D_S^*(t_j^s), Lct(t_j^s)\}, & \text{otherwise.} \end{cases} \tag{30}$$

where $D_S^*(t_j^s)$ is the new sub-deadline for the short term workflow tasks and calculated as:

$$D_S^*(t_i^s) = Est^*(t_i^s) + \frac{Ect^*(t_i^s) - Est^*(t_{i_{min}}^s)}{Ect^*(t_{i_{max}}^s) - Est^*(t_{i_{min}}^s)} \times (Lct(t_{i_{max}}^s) - Est^*(t_{i_{min}}^s)). \tag{31}$$

A new ready task priority queue can be obtained by increasing the order of sub-deadlines of multiple ready tasks.

C. Complexity Analysis

The complexity of Algorithm 1 is analyzed with two sections, i.e multi-workflows preprocessing and task scheduling. In multi-workflows preprocessing stage, the complexity is determined by line 2-11 of Algorithm 1. The time complexity for computing tasks executing time and classify workflows are $O(T)$ and $O(N)$, respectively. The task number of N workflows is denoted T. Then, the time complexity of $O(H)$ is to obtain the task sub-deadline of long term workflows, where H is the task number of long term workflows. The complexity of $O(L)$ is used to obtain the task sub-deadline for short term workflow tasks, where L denotes the task number of short term workflows. Next, the time complexity of $O(RlogR)$ is used to obtain the ready task scheduling ordering, where R denotes the number of ready tasks. It requires $O(R|I|)$ time for task scheduling in Algorithm 2 (line 3-23), where the number of active VMs is denoted as $|I|$. Thus, it requires $O(RlogR) + O(R|I|)$ time to schedule ready tasks. For the scheduling feedback in line $24-30$ of Algorithm 2, it has the complexity of $O(FS)$ to adjust the sub-deadlines and scheduling ordering of new ready tasks, where F and S denote the number of finished tasks and new ready tasks, respectively. Thus, the time complexity of Algorithm 2 and Algorithm 1 are $O(RlogR) + O(RI) + O(FS)$ and $O(T) + O(N) + 2O(H)+2O(L)+O(RlogR)+O(RI)+O(FS)$, respectively.

V. Performance Evaluation

In this section, we use four evaluating metrics, such as total cost, number of VMs, resource utilization and success ratio to demonstrate the experimental results.

A. Experimental Setting

1) Environment Configurations: The experiments are performed via WorkflowSim [36] and conducted on a a PC with 3.00GHz Intel Core i5 processor and 16GB memory, Windows 10, JAVA, Eclipse, and JDK 7.0.

In our experiments, five real scientific workflow applications which are Montage, Epigenomics, Inspiral, CyberShake, and Sipht, are used to evaluate the performance of our algorithm.

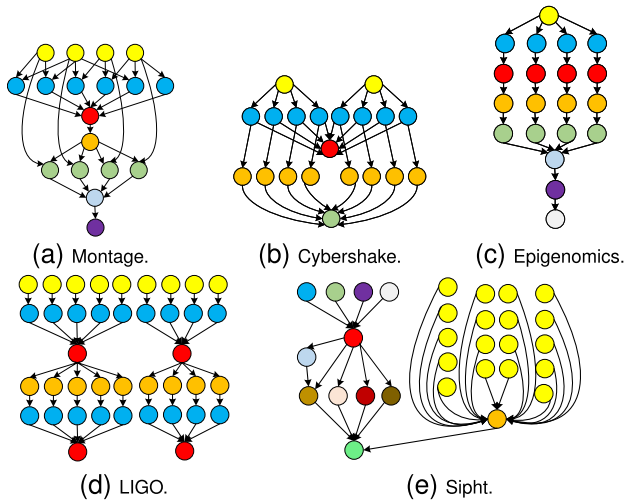


Fig. 2. Structures of five realistic workflows.

TABLE II
VM Instances Parameters of Experiments

VM type	vCPU	Price per time unit($)	VM weight
V^1	8	0.98	1.0
V^2	4	0.49	1.2
V^3	4	0.35	1.3
V^4	2	0.245	1.4
V^5	2	0.175	1.6
V^6	1	0.087	1.8
V^7	1	0.044	2.0

These workflows are widely used in workflow scheduling problems [26] and have different structure and scharacteristics [42]. Specially, we show the structures of five real workflows in Fig. 2. In addition, we take seven different types of VMs from Amazon EC2 to simulate the cloud platform [31]. The prices and processing capabilities of these VMs are shown in Table II, where VM weight are used to characterize the rank of VMs. The average bandwidth among VMs and the billing period (BP) are the same as Amazon EC2 and set to 100 Mbps and 3600 seconds, respectively.

2) Parameter Settings and Baselines: The baseline execution time $ET(t_i^s)$ for task t_i^s is obtained by the task runtime recorded on the highest rank VM from a workflow traces [30]. In this paper, the execution time of task t_i^s is considered as a stochastic variable with the normal distribution $N(ET(t_i^s), \beta ET(t_i^s))$, where the variance factor $\beta \in \{0.6, 0.7, 0.8, 0.9, 1.0\}$ is used to represent the fluctuation of task execution time. The workflow number N for scheduling is set as $\{100, 200, 400\}$ and the workflow arrival rate is a Possion distribution with $\lambda = 1$.

In addition, similar to [5], we evaluate the impact of different deadlines on our algorithm with the baseline schedule S_{fast}^s which is expressed as:

$$S_{fast}^s = \sum_{t_i^s \in W^s} ET_{min}(t_i^s). \tag{32}$$

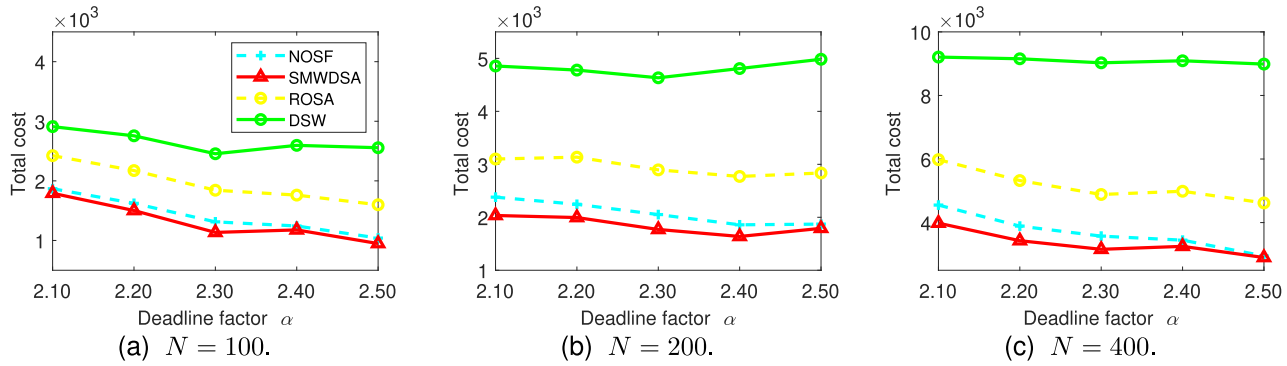


Fig. 3. Total cost of three algorithms with different deadline factors α.

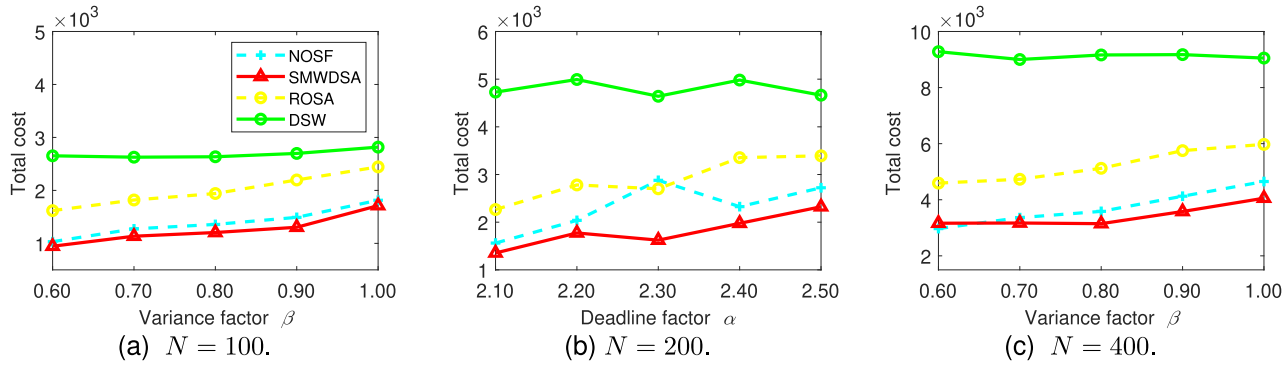


Fig. 4. Total cost of multi-workflows with different variance factors β.

where the task execution time $ET_{min}(t_i^s)$ is obtained by the highest ranking VM. Then, the workflow deadline is obtained by

$$D^s = A^s + \alpha S^s_{fast}, \tag{33}$$

where the deadline factor $\alpha \in \{2.1, 2.2, 2.3, 2.4, 2.5\}$.

In this paper, three related algorithms are used to evaluate the performance of the proposed approach. ROSA [30] is designed to schedule online workflows with stochastic task execution time for minimizing workflow execution cost and satisfying workflow deadlines. Furthermore, NOSF [31] is an new workflow scheduling framework to schedule online multi-workflows which are with the stochastic task execution time and workflow arrival rates, which is based on ROSA. NOSF solved an online workflows dynamic scheduling problem in IaaS clouds and tasks execution times in NOSF are considered as random variables which are following normal distributions. In NOSF, the ready tasks are ranked based on task earliest start time. Then, NOSF schedules these tasks based on the task priorities. Finally, the scheduling order and sub-deadlines of ready tasks can be updated to improve workflow success ratio in the feedback processing. DWS [32] handles the dynamics of multiple deadline constrained workflows arriving randomly and scheduling these workflows with reducing cost.

3) Performance Metrics: In this paper, four metrics are used to evaluate the performance of methods. The details are shown as follows.

- **Total cost (TC):** We calculate the normalized cost with all used VMs as:
$$TC = \sum_{m=1}^{|I|} LT_m P(I_m). \tag{34}$$
where LT_m represents the leased time of VM I_m for executing workflows and the price of VM I_m is denoted as $P(I_m)$.
- **Success ratio (SR):** We use N_f to denote the number of fail workflows whose the completion time exceeds their deadlines. SR is the ratio between N_{succ} and total workflow number N, expressed as:
$$SR = \frac{N_{succ}}{N}. \tag{35}$$
- **Number of VMs:** The number of VMs leased is a important metric for scheduling multi-workflows in cloud and is used to evaluate our algorithm.
- **Resource utilization (RU):** We evaluate these algorithms with the resource utilization of all VMs, which is defined as
$$RU = \frac{\sum_{m=1}^{|I|} ET_m}{\sum_{m=1}^{|I|} LT_m}. \tag{36}$$
where ET_m represents the processing time of VM I_m for executing multi-workflows and LT_m represents the leased time of VM I_m.

B. Experimental Results

1) Total Cost for Scheduling Multi-Workflows: According to Figs. 3 and Fig. 4, the total cost obtained by SMWDSA is less

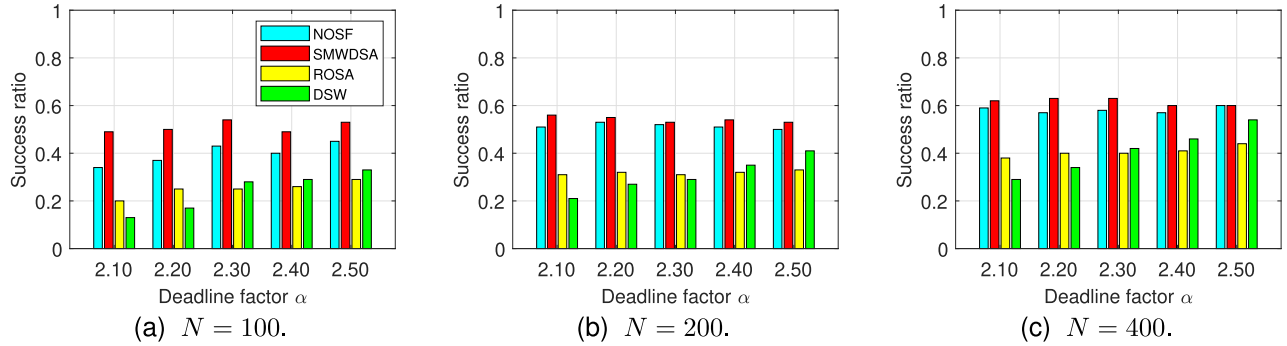


Fig. 5. Success ratio of three algorithms with different deadline factors α.

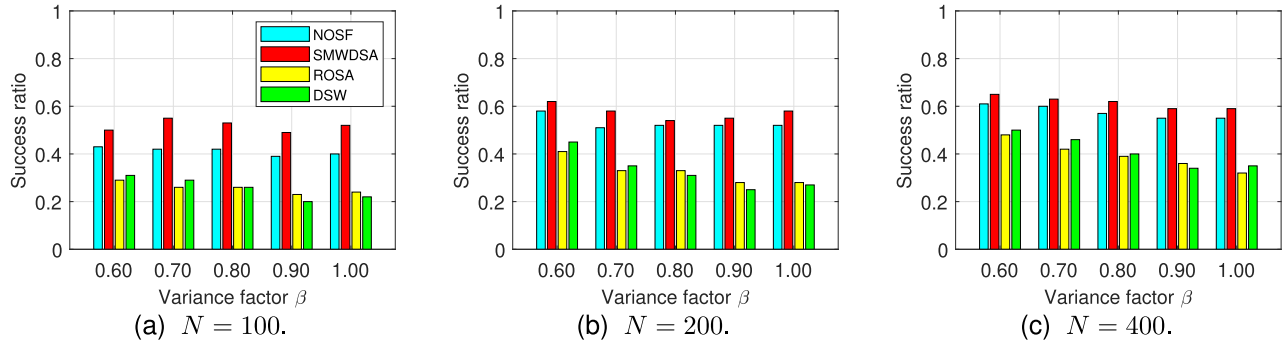


Fig. 6. Success ratio of three algorithms with different variance factors β.

than the cost of the other three algorithms with the deadline factor α and the variance factor β increasing. In Fig. 3, we set the variance factor β as 0.8 and show the total cost by three algorithms with different workflow number N. Fig. 3 illustrates the results of the total cost by four algorithms with varying deadline factors. It slightly decreases when the deadline factor increase. The reason is that when the deadline factor increase, the task are more likely to be assigned to a cheaper VM with meeting its subdeadline. In Fig. 4, we set the deadline factor β as 2.3 and show the total cost obtained by three algorithms with different workflow number N and variance factors. According to Fig. 4, the total cost obtained by SMWDSA are always the best compared with the other three algorithms when changing variance factors. Moreover, the total cost of three algorithms increase when the variance factor increase. This can be explained as follows: When the variance factor increases, the disturbance of task execution time increases, which leads to more cost. Furthermore, Fig. 3(a) and Fig. 4(a) show the total cost for scheduling 100 workflows with three algorithms under different deadline factors and variance factors. The total cost for scheduling 200 workflows with three algorithms under different deadline factors and variance factors are shown in Fig. 3(b) and Fig. 4(b). For 400 workflows, the total cost obtained by three algorithms are shown in Fig. 3(c) and Fig. 4(c). It can be see that SMWDSA outperforms he other three algorithms in all cases.

2) Success Ratio for Scheduling Multi-Workflows: In this subsection, we design the experiments with different deadline factors and variance factors for evaluating the success ratio of our algorithm and the results are shown in Fig. 5 and Fig. 6. In Fig. 5, we set the variance factor β as 0.8 and show results of the success ratio by increasing deadline factors in different workflow number N. As shown in Fig. 5(a), we can see that the success ratio of scheduling 100 workflows are increasing with deadline factors increasing. Fig. 5(b) and Fig. 5(c) show the results of the success ratio with three algorithms for scheduling 200 and 400 workflows under different deadline factors, respectively. From Fig. 5, we can see that the success ratio of SMWDSA can achieve 0.5 when the deadline factor is 0.6 and has a better performance than the other three algorithms when the deadline factor is increasing. In Fig. 6, we set the deadline factor α as 2.3 and show the results of the success ratio by three algorithms with different workflow number N. When the variance factor rises, a decreasing trend is observed in Fig. 6, because the larger variance factor is, the lower the prediction accuracy of task execution time is, which leads to a lower success ratio. From Fig. 6, we can see that SMWDSA outperforms the other three algorithms in all cases.

3) Number of VMs for Scheduling Multi-Workflows: In Fig. 7, we set the variance factor β as 0.8 and show the number of VMs via three methods. The proposed algorithm and other two algorithms show their downward trend of number of VMs when deadline factors become larger. The former has obvious advantage than the latter. In addition, from Fig. 7(a) to Fig. 7(c), we can see that three algorithms show their upward trend of number of VMs when workflow number become larger and the number of VMs by SMWDSA is always less than that of the other three algorithms with different deadline factor α. In Fig. 8, we set the deadline factor β as 2.3 and show the results of the number of VMs under different

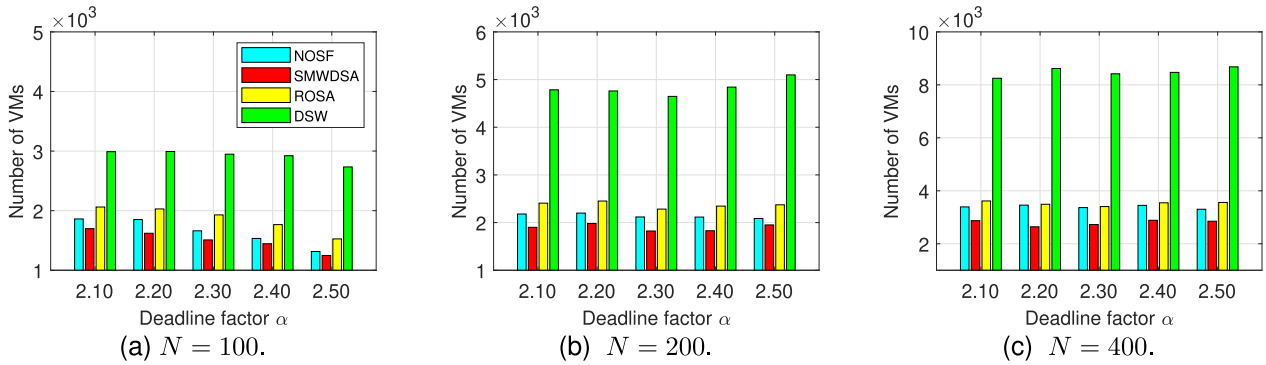


Fig. 7. The number of VMs of three algorithms with different deadline factors α.

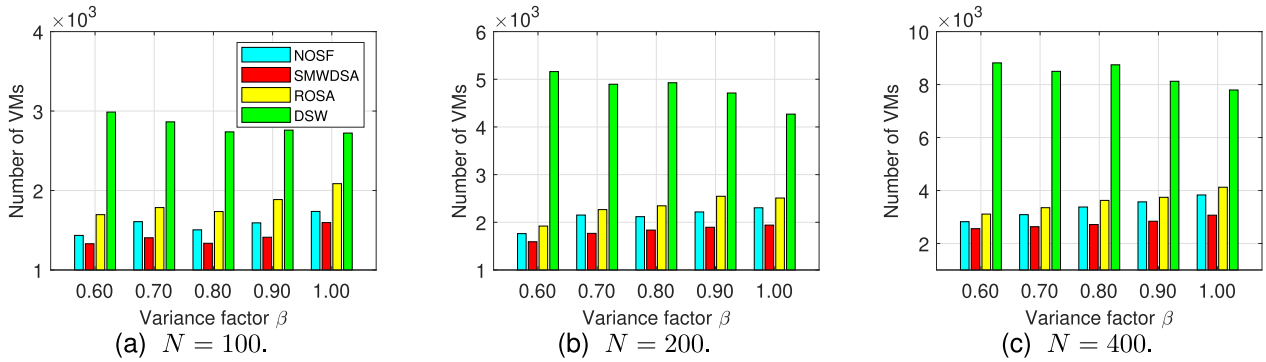


Fig. 8. The number of VMs of three algorithms with different variance factors β.

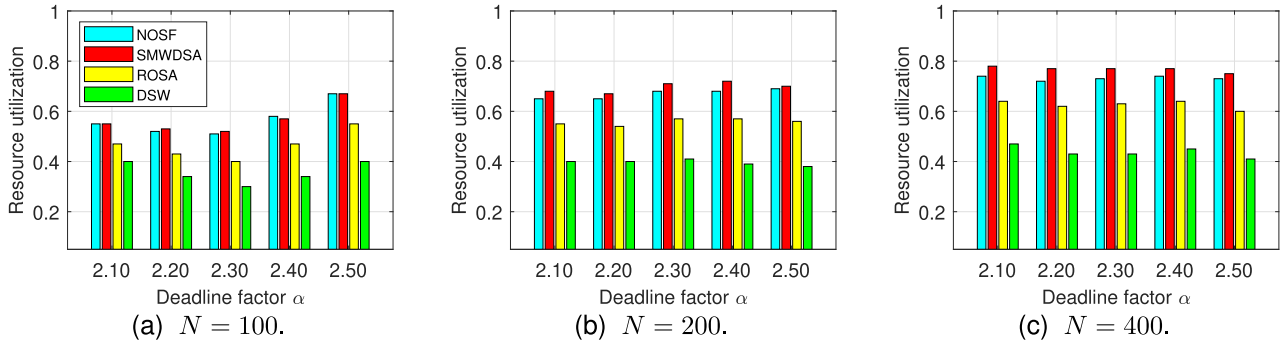
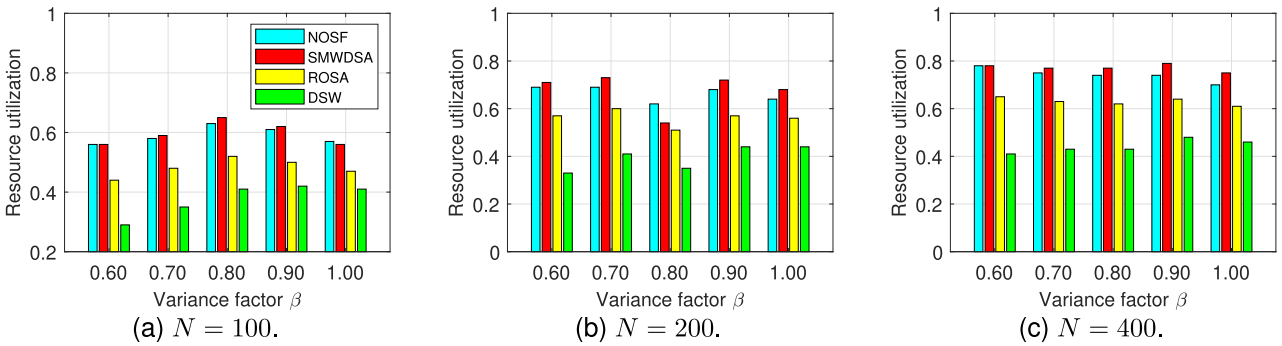


Fig. 9. Resource utilization of three algorithms with different deadline factors α.

variance factors. From Fig. 8, we can see that, as variance factor increases, number of VMs by SMWDSA and other two algorithms increase too. The difference between them is that the proposed algorithm has a significantly smaller number of VMs for increasing variance factor. From Fig. 8(a) to Fig. 8(c), the number of VMs by three algorithms become larger and larger with the increasing workflow number and the number of VMs of SMWDSA are always the lowest compared with the other three algorithms when changing variance factors. According to Fig. 7 and Fig. 8, the experimental results show that SMWDSA can reduce the number of VMs for multi-workflows scheduling in comparison with the other three algorithms.

4) Resource Utilization for Scheduling Multi-Workflows: In this subsection, we design the experiments with different deadline factors and variance factors for evaluating the resource utilization of our algorithm and the results are shown in Fig. 9 and Fig. 10. We set the variance factor β as 0.8 in Fig. 9 which shows the impact of deadline factor on resource utilization. As shown in Fig. 9(a), we can observe that the resource utilization of scheduling 100 workflows slowly increase as the deadline factor increase. Similar to Fig. 9(a), Fig. 9(b) and Fig. 9(c) show the resource utilization of scheduling 200 workflows and 400 workflows under different deadline factors, respectively. From Fig. 9, we can see that the resource utilization of SMWDSA is increasing and has a better performance than he other three algorithms when the workfolow number is increasing. In Fig. 10, we set the deadline factor α as 2.3. Fig. 10 shows the impact of variance factor on resource utilization by three algorithms. From Fig. 10(a), we can see that the proposed and other two methods show their downward trend of resource utilization of scheduling 100 wokflows when variance factor increases. The former has obvious advantage than the latter for different



(a) $N = 100$. (b) $N = 200$. (c) $N = 400$.

Fig. 10. Resource utilization of three algorithms with different variance factors β.

variance factors. Furthermore, Fig. 10(b) and Fig. 10(c) show the resource utilization for scheduling 200 workflows and 400 workflows with three algorithms under different variance factors, respectively. From the above results, we can see that SMWDSA outperforms the other three algorithms.

VI. Conclusion and Future Work

To reduce the impact of VM performance fluctuations and heterogeneity in workflow scheduling in the cloud, this paper proposes a stochastic multi-workflows dynamic scheduling algorithm (SMWDSA). Due to the uncertainty of workflow arrival time, SMWDSA dynamically allocates tasks and divides the scheduling process into three stages which consists of three stages which are multi-workflows preprocessing, multi-workflows scheduling and scheduling feedback. Firstly, we compute the task execution time and obtains the task sub-deadlines constraints without violating workflow deadline constraints in the multi-workflows preprocess stage. Then, we obtain a ready task priority queue from multiple workflows and schedules tasks to the VM with minimal task scheduling cost in the multi-workflows scheduling stage. Finally, we update the original priorities and sub-deadlines of unscheduled tasks to reduce workflows scheduling cost in the scheduling feedback stage. Extensive experiments results demonstrate that SMWDSA is superior to the state-of-the-art algorithms in terms of total toatl cost, resource utilization, success rate and VM number under different conditions. Additionally, we will do the empirical study to make the evaluation part more comprehensive.

We discuss some possible developments of our research work which may inspire future research work. In the future, we will further optimize the performance of the algorithm and consider the scheduling strategy when task execution fails due to hardware faults. In addition, our proposed algorithm focuses on solve the multi-workflows dynamic scheduling problem with a cloud computing system. With the development of cloud computing, the multi-cloud computing system collaborative scheduling has become a development trend. We plan to study the online multi-workflows scheduling in the multi-cloud computing system in the future work.

References

[1] X. Li and Z. Cai, "Elastic resource provisioning for cloud workflow applications," *IEEE Trans. Automat. Sci. Eng.*, vol. 14, no. 2, pp. 1195–1210, Apr. 2017.

[2] M. Adhikari and T. Amgoth, "A survey on scheduling strategies for workflows in cloud environment and emerging trends," *ACM Comput. Surv.*, vol. 52, no. 4, p. 68, 2019.

[3] H. Yan *et al.*, "Cost-efficient consolidating service for Aliyun's cloud-scale computing," *IEEE Trans. Services Comput.*, vol. 12, no. 1, pp. 117–130, Jan./Feb. 2019.

[4] L. Yin, J. Lou, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4712–4721, Oct. 2018.

[5] Q. Wu, F. Ishikawa, Q. Zhu, Y. Xia, and J. Wen, "Deadline-constrained cost optimization approaches for workflow scheduling in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 12, pp. 3401–3412, Aug. 2017.

[6] G. Xie *et al.*, "Minimizing redundancy to satisfy reliability requirement for a parallel application on heterogeneous service-oriented systems," *IEEE Trans. Services Comput.*, vol. 13, no. 5, pp. 871–886, Sep. 2020.

[7] G. Xie, Y. Chen, Y. Liu, and Y. Wei, "Resource consumption cost minimization of reliable parallel applications on heterogeneous embedded systems," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1629–1640, Aug. 2017.

[8] V. Arabnejad, K. Bubendorfer, and B. Ng, "Scheduling deadline constrained scientific workflows on dynamically provisioned cloud resources," *Future Gener. Comput. Syst.*, vol. 75, pp. 348–364, Oct. 2017.

[9] Z.-G. Chen *et al.*, "Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 2912–2926, Aug. 2019.

[10] A. Choudhary, I. Gupta, V. Singh, and P. K. Jana, "A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing," *Future Gener. Comput. Syst.*, vol. 83, pp. 14–26, Jun. 2018.

[11] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proc. ACM Special Interest Group Data Commun.*, pp. 270–288, 2019.

[12] Z. Zhu and X. Tang, "Deadline-constrained workflow scheduling in IaaS clouds with multi-resource packing," *Future Gener. Comput. Syst.*, vol. 101, pp. 880–893, Dec. 2019.

[13] H. Yuan, J. Bi, and M. Zhou, "Profit-sensitive spatial scheduling of multi-application tasks in distributed green clouds," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 3, pp. 1097–1106, Jul. 2020.

[14] 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.

[15] D. Bruneo, "A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 560–569, Mar. 2014.

[16] 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. 2014.

[17] J. Chen, K. Li, H. Rong, K. Bilal, K. Li, and P. S. Yu, "A periodicity-based parallel time series prediction algorithm in cloud computing environments," *Inf. Sci.*, vol. 496, pp. 506–537, Sep. 2019.

[18] 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.

[19] G. Khojasteh Toussi, M. Naghibzadeh, S. Abrishami, H. Taheri, and H. Abrishami, "EDQWS: An enhanced divide and conquer algorithm for workflow scheduling in cloud," *J. Cloud Comput.*, vol. 11, no. 1, pp. 1–18, Dec. 2022.

[20] 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.
[21] V. Singh, I. Gupta, and P. K. Jana, "A novel cost-efficient approach for deadline-constrained workflow scheduling by dynamic provisioning of resources," *Future Gener. Comput. Syst.*, vol. 79, pp. 95–110, Feb. 2018.
[22] J. Sahni and D. P. Vidyarthi, "A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 2–18, Mar. 2015.
[23] 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.
[24] P. Han, C. Du, J. Chen, F. Ling, and X. Du, "Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique," *J. Syst. Archit.*, vol. 112, Jan. 2021, Art. no. 101837.
[25] J. J. Durillo and R. Prodan, "Multi-objective workflow scheduling in Amazon EC2," *Cluster Comput.*, vol. 17, no. 2, pp. 169–189, Jun. 2014.
[26] Q. Wu, M. Zhou, Q. Zhu, Y. Xia, and J. Wen, "MOELS: Multiobjective evolutionary list scheduling for cloud workflows," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 1, pp. 166–176, Jan. 2020.
[27] K. Psychas and J. Ghaderi, "Scheduling jobs with random resource requirements in computing clusters," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 2269–2277.
[28] 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 Gener. Comput. Syst.*, vol. 71, pp. 57–72, Jan. 2017.
[29] H. Chen, J. Zhu, Z. Zhang, M. Ma, and X. Shen, "Real-time workflows oriented online scheduling in uncertain cloud environment," *J. Supercomput.*, vol. 73, no. 11, pp. 4906–4922, Nov. 2017.
[30] H. Chen, X. Zhu, 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, doi: 10.1109/TSC.2018.2866421.
[31] J. Liu *et al.*, "Online multi-workflow scheduling under uncertain task execution time in IaaS clouds," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 1180–1194, Jul. 2021, doi: 10.1109/TCC.2019.2906300.
[32] V. Arabnejad, K. Bubendorfer, and B. Ng, "Dynamic multi-workflow scheduling: A deadline and cost-aware approach for commercial clouds," *Future Gener. Comput. Syst.*, vol. 100, pp. 98–108, Nov. 2019.
[33] H. Gu, X. Li, M. Liu, and S. Wang, "Scheduling method with adaptive learning for microservice workflows with hybrid resource provisioning," *Int. J. Mach. Learn. Cybern.*, vol. 12, pp. 3037–3048, Aug. 2021.
[34] X. Ma, H. Xu, H. Gao, and M. Bian, "Real-time multiple-workflow scheduling in cloud environments," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 4, pp. 4002–4018, Dec. 2021.
[35] T. Dong, F. Xue, C. Xiao, and J. Zhang, "Deep reinforcement learning for dynamic workflow scheduling in cloud environment," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Sep. 2021, pp. 1–9.
[36] Z. Tong, H. Chen, X. Deng, K. Li, and K. Li, "A scheduling scheme in the cloud computing environment using deep Q-learning," *Inf. Sci.*, vol. 512, pp. 1170–1191, Feb. 2020.
[37] P. M. Kebria, A. Khosravi, S. M. Salaken, and S. Nahavandi, "Deep imitation learning for autonomous vehicles based on convolutional neural networks," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 1, pp. 82–95, Jan. 2020.
[38] X. Li, L. Qian, and R. Ruiz, "Cloud workflow scheduling with deadlines and time slot availability," *IEEE Trans. Services Comput.*, vol. 11, no. 2, pp. 329–340, Mar. 2018.
[39] L. Ye, Y. Xia, L. Yang, and C. Yan, "SHWS: Stochastic hybrid workflows dynamic scheduling in cloud container services," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 3, pp. 2620–2636, Jul. 2022, doi: 10.1109/TASE.2021.3093341.
[40] X. Tang, K. Li, G. Liao, K. Fang, and F. Wu, "A stochastic scheduling algorithm for precedence constrained tasks on grid," *Future Gener. Comput. Syst.*, vol. 27, no. 8, pp. 1083–1091, Oct. 2011.
[41] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *Proc. IEEE 8th Int. Conf. E-Sci.*, Oct. 2012, pp. 1–8.
[42] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Gener. Comput. Syst.*, vol. 29, no. 3, pp. 682–692, 2013.

Lingjuan Ye received the B.S. degree from Nanchang University, Nanchang, China, in 2016. She is currently pursuing the Ph.D. degree with the School of Mathematics and Statistics, Beijing Institute of Technology, Beijing, China. Her current research interests include cloud computing, workflow scheduling, and reinforcement learning.

Yuanqing Xia (Senior Member, IEEE) received the M.S. degree in fundamental mathematics from Anhui University, Hefei, China, in 1998, and the Ph.D. degree in control theory and control engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001. He was a Post-Doctoral Research Associate at the Institute of Systems Science, Academy of Mathematics and Systems Science, and the Chinese Academy of Sciences, Beijing, from 2002 to 2003. From 2004 to 2006, he was a Research Fellow at the University of Glamorgan, Pontypridd, U.K. From 2007 to 2008, he was a Guest Professor at Innsbruck Medical University, Innsbruck, Austria. He is currently a Professor with the School of Automation, Beijing Institute of Technology, Beijing. His research interests include cloud control systems, networked control systems, robust control and signal processing, active disturbance rejection control, unmanned system control, and flight control.

Liwen Yang received the B.S. degree in automation from Yantai University, Yantai, China, in 2014, and the M.S. degree from Northeastern University, Shenyang, China, in 2018. She is currently pursuing the Ph.D. degree with the School of Automation, Beijing Institute of Technology, Beijing, China. Her current research interests include multi-objective optimization and cloud workflow scheduling.

Yufeng Zhan received the Ph.D. degree from the Beijing Institute of Technology (BIT), Beijing, China, in 2018. He was a Post-Doctoral Fellow at the Department of Computing, The Hong Kong Polytechnic University. He is currently an Assistant Professor with the School of Automation, BIT. His research interests include networking systems, game theory, and machine learning.