Computing and Informatics, Vol. 45, 2026, 127–153, doi: 10.31577/cai_2026_1_127

RELIABILITY MODEL OF CLOUD COMPUTING JOB SCHEDULING BASED ON DISCRETE TIME MARKOV CHAIN

Ali Sarhadi

Department of Computer Engineering
Mal. C., Islamic Azad University
Malayer, Iran
e-mail: Asarhadi@iau.ac.ir

Abbas Karimi

Department of Computer Engineering
Ka. C., Islamic Azad University
Karaj, Iran
e-mail: Akarimi@iau.ac.ir

Ali Yousefi

Department of Computer Engineering
Ha. C., Islamic Azad University
Hamedan, Iran
e-mail: ali.yousefii@iau.ac.ir

Abstract. Cloud computing primarily focuses on providing secure and reliable access to geographically distributed resources. Cloud computing is a parallel and distributed system that simplifies the virtualization of distributed computing. Job scheduling is one of the most important services in cloud computing. Job scheduling is used to schedule users' jobs to allocate suitable resources in a cloud environment. Recently, several efficient job scheduling algorithms have been proposed for cloud computing. The purpose of designing these algorithms is to achieve time and cost optimization for job execution. On the other hand, it is necessary to provide a model

to assess the reliability of the different job scheduling algorithms. Unfortunately, no accurate and complete model has been proposed to evaluate the reliability of these algorithms. Scheduling failures are inevitable in cloud computing due to the harsh deployment environment, resource migration, non-compliance time or cost, etc. Therefore, providing a reliability model is necessary for successful scheduling algorithms. In this paper, a reliability model based on a discrete-time Markov chain is incorporated for the famous scheduling algorithms MCT, MET, BCO, HCOC, and PPO. We propose an approach to evaluating the reliability of scheduling algorithms based on time and cost constraints. The proposed approach is modeled by a Discrete Time Markov Chain, and the results show the reliability of scheduling algorithms in different states.

Keywords: Cloud computing, scheduling, Markov chain, reliability

1 INTRODUCTION

Cloud computing proposes a new virtual resource, similar to traditional IT services, such as storage, servers, and databases. In cloud computing, computing infrastructure and services should always be available on computing servers. Cloud computing proposes a new infrastructure to establish virtual resources via the internet. The main reasons for its popularity include scalability, fault tolerance, high availability, utilization, reliability, easy to use, and economical [1, 2]. Moreover, cloud computing can also obtain nested services from different service providers and owners [3]. The cloud provides a cost-effective model through a pay-as-you-go model that allows individuals or businesses, or industries to start a cloud-based service with minimum investment [4]. Resource management is the most important activity in a cloud computing environment that comprises different activities, from job submission to job execution. Resource management in the Cloud consists of two steps: a) resource provisioning and b) resource scheduling. Resource provisioning is described as the step to select appropriate resources for a given job based on QoS requirements specified by cloud users. Resource scheduling is the most important part of resource management in parallel and distributed computing. Intensive research has been done in this area, and many results have been widely accepted [5]. Indeed, resource scheduling assigns the jobs to the appropriate and selected resources based on required features. The process of resource scheduling instructs the scheduler to get tasks from the users and asks the cloud information service (CIS) for available resources and their properties. According to the availability of resources and the task scheduling algorithm, the scheduler schedules user-submitted jobs on various resources as per requirements [6, 7]. The overall performance of cloud computing should be enhanced by reducing the completion time and cost of all tasks. The jobs are appropriately distributed over resources so that necessary preferences between tasks are met, and the total time and cost

needed to execute all tasks are minimized. Proper task scheduling improves the efficiency and performance of the cloud environment. To achieve high performance, parameters such as task execution time, resource utilization, etc., should be considered [8]. Cloud computing job scheduling algorithms face challenges in reliability and resource allocation. Markov-based methods can be employed to model cloud service reliability and optimize task scheduling in terms of makespan, execution time, and cost. Reliability evaluation can estimate failure probabilities in cloud environments, enabling reliability-aware scheduling algorithms that outperform traditional methods. The rest of the article is organized as follows. Section 2 deals with Related Work. Cloud computing scheduling is reviewed in Section 3. In Section 4, the concept of a Markov chain is described. In Section 5, cloud computing task scheduling based on a Markov chain model is proposed. In Section 6 the reliability function is described. In Section 7, case studies are described. In Section 8, the experimental setup is described. Experimental results of the proposed method are presented in Section 9, and finally, the Conclusions are provided in Section 10.

1.1 Proposed Approach

Cloud computing job scheduling algorithms are crucial for efficient resource utilization and task completion in distributed computing environments. Therefore, considering the reliability of scheduling algorithms is an important issue. In this paper, we have introduced a Markov chain model for evaluating the reliability of task scheduling algorithms. Reliability of task scheduling with time and cost constraints using a Markov chain introduces new challenges because these constraints require not only a specific task execution time but also a certain task completion cost. Cloud computing reliability and job scheduling are critical challenges that several researchers have addressed. Discrete Markov chain models have shown significant potential in improving job scheduling efficiency and resource allocation in cloud computing environments. Markov-based methods have been applied to model cloud service reliability and optimize task scheduling. Discrete Markov chain modeling has been shown to improve reliability in cloud computing job scheduling. Discrete Markov chain modeling can effectively quantify and predict job scheduling reliability metrics in cloud computing systems. In this paper, we have not presented a new scheduling or load balancing algorithm, but we have provided a method to evaluate the reliability of all existing scheduling algorithms under time and cost constraints. The proposed model comprehensively considers all failure states in task scheduling, including time, cost, or both. The experimental results show a comparison of the reliability of popular scheduling algorithms under the above conditions.

1.2 Contributions

The main contributions can be expressed as follows:

- Reliability model for cloud computing job scheduling is presented.
- The proposed model is based on time and cost constraints in task scheduling in the cloud environment.
- Discrete Markov chain has been used to implement the proposed model.
- We conducted a series of experiments to evaluate the performance of the proposed model on some of the most famous scheduling algorithms.

2 RELATED WORK

Markov chain-based reliability models can significantly enhance job scheduling algorithms in cloud computing environments with varying time and cost constraints. These models enable the accurate analysis of reliability in cloud services. Markov chain models offer a probabilistic framework for predicting system failures and guiding scheduling decisions in cloud computing. Recent research has explored reliability-driven task scheduling in cloud computing environments. In a set of studies, more researchers employed Markov-based techniques (including continuous-time and reward models), and fewer cases used semi-Markov processes. Souravlas et al. [9] used an irreducible finite state Markov process to report improvements in makespan and average response time through fair load balancing. Jiang et al. [10] described a continuous-time MDP approach that minimizes energy consumption when managing randomly arriving, interdependent tasks, and Wang et al. [11] combined a Deep-Q-Network with a Markov game model to achieve better execution time and cost outcomes compared with other metaheuristic approaches. Chang et al. [12] employ an M/M/m/m Markov chain within a Reward-based Adaptive Global Resource Management framework to address reliability issues. Sun et al. [13] facilitated the formulation of scheduling problems as multi-objective optimization tasks. Several studies have proposed models utilizing Markov-based methods to analyze cloud service reliability and optimize task scheduling [13, 14]. Markov-based methods have been applied to model cloud service reliability and optimize task scheduling, considering factors like makespan and flowtime [14]. Some approaches include reliability-based allocation matrices for virtual machine assignment in cloud computing [15] and two-phase methods combining reliability requirements and timing constraints [16]. These studies aim to improve overall system reliability and reduce costs. Other studies use threshold-based policies, uniform budget splitting, and multi-objective fitness measures to refine resource allocation.

Scheduling failures are unavoidable in heterogeneous computing, such as cloud, which affect system reliability. Since the task scheduling algorithm in cloud computing is challenging, we investigate a new reliability model with time and cost

constraints for task scheduling algorithms in this paper. The proposed method is designed to evaluate the reliability of scheduled tasks by using the Discrete Markov chain. The main idea of many job scheduling algorithms in cloud computing systems is reducing the execution time of jobs or applications without considering other factors, such as the budget or deadline [17]. On the other hand, the job scheduling becomes more challenging when we consider multiple QoS parameters.

Time optimization is usually a classic and traditional goal in cloud or distributed computing scheduling [18, 19, 20]. Moreover, the problem of finding the best solution with the optimized time is known as an NP-hard problem [21, 22]. Therefore, using traditional methods to solve this problem is necessary, and many field works in this discourse have been done recently [23, 24].

In [25, 26], a genetic-based job scheduling algorithm to optimize the time of cloud jobs has been suggested. The algorithm suggested in [23] and other similar works [5, 27] only focus on minimizing time, but as in large-scale jobs, reliability of computational resources is a basic issue that must be carefully considered to be able to offer applicable scheduling solutions for that kind of job.

However, cost, budget, and economic criteria are other important parameters in the cloud computing scheduling mechanism. For cost-critical parallel applications, cost-aware scheduling algorithms have been proposed for minimizing execution cost or satisfying the budget constraint on heterogeneous systems [28, 29]. However, very few papers have targeted the cloud computing environment and designs for minimizing the schedule length of budget-constrained applications. In [30], a heuristic algorithm of the deadline early tree was presented. It minimized the cost of deadline-constrained applications without considering the communication time between tasks. As the job scheduling constraints in this paper are time and cost, we only consider these two QoS parameters in our review of previous work.

Reliability, time, and cost are three different QoS parameters that are considered in the algorithms. Users are allowed to define QoS constraints to guarantee the quality of the schedule. In [31, 32] the Dynamic Constraint Algorithm (DCA) was proposed as an improvement of the Multiple-Choice Knapsack Problem (MCKP), to optimize two criteria for jobs, e.g., execution time and budget. The Hybrid Cloud Optimized Cost scheduling algorithm (HCOC), proposed in [33], and a cost-based job scheduling algorithm called Deadline-MDP (Markov Decision Process), proposed in [34], address the problem of minimizing cost while constrained by a deadline. Although these models could have applicability in a utility computing paradigm, we do not consider such a paradigm in this paper. On the other hand, the reliability of the cloud system and scheduling is affected by several factors, such as hardware failure, software failure, etc. Reliability of computational hardware, software, and data resources that contain the cloud and provide the means to execute user jobs, and reliability of cloud networks for messaging and data transport are important and should be met [35]. Regardless of the Cloud reliability features can cause reduced application performance, such as scheduling time and speedup, due to idle operations [36]. He et al. [37] proposed the relia-

bility of the cloud computing infrastructure based on the failure rate of the processors and the links between them. These failure rates can be taken from cloud resource profiling, system logs, and statistical prediction techniques. Srinivasan and Jha [38] proffered reliability of a system with envisage to a task set as the probability that the system can run the task set without any failure. The target of the work [39] is to provide a reliability-driven scheduling system to evaluate system reliability, based on a best reliability communication link search algorithm to find the shortest path, and then they provide reliability priority rank (Rank) to evaluate the task's priority by focusing on reliability overheads RASD. Two heuristic algorithms are proposed in [40], namely Minimum Cost Scheduling and Development of Reliability Maximization Schedule. Poola et al. [41] proposed a fault-tolerant algorithm against the premature termination of spot instances, as well as robustness against performance variations of Cloud resources. A spot instance in cloud computing helps a user to utilize resources with less expensive cost, even if it is unreliable [42]. This paper proposed a workflow scheduling scheme that reduces the task waiting time when an instance occurs in the out-of-bid situation.

Also, learning based techniques such as Reinforcement Learning (RL) are broadly used to solve problems in partially visible environments. As the most recent state-of-the-art (SOTA) reinforcement learning (RL) algorithms, they have many advantages in solving problems. Asynchronous Advantage Actor-Critic (A3C) [43], Proximal Policy Optimization (PPO) [44], and Soft Actor-Critic (SAC) [45] are the most important models of SOTA algorithms. Those models are already being used to solve various problems, such as robot navigation, elevator controls, military applications, medical diagnostics, task scheduling, and education [46]. Proximal Policy Optimization (PPO) is one of the SOTA models and is a type of reinforcement learning algorithm. The PPO algorithm is a gradient policy algorithm that is suitable for continuous control problems. PPO is an offline learning method, and the strategy it adopts to interact with the environment and the strategy to be learned differ. The main idea of the PPO algorithm is to transfer online learning to offline learning based on importance sampling and adopt two networks to improve the network convergence rate [47].

3 CLOUD COMPUTING SCHEDULING

Parallel computing environments have made progress toward new scheduling services that go beyond classical scheduling theory. Unlike the traditional scheduling services assumptions, which do not allow processing a task by several machines at a time [27], a new feature of multiprocessor computations is the ability to break tasks into several parts and to process them simultaneously by different processors. A full survey of the relevant models and solution techniques can be found in [48, 49]. A main aspect of a cloud scheduling system is efficiently allocating users' tasks to cloud resources. Recent works in cloud computing include Quality of Service prepa-

ration, which ensures that an application is performed within a guaranteed time at a pre-specified cost. Often users need the time or cost parameters optimization, as quality of services, the goals of scheduling algorithms can be thought of as time or cost optimization [50]. Regarding time optimization, the user expects the scheduling algorithms to complete the task in the most efficient time possible. In cost optimization, the user also requires a scheduling system with the lowest possible cost. As a result, the resource scheduling system must use a mechanism that meets these requirements.

3.1 Time and Budget Constraint Scheduling

In the cloud environment, users must pay to use the resources to execute their jobs. As mentioned in the related work section, users determine their deadlines and then request cost or time optimization. The deadline of a job is defined as the maximum finishing time of its last task to be executed. A scheduling algorithm that focuses on cost optimization strategy should allocate heterogeneous cloud resources to heterogeneous users' jobs so that their execution finishes in the specified deadline with minimum cost, and a scheduling algorithm that adopts a time optimization strategy should allocate heterogeneous cloud resources to heterogeneous user jobs so that their execution finishes within specified budget and in minimum time. Amini et al. [51] proposed a method that is a cost-optimizer and tries to grow the opportunity of meeting job deadlines. Sahni and Vidyarthi [52] introduced an effective cost-effective deadline-constrained algorithm, namely JIT-C, for scheduling a scientific workflow in a public Cloud. Budget is defined as the maximum amount that a user wants to pay for executing a workflow application on computing resources. In [25, 30], the authors proposed heuristic-based scheduling algorithms to minimize end-to-end execution under a user-specified financial cost constraint.

In this paper, we assumed that the scheduling policy may fail for the following reasons:

1. Time constraint is not satisfied (λ_t).
2. Budget constraint is not satisfied (λ_c).
3. Failed state (both of above options λ_t and λ_c).

Based on these assumptions, the reliability model of scheduling algorithms can be designed.

4 CONCEPTION OF MARKOV CHAIN

A Markov chain is a stochastic process consisting of a finite number of states with the feature that the future state only depends on the current state, and is independent of the past state [53]. Consider, there is a set of states S, S=$\{s_1, s_2, ..., s_n\}$, the

probability equation can be illustrated by Equation (1):

$$prob\left(X_{(k+1)} = x_{(K+1)} \mid X_1 = x_1, \ldots, X_k = x_k\right) = prob\left(X_{(K+1)} = x_{(k+1)} \mid X_k = x_k\right). \tag{1}$$

If the chain is right now in state x_i, then it goes to state x_j at the next step with a probability defined by p_{ij}, which is called the transition probability, and it does not depend on all previous states. Therefore, we can create a transition matrix P as below, in which n is the total number of states. We can then define the law of "probability of transition" from a state (i) towards another state (j) by:

$$P = \begin{bmatrix} p_{1,1} & \cdots & p_{1,n} \\ \vdots & \ddots & \vdots \\ p_{n,1} & \cdots & p_{n,n} \end{bmatrix},$$

$$P_{ij} \in [0,1], \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, n, \quad \sum_{j=1}^{n} P_{ij} = 1.$$

A Markov process is called a Markov chain if its state space and time space are both discrete. Then we have:

$$S(K) = s(K-1).p = S(K-2).p^2 = ... = s(0).p^k$$

Here's (k) introduces the state at time k, and $s(0)$ means the initial state when the system starts [54].

4.1 Markov Model Discretion

For each system, a Markov model includes a list of the possible states of that system, the possible transition paths between those states, and the rate parameters of those transitions. In a reliability model, the transitions usually consist of failures and repairs. When presenting a Markov model graphically, each state is usually depicted as a "bubble", with arrows denoting the transition links between states, as shown in Figure 1, for a single component that has just two states: successes and failures. The symbols P and q specified the probability of the transition from State Good to Bad [55].

Fortunately, the non-recursive effects in Markov models for reliability analysis have been proved, so it is possible to utilize the recursive solution method, solving for each state probability ratio using its respective state equation as if all the other state probability ratios were known, and arrive at a first-order approximate solution. Then this process can be repeated, using the newly computed probability ratios, to arrive at an even better approximation, and so on.

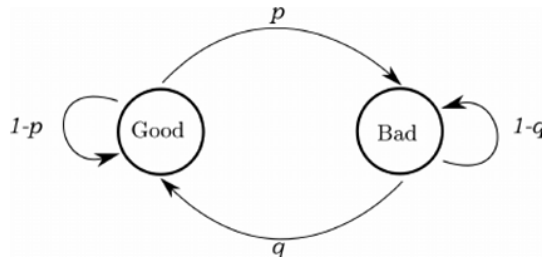

Figure 1. Transition between states

5 CLOUD COMPUTING TASK SCHEDULING BASED ON MARKOV CHAIN MODEL

This section describes an efficient reliability model for cloud computing task scheduling. Our assumption is that each mapping of resources to tasks can put the system in one of the four modes that are described below:

State 0: Success scheduling (time and cost are satisfied).

State 1: Unsatisfied time constraint.

State 2: Unsatisfied cost constraint.

State 3: Fail state (unsatisfied time and cost constraint).

Figure 2 shows the reliability state diagram of cloud computing scheduling.

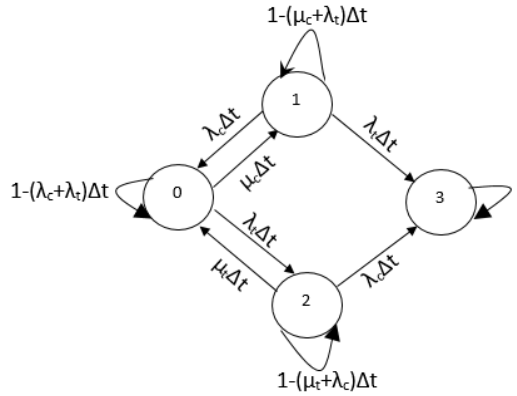

Figure 2. Cloud computing task scheduling state diagram

5.1 Reliability Modeling

The function of $R(t)$ can also be obtained using a Markov process by defining a D transition matrix to compute the system reliability. The transition matrix D is the overall Transition Rate Matrix that combines the graph. This matrix represents all the transition rates between states (both transient and absorbing) in the discrete-time Markov chain [56]. The D transition matrix for reliability modeling in the case of a four-unit system would be given by:

$$[D]_{Rel} = \begin{bmatrix} 1-(\lambda_c+\lambda_t) & \lambda_c & \lambda_t & 0 \\ \mu_c & 1-(\mu_c+\lambda_t) & 0 & 0 \\ \mu_t & 0 & 1-(\lambda_c+\mu_t) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

This matrix that describes the Markov chain is called the transition matrix. It is the most important tool for analyzing Markov chains. The transition matrix D must list all possible states in the state space S. Once the system enters state 3, we do not carry out system repair, and the system remains permanently in state 3 (failed state). Once the D matrix is available, we can find the B matrix directly from D through the following formula. The B matrix, also known as the fundamental matrix, quantifies the expected number of visits to transient states before absorption [56]. In fact, this matrix characterizes the expected behavior of the system across transient states before failure. The element B_{ij} denotes the expected number of times the process will be in transient state j, given that it started in transient state i, before eventual absorption.

$$[B] = [D-I]^{TR},$$

where the TR symbol stands for the transpose of the matrix $[D-I]$, and I is an identity matrix [57].

$$[B]_{Rel} = \begin{bmatrix} -(\lambda_c+\lambda_t) & \mu_c & \mu_t & 0 \\ \lambda_c & -(\mu_c+\lambda_t) & 0 & 0 \\ \lambda_t & 0 & -(\lambda_c+\mu_t) & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The C matrix provides the absorption probabilities, i.e., the probability that the system will ultimately enter a given absorbing state, and we can compute the C matrix from B through the following formula. Here the element C_{ij} represents the probability that the system, starting from transient state i, will eventually be absorbed in absorbing state j [58].

$$[C] = S[I] - [B],$$

where I is the identity matrix, therefore, the C matrix is defined as follows:

$$[C]_{Rel} = \begin{bmatrix} s+\lambda_c+\lambda_t & -\mu_c & -\mu_t & 0 \\ -\lambda_c & s+\mu_c+\lambda_t & 0 & 0 \\ -\lambda_t & 0 & s+\lambda_c+\mu_t & 0 \\ 0 & 0 & 0 & s \end{bmatrix}.$$

The C matrix quantifies the probabilities of different failure modes, depending on the absorbing state reached. Let $P_i(t)$ denote the probability of being in state i at time t. The Kolmogorov forward differential equations generate the Laplace form of each probability state. The use of the Laplace transform in solving state probabilities of a Markov reliability model is motivated by several advantages over direct time-domain analysis. First, the Kolmogorov forward differential equations governing the state probabilities form a set of coupled linear differential equations. Solving these equations directly in the time domain can become algebraically cumbersome, especially for systems with multiple degradation states. The Laplace transform converts these differential equations into a set of algebraic equations, which are considerably easier to manipulate and solve. Second, the Laplace transform inherently incorporates the initial conditions into the transformed equations. This eliminates the need for manual handling of boundary conditions, thereby streamlining the solution process.

Third, the Laplace domain representation often yields compact closed-form expressions for the transformed state probabilities. Finally, the Laplace transform facilitates the derivation of important reliability measures such as the reliability function $R(t)$, the mean time to failure (MTTF), and availability measures. The transform-domain framework provides a systematic and scalable approach that remains efficient even for higher-order Markov models.

For these reasons, the Laplace transform is widely regarded as the standard technique in the reliability analysis of Markov models, as also emphasized in Misra's formulation [58]. By following the outlined earlier procedure, various $P_i(s)$, $i = 0, 1, 2, 3$ could be determined as below based on Komologorov forward differential equations. The denominator polynomials for each $P_i(s)$ arise from $det(sI - B)$, which is of degree for a four-state system. The numerators correspond to minors of this matrix. Explicitly, one may write:

$$P_i(s) = \frac{N_i(s)}{det(sI - B)}, \tag{2}$$

where $N_i(s)$ is obtained from the adjugate matrix or directly via block inversion. This formulation naturally reproduces rational forms of the Laplace transforms for each state probability.

$$P_0(s) = \frac{s^3 + s^2(\lambda_c + \mu_t)(\mu_c + \lambda_t) + s(\lambda_c + \mu_t)(\mu_c + \lambda_t)}{\Delta}, \tag{3}$$

$$P_1(s) = \frac{\lambda_c s^2 + \lambda_c^2 s}{\Delta}, \tag{4}$$

$$P_2(s) = \frac{\lambda_t s^2 + \lambda_t^2 s)}{\Delta}, \tag{5}$$

$$P_3(s) = \frac{2\lambda_c\lambda_t s + \lambda_c^2\lambda_t + \lambda_t^2\lambda_c}{\Delta}. \tag{6}$$

In Equations (3), (4), (5) and (6), Delta [59] is calculated by Equation (7):

$$\Delta = s\Big[s^3 + s^2\{(\lambda_c + \mu_t) + (\mu_c + \lambda_t) + \lambda_c + \lambda_t\} + s\{(\lambda_c + \mu_t)(\mu_c + \lambda_t) + \lambda_c(\lambda_c + \mu_t) + \lambda_t(\mu_c + \lambda_t) + 2\lambda_c\lambda_t\} + \lambda_c\lambda_t(\lambda_c + \mu_t) + \lambda_c\lambda_t(\mu_c + \lambda_t)\Big]. \tag{7}$$

With a little difficulty, one can find the inverse Laplace transform of $P_i(s)$ in Equations (3), (4), (5) and (6) to find the time-dependent probabilities. Once again, if we define the coefficients of the numerator of $P_i(s)$ as a_{ij} where i = 0, 1, 2, 3 and j corresponds to the j^{th} power of s in the numerator, then from Equations (3), (4), (5) and (6), we have:

$$a_{01} = (\lambda_c + \mu_t)\mu_c + \lambda_t)a_{11} = \lambda_c^2,$$

$$a_{21} = \lambda_t^2 a_{03} = \lambda_c\lambda_t(\lambda_c + \lambda_t),$$

$$a_{02} = (\lambda_c + \mu_t)\mu_c + \lambda_t)a_{12} = \lambda_c,$$

$$a_{22} = \lambda_t a_{31} = 2\lambda_c\lambda_t a_{03} = 1,$$

$$P_0(s) = \frac{s^3 + a_{02}s^2 + a_{01}s}{\Delta}, \tag{8}$$

$$P_1(s) = \frac{a_{12}s^2 + a_{11}s}{\Delta}, \tag{9}$$

$$P_2(s) = \frac{a_{22}s^2 + a_{21}s}{\Delta}, \tag{10}$$

$$P_3(s) = \frac{a_{31}s + a_{30}}{\Delta}. \tag{11}$$

And we have three non-zero real roots as s_1, s_2, and s_3 [60]. Now, taking the Laplace inverse transform, we can easily write:

$$P_0(t) = \sum_{i=0}^{10} \frac{a_{03}s_i^2 + a_{02}s_i + a_{01}}{\prod_{i=1,j\neq i}^{3}(s_i - s_j)} e^{s_i t}, \tag{12}$$

$$P_1(t) = \sum_{i=0}^{3} \frac{a_{12}s_i + a_{11}}{\prod_{i=1,j\neq i}^{3}(s_i - s_j)} e^{s_i t}, \tag{13}$$

$$P_2(t) = \sum_{i=0}^{3} \frac{a_{22}s_i + a_{21}}{\prod_{i=1,j\neq i}^{3}(s_i - s_j)} e^{s_i t}, \tag{14}$$

$$P_3(t) = 1 - \sum_{i=0}^{3} \frac{a_{31}s_i + a_{30}}{s_i \prod_{i=1,j\neq i}^{3}(s_i - s_j)} e^{s_i t}. \tag{15}$$

6 RELIABILITY FUNCTION

The reliability of a system as a function of time t is given as R(T) is the probability that the system operates without failure in the interval [0, t], as respects the system was performing correctly at time 0. λ_c, λ_t are the failure rate that is the expected cost and time failure in task scheduling. During the useful life phase of the system, failure rate parameters are assumed to have a constant value λ_c, λ_t. Then, the reliability of the system varies exponentially as a function of time, shown in formula (16):

$$R(t) = 1 - p_3(t) = \sum_{i=0}^{3} \frac{a_{31}s_i + a_{30}}{s_i \prod_{i=1,j\neq i}^{3}(s_i - s_j)} e^{s_i t}. \tag{16}$$

7 CASE STUDIES

To compare the impact of the proffered reliability model on the task scheduling algorithms in the cloud environment, the reliability of such famous scheduling algorithms in the form of different case studies named BCO [61], HCOC [62], MET [63], MCT [64] and PPO [65] are compared. Due to the popularity of these scheduling algorithms, their code is not mentioned, but a description of the algorithms is provided below.

BCO: In the BCO algorithm, a designation queue is allocated to each resource. The users' requests are set in the designation queue in an ascending order of time required to resource. The resource consumers are registered in the resource queue based on an ascending order from the cheapest one, and the number of customers that each resource can fulfill. These consumers are deleted from the designation queue, and the resources are apportioned to them in turn. This

trend continues until the next round of bidding. More details are described in [61].

HCOC: The Hybrid Cloud Optimized Cost scheduling algorithm. HCOC decides which resources should be leased from the public cloud and aggregated to the private cloud to provide enough processing power to execute a workflow within a given execution time. The Hybrid Cloud Optimized Cost scheduling algorithm. HCOC decides whether a task from a workflow is to be executed in a private resource or in a publicly charged resource. The algorithm assumes that the user stipulates a maximum desired execution time (or deadline) D for the workflow, and our algorithm tries to optimize the monetary execution costs while maintaining the execution time lower than D. Therefore, the objectives of this algorithm are:

1. minimize the monetary costs;
2. minimize the schedule length (makespan); and
3. reduce, over time, the number of schedules with a makespan higher than D; in fact, this algorithm is proposed for clouds with the objective of not only cost optimization but also speeding up the execution of jobs to meet a deadline constraint.

Given a hybrid cloud, it determines which resources should be leased from the public cloud and aggregated to the private cloud to have sufficient processing capacity to execute a workflow within its deadline [33].

MCT: The Minimum Completion Time (MCT) algorithm assigns tasks to VMs or resources based on the best predictable completion time for that task in random order. Each task is assigned to the VM or resource that has the earliest completion time. More details are mentioned in [66].

MET: The Minimum Execution Time (MET) algorithm tasks are assigned to VMs or resources based on the best predictable completion time for each task, without regard to resource availability. More details are mentioned in [66, 67].

PPO: PPO is an offline learning method, and the strategy it adopts to interact with the environment and the strategy to be learned differ. The main idea of the PPO algorithm is to transfer online learning to offline learning based on importance sampling and adopt two networks to improve the network convergence rate [47].

8 EXPERIMENTAL SETUP

Let us presume that the cloud system is composed of several machines (Each host has between 20 and 100 virtual machines). And all experiment results are done for 500 tasks, illustrated in Table 1. To evaluate the proposed algorithm, a Möbius tool [68] is used to simulate and analyze. The Möbius tool currently has two discrete event emulators: a transient simulator and a steady-state simulator.

The transient simulator employs the independent replication technique to collect statistical data on specific reward variables. During the initial phase of the system's operation, focus on short-term performance indicators such as task completion time and system load. In contrast, the steady-state simulator is used for evaluating long-term reliability, using batch means with the deletion of an initial transient period to obtain accurate steady-state variables. The mean and variance from the simulations provide confidence intervals for the results.

For this study, we assumed a cloud environment with the specified number of machines and jobs as illustrated in Table 1 (note that the proposed model can be adapted to any cloud environment with any number of machines and jobs). The reliability model was constructed using a state diagram model (Figure 1) to represent the four possible states of the system (0, 1, 2, 3). The reliability function R(T) was then evaluated for different scheduling algorithms under three scenarios: when the time constraint is violated, when the cost constraint is violated, and when both constraints are violated.

Two distinct cloud environments were simulated: homogeneous and heterogeneous. In the homogeneous environment, all machines had identical processing power, while in the heterogeneous setup, the machines exhibited varying performance, more closely mimicking real-world cloud configurations.

Parameters	Value	Parameters	Value	Parameters	Value
Tasks (cloudlets)		Virtual Machine		Datacenter	
Length of task	1 000–20 000	Number of VMs	20–100	Number of Datacenter	6
Number of task	500	MIPS	500–2 000	Number of Host	3–6
File Size	1–500	VM memory (RAM)	256–2 048		
Output Size	1–500	Bandwidth	500–1 000		

Table 1. Task and resource specifications

The following parameters were used in the experiments:

Number of Tasks (Cloudlets): 500 tasks with lengths ranging from 1 000 to 20 000 units. (The number of tasks was chosen to be 500 to provide a sufficient sample size for accurate performance evaluation. This allows the algorithm to be tested across a variety of task completion times, load conditions, and execution scenarios, ensuring robust results. A larger number of tasks ensures that the effects of outliers and random variability are minimized, providing more reliable statistical conclusions.)

Virtual Machines (VMs): Each host machine supports between 20 and 100 virtual machines. The VMs are configured to simulate different levels of compu-

tational power and memory availability. (The virtual machines were configured with varying processing capabilities (500–2 000 MIPS) and memory sizes (256–2 048 MB) to simulate heterogeneous cloud environments. This variation allows us to evaluate the performance of the scheduling algorithm under different resource conditions, from highly uniform systems to those with significant resource disparities.)

Datacenters: The cloud environment consists of 6 data centers, each with multiple hosts and VMs. (The cloud system was modeled with 6 datacenters to simulate a larger-scale cloud environment, which is more representative of commercial cloud services. The inclusion of multiple datacenters allows the algorithm to be evaluated under conditions of load balancing, resource distribution, and potential network failures, providing insights into the algorithm's scalability and resilience.)

Hosts: Between 3 to 6 physical machines are assumed for each datacenter, and each machine is capable of hosting multiple VMs. (The number of hosts was varied between 3 and 6 to model cloud systems with different levels of hardware availability. This range enables testing the algorithm under both small and medium-sized cloud environments, ensuring that the proposed approach is flexible and scalable across various system configurations.)

MIPS (Million Instructions Per Second): Each machine in the system is equipped with a processing unit that operates between 500 to 2 000 MIPS, which determines the processing speed for task execution.

VM Memory (RAM): Ranges from 256 MB to 2 048 MB, with memory resources allocated to each virtual machine based on workload requirements.

Bandwidth: The network bandwidth between the machines and data centers ranges from 500 to 1 000 Mbps, which affects communication speed and task data transfer times.

9 EXPERIMENTAL RESULT

The reliability of the mentioned algorithms is one of the most important aspects of those algorithms, and it can be defined as the efficiency of the task scheduling service in resource selection based on users' parameters (cost and time). To demonstrate the efficiency of the proposed reliability model, the obtained results were compared with the latest versions of MET, MCT, HCOC, BCO, and PPO algorithms. Based on the above assumption and simulation performed, the following results are obtained.

Figure 3 shows that the reliability of MCT algorithms under a homogeneous Cloud environment is better than other algorithms, and the PPO algorithm has the least reliability in this case.

Figure 4 shows that in the heterogeneous environment, there are still the same results as before (MCT has the highest reliability and PPO is the lowest case).

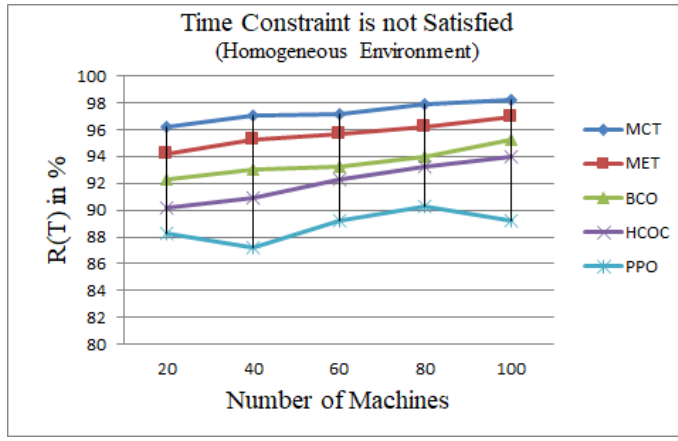

Figure 3. Reliability comparison when Time constraint is not satisfied in homogeneous environment

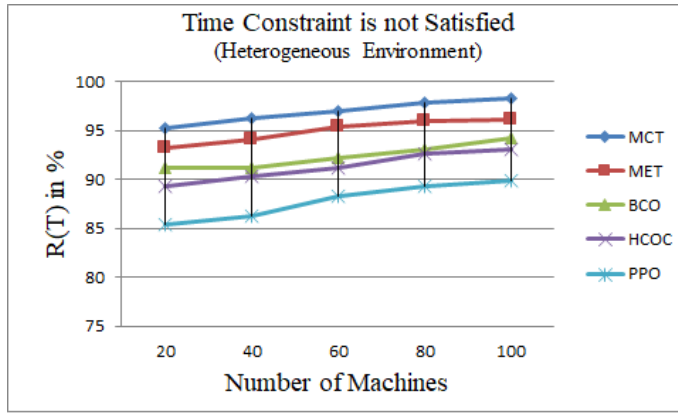

Figure 4. Reliability comparison when Time constraint is not satisfied in heterogeneous environment

Figure 5 shows that when the cost constraint is not satisfied under a homogeneous environment, the BCO algorithm has better reliability, and the MCT algorithm has the weakest reliability.

Figure 6 shows that in the heterogeneous environment HCOC algorithm has better reliability, and the MCT algorithm has the weakest reliability.

And finally, Figure 7 indicates that when the time and cost constraints are not satisfied, BCO has better reliability, and PPO is the lowest case. But in this model, the results are almost close in homogeneous and heterogeneous environments.

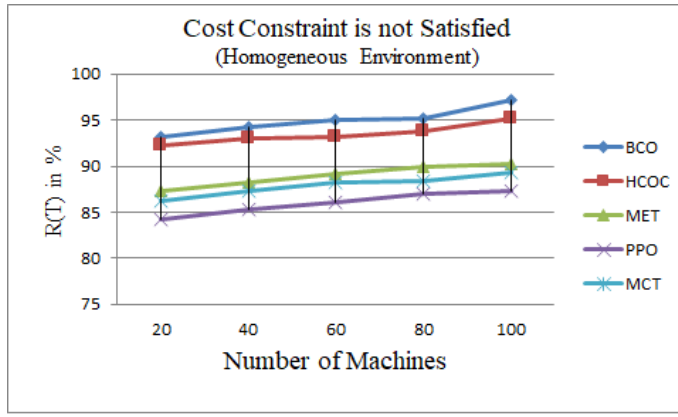

Figure 5. Reliability comparison when Cost constraint is not satisfied in homogeneous environment

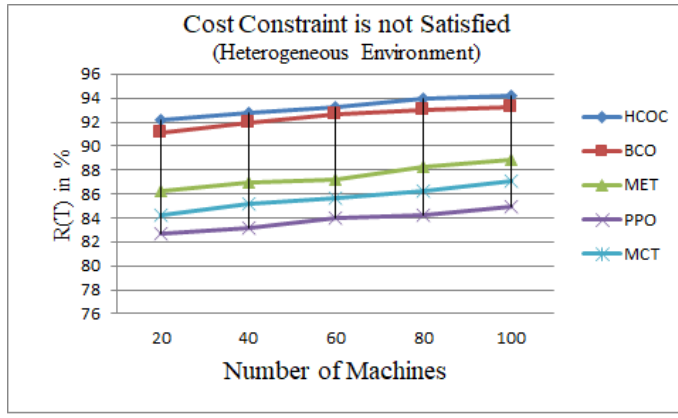

Figure 6. Reliability comparison when Cost constraint is not satisfied in heterogeneous environment

9.1 Detailed Discussion of Experimental Results

The experimental evaluation provides clear insights into the reliability characteristics of different scheduling algorithms under various constraint conditions. When the time constraint was not satisfied, both the MCT and MET algorithms achieved the highest reliability values (approximately 96–97 %) in both homogeneous and heterogeneous cloud environments. This superior performance arises from their fundamental design philosophy, which focuses on minimizing task completion time. Since the proposed Markov-based reliability model quantifies the probability of entering fail-

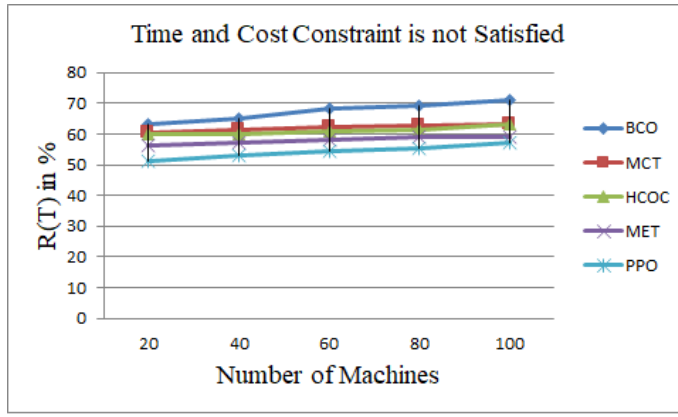

Figure 7. Reliability comparison when the Cost and Time constraints are not satisfied

ure states through the time-related failure rate (λ_t), algorithms emphasizing time minimization naturally achieve lower transition probabilities to failure states. In contrast, the PPO algorithm showed the lowest reliability in this case, mainly due to its reinforcement learning mechanism, which requires iterative exploration and may not adapt quickly under tight time constraints.

In the cost constraint violation scenario, the BCO and HCOC algorithms displayed higher reliability compared to MCT and MET. The BCO algorithm's bidding-based strategy efficiently allocates resources according to both price and workload distribution, effectively reducing cost-related failure rates (λ_c). Similarly, the HCOC algorithm leverages hybrid cloud cost optimization by dynamically balancing resource allocation between private and public clouds, which stabilizes cost performance and improves reliability. The relatively weak reliability of MCT under cost constraints reflects its purely time-focused mechanism, which does not explicitly consider budget factors during task assignment.

When both time and cost constraints were violated, the BCO and MCT algorithms maintained better reliability than the other algorithms, while PPO and MET exhibited the lowest reliability. This suggests that the combination of deterministic scheduling strategies (such as MCT) with adaptive cost control mechanisms (such as BCO) achieves a more balanced trade-off between conflicting QoS parameters. Algorithms that target a single optimization goal or depend on adaptive policy learning (like PPO) tend to face instability when time and cost objectives interact.

Overall, the results demonstrate that the proposed Discrete-Time Markov Chain (DTMC) model effectively captures the probabilistic transitions among success and failure states for each scheduling algorithm. It quantitatively represents how algorithmic design factors – such as time-awareness (MCT/MET), cost-adaptiveness (BCO/HCOC), and learning-based adaptation (PPO) – influence overall system reliability. Moreover, the similarity of results in both homogeneous and heterogeneous

cloud environments verifies the robustness of the proposed reliability model against variations in task and resource characteristics.

In summary, these findings confirm that reliability evaluation should be regarded as an essential criterion in the design of future cloud scheduling algorithms. Integrating reliability-aware models into scheduling frameworks can lead to more stable and predictable performance across diverse cloud computing scenarios.

10 CONCLUSION

Reliability of the cloud environment to the successful execution of users' tasks and service requests are one of the most important QoS factors in the cloud context. A pervasive model is required to evaluate the reliability of the cloud scheduling algorithms and study the impact of the task scheduling algorithms on the entire system reliability. In this paper, a model based on users' restrictions (Time and Cost) has been presented to evaluate the reliability of the famous cloud scheduling algorithms (BCO, MCT, MET, HCOC, and PPO). The model takes into account the failure state of the job scheduling algorithms and cloud resources, influencing the reliability of the system. Overall, we proposed a model that enables the evaluation of the reliability of any scheduling algorithm in cloud environments with respect to time and cost parameters. There are some research backgrounds remaining for future work. One of them can model and proffer other QoS measures, such as the availability and performance of the cloud environments using a discrete-time Markov chain. Modeling the reliability and availability of the cloud scheduling services and cloud resources, and then computing the performance of the cloud environment, is one of the most interesting issues in this field.

REFERENCES

[1] Sunyaev, A.: Cloud Computing. Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies, Springer, Cham, 2024, pp. 165–209, doi: 10.1007/978-3-030-34957-8_7.

[2] Alam, T.: Cloud Computing and Its Role in the Information Technology. IAIC Transactions on Sustainable Digital Innovation (ITSDI), Vol. 1, 2020, No. 2, pp. 108–115.

[3] Liu, J.—Wang, S.—Zhou, A.—Xu, J.—Yang, F.: SLA-Driven Container Consolidation with Usage Prediction for Green Cloud Computing. Frontiers of Computer Science, Vol. 14, 2020, No. 1, pp. 42–52, doi: 10.1007/s11704-018-7172-3.

[4] Deelman, E.—Blythe, J.—Gil, Y.—Kesselman, C.—Mehta, G.—Patil, S.—Su, M. H.—Vahi, K.—Livny, M.: Pegasus: Mapping Scientific Workflows Onto the Grid. In: Dikaiakos, M. D. (Ed.): Grid Computing. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 3165, 2004, pp. 11–12, doi: 10.1007/978-3-540-28642-4_2.

[5] Shahidinejad, A.—Ghobaei-Arani, M.—Masdari, M.: Resource Provisioning Using Workload Clustering in Cloud Computing Environment: A Hybrid Approach. Cluster Computing, Vol. 24, 2021, No. 1, pp. 319–342, doi: 10.1007/s10586-020-03107-0.

[6] Shukur, H. M.—Zeebaree, S. R. M.—Zebari, R. R.—Zeebaree, D. Q.—Ahmed, O. M.—Salih, A. A.: Cloud Computing Virtualization of Resources Allocation for Distributed Systems. Journal of Applied Science and Technology Trends (JASTT), Vol. 1, 2020, No. 2, pp. 98–105, doi: 10.38094/jastt1331.

[7] Amalarethinam, D. I. G.—Muthulakshmi, P.: An Overview of the Scheduling Policies and Algorithms in Grid Computing. International Journal of Research and Reviews in Computer Science, Vol. 2, 2011, No. 2, pp. 280–294.

[8] Abid, A.—Manzoor, M. F.—Farooq, M. S.—Farooq, U.—Hussain, M.: Challenges and Issues of Resource Allocation Techniques in Cloud Computing. KSII Transactions on Internet and Information Systems (TIIS), Vol. 14, 2020, No. 7, pp. 2815–2839, doi: 10.3837/tiis.2020.07.005.

[9] Souravlas, S.—Anastasiadou, S. D.—Tantalaki, N.—Katsavounis, S.: A Fair, Dynamic Load Balanced Task Distribution Strategy for Heterogeneous Cloud Platforms Based on Markov Process Modeling. IEEE Access, Vol. 10, 2022, pp. 26149–26162, doi: 10.1109/ACCESS.2022.3157435.

[10] Zhou, H.—Jiang, K.—Liu, X.—Li, X.—Leung, V. C. M.: Deep Reinforcement Learning for Energy-Efficient Computation Offloading in Mobile-Edge Computing. IEEE Internet of Things Journal, Vol. 9, 2022, No. 2, pp. 1517–1530, doi: 10.1109/JIOT.2021.3091142.

[11] Wang, Y.—Zhou, X.—Zhou, H.—Chen, L.—Zheng, Z.—Zeng, Q.—Cai, S.—Wang, Q.: Transmission Network Dynamic Planning Based on a Double Deep-Q Network with Deep ResNet. IEEE Access, Vol. 9, 2021, pp. 76921–76937, doi: 10.1109/ACCESS.2021.3083266.

[12] Chang, B. J.—Lin, Y.—Hung, W.: Reward-Based Markov Chain Analysis of Slicing Flows for Inter-Cloud Virtual Resources Allocation in 5G Cellular Network. 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2019, pp. 375–381, doi: 10.1109/IEMCON.2019.8936273.

[13] Sun, P.—Wu, D.—Qiu, X.—Luo, L.—Li, H.: Performance Analysis of Cloud Service Considering Reliability. 2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2016, pp. 339–343, doi: 10.1109/QRS-C.2016.52.

[14] Cui, H.—Li, Y.—Liu, X.—Ansari, N.—Liu, Y.: Cloud Service Reliability Modelling and Optimal Task Scheduling. IET Communications, Vol. 11, 2017, No. 2, pp. 161–167, doi: 10.1049/iet-com.2016.0417.

[15] Chowdhury, A.—Tripathi, P.: A Novel Attempt Towards Effective Scheduling Based on Reliability in Cloud Environment. Proceedings of the 2014 International Conference on Information and Communication Technology for Competitive Strategies (ICTCS '14), 2014, doi: 10.1145/2677855.2677891.

[16] Jing, W.—Chen, G.: An Improved Reliability-Driven Task Scheduling Algorithm in Cloud System. 2016 International Conference on Identification, Informa-

tion and Knowledge in the Internet of Things (IIKI), 2016, pp. 167–172, doi: 10.1109/IIKI.2016.54.

[17] Taghinezhad-Niar, A.—Pashazadeh, S.—Taheri, J.: Energy-Efficient Workflow Scheduling with Budget-Deadline Constraints for Cloud. Computing, Vol. 104, 2022, No. 3, pp. 601–625, doi: 10.1007/s00607-021-01030-9.

[18] Meng, S.—Huang, W.—Yin, X.—Khosravi, M. R.—Li, Q.—Wan, S.—Qi, L.: Security-Aware Dynamic Scheduling for Real-Time Optimization in Cloud-Based Industrial Applications. IEEE Transactions on Industrial Informatics, Vol. 17, 2021, No. 6, pp. 4219–4228, doi: 10.1109/TII.2020.2995348.

[19] Ghafari, R.—Kabutarkhani, F. H.—Mansouri, N.: Task Scheduling Algorithms for Energy Optimization in Cloud Environment: A Comprehensive Review. Cluster Computing, Vol. 25, 2022, No. 2, pp. 1035–1093, doi: 10.1007/s10586-021-03512-z.

[20] Guo, X.: Multi-Objective Task Scheduling Optimization in Cloud Computing Based on Fuzzy Self-Defense Algorithm. Alexandria Engineering Journal, Vol. 60, 2021, No. 6, pp. 5603–5609, doi: 10.1016/j.aej.2021.04.051.

[21] Liu, S.—Wang, N.: Collaborative Optimization Scheduling of Cloud Service Resources Based on Improved Genetic Algorithm. IEEE Access, Vol. 8, 2020, pp. 150878–150890, doi: 10.1109/ACCESS.2020.3016762.

[22] Chhabra, A.—Singh, G.—Kahlon, K. S.: Multi-Criteria HPC Task Scheduling on IaaS Cloud Infrastructures Using Meta-Heuristics. Cluster Computing, Vol. 24, 2022, No. 2, pp. 885–918, doi: 10.1007/s10586-020-03168-1.

[23] Argungu, S. M.—Arif, S.—Omar, M. H.: Survey on Job Scheduling Mechanism in Grid Environment. ARPN Journal of Engineering and Applied Sciences, Vol. 10, 2015, No. 15, pp. 6654–6661.

[24] Zahoor, S.—Mir, R. N.: Resource Management in Pervasive Internet of Things: A Survey. Journal of King Saud University – Computer and Information Sciences, Vol. 33, 2022, No. 8, pp. 921–935, doi: 10.1016/j.jksuci.2018.08.014.

[25] Al-Khanak, E. N.—Lee, S. P.—Khan, S. U. R.—Behboodian, N.—Khalaf, O. I.—Verbraeck, A.—van Lint, H.: A Heuristics-Based Cost Model for Scientific Workflow Scheduling in Cloud. Computers, Materials and Continua, Vol. 67, 2021, No. 3, pp. 3265–3282, doi: 10.32604/cmc.2021.015409.

[26] Kardani-Moghaddam, S.—Khodadadi, F.—Entezari-Maleki, R.—Movaghar, A.: A Hybrid Genetic Algorithm and Variable Neighborhood Search for Task Scheduling Problem in Grid Environment. Procedia Engineering, Vol. 29, 2012, pp. 3808–3814, doi: 10.1016/j.proeng.2012.01.575.

[27] Mousavinasab, Z.—Entezari-Maleki, R.—Movaghar, A.: A Bee Colony Task Scheduling Algorithm in Computational Grids. In: Snasel, V., Platos, J., El-Qawasmeh, E. (Eds.): Digital Information Processing and Communication (ICDIPC 2011). Springer, Berlin, Heidelberg, Communications in Computer and Information Science, Vol. 188, 2019, pp. 200–210, doi: 10.1007/978-3-030-11263-0_18.

[28] Toporkov, V.—Bobchenkov, A.—Toporkova, A.—Tselishchev, A.—Yemelyanov, D.: Slot Selection and Co-Allocation for Economic Scheduling in Distributed Computing. In: Malyshkin, V. (Ed.): Parallel Computing Technolo-

gies (PaCT 2011). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6873, 2011, pp. 368–383, doi: 10.1007/978-3-642-23178-0_32.

[29] Toporkov, V.—Toporkova, A.—Bobchenkov, A.—Yemelyanov, D.: Resource Selection Algorithms for Economic Scheduling in Distributed Systems. Procedia Computer Science, Vol. 4, 2011, pp. 2267–2276, doi: 10.1016/j.procs.2011.04.247.

[30] Thanasias, V.—Lee, C.—Hanif, M.—Kim, E.—Sumi, H.: VM Capacity-Aware Scheduling Within Budget Constraints in IaaS Clouds. PLoS ONE, Vol. 11, 2016, No. 8, e0160456 pp., doi: 10.1371/journal.pone.0160456.

[31] Prodan, R.—Wieczorek, M.: Bi-Criteria Scheduling of Scientific Grid Workflows. IEEE Transactions on Automation Science and Engineering, Vol. 7, 2010, No. 2, pp. 364–376, doi: 10.1109/TASE.2009.2014643.

[32] Yu, J.—Buyya, R.: A Budget Constrained Scheduling of Workflow Applications on Utility Grids Using Genetic Algorithms. 2006 Workshop on Workflows in Support of Large-Scale Science, 2006, pp. 1–10, doi: 10.1109/WORKS.2006.5282330.

[33] Bittencourt, L. F.—Madeira, E. R. M.: HCOC: A Cost Optimization Algorithm for Workflow Scheduling in Hybrid Clouds. Journal of Internet Services and Applications, Vol. 2, 2011, No. 3, pp. 207–227, doi: 10.1007/s13174-011-0032-0.

[34] Hu, Z.—Wu, K.—Huang, J.: An Utility-Based Job Scheduling Algorithm for Current Computing Cloud Considering Reliability Factor. IEEE International Conference on Computer Science and Automation Engineering, 2012, pp. 296–299, doi: 10.1109/ICSESS.2012.6269464.

[35] Dabrowski, C.: Reliability in Grid Computing Systems. Concurrency and Computation: Practice and Experience, Vol. 21, 2009, No. 8, pp. 927–959, doi: 10.1002/cpe.1410.

[36] Doğan, A.—Özgüner, F.: Biobjective Scheduling Algorithms for Execution Time-Reliability Trade-Off in Heterogeneous Computing Systems. The Computer Journal, Vol. 48, 2005, No. 3, pp. 300–314, doi: 10.1093/comjnl/bxh086.

[37] He, Y.—Shao, Z.—Xiao, B.—Zhuge, Q.—Sha, E.: Reliability Driven Task Scheduling for Heterogeneous Systems. Proceedings of the Fifteenth IASTED International Conference on Parallel and Distributed Computing and Sytems, 2003, pp. 465–470, `https://www4.comp.polyu.edu.hk/~csbxiao/paper/2003%20and%20before/iasted03.pdf`.

[38] Srinivasan, S.—Jha, N. K.: Safety and Reliability Driven Tasks Allocation in Distributed Systems. IEEE Transactions on Parallel and Distributed Systems, Vol. 10, 1999, No. 3, pp. 238–251, doi: 10.1109/71.755824.

[39] Tang, X.—Li, K.—Li, R.—Veeravalli, B.: Reliability-Aware Scheduling Strategy for Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing, Vol. 70, 2010, No. 9, pp. 941–952, doi: 10.1016/j.jpdc.2010.05.002.

[40] Shao, F. M.—Shen, X.—Ho, P. H.: Reliability Optimization of Distributed Access Networks with Constrained Total Cost. IEEE Transactions on Reliability, Vol. 54, 2005, No. 3, pp. 421–430, doi: 10.1109/TR.2005.853440.

[41] Poola, D.—Ramamohanarao, K.—Buyya, R.: Fault-Tolerant Workflow Scheduling Using Spot Instances on Clouds. Procedia Computer Science, Vol. 29,

2014, pp. 523–533, doi: 10.1016/j.procs.2014.05.047.

[42] Jung, D.—Lim, J.—Yu, H.—Gil, J.—Lee, E.: A Workflow Scheduling Technique for Task Distribution in Spot Instance-Based Cloud. In: Jeong, Y. S., Park, Y. H., Hsu, C. H., Park, J. J. (Eds.): Ubiquitous Information Technologies and Applications (CUTE 2013). Springer, Berlin, Heidelberg, Lecture Notes in Electrical Engineering, Vol. 280, 2014, pp. 409–416, doi: 10.1007/978-3-642-41671-2_52.

[43] Mnih, V.—Badia, A. P.—Mirza, M.—Graves, A.—Lillicrap, T.—Harley, T.—Silver, D.—Kavukcuoglu, K.: Asynchronous Methods for Deep Reinforcement Learning. In: Balcan, M. F., Weinberger, K. Q. (Eds.): Proceedings of the 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research (PMLR), Vol. 48, 2016, pp. 1928–1937, https://proceedings.mlr.press/v48/mniha16.html.

[44] Gu, Y.—Cheng, Y.—Chen, C. L. P.—Wang, X.: Proximal Policy Optimization with Policy Feedback. IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 52, 2022, No. 7, pp. 4600–4610, doi: 10.1109/TSMC.2021.3098451.

[45] Chavali, L.—Gupta, T.—Saxena, P.: SAC-AP: Soft Actor Critic Based Deep Reinforcement Learning for Alert Prioritization. 2022 IEEE Congress on Evolutionary Computation (CEC), 2022, pp. 1–8, doi: 10.1109/CEC55065.2022.9870423.

[46] Cassandra, A. R.: A Survey of POMDP Applications. Working Notes of AAAI 1998 Fall Symposium on Planning with Partially Observable Markov Decision Processes, 1998, https://pomdp.org/papers/applications.pdf.

[47] Mnih, V.—Kavukcuoglu, K.—Silver, D.—Rusu, A. A.—Veness, J. et al.: Human-Level Control Through Deep Reinforcement Learning. Nature, Vol. 518, 2015, No. 7540, pp. 529–533, doi: 10.1038/nature14236.

[48] Kumar, A.—Rani, K.—Kumar, S.: An Overview of Cloud Scheduling Algorithms. Vidyabharati International Interdisciplinary Research Journal, Special Issue on Emerging Techniques in Interdisciplinary Sciences, 2021, pp. 2778–2782.

[49] Yahia, H. S.—Zeebaree, S. R. M.—Sadeeq, M. A. M.—Salim, N. O. M.—Kak, S. F.—Al-Zebari, A.—Salih, A. A.—Hussein, H. A.: Comprehensive Survey for Cloud Computing Based Nature-Inspired Algorithms Optimization Scheduling. Asian Journal of Research in Computer Science, Vol. 8, 2021, No. 2, pp. 1–16, doi: 10.9734/ajrcos/2021/v8i230195.

[50] Zhu, Q. H.—Tang, H.—Huang, J. J.—Hou, Y.: Task Scheduling for Multi-Cloud Computing Subject to Security and Reliability Constraints. IEEE/CAA Journal of Automatica Sinica, Vol. 8, 2021, No. 4, pp. 848–865, doi: 10.1109/JAS.2021.1003934.

[51] Amini Motlagh, A.—Movaghar, A.—Rahmani, A. M.: Task Scheduling Mechanisms in Cloud Computing: A Systematic Review. International Journal of Communication Systems, Vol. 33, 2020, No. 6, Art. No. e4302, doi: 10.1002/dac.4302.

[52] Sahni, J.—Vidyarthi, D. P.: A Cost-Effective Deadline-Constrained Dynamic Scheduling Algorithm for Scientific Workflows in a Cloud Environment. IEEE Transactions on Cloud Computing, Vol. 6, 2015, No. 1, pp. 2–18, doi: 10.1109/TCC.2015.2451649.

[53] Stewart, W. J.: MARCA: Markov Chain Analyzer, A Software Package for Markov Modeling. Numerical Solution of Markov Chains, CRC Press, 1991, pp. 37–61.

[54] Bhattacharya, R. N.—Waymire, E. C.: Stochastic Processes with Applications. SIAM, 2009.

[55] Vermeer, S.—Trilling, D.: Toward a Better Understanding of News User Journeys: A Markov Chain Approach. Journalism Studies, Vol. 21, 2020, No. 7, pp. 879–894, doi: 10.1080/1461670X.2020.1722958.

[56] Balagurusamy, E.—Misra, K. B.: Reliability & Mean Life of a Parallel System with Nonidentical Units. IEEE Transactions on Reliability, Vol. R-24, 1975, No. 5, pp. 340–341, doi: 10.1109/TR.1975.5214926.

[57] Misra, K. B.—Balagurusamy, E.: A Note on Sandler's Dependency Model. IEEE Transactions on Reliability, Vol. R-24, 1975, No. 5, pp. 343–343, doi: 10.1109/TR.1975.5214928.

[58] Misra, K. B.—Prasad, P.: Comment on "Reliability Evaluation of a Flow Network". IEEE Transactions on Reliability, Vol. R-31, 1982, No. 2, pp. 174–176, doi: 10.1109/TR.1982.5221288.

[59] Gadani, J. P.—Misra, K. B.: Network Reliability Evaluation of Three-State Devices Using Transformation Technique. Microelectronics Reliability, Vol. 21, 1981, No. 2, pp. 231–234, doi: 10.1016/0026-2714(81)90394-2.

[60] Gadani, J. P.—Misra, K. B.: Reliability Evaluation of a System with Imperfect Nodes and Links Using Network Approach. Systems Science, Vol. 5, 1979, No. 3, pp. 265–274.

[61] Buyya, R.: Economic-Based Distributed Resource Management and Scheduling for Grid Computing. Ph.D. Thesis. 2002, doi: 10.48550/arXiv.cs/0204048.

[62] Sarhadi, A.—Torkestani, J. A.: Cost-Effective Scheduling and Load Balancing Algorithms in Cloud Computing Using Learning Automata. Computing and Informatics, Vol. 42, 2023, No. 1, pp. 37–74, doi: 10.31577/cai_2023_1_37.

[63] Asghari Alaie, Y.—Hosseini Shirvani, M.—Rahmani, A. M.: A Hybrid Bi-Objective Scheduling Algorithm for Execution of Scientific Workflows on Cloud Platforms with Execution Time and Reliability Approach. The Journal of Supercomputing, Vol. 79, 2023, No. 2, pp. 1451–1503, doi: 10.1007/s11227-022-04703-0.

[64] Alsaidy, S. A.—Abbood, A. D.—Sahib, M. A.: Heuristic Initialization of PSO Task Scheduling Algorithm in Cloud Computing. Journal of King Saud University – Computer and Information Sciences, Vol. 34, 2022, No. 6, Part A, pp. 2370–2382, doi: 10.1016/j.jksuci.2020.11.002.

[65] Lin, L.—Pan, L.—Liu, S.: SpotDAG: An RL-Based Algorithm for DAG Workflow Scheduling in Heterogeneous Cloud Environments. IEEE Transactions on Services Computing, Vol. 34, 2024, No. 5, pp. 2904–2917, doi: 10.1109/TSC.2024.3422828.

[66] Freund, R. F.—Gherrity, M.—Ambrosius, S.—Campbell, M.—Halderman, M.—Hensgen, D.—Keith, E.—Kidd, T.—Kussow, M.—Lima, J. D.—Mirabile, F.—Moore, L.—Rust, B.—Siegel, H. J.: Scheduling Resources in Multi-User, Heterogeneous Computing Environments with SmartNet. Proceedings Seventh IEEE Heterogeneous Computing Workshop (HCW '98), 1998, pp. 184–199, doi: 10.1109/HCW.1998.666558.

[67] Armstrong, R.—Hensgen, D.—Kidd, T.: The Relative Performance of Various Mapping Algorithms Is Independent of Sizable Variances in Run-Time Predictions.

Proceedings Seventh IEEE Heterogeneous Computing Workshop (HCW '98), 1998, pp. 79–87, doi: 10.1109/HCW.1998.666547.

[68] Norling, M. D.—Jackson-Blake, L. A.—Calidonio, J. L. G.—Sample, J. E.: Rapid Development of Fast and Flexible Environmental Models: The Mobius Framework v1.0. Geoscientific Model Development, Vol. 14, 2021, No. 4, pp. 1885–1897, doi: 10.5194/gmd-14-1885-2021.

Ali Sarhadi received his B.Sc. and M.Sc. degrees in computer software engineering. He was graduated with Ph.D. degree of software systems (learning automata, cloud computing, scheduling, deep learning). He is Associate Professor of computer engineering with his Ph.D. in artificial intelligence. He is a faculty member with extensive research and teaching experience in machine learning, intelligent systems, cloud computing, and expert systems. His research has particularly focused on cost-effective job scheduling, load balancing, and reliability modeling in cloud computing environments. He has published more than 40 peer-reviewed journal articles in reputable international journals. His work has contributed to the development of learning-based and economically efficient resource management algorithms for large-scale distributed and cloud systems.

Abbas Karimi received his B.Sc. degree in computer hardware engineering and his M.Sc. degree in computer software engineering. He graduated with a Ph.D. degree in computer system engineering (neuro-fuzzy, AI, and fault-tolerant). He received a postdoctoral of smart mobile networks and a postdoctoral of medical imaging systems from UPM, Malaysia. He is currently working as the Vice Chancellor of Education and Research and Associate Professor with the Department of Artificial Intelligence, IAU-Karaj and Dubai Branch. Apart from lecturing and supervising students, he also plays an active role in research and development. He was the leader of multiple research projects, and the author of three textbooks, many journals and actively participated in seminars and conferences conducted at national and international levels. He is a senior member of IACSIT (International Association of Computer Science and Information Technology), senior member of IEEE (the Institute of Electrical and Electronics Engineers).

Ali Yousefi received his B.Sc. in computer software from the IAU Qazvin Branch, Qazvin, Iran, in 1999 and his M.Sc. degree in artificial intelligence from the IAU Science and Research Branch, Tehran, Iran, in 2004. He received his Ph.D. degree in artificial intelligence from the IAU Science and Research Branch, Tehran, Iran, in 2023. Currently, he is Lecturer and Head of Computer Engineering Department of the IAU Hamedan Branch. He is interested in a wide range of topics that fall under computer science, artificial intelligence, soft computing, cognitive science and machine learning.