The Journal of Systems & Software 184 (2022) 111123

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

A stochastic algorithm for scheduling bag-of-tasks applications on hybrid clouds under task duration variations☆,☆☆

Lu Yin, Junlong Zhou, Jin Sun *

School of Computer Science and Engineering, Nanjing University of Science and Technology, 200 Xiaolingwei Street, Nanjing 210094, China

ARTICLE INFO

Article history:
Received 12 September 2021
Accepted 11 October 2021
Available online 21 October 2021

Keywords:
Bag-of-tasks applications
Hybrid clouds
Stochastic task scheduling
Probabilistic constraint
Profit maximization

ABSTRACT

Hybrid cloud computing, which typically involves a hybrid architecture of public and private clouds, is an ideal platform for executing bag-of-tasks (BoT) applications. Most existing BoT scheduling algorithms ignore the uncertainty of task execution times in practical scenarios and schedule tasks by assuming that the task durations can be determined accurately in advance, often leading to the violation of the deadline constraint. In view of this fact, this paper devotes to maximizing the profit of the private cloud provider while guaranteeing the quality-of-service provided by the cloud platform, through designing an effective stochastic BoT scheduling algorithm based on the distribution of task duration variations. With the varying task execution times modeled as random variables, we formulate a stochastic scheduling framework that incorporates a probabilistic constraint upon the makespans of BoT applications. The resultant stochastic optimization model is capable of characterizing the complete distribution information of makespan variations and satisfying the deadline constraint in a probabilistic sense. We further design an immune algorithm-based metaheuristic to solve this stochastic optimization problem. Simulations results justify that our proposed algorithm outperforms several competing algorithms in maximizing the cloud provider's profit while satisfying the user-specified deadline constraint under the impact of uncertain task durations.

1. Introduction

As a popular service-based paradigm, cloud computing can provide large-scale computational resources to customers in a pay-as-you-go fashion. As the technical foundation of cloud computing, virtualization technology enables cloud providers to provide multiple types of virtual machines (VMs). Each VM is deployed with different amounts of computing resources with different prices. From the perspective of cloud providers, the tasks submitted need to be performed on the types of VMs specified by customers, and charged to the customers based on the usage of the VM instances. Due to the promised quality-of-service (QoS) described in service-level agreements (SLAs) (Cao et al., 2018), when the resources of the private cloud cannot afford the workload of the tasks submitted by customers, some tasks have to be outsourced to public clouds. Based on the hybrid cloud construction solution that administrators and workloads of the private cloud can use public cloud resources through unified tools or interfaces, the private cloud and its public cloud extensions can use the same virtualization technique to ensure that the same types of VM instances in the public extensions (Zhang and Sun, 2017; Zhang et al., 2019, 2017).

A bag-of-tasks (BoT) application in general consists of a set of independent tasks, each of which can be processed in parallel without synchronization and communication (Hu and Veeravalli, 2013). As a common type of workload, BoT applications are widely spreading in a variety of fields, such as Monte Carlo simulations, data mining algorithms, and parameter sweep applications (Long et al., 2018; Varshney and Simmhan, 2019). Applications in these fields generally consist of many atomic processes that can be performed in parallel, which can be considered as independent tasks in BoT applications. For example, a Monte-Carlo simulation program needs to be repeated hundreds of times independently with numerous input values to obtain the distribution of simulation results. Since the execution of BoT applications often requires expensive investments in computing infrastructure, and cloud computing delivers large-scale computing resources in a pay-as-you-go fashion, hybrid cloud computing becomes an ideal and economical platform for processing BoT applications (Gutierrez-Garcia and Sim, 2015).

☆ Editor: J.C. Duenas.

☆☆ This work was supported in part by the National Natural Science Foundation of China under Grants 61872185 and 61802185 and in part by the Fundamental Research Funds for the Central Universities under Grants 30919011233, 30919011402, and 30920021132.

* Corresponding author.
E-mail addresses: ylu@njust.edu.cn (L. Yin), jlzhou@njust.edu.cn (J. Zhou), sunj@njust.edu.cn (J. Sun).

https://doi.org/10.1016/j.jss.2021.111123
0164-1212/

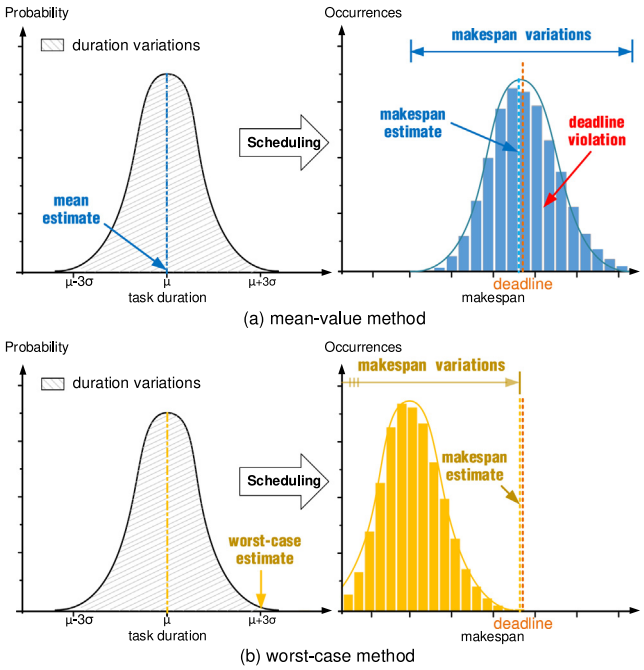

Fig. 1. The limitations mean-case and worst-case methods under the impact of task duration variations.

Motivation. Scheduling is an important issue in cloud computing since an appropriate solution of task scheduling would have a direct impact on the QoS promised by cloud providers. In the literature, there exist a tremendous number of approaches for solving scheduling problems in cloud computing environments (Dong et al., 2019; He et al., 2021; Houssein et al., 2021; Alam and Raza, 2018). In general, the scheduling approaches can be classified into two categories according to their application scenarios. Static scheduling assumes task characteristics are known in advance and seeks for an offline schedule of task execution (Keshanchi et al., 2017). On the contrary, dynamic scheduling performs online when real-time tasks arrive (Zheng et al., 2021a; Wu et al., 2017). This work studies a static scheduling problem that assigns the tasks of BoT applications to VMs provided by hybrid clouds.

Most existing static approaches assume that the task execution time is simply a pre-determined fixed value (Zhang et al., 2015; Oprescu and Kielmann, 2010; Long et al., 2015). However, in practical scenarios, it is challenging and even unrealistic to know the completely accurate execution time of the task before the task is completed. In addition to the inherent uncertainty of program execution time, this limitation mainly stems from the following two reasons. On the one hand, task programs generally involve conditional instructions and loop operations. Affected by the conditional instructions, different inputs lead to different program branches and different number of loops, resulting in different amounts of computations of the task program. Therefore, the task duration may vary under different inputs (Li et al., 2013b; Lee and Zomaya, 2010). On the other hand, the processing speeds of VM instances in cloud platforms may fluctuate over time, due to the fact that there are multiple VMs deployed on the same physical server and these VMs use the server's computing and storage resources synchronously. The fluctuation of VM performance in turn results in the uncertainty of task execution time (Chen et al., 2015; Tang et al., 2011; Cao et al., 2020).

This paper concentrates on the BoT scheduling problem that maximizes the private cloud provider's profit under user-specified deadline constraints. The profit is defined as the private cloud provider's income (paid by users submitting their BoT applications) minus public cloud usage cost (for outsourcing certain tasks to public clouds in case of insufficient computing resources). The scheduling problem imposes a user-specified deadline constraint upon the makespan of BoT applications as well as the resource constraints for the VMs on the private cloud. Under the impact of task duration variations, existing task scheduling algorithms, in which task execution times are regarded as fixed values (e.g., mean values or worst-case estimates), tend to cause violation of deadline constraint or over-provisioning of public cloud resources. To illustrate the limitations of deterministic methods, we conduct experiments on a specific BoT application with normally distributed task execution times using mean-value and worst-case methods. The mean-value method uses the mean values of random variables as task duration estimates, and the worst-case method takes the maximum values of random variables as task duration estimates. Accordingly, we can obtain the scheduling solutions generated based upon the corresponding deterministic task durations. Affected by the uncertainty of task durations, however, the makespan of the BoT application also exhibits a stochastic behavior. On the one hand, as revealed in Fig. 1(a), the mean-value estimates would lead to an optimistic estimation of the application's makespan that is close to the deadline value. The makespan variations could cause severe deadline violation under the uncertainty of task durations. On the other hand, Fig. 1(b) indicates that the worst-case method results in a conservative makespan estimate by assuming that all makespan variations are shorter than the deadline value. This conservative scheduling solution, which is obtained by using worst-case task duration estimates, would outsource more tasks to public clouds and incur extra costs to guarantee the satisfaction of the deadline constraint.

To overcome the limitations of deterministic methods, in this work we assume the task execution time is a random variable subject to a normal distribution (Li et al., 2013a; Sarin et al., 2010; Gu et al., 2009). Under this assumption of uncertain task durations, the makespans of BoT applications also become random variables according to probability theory. As the fundamental part of our problem formulation, we introduce a probabilistic constraint on the makespan of each BoT application. The purpose of this constraint is to guarantee that the makespan variations caused by uncertain task durations can meet the deadline with a certain probability. Distinct from the deterministic methods in which task execution times are modeled as fixed values, our scheduling framework characterizes the complete distribution information of makespan variations by incorporating the probabilistic constraint. To solve the resultant stochastic optimization problem, we propose an effective metaheuristic called Immune Algorithm-based BoT Scheduling (IABS). In this metaheuristic, each individual antibody is considered as a scheduling solution. We develop a task assignment strategy to link each antibody with a feasible solution that satisfies the probabilistic deadline constraint under the uncertainty of task execution times. Furthermore, we modify the antibody updating scheme in the standard immune algorithm (IA) to enhance the capability of our IABS in solving the stochastic BoT scheduling problem.

Contributions. The main contributions of this paper are summarized as follows.

- We propose a stochastic BoT scheduling framework for hybrid clouds that models the varying task durations as random variables and incorporates a probabilistic deadline constraint upon the makespan of BoT applications.
- We develop an uncertainty-aware task assignment strategy that relies on an analytical approach to evaluate the feasibility of a candidate solution satisfying the probabilistic constraint.

- We design an effective metaheuristic IABS to solve the stochastic scheduling problem by employing a solution initialization rule and a novel antibody updating operator.

We create a variety of testing instances by profiling real-world BoT applications and run extensive simulation experiments to evaluate the efficacy of the proposed IABS algorithm. Evaluation results demonstrate that compared with competing scheduling algorithms, our proposed IABS increases the private cloud provider's profit by up to 21%.

The reminder of this paper is organized as follows. Section 2 discusses existing scheduling approaches that are related to this work. Section 3 presents the problem description and formulates the stochastic BoT scheduling model. Section 4 discusses how to estimate the expected value and variance of each application's makespan under uncertain task durations. Section 5 details our proposed algorithm. Experimental results and concluding remarks are provided in Sections 6 and 7 , respectively.

2. Related work

In recent years, a considerable number of research efforts have been made in solving BoT scheduling problems in cloud computing. In the literature, a number of approaches (Wang et al., 2016; Pelaez et al., 2016; Abdi et al., 2017) were oriented toward addressing deadline-constrained BoT scheduling problem with an objective of cost minimization, whereas another category of approaches (Maurya and Tripathi, 2018; Zhang et al., 2019; Cao et al., 2016) aim at minimizing the makespan of BoT applications while satisfying deadline or budget constraints. In addition, multi-objective optimization techniques are also commonly used in developing BoT scheduling algorithms (Sindhu and Mukherjee, 2018; Tarplee et al., 2015; Duan et al., 2014). Despite extensive efforts that have been made in this research direction, the aforementioned methods are based on the assumption that the task execution times are deterministic values. However, in practical scenarios, actual task execution times cannot be accurately predicted due to the uncertainty of virtual machine performance and complex task properties (Malawski et al., 2015). The deterministic scheduling algorithms are not applicable for the task scheduling problem with stochastic tasks. In what follows, we lay emphasis on discussing the approaches for solving uncertainty-aware scheduling problems.

A number of scheduling algorithms rely on task execution status to determine an accurate estimation of task duration (Zheng et al., 2021b; Islam et al., 2020). For instance, Cai et al. (2017) proposed a cloud resource management and scheduling algorithm for BoT applications with varying execution times. In order to reduce the resource rental cost, the decision making of VM renting takes place whenever a batch of tasks in a BoT application is ready for execution. Chen et al. (2016) designed an uncertainty-aware scheduling algorithm for real-time workflows to reduce the uncertainty propagated during the process of task allocation. The same authors (Chen et al., 2018) further extended their work by taking the uncertainty of data transfer time into consideration. Yan et al. (2019) proposed a runtime estimation approach and designed a DEFT algorithm for scheduling real-time tasks in the cloud environment considering system performance volatility. The main drawback of these algorithms is their limited scheduling performance due to myopic optimization and inaccurate estimation of the uncertainty of task durations (Malawski et al., 2015).

There also exist many approaches that characterize task execution times by using measuring or profiling tools to tolerate the uncertainty in real-time task execution (Chen et al., 2020). Doğan and özgüner (2004) revealed that scheduling algorithms maximizing the expected value of the scheduling objective may produce inefficient schedules where task execution times are subject to variations. Novak et al. (2019) investigated a makespan-minimization scheduling problem with uncertain task execution times in mixed-criticality systems. An F-shape model is utilized to overcome such uncertainty, in which each task contains a set of alternative execution times. Jia et al. (2021) developed an estimation model of task execution time according to virtual machine settings and historical data from practical workflows. The authors further proposed an adaptive ant colony optimization algorithm based on the estimation model to schedule workflow applications under deadline constraints. To schedule precedence-constrained tasks with stochastic execution times, Tang et al. (2011) formulated a stochastic scheduling model and proposed a SHEFT algorithm to schedule stochastic tasks relying on the expected value and variance of execution times. The same authors (Tang et al., 2017) further proposed a time and cost multi-objective task scheduling genetic algorithm that determines the stochastic task finish times based on an online feedback mechanism. Haidri et al. (2019) developed a cost-effective deadline-aware resource scheduler, which is also a multi-objective method, for precedence-constrained stochastic tasks. Following the calculation method of stochastic task duration's approximate weight proposed by Tang et al. (2011) and Yang et al. (2015) designed an energy-aware stochastic task scheduling algorithm. This algorithm introduces a metric of importance-ratio of makespan to energy and improves the scheduling quality by optimizing this metric. The essential idea of those methods is to reduce the uncertainty-aware scheduling problems to easy-to-solve deterministic optimization problems. A distinguishing property of our work is that, by incorporating the probabilistic deadline constraint, our scheduling framework of stochastic maintains the essence of stochastic optimization by taking into account the complete distribution information of makespan variations.

It is worth mentioning the uncertainty-aware scheduling algorithm for solving the BoT scheduling problem with stochastic tasks in heterogeneous computing systems (Li et al., 2013a). The scheduling objectives include the BoT application's makespan and energy consumption. A stochastic task scheduling heuristic is proposed to optimize the weighted probability of these two metrics. However, this approach is developed specifically for scheduling a single BoT application, and therefore cannot cope with the scenario of scheduling multiple BoT applications as in our work. In addition, this approach is oriented toward a heterogeneous computing environment, whereas our scheduling problem for hybrid clouds needs to take into account the cost for renting resources of public clouds, leading to a more complicated optimization model.

3. Problem description

This section first describes the models of BoT applications and hybrid cloud environment, and then formulates the stochastic task scheduling problem as an integer programming problem. For ease of illustration, all notations used in problem description and their definitions are provided in Table 1.

3.1. Hybrid clouds model

Hybrid clouds refer to the cloud infrastructure environment that is a combination of both public and private computing resources (Genez et al., 2019). We assume that CP denotes the hybrid cloud environment consisting of $m+1$ cloud providers, in which CP_0 is the private cloud provider and $CP_1, CP_2, \ldots, CP_m$ are public cloud providers. CP_0 can provide k types of VM instance represented by $VM_1, VM_2, \ldots, VM_k$. According to hybrid cloud construction solutions, public clouds can create an extension of

Table 1
Notations for problem formulation.

Notation	Definition	Notation	Definition
CP_0	The private cloud provider	CP_h	The hth public cloud provider
m	Total number of public cloud providers	VM_q	The qth VM type
k	Total number of VM types	CPU_q	The number of CPUs on VM_q
Mem_q	The amount of memory on VM_q	p_{qh}	The unit price of VM_q provided by CP_h
$CPU_{\max}$	Available CPUs on CP_0	$Mem_{\max}$	Available memory on CP_0
$cost_q$	The unit cost of VM_q instance on CP_0	n	The total number of BoT applications
a_i	The ith BoT application	T_i	The number of tasks in application a_i
d_i	The deadline for application a_i	x_i	The index of VM type required by a_i
t_{ij}	The jth task of a_i	$\tilde{et}_{ij}$	The execution time of t_{ij} on VM_{x_i}
μ_{ij}	The expected value of t_{ij}	σ_{ij}^2	The variance of t_{ij}
$\tilde{C}_i$	The makespan of application a_i	$\tilde{eft}_{ij}$	The estimated finish time of t_{ij}

the private cloud by using the same virtualization technique. In other words, public cloud providers can provide the same VM types as the private cloud can. For this reason, we assume that the m public clouds also provide the k VM types. Different VM types are associated with various configurations and prices. For each VM type VM_q, cpu_q, and mem_q represent the number of CPUs and the amount of memory it requires, respectively. When outsourcing tasks to public clouds, p_{qh} denotes the price for creating and using the instance of VM_q provided by CP_h per time unit. Following the model in Zhang and Sun (2017), we set the time unit as an hour. From the private cloud provider's perspective, the use of a VM instance is not free due to the costs of power consumption, cooling, etc. We use $cost_q$ to represent the cost of the private cloud providing an instance of VM_q per time unit. Obviously, the $cost_q$ of private cloud provider using its own virtual machine instances should be less than p_{qh} which is the fee charged by the public cloud providers, i.e., $cost_q < p_{qh}$.

3.2. BoT application model

In this paper, users submit BoT applications to a private cloud provider, where the tasks of each BoT application are independent and atomic. Assuming that there are n BoT applications in finite set App that need to be scheduled, the set App can be denoted as $App = \{a_1, a_2, \ldots, a_n\}$. For each BoT application, we represent a_i as $a_i = \langle x_i, d_i, T_i \rangle$, where x_i is the index of user-specified VM type required by a_i, d_i is the deadline for a_i, and T_i is the number of tasks in a_i. Accordingly, the BoT application a_i can be denoted as $a_i = \{t_{i1}, t_{i2}, \ldots, t_{iT_i}\}$.

Different from deterministic scheduling methods which assume that task execution times are fixed and known in advance, in this work, the varying task execution time is modeled as a random variable characterized by its probability distribution. Following Li et al. (2013a), Ando et al. (2009) and Möhring et al. (1999), for a specific task t_{ij}, we assume its execution time is subject to a normal distribution, i.e., $\tilde{et}_{ij} \sim N(\mu_{ij}, \sigma_{ij}^2)$ where μ_{ij} and σ_{ij}^2 denote the expected value and variance of execution time, respectively.

3.3. BoT scheduling model

To solve the BoT scheduling problem with stochastic tasks, we propose a scheduling framework with profit optimization on hybrid clouds, as shown in Fig. 2. From the perspective of the private cloud provider, the proposed framework can be divided into four parts: users, BoT applications, task scheduler, and hybrid cloud computing resources. As described previously, the scheduling problem is to schedule the tasks of BoT applications submitted by users (Wu et al., 2021). Each BoT application contains multiple independent tasks with varying task execution times. The hybrid clouds, including the private and public clouds, can provide various types of VM instance to process these BoT applications. The public cloud can provide the same types of VM instances as the private cloud can by using virtualization technology. As the most important part of the framework, the task scheduler will generate the profit-optimized scheduling result, while satisfying the QoS constraints and the limitations of private cloud computing resources. Moreover, since cloud providers deliver VM services in an on-demand manner, we assume that each application requires a user-specified VM type. In other words, users can select any VM type for executing a task among the available VM types provided by cloud providers. The VM instance is created when an application's task starts execution, and is released as long as the task finishes execution. Similar assumptions have been used in Zhang and Sun (2017), Zhang et al. (2019, 2017), Zuo et al. (2013) to solve BoT scheduling problems on hybrid clouds.

First, we define two sets of decision variables for problem formulation. The decision variable $y_{ij,h}$ indicates whether a task is scheduled to a cloud provider. To be specific, if t_{ij} is dispatched to CP_h, $y_{ij,h}$ would be 1. Otherwise, $y_{ij,h}$ equals 0. That is,

$$y_{ij,h} = \begin{cases} 1, & \text{if } t_{ij} \text{ is dispatched to } CP_h. \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The other decision variable we defined is $z_{ij,s}$ which is used to denote task t_{ij}'s execution time on the private cloud. As existing studies (Van den Bossche et al., 2011; Zuo et al., 2013), we regard the whole time frame as a time axis that can be divided into many time slots. Accordingly, if task t_{ij} is in execution at time slot s on the private cloud CP_0, $z_{ij,s}$ equals 1. Otherwise, $z_{ij,s}$ equals 0, i.e.,

$$z_{ij,s} = \begin{cases} 1, & \text{if } t_{ij} \text{ is in execution at } s \text{ on } CP_0 \\ 0, & \text{otherwise} \end{cases}. \tag{2}$$

Following the definition of $z_{ij,s}$ in Eq. (2), the start time of task t_{ij} can be represented by $st_{ij} = \arg\min\{s | z_{ij,s} = 1\}$. Due to the limited resources of the private cloud, we impose resource constraints for the usage of CPU and memory. For a specific task t_{ij}, all time slots within the range $[st_{ij}, st_{ij} + et_{ij}]$, where et_{ij} is the task execution time, are naturally identified as busy slots. Therefore, it is only necessary to ensure the satisfaction of resource constraints at each time point when a task starts its execution (i.e., $s \in \{st_{ij} | 1 \le i \le n,\ 1 \le j \le T_i\}$. Therefore, we have the following resource constraints:

$$\sum_{i=1}^{n} \sum_{j=1}^{T_i} z_{ij,s} CPU_{x_i} \le CPU_{\max},\ \forall s \in \{st_{ij} | 1 \le i \le n,\ 1 \le j \le T_i\}, \tag{3}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{T_i} z_{ij,s} MEM_{x_i} \le MEM_{\max},\ \forall s \in \{st_{ij} | 1 \le i \le n,\ 1 \le j \le T_i\}. \tag{4}$$

where $CPU_{\max}$ and $MEM_{\max}$ are the upper-bound limits for CPU and memory usages, respectively. Here, x_i denotes the index of

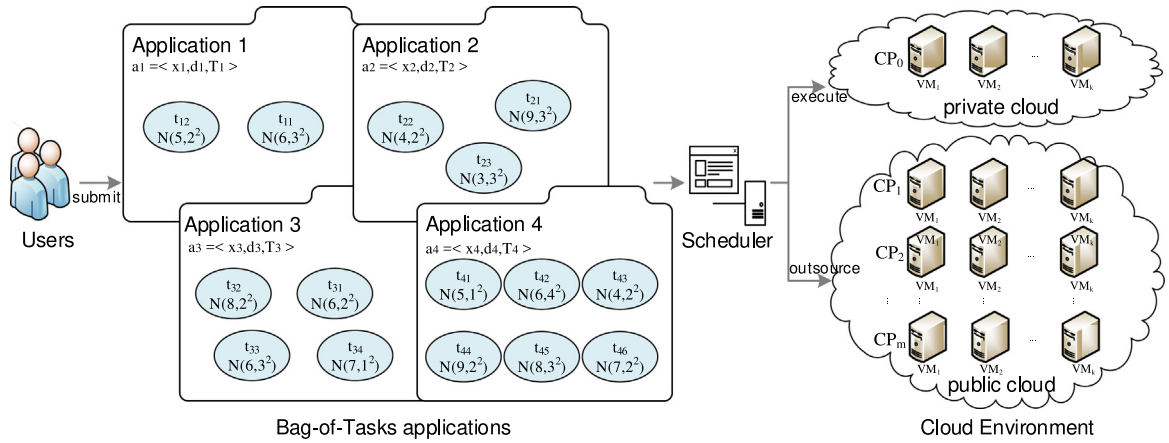

Fig. 2. The scheduling framework with profit optimization on a hybrid cloud environment.

the VM type required by the ith application's tasks. It is worth emphasizing that, to reduce the problem size, Eqs. (3) and (4) are required to hold for only a set of particular time slots rather than all time slots.

In addition to the resource constraints of the private cloud, there also exist deadline constraints from users upon the makespans of the BoT applications they submitted. To represent the deadline constraint, we first define eft_{ij} to describe the estimated finish time of t_{ij}, which is given by

$$\tilde{eft}_{ij} = \begin{cases} st_{ij} + \tilde{et}_{ij}, & \text{if } y_{ij,0} = 1 \\ \tilde{et}_{ij}, & \text{otherwise} \end{cases}. \tag{5}$$

Eq. (5) indicates that if t_{ij} is scheduled for execution on the private cloud, its estimated finish time $\tilde{eft}_{ij}$ is its start time st_{ij} plus its expected execution time $\tilde{et}_{ij}$. Similar to the scheduling models in Van den Bossche et al. (2011), Zuo et al. (2013), Zhang and Sun (2017), Zhang et al. (2019), if t_{ij} is dispatched to a public cloud, we set its start time to 0 due to the following two facts. First, public clouds are regarded as having infinite available computing resources. Second, the tasks' execution times in this work are evaluated in the granularity of hours. Currently, there are cloud providers that can deliver VM instances in minutes (e.g., Amazon EC2[1]) and even in seconds (e.g., QingCloud[2]). As such, the time overhead for VM instantiation is negligible compared with task execution times.

Since a BoT application contains a set of independent tasks, the estimated finish time of a BoT application, i.e., its makespan, is the maximum value of all tasks' finish times (Sun et al., 2020). In other words, we can define the estimated finish time of this BoT application a_i as

$$\tilde{\mathcal{C}}_i = \max_{j=1}^{T_i} \left\{ \tilde{eft}_{ij} \right\} \tag{6}$$

where $\tilde{\mathcal{C}}_i$ denotes the maximum value of multiple normal random variables. Let $E[\tilde{\mathcal{C}}_i]$ and $Var[\tilde{\mathcal{C}}_i]$ be the expected value and variance of $\tilde{\mathcal{C}}_i$, respectively. The probabilistic constraint upon application's makespan can be defined as

$$\text{Prob}\left\{\tilde{\mathcal{C}}_i < d_i\right\} \geq \alpha, \ \forall 0 \leq i \leq n \tag{7}$$

where α is a pre-specified threshold value for the probability of BoT applications meeting their deadline constraints. Since the task execution time is stochastic, it is unrealistic to guarantee that each application is completed before the corresponding deadline with a 100% probability. Therefore, the probabilistic constraint in Eq. (7) guarantees that each application's makespan satisfies the deadline requirement in a stochastic sense.

The scheduling objective is to maximize the profit of the private cloud provider, which is the difference between its income and cost and is given by

$$profit = income - cost \tag{8}$$

where $income$ is the monetary cost charged by the cloud provider for completing the task users submit, which can be expressed as

$$income = \sum_{i=1}^{n} \sum_{j=1}^{T_i} \mu_{ij} p_{x_i 0}. \tag{9}$$

The cost of the private cloud provider includes the cost of using the private cloud resources and the cost of outsourcing tasks to public clouds. Thus, we arrive at

$$cost = \sum_{i=1}^{n} \sum_{j=1}^{T_i} y_{ij,0} \mu_{ij} cost_{x_i} + \sum_{i=1}^{n} \sum_{j=1}^{T_i} \sum_{h=1}^{m} y_{ij,h} \mu_{ij} p_{x_i h} \tag{10}$$

where x_i is the index of VM type required by task t_{ij}. Accordingly, $p_{x_i h}$ $(q = 1, 2, \ldots, k;\ h = 0, 1, \ldots, m)$ represents the price (per time unit) for using an instance of VM_{x_i} provided by CP_h, and $cost_{x_i}$ represents the induced cost (per time unit) of a VM type VM_{x_i} provided by the private cloud.

Putting it all together, the BoT scheduling problem in this work can be formulated as the following optimization model:

$$\text{maximize} \quad profit = income - cost \tag{11}$$

$$\text{subject to} \quad Prob\{\tilde{\mathcal{C}}_i < d_i\} \geq \alpha \tag{12}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{T_i} z_{ij,s} CPU_{x_i} \leq CPU_{\max},$$

$$\forall s \in \{st_{ij} | 1 \leq i \leq n, 1 \leq j \leq T_i\} \tag{13}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{T_i} z_{ij,s} MEM_{x_i} \leq MEM_{\max},$$

$$\forall s \in \{st_{ij} | 1 \leq i \leq n, 1 \leq j \leq T_i\} \tag{14}$$

$$\sum_{h=0}^{m} y_{ij,h} = 1, \forall 1 \leq i \leq n, 1 \leq j \leq T_i \tag{15}$$

$$\text{variables} \quad y_{ij,h}, z_{ij,s} \tag{16}$$

In this model, Eq. (11) is the optimization objective, which is to maximize the profit of the private cloud provider. Eq. (12)

[1] https://aws.amazon.com.

[2] https://www.qingcloud.com.

specifies the probabilistic constraint that each BoT application can be completed no later than its deadline with a certain probability. Eqs. (13) and (14) describe the constraints on CPU and memory resources of the private cloud, respectively. Eq. (15) guarantees that a task can be scheduled to only one cloud. Eq. (16) denotes the decision variables in the optimization model, including the mapping relationships of tasks onto cloud resources, as well as the execution time slots of tasks on the private cloud. In summary, our optimization model requires that the scheduling solution not only maximizes the profit of the private cloud provider, but also satisfies the probabilistic deadline constraint under the variations of task durations.

In the literature, deterministic BoT scheduling problems are in general formulated as NP-hard integer programs (IPs) (Brucker, 2004; Sun et al., 2019). Considering the stochastic properties of task durations, the problem studied in this paper is more difficult to solve than the deterministic counterparts. Under the uncertainty of task durations, there exist two most challenging issues involved in the formulated stochastic model: (a) how to obtain the probability distribution of each BoT application's estimated finish time $\tilde{C}_i$, which is determined as the maximum of multiple normal random variables; and (b) how to perform the task assignment under the circumstance of varying task durations and a fixed deadline. In the subsequent sections, we will discuss the solutions to the above-mentioned problems in detail.

4. Distribution of makespan variations

A main challenge in the proposed stochastic framework is to model the distribution of each BoT application's estimated finish time. As described in Eq. (6), the estimated finish time $\tilde{C}_i$ of BoT application a_i is the maximum of finish times of all tasks a_i contains. Because the estimated finish time of t_{ij} is a normally distributed variable, the key challenge is to determine the distribution of application's makespan, which turns to be the maximum of multiple normal random variables (Sinha et al., 2012). Following the method by Li et al. in which the expected value and variance of uncertainty-affected makespan can be derived by Clark's equations (Li et al., 2013a), we calculate the expected value and variance of $\tilde{C}_i$ in an iterative manner. To be specific, for the ith BoT application, in order to obtain the distribution of $\tilde{C}_i$, we first consider the estimated finish times of two tasks, e.g., $e\tilde{ft}_{i1}$ and $e\tilde{ft}_{i2}$, and predict the distribution of the maximum of these two variables. Repeating this procedure for the remaining tasks, we can eventually obtain the distribution of $\tilde{C}_i$ as

$$\begin{aligned}\tilde{C}_i &= \max_{j=1}^{T_i}\left\{e\tilde{ft}_{ij}\right\}\\ &= \max\left\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}, \ldots, e\tilde{ft}_{iT_i}\right\}\\ &= \max\left\{\max\left\{\max\left\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}\right\}, \ldots, e\tilde{ft}_{i(j-1)}\right\}, e\tilde{ft}_{ij}\right\}.\end{aligned} \tag{17}$$

In what follows, we will detail how to calculate the distribution of the maximum of two normal variables. First, it is worth mentioning that, because the VM setup time is 0 and BoT tasks are independent, any two tasks do not interfere with each other no matter they are executed on public clouds or the private cloud. In other words, the estimated finish times of any two tasks are independent, i.e., $\rho(e\tilde{ft}_{iu}, e\tilde{ft}_{iv}) = 0,\ \forall 1 \le u \ne v \le T_i$. Here, $\rho(e\tilde{ft}_{iu}, e\tilde{ft}_{iv})$ represents the correlation coefficient between two tasks' estimated finish times. Then, according to Clark's equations (Li et al., 2013a), the expected value and variance of $\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}\}$ with $\rho(e\tilde{ft}_{i1}, e\tilde{ft}_{i2}) = 0$ can be calculated as

$$\begin{aligned}E[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}\}] &= E[e\tilde{ft}_{i1}]\phi(\xi_{1,2}) + E[e\tilde{ft}_{i2}]\phi(-\xi_{1,2})\\ &\quad + \varepsilon_{1,2}\psi(\xi_{1,2}),\end{aligned} \tag{18}$$

$$\begin{aligned}&Var[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}\}]\\ &= E[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}\}^2] - E^2[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}\}]\\ &= (E^2[e\tilde{ft}_{i1}] + Var[e\tilde{ft}_{i1}])\phi(\xi_{1,2})\\ &\quad + (E^2[e\tilde{ft}_{i2}] + Var[e\tilde{ft}_{i2}])\phi(-\xi_{1,2})\\ &\quad + (E[e\tilde{ft}_{i1}] + E[e\tilde{ft}_{i2}])\varepsilon_{1,2}\psi(\xi_{1,2})\\ &\quad - E^2[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}\}],\end{aligned} \tag{19}$$

where $\xi_{1,2} = \frac{1}{\varepsilon_{1,2}}\left(E[e\tilde{ft}_{i1}] - E[e\tilde{ft}_{i2}]\right)$, $\varepsilon_{1,2} = \sqrt{Var[e\tilde{ft}_{i1}] + Var[e\tilde{ft}_{i2}] - 2\rho_{1,2}\chi_{1,2}}$, $\rho_{1,2} = 0$, $\chi_{1,2} = \sqrt{Var[e\tilde{ft}_{i1}]Var[e\tilde{ft}_{i2}]}$, $\psi(t) = \frac{1}{\sqrt{2\pi}}e^{-\frac{t^2}{2}}$, and $\phi(x) = \int_{-\infty}^{x}\psi(t)dt = \int_{-\infty}^{x}\frac{1}{\sqrt{2\pi}}e^{-\frac{t^2}{2}}dt$.

Taking $\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}, e\tilde{ft}_{i3}\}$ for example, we use the following iterative method to calculate the expected value and variance of the maximum of multiple tasks' expected finish time. As mentioned earlier, the correlation coefficient of these three tasks' finish times is also zero, i.e., $\rho(e\tilde{ft}_{i1}, e\tilde{ft}_{i2}, e\tilde{ft}_{i3}) = 0$. The expected value and variance of $\max\{eft_{i1}, eft_{i2}, eft_{i3}\}$ can be calculated as

$$\begin{aligned}&E[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}, e\tilde{ft}_{i3}\}]\\ &= E[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}\}]\phi(\xi_{1,2,3}) + E[e\tilde{ft}_{i3}]\phi(-\xi_{1,2,3})\\ &\quad + \varepsilon_{1,2,3}\psi(\xi_{1,2,3}),\end{aligned} \tag{20}$$

$$\begin{aligned}&Var[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}, e\tilde{ft}_{i3}\}]\\ &= E[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}, e\tilde{ft}_{i3}\}^2] - E^2[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}, e\tilde{ft}_{i3}\}]\\ &= (E^2[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}\}] + Var[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}\}])\phi(\xi_{1,2,3})\\ &\quad + (E^2[e\tilde{ft}_{i3}] + Var[e\tilde{ft}_{i3}])\phi(-\xi_{1,2,3})\\ &\quad + (E[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}\}] + E[e\tilde{ft}_{i3}])\varepsilon_{1,2,3}\psi(\xi_{1,2,3})\\ &\quad - E^2[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}, e\tilde{ft}_{i3}\}]\end{aligned} \tag{21}$$

where $\varepsilon_{1,2,3} = \sqrt{Var[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}\}] + Var[e\tilde{ft}_{i3}]}$ and $\xi_{1,2,3} = (E[\max\{e\tilde{ft}_{i1}, e\tilde{ft}_{i2}\}] - E[e\tilde{ft}_{i3}])/\varepsilon_{1,2,3}$.

In summary, for a BoT application with T_i tasks, we can calculate the expected value and variance of $\tilde{C}_i$ by iteratively using Clark's equations $T_i - 1$ times. Once we have the distribution of $\tilde{C}_i$, its expected value $E[\tilde{C}_i]$ and variance $Var[\tilde{C}_i]$ can be easily determined, based on which we can determine whether the makespan under task duration variations can satisfy the probabilistic deadline constraint in Eq. (7).

On the other hand, due to the fact that task execution time is no longer a fixed value, it is also of great importance to develop a task assignment strategy that can tolerance the uncertainty of task execution times. When dispatching a task to the private cloud, we need to take into account the variations of execution times of the tasks ahead of this task. In our proposed stochastic scheduling framework, we design an uncertainty-aware task assignment strategy based on the distribution information of task execution times, which will be detailed later.

5. The proposed scheduling algorithm

In this section, we present our proposed metaheuristic algorithm IABS for solving the stochastic BoT scheduling problem formulated in Eqs. (11)–(16). IABS seeks for the scheduling solution leading to the maximized profit based on the search strategy of immune algorithm (IA), which is inspired by the immune response of organisms in nature. In the IA framework, the optimization objective corresponds to the antigen in the

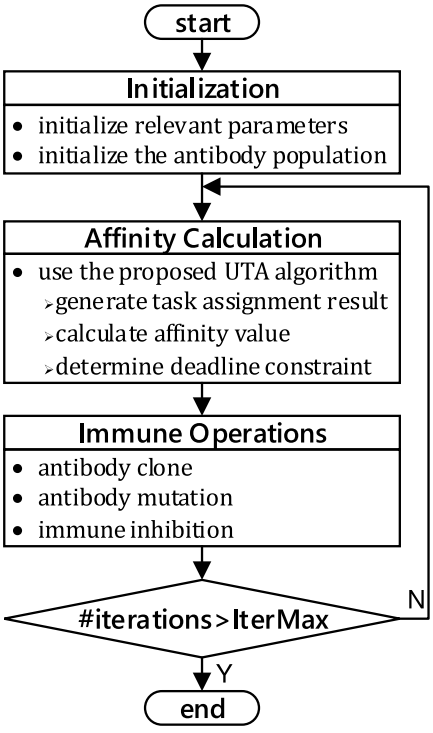

Fig. 3. The flowchart of the proposed stochastic BoT scheduling algorithm.

immune response, the feasible solution corresponds to the antibody, and the quality of a feasible solution corresponds to the affinity between antibody and antigen. For ease of interpretation, Fig. 3 illustrates the flowchart of the proposed IABS algorithm. Specifically, IABS starts with initializing a population of antibody individuals and the associated parameters. For each antibody in the population, we need to calculate its affinity value, i.e., the profit of the private cloud provider. This step is crucial in the stochastic scheduling framework, since we need to determine an appropriate task assignment result considering both stochastic tasks and the deadline constraint. Then, we use a series of immune operations to update the antibody population. IABS iteratively perform the affinity calculation and immune operations to obtain the final solution to the stochastic BoT scheduling problem.

5.1. Solution representation

Referring to the fundamental mechanism of standard IA, the antigen refers to the stochastic BoT scheduling problem to be solved, and each antibody stands for a candidate solution to the problem. In this work, we use a task sequence, i.e., a permutation of all BoT tasks to be scheduled, to represent the scheduling solution. To be specific, for any antibody A_i in the population $P = \{A_1, A_2, \ldots, A_{|P|}\}$, we define A_i as $A_i = \{A_{i1}, A_{i2}, \ldots, A_{iT}\}$, which is a T-dimensional tuple that can be regarded as a task sequence, where T is the total number of tasks. For example, assume there are two BoT applications $a_1 = \{t_{11}, t_{12}, t_{13}\}$ and $a_2 = \{t_{21}, t_{22}\}$ to be scheduled. Thus, we have $T = 5$. The following antibodies $A_1 = \{t_{11}, t_{12}, t_{13}, t_{21}, t_{22}\}$ and $A_2 = \{t_{11}, t_{21}, t_{12}, t_{22}, t_{13}\}$ both are valid task sequences.

In addition, IABS considers the affinity value to evaluate the quality of the antibody. As the optimization objective of our problem is to maximize the profit of the private cloud provider, we define antibody A_i's affinity value aff_i as the profit value that A_i finally leads to. Note that, in order to evaluate the quality of the solution, we need to first generate the task assignment result according to each antibody.

5.2. Solution initialization

A good initial solution can accelerate the optimization process of the proposed metaheuristic algorithm and improve the quality of the final solution. For this reason, we develop an initialization method, including the generation of the initial solution and the construction of the initial antibody population.

First, we discuss the generation of the initial solution. Considering the probabilistic deadline constraint in the stochastic scheduling framework, for a stochastic task t_{ij} with $\tilde{et}_{ij} \sim (\mu_{ij}, \sigma_{ij}^2)$, we estimate the deadline violation probability (DVP) as

$$DVP_{ij} = \int_{d_i}^{+\infty} \tilde{et}_{ij}(t)\, dt. \tag{22}$$

According to the value of DVP, we sort all the tasks in a non-ascending order of the DVP to generate an initial task sequence A_{init}. In other words, the task with a higher probability of deadline violation would be assigned a higher priority when performing task assignment.

Then, we construct the initial antibody population P_{init} based on the generated initial solution A_{init} in two steps. The first step is to add the initial solution A_{init} into P_{init}. In the second step, we perform a single-point crossover operation on A_{init} to generate other antibodies in P_{init}. After N-1 rounds of single-point crossover operation, we can obtain the initial antibody population P_{init} consisting of A_{init} and N-1 crossover antibodies.

5.3. Solution updating

We designed a novel antibody population updating scheme specifically for solving the stochastic BoT scheduling problem studied in this paper. The update of antibody population can be divided into the following steps: (a) antibody cloning, (b) antibody mutation, and (c) immune inhibition. We discuss below the details about these steps.

- antibody cloning

As the first step of the updating scheme, IABS performs an antibody cloning operation on the source population. Here, we use $P = \{A_1, A_2, \ldots, A_N\}$ to represent the source antibody population. For each source antibody A_i, $CP_i = \{cA_i^1, cA_i^2, \ldots, cA_i^{cn_i}\}$ represents the clone antibody sub-population generated by A_i where cn_i denotes the clone number of antibody A_i. We define cn_i as

$$cn_i = \left\lceil \frac{aff_i}{\sum_{i=1}^{N} aff_i} \times N \right\rceil \tag{23}$$

where N is the size of source population to be cloned. The greater the affinity of the individual is, the larger number of sub-individuals that will be cloned, for the purpose of protecting good genes and in turn speeding up the convergence rate of our algorithm. After all the source antibodies have generated their corresponding clone antibody sub-populations, we can obtain the clone population $CP = \{CP_1, CP_2, \ldots, CP_N\}$.

- antibody mutation

We propose a novel antibody mutation scheme that relies on not only the affinity of the antibody but also the current best solution to generate new solutions for the next-generation population. This scheme employs a probabilistic model in which the antibody with a larger affinity value has a higher probability to maintain the current position of the tasks. Based on this mutation strategy, good orders of tasks contained by the antibody could be inherited to explore more high-quality solutions.

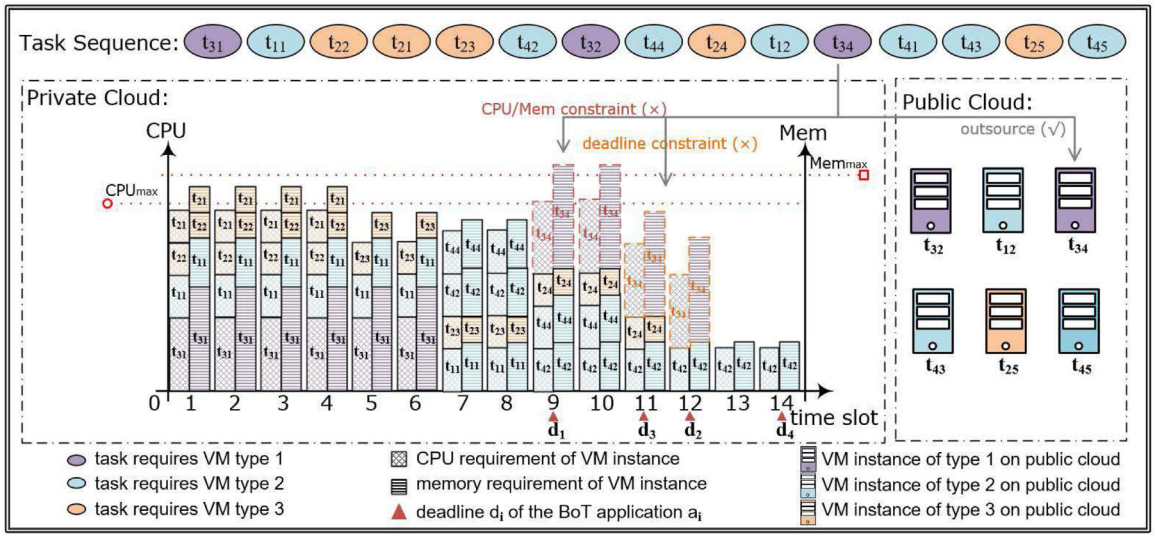

Fig. 4. An illustrative example of task assignment.

The antibody mutation operation starts with generating a mutation sub-population MP_i for each clone sub-population CP_i. For each antibody in CP_i, the affinity value of cloned antibody cA_i^l $(l = 1, 2, \ldots, cn_i)$ is aff_i. We define $diff_i$ to denote the affinity difference between cA_i^l and A_{best}, i.e., $diff_i = aff_{\text{best}} - aff_i$ where A_{best} is the current best antibody in P and aff_{best} is its affinity value. Next, each task in cA_i^l will remain in its original position with the probability

$$p_i = \exp\left\{-\frac{diff_i}{aff_{\text{best}}}\right\}. \tag{24}$$

As $diff_i \geq 0$, we have $p_i \in (0, 1]$. Note that, since the mutation probability p of the current best antibody A_{best} is 1, we perform a single-point crossover operation on A_{best} to generate mutation sub-population MP_{best}. According to Eq. (24), for antibodies with larger affinity values, the task orders in their corresponding task sequences are more likely to be maintained. For tasks whose positions have been changed, we determine the order of these tasks at random, and follow this order to arrange them in the empty positions of the task sequence.

Based on the steps described above, a mutated antibody mA_i^l is generated. Similarly, we can obtain the mutation sub-population $MP_i = \{mA_i^1, mA_i^2, \ldots, mA_i^{cn_i}\}$ and accordingly the mutation population $MP = \{MP_1, MP_2, \ldots, MP_N\}$.

- immune inhibition and population updating

During the immune inhibition operation, the clone inhibition population, denoted by CIP, is composed of N clone inhibition sub-populations $\{CIP_1, CIP_2, \ldots, CIP_N\}$. For each clone inhibition sub-population, CIP_i consists of source antibody A_i and all the mutated antibodies in mutation sub-population MP_i. The clone inhibition operation first selects an antibody with the highest affinity value from each clone inhibition sub-population, and adds this antibody to the new population. When this selection procedure is completed for all N sub-populations, the new population NP is generated and will be used in the next iteration.

5.4. Solution evaluation

To evaluate the quality of each candidate solution, we need to determine the assignment of all tasks in the task sequence onto the VMs of hybrid clouds. Fig. 4 illustrates the general flow of the task assignment process. Based on the task order in a given task sequence, the task assignment strategy determines whether each task should be outsourced to the public cloud or be executed on the private cloud. As such, we need to investigate the following three conditions: (a) the available CPU resources of the private cloud, (b) the available memory resources of the private cloud, and (c) the deadline constraint of each task.

In order to maximize the private cloud provider's profit, we need to minimize the outsourcing cost of using the public cloud resource. Here, we introduce the concept of minimum outsourcing cost (MOC) to define the lowest cost of renting the required VM instance for a task to be outsourced to public clouds. When a task has to be outsourced, it should be allocated to the public cloud leading to the MOC value. Specifically, for a task t_{ij}, if the index of the application-defined VM type is v, its MOC can be calculate as

$$MOC_{ij} = \min_{h=1}^{m} \mu_{ij} p_{vh}. \tag{25}$$

Accordingly, we define the public cloud dispatched for executing task t_{ij} as the optimal public cloud (OPC). i.e.,

$$OPC_{ij} = \arg\min\{h | \mu_{ij} p_{vh}\}. \tag{26}$$

In short, if our algorithm decides to outsource a task to public clouds, this task should be executed on the public cloud with the lowest outsourcing cost.

In order to calculate the affinity values of antibodies, we propose an uncertainty-aware task assignment (UTA) strategy. The proposed UTA takes into account not only the resource limitations of the private cloud but also the stochastic of the task duration. Algorithm 1 shows the pseudocode of our UTA algorithm. The first step is to initialize relevant parameters, including profit $Profit$, the maximum value of time slot S, and the available cpu_s and mem_s per time slot (lines 1–4). In line 5, we define a two-dimensional matrix $StartTime$ to record the start time of each task. Next, for each task t_{ij} in the given task sequence, we scan the remaining resources of the private cloud starting from time slot $s = 0$ to find whether there exists a continuous execution period that satisfies the task execution condition. To be specific, the above-mentioned task execution condition indicates that the remaining available CPU and memory resources of the private cloud are sufficient for the VM instance executing this task (lines 15–18). If we could find a frame of continuous time slots on the private cloud that satisfy the CPU and memory requirements, we will

Algorithm 1: Uncertainty-Aware Task Assignment (UTA)

Input: a task sequence ts;
Output: profit $Profit$;
1 set $S \leftarrow Max$, $P \leftarrow 0$;
2 **for** *each time slot* $s \in \{0, 1, \cdots, S\}$ **do**
3 set $CPU_s \leftarrow CPU_{\max}$;
4 set $Mem_s \leftarrow MEM_{\max}$;
5 initialize $StartTime[n][\max_{i=1}^{n} T_i]$;
6 **for** *each task* t_{ij} *in ts* **do**
7 set $st_{ij} \leftarrow 0$;
8 set $ad_{ij} \leftarrow \phi^{-1}(\alpha) \times \sigma_{ij} + \mu_{ij}$;
9 set $DispatchToPrivateCloud \leftarrow$ FALSE;
10 **while** *TRUE* **do**
11 $DispatchToPrivateCloud \leftarrow$ TRUE;
12 **if** $st_{ij} + ad_{ij} > d_i$ **then**
13 $DispatchToPrivateCloud \leftarrow$ FALSE;
14 **break**;
15 **for** *each time slot* $s \in \{st_{ij}, st_{ij}+1, \cdots, st_{ij}+ad_{ij}-1\}$ **do**
16 **if** ($cpu_{x_i} > CPU_s$ ***OR*** $mem_{x_i} > Mem_s$) **then**
17 $DispatchToPrivateCloud \leftarrow$ FALSE;
18 **break**;
19 **if** *DispatchToPrivateCloud* **then**
20 **for** *each time slot* $s \in \{st_{ij}, st_{ij}+1, \cdots, st_{ij}+dt_{ij}-1\}$ **do**
21 $CPU_s \leftarrow cpu_s - cpu_{x_i}$;
22 $Mem_s \leftarrow mem_s - mem_{x_i}$;
23 $StartTime[i][j] \leftarrow st_{ij}$;
24 **break**;
25 **else**
26 $st_{ij} \leftarrow st_{ij} + 1$;
27 **if** *DispatchToPrivateCloud* **then**
28 set $cost \leftarrow cost + \mu_{ij} \cdot cost_{x_i}$;
29 **else**
30 $StartTime[i][j] = 0$;
31 $cost \leftarrow cost + MOC_{ij}$;
32 **if** *MeetDeadlineConstraint(StartTime) is FALSE* **then**
33 $Profit \leftarrow -1$;
34 **else**
35 calculate *Profit* according to Eq. (8);
36 **return** *Profit*;

Algorithm 2: Deadline Constraint Satisfaction (DCS)

Input: *StartTime*;
Output: *MeetDeadlineConstraint*;
1 set *MeetDeadlineConstraint* $\leftarrow$ TRUE;
2 **for** *each task* t_{ij} *(i=1 to n; j=1 to* T_i*)* **do**
3 $st_{ij} \leftarrow StartTime[i][j]$;
4 calculate eft_{ij} according to Eq. (5);
5 **for** *each application* a_i *(i=1 to n)* **do**
6 **for** *each task* t_{ij} *(j=1 to* T_i*)* **do**
7 **if** *(j==1* ***OR*** *2)* **then**
8 calculate $mean_i$ of $\max\{eft_{i1}, eft_{i2}\}$ according to Eq. (18);
9 calculate $variance_i$ of $\max\{eft_{i1}, eft_{i2}\}$ according to Eq. (19);
10 **else**
11 update $mean_i$ of $\max\{\max\{eft_{i(j-2)}, eft_{i(j-1)}\}, eft_{ij}\}$ according to Eq. (20);
12 update $variance_i$ of $\max\{\max\{eft_{i(j-2)}, eft_{i(j-1)}\}, eft_{ij}\}$ according to Eq. (21);
13 $j = j + 1$;
14 determine the probability distribution of $makespan_i$ based on $mean_i$ and $variance_i$;
15 **if** $(Prob\{makespan_i < d_i\} < \alpha)$ **then**
16 *MeetDeadlineConstraint* $\leftarrow$ FALSE;
17 **return** *MeetDeadlineConstraint*;

assign t_{ij} to the private cloud and update the remaining available resources of the private cloud on these time slots (lines 19–23). Otherwise, t_{ij} will be outsourced to public clouds. The cost of executing t_{ij} is calculated as in lines 27–31. If t_{ij} is dispatched to the private cloud, the execution cost is $\mu_{ij} cost_{x_i}$. Otherwise, this task will be executed on public cloud OPC_{ij} with the cost of MOC_{ij}. After all tasks in the task sequence have been assigned, we need to determine whether the generated assignment result satisfies the probabilistic deadline constraint described in Eq. (13). If the probabilistic constraint is satisfied, this task sequence is identified as a feasible solution and the profit can be calculated by Eq. (8). Otherwise, the profit of infeasible solutions will be set to an invalid value (line 32–35).

It is worth emphasizing the following two issues in UTA: (a) how to calculate the task duration on the dispatched VM instance; and (b) how to evaluate the feasibility of the assignment result satisfying the probabilistic constraint on each application's makespan. To solve the first problem, we design a task duration calculation method based on the probability distribution of et_{ij}. Given a specific threshold value α in the probabilistic constraint (refer to Eq. (7)), we define ad_{ij} as the assigned duration of task t_{ij} constrained by this threshold value, which can be determined as

$$ad_{ij} = norminv(\alpha, \mu_{ij}, \sigma_{ij}) \tag{27}$$

where the definitions of α, μ and σ can be referred to in Section 3.3. Here, we use function $norminv(\alpha, \mu_{ij}, \sigma_{ij})$ to calculate the quantile of the normally distributed variable $\tilde{et}_{ij} \sim N(\mu_{ij}, \sigma_{ij}^2)$ given a probability α. The quantile of the standard normal distribution can be known by looking up the table of the standard normal cumulative distribution function (Papoulis and Pillai, 2004). For random variables that are not standard normal, e.g., $\tilde{et}_{ij}$, we can obtain the α quantile by the following theorem.

Theorem 1. *For a non-standard normal distribution variable* $\tilde{et}_{ij} \sim N(\mu_{ij}, \sigma_{ij}^2)$*, its* α *quantile* et_{ij}^{α} *can be calculated by*

$$\begin{aligned} et_{ij}^{\alpha} &= norminv(\alpha, 0, 1) \times \sigma_{ij} + \mu_{ij} \\ &= \phi^{-1}(\alpha) \times \sigma_{ij} + \mu_{ij} \end{aligned} \tag{28}$$

where $norminv(\alpha, 0, 1)$ represents the inverse cumulative distribution function of the zero-mean and unit-variance standard normal variable. For simplicity, we use $\phi^{-1}(\alpha)$ to denote the α quantile of $\tilde{et}_{ij}$.

Proof. Suppose $\phi(x)$ denotes the cumulative distribution function of standard normal distribution, i.e., $\phi(x) = \int_{-\infty}^{x} \psi(t)dt = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt$. The inverse function of $\phi(x)$ can be represented by $\phi^{-1}(\alpha)$. In other words, the value of $\phi^{-1}(\alpha)$ is the α quantile of standard normal distribution. For non-standard normal

distribution variable $\tilde{et}_{ij}$, we can derive

$$P\{\tilde{et}_{ij} < et_{ij}^{\alpha}\} = P\{\frac{\tilde{et}_{ij} - \mu_{ij}}{\sigma_{ij}} < \frac{et_{ij}^{\alpha} - \mu_{ij}}{\sigma_{ij}}\} = P\{\widehat{X} < \frac{et_{ij}^{\alpha} - \mu_{ij}}{\sigma_{ij}}\} = \alpha \quad (29)$$

where $\widehat{X} \sim N(0, 1)$ is subject to the standard normal distribution. Therefore, we have $\frac{et_{ij}^{\alpha} - \mu_{ij}}{\sigma_{ij}} = \phi^{-1}(\alpha)$, i.e., $et_{ij}^{\alpha} = \phi^{-1}(\alpha) \times \sigma_{ij} + \mu_{ij}$. □

Following Theorem 1, for any task t_{ij}, we can calculate its assigned duration ad_{ij} by $\phi^{-1}(\alpha) \times \sigma_{ij} + \mu_{ij}$, where the value of $\phi^{-1}(\alpha)$ can be obtained by looking up the cumulative standard normal distribution table directly. Note that by determining the assigned duration ad_{ij} for task t_{ij}, we in fact extend t_{ij}'s duration during the task assignment procedure, in order to tolerate the variations of task duration in practical scenarios. The obtained ad_{ij} will be applied to the subsequent steps of UTA algorithm, such as the determination of CPU and memory constraints and the update of CPU_s and Mem_s.

To evaluate the feasibility of the assignment result produced by UTA, we design a deadline constraint satisfaction (DCS) algorithm to determine whether the probabilistic constraints on all applications' makespans are satisfied. As described in Algorithm 2, DCS first calculates the expected finish time $\tilde{eft}_{ij}$ for each task based on its execution time $\tilde{et}_{ij}$ and the start time specified in UTA (lines 2–4). Next, by using the method detailed in Section 3.3, for each application a_i, we calculate the probability distribution of $\tilde{makespan}_i$ in an iterative fashion (lines 5–14). Then, based on the obtained $mean_i$ and $variance_i$ of $\tilde{makespan}_i$, we investigate whether the generated assignment result satisfies the probabilistic constraint of the application's deadline (lines 14–16). If the probability of all applications completing their executions before their corresponding deadlines is no less than α, the DCS algorithm returns TRUE, and returns FALSE otherwise.

5.5. The proposed IABS

In this paper, we propose a metaheuristic IABS to solve the formulated stochastic BoT scheduling problem. Combined with the solution representation, initialization, updating, and evaluation operators that have been elaborated previously, IABS searches for the optimal scheduling solution with the maximized profit in an iterative manner. IABS begins with generating an initial antibody population, where each antibody is a candidate solution represented by a task sequence of all BoT tasks. During the iterative update process, IABS uses the UTA algorithm to evaluate the quality of each antibody by calculating the affinity value. After evaluating all antibody individuals, the antibody with the highest affinity value will be identified as the best solution in the current population. To produce the next-generation population, IABS uses a series of immune operations, including antibody clone, antibody mutation, and immune inhibiting, to update all antibodies. This procedure is repeated until the termination criterion is met. Algorithm 3 summarizes the overall flow of the proposed IABS algorithm.

IABS starts with initializing several critical parameters and generating the initial antibody population (see Section 5.2). During each iteration, we first calculate the affinity values of all antibodies in P by performing Algorithm 1 (see Section 5.4), and update the *BestAff* and *BestAnti* accordingly (lines 4–7). Based upon the proposed solution updating scheme (see Section 5.3), the antibody population P is updated as follows. The first step is antibody cloning (lines 8–11). In this step, we determine the clone number of each antibody by Eq. (23), and generate the clone antibody sub-population. The second step is antibody mutation (lines 12–29). In this step, we generate the mutation sub-population MP_i for each clone sub-population CP_i by employing a probabilistic model, which can inherit the good orders of tasks to explore more high-quality solutions. The third step is immune inhibiting (lines 30–35), in which we construct the clone inhibition sub-population CIP_i and then add the best antibody in each CIP_i to the new population. Accordingly, the new population NP will be generated when the clone inhibition operation of N sub-populations CIP_i is completed. When the iteration terminates, the value of *BestAff* is returned as the optimal profit of the private cloud provider.

Algorithm 3: IA-Based BoT Scheduling (IABS)

Input: task set including all applications' tasks Ψ;
Output: the profit of the best solution *Profit*;
1 initialize parameters *Iter*, *IterMax*, $|P|$, and *BestAff*;
2 initialize antibody population $P = \{A_1, A_2, \cdots, A_{|P|}\}$;
3 **while** *Iter* < *IterMax* **do**
4 **for** *i=1 to* $|P|$ **do**
5 calculate the aff_i by performing UTA on A_i;
6 **if** (aff_i > *BestAff*) **then**
7 set *BestAff* ← aff_i and *BestAnti* ← A_i;
8 sort the antibodies in P in a non-ascending order of affinity values;
9 **for** *each source antibody* A_i *(i=1 to N)* **do**
10 calculate cn_i by Eq. (23);
11 generate clone antibody sub-population CP_i;
12 **for** *each* CP_i *(i = 1 to N)* **do**
13 calculate $diff_i \leftarrow aff_{\text{best}} - aff_i$;
14 calculate $p_i \leftarrow \exp\left\{-\frac{diff_i}{aff_{\text{best}}}\right\}$;
15 **for** *each cloned antibody* cA_i^l *(l = 1 to* cn_i*)* **do**
16 **if** (p_i *is equal to 1*) **then**
17 randomly select a position within $[1, T]$;
18 perform a single-point crossover operation;
19 **else**
20 **for** *each task t in* cA_i^l *(t = 1 to T)* **do**
21 **if** (*random(0, 1)* ≤ p_i) **then**
22 keep task t unchanged;
23 **else**
24 remove task t from cA_i^l;
25 empty the t-th position of cA_i^l;
26 sort all removed tasks at random and arrange them in the empty positions of cA_i^l;
27 obtain mutated antibody mA_i^l;
28 $MP_i = \{mA_i^1, mA_i^2, \cdots, mA_i^{cn_i}\}$;
29 clear the new population NP;
30 **for** *each* CIP_i *(i = 1 to N)* **do**
31 add A_i to MP_i to form CIP_i;
32 sort the antibodies in CIP_i in a non-ascending order of affinity values;
33 add the first antibody in CIP_i into NP;
34 set $P \leftarrow NP$;
35 set *Profit* ← *BestAff*;
36 **return** *Profit*;

Table 2
VM types and prices of the hybrid cloud environment.

VM type	#CPUs	Memory (GB)	Public cloud price ($/h)			Private cloud	
			EC2 us-east	EC2 us-west	EC2 eu-east	Income ($/h)	Cost ($/h)
t2.micro	1	1	0.013	0.017	0.014	0.015	0.003
t2.small	1	2	0.026	0.034	0.028	0.030	0.006
t2.medium	2	4	0.052	0.068	0.056	0.060	0.012
m3.medium	1	3.75	0.070	0.077	0.077	0.080	0.007
m3.large	2	7.5	0.140	0.154	0.154	0.160	0.014
m3.xlarge	4	15	0.280	0.308	0.308	0.320	0.028
m3.2xlarge	8	30	0.560	0.616	0.616	0.640	0.056

5.6. Complexity analysis

We analyze the computational complexity of the proposed algorithm as follows. First, since we use quick sort to arrange all the tasks in non-ascending order of DVP value, the complexity of algorithm initialization is $\mathcal{O}(T \cdot \log T)$, where T is the total number of tasks. In the solution updating procedure during each iteration, the operations of antibody cloning, antibody mutation, and immune inhibition all have linear complexities $\mathcal{O}(T)$. In addition, the UTA strategy (Algorithm 1) is used iteratively for evaluating the quality of a candidate solution. The computation cost of UTA mainly consists of the execution of the "while" loop (lines 10–26) and the calculation of deadline constraint satisfaction (Algorithm 2). However, it is not practical to determine the complexity of this algorithm since it depends on how many rounds in the "while" loop. Alternatively, we provide the lower- and upper-bound complexities of the "while" loop in UTA, which are $\mu_{\max} T$ and $\mu_{\max} st_{\max} T$, respectively, where $\mu_{\max} = \max\{\mu_{ij} \mid i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, T_i\}$ and $st_{\max} = \max\{st_{ij} \mid i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, T_i\}$. In addition, Algorithm 2 has a linear complexity $\mathcal{O}(T)$. The total computation cost of UTA is bounded by $\mu_{\max} T + T$ and $\mu_{\max} st_{\max} T + T$. Thus, the lower- and upper-bound complexities of UTA are determined as $\mathcal{O}(\text{UTA})_{\text{lower}} = \mathcal{O}(T)$ and $\mathcal{O}(\text{UTA})_{\text{upper}} = \mathcal{O}(st_{\max} T)$, respectively.

According to the preceding analysis, the amount of computations in the entire flow of our proposed IABS is $T \cdot \log T + 3T + IterMax \cdot |P| \cdot (\mathcal{O}(\text{UTA}))$, where $IterMax$ and $|P|$ represent the maximum number of iterations and population size, respectively. We draw the conclusion that the lower- and upper-bound complexities of IABS are given by $\mathcal{O}(\text{IABS})_{\text{lower}} = \mathcal{O}(IterMax \cdot |P| \cdot T)$ and $\mathcal{O}(\text{IABS})_{\text{upper}} = \mathcal{O}(IterMax \cdot |P| \cdot st_{\max} T)$, respectively.

6. Performance evaluation

This section presents the performance evaluation results. To verify the scheduling quality of our IABS algorithm, we compare IABS with two deterministic scheduling methods and three uncertainty-aware algorithms. Moreover, we also investigate the impacts of important algorithm factors upon the scheduling performance. The evaluation system for scheduling BoT applications on hybrid clouds is implemented all by Java programming. All simulation experiments were performed on a desktop machine equipped with a 10-core CPU operating at 2.8 GHz and 32-GB memory.

6.1. Experimental setup

In experiments, the hybrid cloud environment consists of one private cloud and three public clouds provided by Amazon EC2, which are represented by us-east, us-west, and eu-east, respectively. As aforementioned, due to the hybrid cloud construction solution, the private cloud and public clouds are capable of offering the same VM types. Table 2 describes the configuration information of the hybrid cloud environment used for performance evaluation. The available VM types on public clouds and the corresponding prices are obtained from the official website of Amazon Web Services.[3] The information of private cloud configuration comes from a local private cloud provider. Specifically, there are seven VM types provided by each cloud provider, and different VM types require different amounts of CPU and memory resources.

To simulate the stochastic BoT scheduling problem, we design a testing instance set (TIS) in which task durations are modeled as normally distributed random variables rather than fixed values. The TIS includes different numbers of BoT applications consisting of varying numbers of tasks. These numbers, together with the nominal values of task durations, are carefully selected according to real-world BoT applications. To be specific, the number of BoT applications is $\omega \in \{5, 10, 20\}$, and the number of tasks in each application is defined as an identical value λ, i.e., $T_1 = T_2 = \cdots = T_n = \lambda$ where $\lambda \in \{5, 10, 20\}$. Therefore, there are nine groups of testing BoT applications, which can be denoted by $\omega \times \lambda$, i.e., 5×5, 5×10, 5×20, 10×5, 10×10, 10×20, 20×5, 20×10, and 20×20. In addition, since each group consists of 5 different testing instances, there are in total 45 testing instances in our TIS. To generate the stochastic task durations, for each task t_{ij}, we set the expected value of its execution time $\tilde{et}_{ij}$ within the interval $E[\tilde{et}_{ij}] \in [10, 50]$ hours, and specify its variance within the interval $Var[\tilde{et}_{ij}] \in \left[0.1 \times E[\tilde{et}_{ij}], 0.4 \times E[\tilde{et}_{ij}]\right]$. Based on the values of $E[\tilde{et}_{ij}]$ and $Var[\tilde{et}_{ij}]$, in our sampling experiments, we truncate the distributions of execution times at their 3σ values. In other words, the sample duration values are bounded by the interval $[\mu - 3\sigma, \mu + 3\sigma]$.

6.2. Evaluation of scheduling quality

In this section, we evaluate the scheduling quality of our stochastic IABS algorithm. For comparison purposes, we consider the following two deterministic scheduling scenarios under the impact of task duration variations:

- Mean-case scheduling: the expected value of uncertain task execution time $\tilde{et}_{ij}$ is used in the task scheduling framework to generate an uncertainty-unaware deterministic scheduling solution.
- Worst-case scheduling: the uncertain task execution time $\tilde{et}_{ij}$ is truncated at its 3σ values, and the maximum value is used in the task scheduling framework to generate an uncertainty-unaware deterministic scheduling solution.

We first investigate the success rate of scheduling solutions generated by different algorithms. For each testing instance ($\omega \times \lambda$), we randomly generate ϱ groups of BoT application samples with fixed task durations according to the expected value $E[\tilde{et}_{ij}]$ and variance $Var[\tilde{et}_{ij}]$. We define success rate (SR) to estimate the feasibility of a candidate solution as follows

$$SR = \frac{\sum_{i=1}^{\varrho} \frac{\omega_s}{\omega}}{\varrho} \times 100\% \tag{30}$$

[3] https://aws.amazon.com.

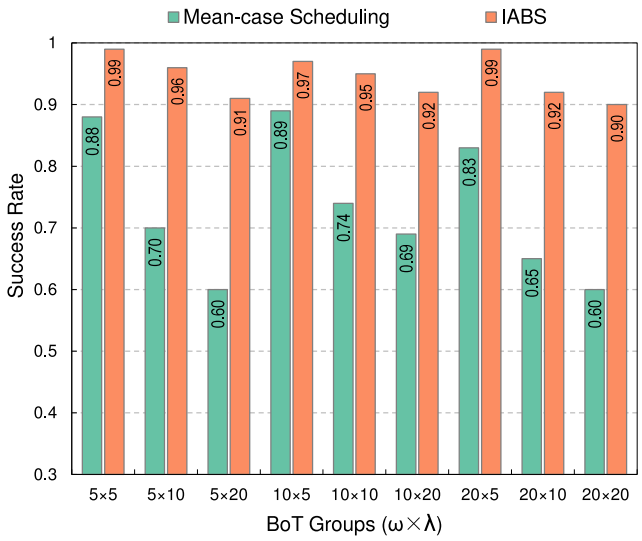

Fig. 5. Comparison of success rates between mean-case scheduling and IABS.

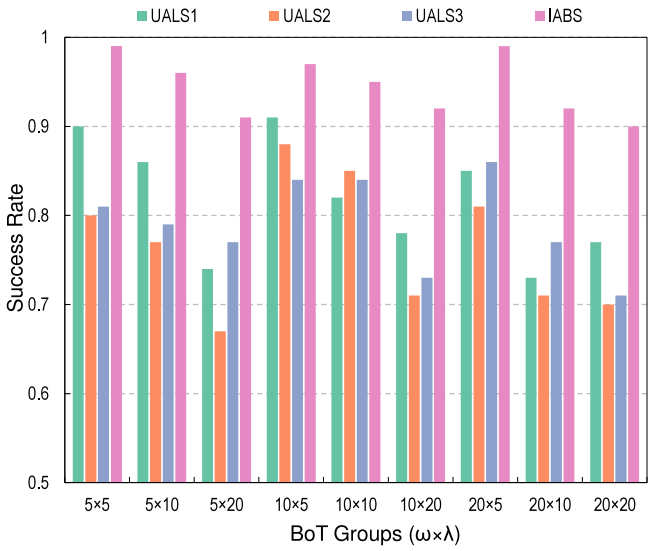

Fig. 6. Comparison of success rates between UALS1–UALS3 and IABS.

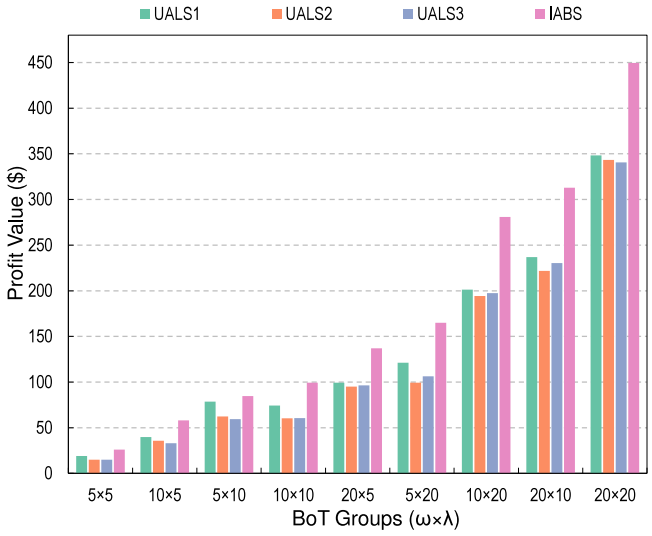

Fig. 7. Comparison of profit values between UALS1–UALS3 and IABS.

Table 3
Comparison between worst-case scheduling and IABS.

ω	λ	Worst-case		IABS		Profit increase
		Profit ($)	Success rate	Profit($)	Success rate	
5	5	10.882	1.00	12.299	0.99	13.02%
	10	31.626	0.99	36.370	0.96	15.00%
	20	144.187	0.97	164.989	0.91	14.43%
10	5	47.982	1.00	58.064	0.97	21.01%
	10	87.123	0.99	99.23	0.95	13.90%
	20	242.868	0.98	280.782	0.92	15.61%
20	5	50.221	0.99	58.146	0.99	15.78%
	10	277.451	0.98	312.902	0.92	12.78%
	20	390.091	0.97	448.54	0.90	14.98%

where ω_s is the number of BoT applications completed before their deadlines, and ϱ is set as 1000. Note that only when all tasks in a BoT application finish their execution before the deadline, the BoT application is considered to be successfully completed. According to Eq. (30), ω_s/ω indicates the proportion of applications completed before the deadline among all applications, and the obtained *SR* is the average success rate of 1000 BoT samples. Fig. 5 shows the success rates achieved by using the deterministic mean-case scheduling and our IABS algorithm with $\alpha = 0.9$. We can observe that for BoT instances of different sizes, the success rate of our stochastic IABS is above 90%. On the other hand, the deterministic mean-case scheduling method leads to significant deteriorations of success rate results compared with our method. To be specific, according to the results on the nine groups, the success rate of the deterministic method is on average 21.44% lower than that of our IABS method. The reason is that since the uncertainty of task duration is not taken into account, the scheduling solution of the deterministic mean-case method is prone to cause the violation of applications' deadlines. To conclude, our algorithm significantly improves the feasibility in satisfying the probabilistic constraint, guaranteeing the QoS of the private cloud provider.

We further investigate the profit values obtained by our IABS algorithm and by the deterministic scheduling, respectively. Since the solution generated by deterministic scheduling using mean-case method has a high probability of violating the deadline constraint, we only compare the profit value between the deterministic scheduling using worst-case design and our stochastic IABS method. For BoT testing instances of different sizes, Table 3 provides the success rates and profit values achieved by using the deterministic worst-case method and IABS, respectively. Specifically, for the group of 20×20, the profit value by the worst-case method is \$390.091, whereas the profit value by our stochastic method reaches \$448.540, indicating a profit increase of 15.78%. This improvement is due to the fact that the conservative solutions generated by the deterministic scheduling using worst-case design would cause higher outsourcing costs for performing tasks as early as possible. On average, compared with the deterministic method, our stochastic algorithm improves the profit value by 15.17% with an insignificant 3.33% decrease of success rate. Therefore, the stochastic IABS algorithm is more appropriate for solving the uncertainty-affected scheduling problem by taking into account both the profit and the probability of completing BoT applications before the deadline.

The previous set of experiments verify the importance of stochastic scheduling under the uncertainty of task durations. Since very few stochastic scheduling exist in the literature, for comparison purposes, we implement three uncertainty-aware methods considering task duration variations. As a common task

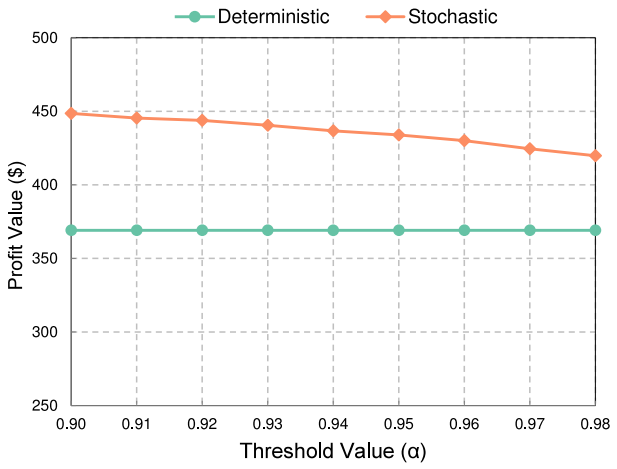

Fig. 8. Profit values for different threshold values.

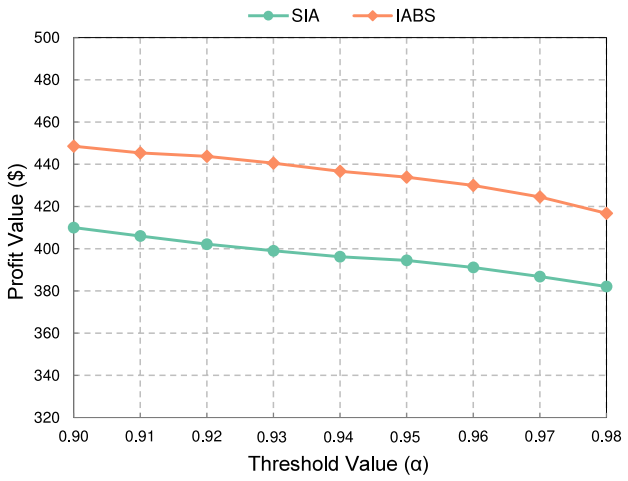

Fig. 9. Profit values by IABS and standard IA for different threshold values.

scheduling strategy, list scheduling is to construct a list by sorting the priorities of tasks, and to schedule tasks one by one according to the order in the list. Based upon different sorting rules of tasks' priorities, we implement three uncertainty-aware list scheduling algorithms, namely UALS1, UALS2, and UALS3, to solve the stochastic BoT scheduling problem studied in this work. Considering the uncertainty of task durations, UALS1 uses the listing strategy in the ESTS algorithm (Li et al., 2013a) to obtain an assignment result. Since Max–Min and Min–Max are two traditional list scheduling algorithms, we perform UALS2 and UALS3 by sorting the tasks according to the $\mu + \sigma$ value of $\tilde{et}_{ij}$ in a non-ascending and non-descending manner, respectively. Fig. 6 shows the success rates by using UALS1–UALS3 and our IABS. With the threshold value α fixed at 90%, the success rates by our algorithm are guaranteed to satisfy this requirement, whereas the success rates of other three algorithms to be compared are below 90% to different extents, with the lowest success rate of 67%. The comparison results in Fig. 6 demonstrate that compared with other uncertainty-aware list scheduling algorithms, our algorithm guarantees the feasibility of scheduling solutions such that the execution of all BoT applications can be completed before the deadline. On the one hand, for complex BoT scheduling problems in a hybrid cloud environment, our IABS metaheuristic outperforms list scheduling in terms of scheduling performance. On the other hand, the probabilistic constraint in our method ensures that the BoT applications can fulfill the QoS requirement under the impact of uncertain task durations. Fig. 7 investigates the profit values of the scheduling solutions obtained by performing UALS1-UALS3 and IABS. There is no significant difference in the profit values among UALS1, UALS2, and UALS3, whereas our IABS leads to significantly higher profits. IABS achieves the maximum profit improvement for the configuration of $\omega = 20$ and $\lambda = 20$. The profit values by IABS and UALS3 are \$449.540 and \$340.569, respectively, indicating a profit increase of 31.99%.

6.3. Evaluation of important factors

Having justified the importance of our stochastic IABS algorithm,we now evaluate the scheduling quality of our IABS by selecting different threshold values for BoT applications to meet their deadline constraints. Fig. 8 presents the profit values achieved by IABS and the worst-case deterministic method. Here, the configurations of ω and λ are 20 and 20, respectively. For different α values, our stochastic method is capable of generating more effective task scheduling solutions that satisfy the probabilistic deadline constraint under the impact of uncertain task durations. However, the profit value of the stochastic method decreases as the deadline probability constraint becomes stringent, since some tasks performed on the private cloud can no longer meet the probabilistic deadline constraint and have to be outsourced to public clouds. In other words, with relatively smaller α values for the probabilistic constraint, IABS can produce more profit-efficient solutions to the stochastic scheduling problem.

We further investigate the impacts of key strategies in our stochastic algorithm. Since we incorporated a solution initialization rule and a new antibody mutation operator in our IABS, we develop a competing stochastic method by using the standard IA (SIA) framework. Different from our IABS, SIA takes no account of the above-mentioned two strategies. Instead, SIA initializes the antibody population in a random manner, and the antibody mutation only uses a single-point crossover approach. Fig. 9 shows the profit values achieved by our IABS and the SIA with various threshold values. The number of iterations and the number of antibodies the population in both algorithms are set as 30 and 100, respectively. Specifically, for the configuration of $\alpha = 0.93$, the profit value by IABS is \$440.509, whereas the profit value of SIA is \$392.017. A maximum profit improvement of 12.37% is achieved. For the nine cases of threshold value α, IABS outperform SIA by an average profit improvement of 9.87%. We draw the conclusion that the solution initialization and antibody mutation strategy incorporated in our IABS enhance the performance of the scheduling metaheuristic, leading to higher profit values.

To fully verify the scheduling performance in a statistical sense, we use the well-known analysis of variance (ANOVA) method to analyze the scheduling results by using IABS and SIA. We introduce relative error (RE) to quantify the algorithm's effectiveness, which can be defined as

$$RE = \frac{1}{R}\left(\sum_{r=1}^{R}\frac{S^* - S_r}{S^*}\right) \times 100\% \tag{31}$$

where R means that the algorithm to be evaluated is repeated R times, and S_r is the performance metric obtained in the rth round, and S^* denotes the best result obtained by all algorithms during all rounds. In our experiments, each algorithm is performed five times, i.e., $R = 5$. It is obvious that a lower RE value indicates a higher quality of the scheduling solutions produced by a scheduling algorithm. Fig. 10 provides the mean values and least-significant difference (LSD) intervals of REs by using IABS and SIA, respectively. We can observe that for different α values ranging from 0.90 to 0.98, our IABS achieves lower REs as compared to

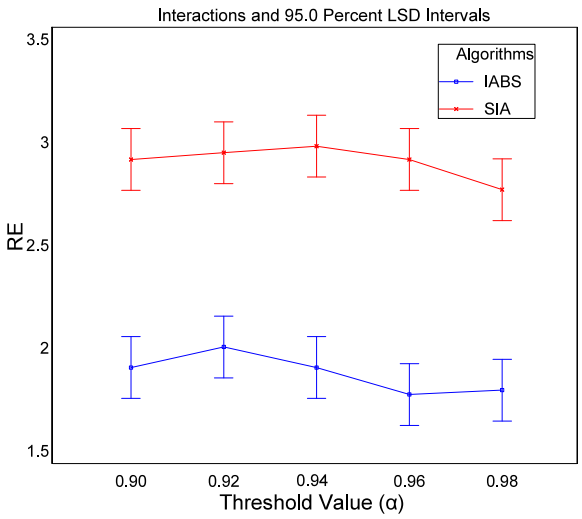

Fig. 10. Comparison of ANOVA analysis results between IABS and SIA.

SIA. More importantly, the non-overlapping LSD intervals justify that the performance of IABS is significantly better than that of SIA. The profit values obtained by IABS are higher than those of SIA under the influence of random factors on the experimental results.

7. Conclusion

This paper addresses the BoT scheduling problem on hybrid clouds with the objective of profit maximization under uncertain task durations and the deadline constraint. We model the execution times of BoT applications as random variables and formulate the resultant task scheduling problem as a stochastic optimization problem. A distinguishing property of this formulation is a probabilistic deadline constraint upon the probability distribution of BoT applications' makespan. An IA-based metaheuristic algorithm IABS is proposed to determine the most optimal solution that maximizes the profit of the private cloud provider under the probabilistic and resource constraints. To further enhance the solution exploration capability, we integrate a solution initialization procedure and an antibody mutation strategy in the IABS framework. Experimental results justify the performance of our IABS algorithm, in terms of solution feasibility and effectiveness.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abdi, S., PourKarimi, L., Ahmadi, M., Zargari, F., 2017. Cost minimization for deadline-constrained bag-of-tasks applications in federated hybrid clouds. Future Gener. Comput. Syst. 71, 113–128.

Alam, T., Raza, Z., 2018. Quantum genetic algorithm based scheduler for batch of precedence constrained jobs on heterogeneous computing systems. J. Syst. Softw. 135, 126–142.

Ando, E., Nakata, T., Yamashita, M., 2009. Approximating the longest path length of a stochastic DAG by a normal distribution in linear time. J. Discrete Algorithms 7 (4), 420–438.

Van den Bossche, R., Vanmechelen, K., Broeckhove, J., 2011. Cost-efficient scheduling heuristics for deadline constrained workloads on hybrid clouds. In: International Conference on Cloud Computing Technology and Science. pp. 320–327.

Brucker, P., 2004. Scheduling Algorithms. Springer-Verlag, Berlin, Germany.

Cai, Z., Li, X., Ruiz, R., Li, Q., 2017. A delay-based dynamic scheduling algorithm for bag-of-task workflows with stochastic task execution times in clouds. Future Gener. Comput. Syst. 71, 57–72.

Cao, K., Li, L., Cui, Y., Wei, T., Hu, S., 2020. Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing. IEEE Trans. Ind. Inf. 17 (1), 494–503.

Cao, K., Xu, G., Zhou, J., Wei, T., Chen, M., Hu, S., 2018. Qos-adaptive approximate real-time computation for mobility-aware IoT lifetime optimization. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 38 (10), 1799–1810.

Cao, K., Zhou, J., Yin, M., Wei, T., Chen, M., 2016. Static thermal-aware task assignment and scheduling for makespan minimization in heterogeneous real-time mpsocs. In: International Symposium on System and Software Reliability. pp. 111–118.

Chen, Y., Wang, L., Chen, X., Ranjan, R., Zomaya, A.Y., Zhou, Y., Hu, S., 2020. Stochastic workload scheduling for uncoordinated datacenter clouds with multiple qos constraints. IEEE Trans. Cloud Comput. 8 (4), 1284–1295.

Chen, H., Zhu, X., Guo, H., Zhu, J., Qin, X., Wu, J., 2015. Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment. J. Syst. Softw. 99, 20–35.

Chen, H., Zhu, X., Liu, G., Pedrycz, W., 2018. Uncertainty-aware online scheduling for real-time workflows in cloud service environment. IEEE Trans. Serv. Comput. http://dx.doi.org/10.1109/TSC.2018.2866421.

Chen, H., Zhu, X., Qiu, D., Liu, L., 2016. Uncertainty-aware real-time workflow scheduling in the cloud. In: International Conference on Cloud Computing. pp. 577–584.

Dong, M., Fan, L., Jing, C., 2019. ECOS: An efficient task-clustering based cost-effective aware scheduling algorithm for scientific workflows execution on heterogeneous cloud systems. J. Syst. Softw. 158, 110405.

Doğan, A., özgüner, F., 2004. Genetic algorithm based scheduling of meta-tasks with stochastic execution times in heterogeneous computing systems. Cluster Comput. 7 (2), 177–190.

Duan, R., Prodan, R., Li, X., 2014. Multi-objective game theoretic scheduling of bag-of-tasks workflows on hybrid clouds. IEEE Trans. Cloud Comput. 2 (1), 29–42.

Genez, T.A.L., Bittencourt, L., Fonseca, N., Madeira, E., 2019. Estimation of the available bandwidth in inter-cloud links for task scheduling in hybrid clouds. IEEE Trans. Cloud Comput. 7 (1), 62–74.

Gu, J., Gu, X., Gu, M., 2009. A novel parallel quantum genetic algorithm for stochastic job shop scheduling. J. Math. Anal. Appl. 355 (1), 63–81.

Gutierrez-Garcia, J.O., Sim, K.M., 2015. Agent-based cloud bag-of-tasks execution. J. Syst. Softw. 104, 17–31.

Haidri, R.A., Katti, C.P., Saxena, P.C., 2019. Cost-effective deadline-aware stochastic scheduling strategy for workflow applications on virtual machines in cloud computing. Concurr. Comput.: Pract. Exper. 31 (7), e5006:1–e5006:24.

He, T., Toosi, A.N., Buyya, R., 2021. SLA-aware multiple migration planning and scheduling in SDN-NFV-enabled clouds. J. Syst. Softw. 176, 110943.

Houssein, E.H., Gad, A.G., Wazery, Y.M., Suganthan, P.N., 2021. Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends. Swarm Evol. Comput. 62, 100841.

Hu, M., Veeravalli, B., 2013. Requirement-aware scheduling of bag-of-tasks applications on grids with dynamic resilience. IEEE Trans. Comput. 62 (10), 2108–2114.

Islam, M.T., Srirama, S.N., Karunasekera, S., Buyya, R., 2020. Cost-efficient dynamic scheduling of big data applications in apache spark on cloud. J. Syst. Softw. 162, 110515.

Jia, Y., Chen, W., Yuan, H., Gu, T., Zhang, H., Gao, Y., Zhang, J., 2021. An intelligent cloud workflow scheduling system with time estimation and adaptive ant colony optimization. IEEE Trans. Syst. Man Cybern.: Syst. 51 (1), 634–649.

Keshanchi, B., Souri, A., Navimipour, N.J., 2017. An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing. J. Syst. Softw. 124, 1–21.

Lee, Y.C., Zomaya, A.Y., 2010. Energy conscious scheduling for distributed computing systems under different operating conditions. IEEE Trans. Parallel Distrib. Syst. 22 (8), 1374–1381.

Li, K., Tang, X., Li, K., 2013a. Energy-efficient stochastic task scheduling on heterogeneous computing systems. IEEE Trans. Parallel Distrib. Syst. 25 (11), 2867–2876.

Li, K., Tang, X., Veeravalli, B., Li, K., 2013b. Scheduling precedence constrained stochastic tasks on heterogeneous cluster systems. IEEE Trans. Comput. 64 (1), 191–204.

Long, T., Varghese, B., Barker, A., 2015. Budget constrained execution of multiple bag-of-tasks applications on the cloud. In: International Conference on Cloud Computing. pp. 975–980.

Long, T., Varghese, B., Barker, A., 2018. A survey and taxonomy of resource optimisation for executing bag-of-task applications on public clouds. Future Gener. Comput. Syst. 82, 1–11.

Malawski, M., Juve, G., Deelman, E., Nabrzyski, J., 2015. Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. Future Gener. Comput. Syst. 48, 1–18.

Maurya, A.K., Tripathi, A.K., 2018. Deadline-constrained algorithms for scheduling of bag-of-tasks and workflows in cloud computing environments. In: International Conference on High Performance Compilation, Computing and Communications. pp. 6–10.
Möhring, R.H., Schulz, A.S., Uetz, M., 1999. Approximation in stochastic scheduling: the power of LP-based priority policies. J. ACM 46 (6), 924–942.
Novak, A., Sucha, P., Hanzalek, Z., 2019. Scheduling with uncertain processing times in mixed-criticality systems. European J. Oper. Res. 279 (3), 687–703.
Oprescu, A.M., Kielmann, T., 2010. Bag-of-tasks scheduling under budget constraints. In: International Conference on Cloud Computing Technology and Science. pp. 351–359.
Papoulis, A., Pillai, S.U., 2004. Probability, Random Variables and Stochastic Processes. Tata McGraw-Hill, New Delhi, India.
Pelaez, V., Campos, A., Garcia, D.F., Entrialgo, J., 2016. Autonomic scheduling of deadline-constrained bag of tasks in hybrid clouds. In: International Symposium on Performance Evaluation of Computer and Telecommunication Systems. pp. 1–8.
Sarin, S.C., Nagarajan, B., Liao, L., 2010. Stochastic Scheduling: Expectation-Variance Analysis of a Schedule. Cambridge University Press, London, UK.
Sindhu, S., Mukherjee, S., 2018. An evolutionary approach to schedule deadline constrained bag of tasks in a cloud. Int. J. Bio-Inspired Comput. 11 (4), 229–238.
Sinha, D., Visweswariah, C., Venkateswaran, N., Xiong, J., Zolotov, V., 2012. Reversible statistical max/min operation: Theory and applications to timing. In: Design Automation Conference. pp. 1067–1073.
Sun, J., Yin, L., Zou, M., Zhang, Y., Zhang, T., Zhou, J., 2020. Makespan-minimization workflow scheduling for complex networks with social groups in edge computing. J. Syst. Archit. 108, 101799.
Sun, J., Zhang, Y., Wu, Z., Zhu, Y., Plaza, A., 2019. An efficient and scalable framework for processing remotely sensed big data in cloud computing environments. IEEE Trans. Geosci. Remote Sens. 47 (7), 4294–4308.
Tang, X., Li, X., Fu, Z., 2017. Budget-constraint stochastic task scheduling on heterogeneous cloud systems. Concurr. Comput.: Pract. Exper. 29 (19), e4210:1–e4210:13.
Tang, X., Li, K., Liao, G., Fang, K., Wu, F., 2011. A stochastic scheduling algorithm for precedence constrained tasks on grid. Future Gener. Comput. Syst. 27 (8), 1083–1091.
Tarplee, K.M., Friese, R., Maciejewski, A.A., Siegel, H.J., Chong, E.K., 2015. Energy and makespan tradeoffs in heterogeneous computing systems using efficient linear programming techniques. IEEE Trans. Parallel Distrib. Syst. 27 (6), 1633–1646.
Varshney, P., Simmhan, Y., 2019. AutoBoT: Resilient and cost-effective scheduling of a bag of tasks on spot VMs. IEEE Trans. Parallel Distrib. Syst. 30 (7), 1512–1527.
Wang, B., Song, Y., Sun, Y., Liu, J., 2016. Managing deadline-constrained bag-of-tasks jobs on hybrid clouds. In: High Performance Computing Symposium. pp. 1–8.
Wu, G., Chen, J., Bao, W., Zhu, X., Xiao, W., Wang, J., 2017. Towards collaborative storage scheduling using alternating direction method of multipliers for mobile edge cloud. J. Syst. Softw. 134, 29–43.
Wu, Z., Sun, J., Zhang, Y., Zhu, Y., Li, J., Plaza, A., Benediktsson, J.A., Wei, Z., 2021. Scheduling-guided automatic processing of massive hyperspectral image classification on cloud computing architectures. IEEE Trans. Cybern. 51 (7), 3588–3601.
Yan, H., Zhu, X., Chen, H., Guo, H., Zhou, W., Bao, W., 2019. DEFT: Dynamic fault-tolerant elastic scheduling for tasks with uncertain runtime in cloud. Inform. Sci. 477, 30–46.
Yang, Y., Lu, X., Jin, H., Liao, X., 2015. A stochastic task scheduling algorithm based on importance-ratio of makespan to energy for heterogeneous parallel systems. In: International Conference on High Performance Computing & Communications. pp. 390–396.
Zhang, Y., Sun, J., 2017. Novel efficient particle swarm optimization algorithms for solving qos-demanded bag-of-tasks scheduling problems with profit maximization on hybrid clouds. Concurr. Comput.: Pract. Exper. 29 (21), e4249:1–e4249:19.
Zhang, Y., Sun, J., Wu, Z., 2017. An heuristic for bag-of-tasks scheduling problems with resource demands and budget constraints to minimize makespan on hybrid clouds. In: 2017 Fifth International Conference on Advanced Cloud and Big Data (CBD). pp. 39–44.
Zhang, Y., Yun, W., Cheng, H., 2015. CloudFreq: Elastic energy-efficient bag-of-tasks scheduling in DVFS-enabled clouds. In: IEEE International Conference on Parallel & Distributed Systems. pp. 585–592.
Zhang, Y., Zhou, J., Sun, J., 2019. Scheduling bag-of-tasks applications on hybrid clouds under due date constraints. J. Syst. Archit. 101, 101654:1–101654:15.
Zhang, Y., Zhou, J., Sun, L., Mao, J., Sun, J., 2019. A novel firefly algorithm for scheduling bag-of-tasks applications under budget constraints on hybrid clouds. IEEE Access 7, 151888–151901.
Zheng, B., Pan, L., Liu, S., 2021a. Market-oriented online bi-objective service scheduling for pleasingly parallel jobs with variable resources in cloud environments. J. Syst. Softw. 176, 110934.
Zheng, B., Pan, L., Liu, S., 2021b. Market-oriented online bi-objective service scheduling for pleasingly parallel jobs with variable resources in cloud environments. J. Syst. Softw. 176, 110934.
Zuo, X., Zhang, G., Tan, W., 2013. Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud. IEEE Trans. Autom. Sci. Eng. 11 (2), 564–573.

Lu Yin received the B.S. degree in network engineering from Nanjing University of Science and Technology, Nanjing, China, in 2018. She is currently working toward the Ph.D. degree in the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. Her research interests include cloud computing and edge computing.

Junlong Zhou received the Ph.D. degree in Computer Science and Technology from East China Normal University, Shanghai, China, in 2017. He was a Visiting Scholar with the University of Notre Dame, Notre Dame, IN, USA, during 2014–2015. He is currently an Associate Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include embedded systems, cloud-edge-IoT, and cyber–physical systems, where he has published 2 book chapters and more than 80 refereed papers, including 25+ IEEE/ACM Transactions. He has been an Associate Editor for the Journal of Circuits, Systems, and Computers and the IET Cyber–Physical Systems: Theory & Applications, a Subject Area Editor for the Journal of Systems Architecture, and a Guest Editor for 6 ACM/IET/Elsevier/Wiley Journals such as ACM Transactions on Cyber–Physical Systems.

Jin Sun received the B.S. and M.S. degrees in computer science from Nanjing University of Science and Technology, Nanjing, China, in 2004 and 2006, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Arizona in 2011. He is currently a Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include high-performance computing and electronic design automation.