A Learning Automata-based Scheduling for Deadline Sensitive Task in The Cloud

Sampa Sahoo, *Member, IEEE,* Bibhudatta Sahoo, *Member, IEEE,* and Ashok Kumar Turuk, *Member, IEEE*

Abstract—The evolution of cloud computing facilitates applications with varying demands to operate in a virtualized environment. For instance, applications like the healthcare system, video streaming, Internet of Things (IoT) that are moving to the cloud, demand responses within a particular time limit, i.e., deadline. However, the cloud computing system consumes a considerable amount of electric energy while providing services to these type of applications, which in turn contribute to the high operational cost. Specifically, it becomes cumbersome to offer services to deadline sensitive task while minimizing energy consumption. In this regard, efficient task scheduling is an attractive way to cut down energy usage while ensuring satisfactory services for cloud users. In this paper, the task scheduling problem is considered as a bi-objective minimization problem which includes minimization of energy consumption and makespan. First, we proposed a novel learning automata-based scheduling framework for deadline sensitive tasks in the cloud. Learning automata (LA) is an adaptive decision-making unit that helps the scheduler to select the best responses. Later, the LA-based Scheduling (LAS) algorithm is introduced which exploits the heterogeneity of tasks and virtual machines (VMs) while ensuring the timing requirements of the tasks. Extensive simulation is carried out to designate the effectiveness and applicability of LAS for deadline sensitive task scheduling in the heterogeneous cloud environment.

Index Terms—Cloud computing, Deadline, Energy consumption, Makespan, Learning Automata (LA)

1 Introduction

Cloud computing is a revolutionary paradigm, which uses data centers to provide services over the Internet. Virtualization is the key technology behind the cloud that creates an illusion of infinite computing resources. Running applications on virtual cloud resources (or virtual machines) contribute to the scalability and cost-efficiency feature of the cloud. Further, cloud-based services like SaaS, PaaS, and IaaS intend to extend support for a wide range of applications. Noticeably many applications, e.g., healthcare system, video streaming, IoT, the financial transaction system employed in the virtualized cloud are with real-time nature, in which the response to a request is a function of both computational outcome and time instant at which it is produced. As these applications need a timely response, depending on their nature (whether emergent or not) the cloud system decides whether to expand or shrink the count on cloud resources. For instance, a task of a healthcare application demand guarantee of timeliness strictly whereas application like video streaming can withstand some relaxation in timing constraint. So, to procure the need for such applications a cloud service provider must provide a sufficient number of cloud resources that satisfy the timing requirement. Meanwhile, the evergrowing demand of applications forces a Cloud Service provider (CSP) to deploy more and more cloud resources. Inevitably, the massive count of cloud resources in cloud data center consumes a tremendous amount of energy [1], [2]. Further, the report presented in [2], [3] also states that data centers consume 1.5 % of global electric energy in the year 2010, which will be doubled by 2020 if current trends continue.

The following facts can attribute the reasons for high energy consumption in cloud data centers; (i) Some computing resources are inevitably idle during different time slots which indeed lowers resource utilization and raises energy consumption. In [1], [2], [4] low utilization of computing resources is affirmed as one of the crucial element contributing to high energy consumption. Moreover, a study presented in [1], [5] shows that average resource utilization in a cloud system is not more than 30 %, but the energy used by idle resources is at least 60-70 % of peak energy. (ii) To meet the worst-case demand of users cloud resources are over-provisioned. The significantly higher number of resources reservation, in turn, increases the energy consumption. Besides inefficient and improper resource scheduling leads to the selection of VMs that will cause high energy consumption while ensuring QoS demand (e.g., deadline) of the applications. But, high energy consumption has a negative impact on the environment. Further, the high energy consumption also affects the operational cost of the CSP, as the rise in energy usage raises the electric cost. Reliability of the system is also affected by high energy consumption. This can be supported by Arrhenius life-stress model which says that "for every 10 ^oC increase in temperature, the failure rate of electric devices rises by a factor of two" [1]. Hence, it is hugely requisite to employ some means to lessen the energy consumption of the cloud system and make it energy efficient.

In this regard, scheduling plays a vital role in deadline sensitive application. These applications need task completion within the deadline while efficiently utilizing cloud resources [7], [8]. More specifically, the task scheduling problem can be defined as follows; for a given set of VMs

- *S. Sahoo, B. D. Sahoo and A. K. Turuk are with the Department of Computer Science and Engineering, National Institute of Technology, Rourkela, Odisha, India.*
E-mail:{sampaa2004, bibhudatta.sahoo, akturuk}@gmail.com

and tasks, we need to obtain the best mapping of tasks to VMs such that the system performance metrics like energy consumption, makespan, etc. can be optimized. But the task scheduling problem becomes more challenging for the multi-objective case, i.e., when more than one contrary metrics need to be simultaneously optimized. For instance, to minimize the cost and response time simultaneously, the scheduler must make the best decision possible otherwise reduction in response time may lead to a rise in cost or vice-versa. Furthermore, deadline constraint of applications and heterogeneity of cloud resources contribute to the complexity of scheduling. In this context, we used learning automata (LA) concept to solve bi-objective deadline sensitive task scheduling problem in the heterogeneous cloud environment. LA is an adaptive decision-making unit that learns from its action applied to a dynamic environment [9], [10], [11], [12]. The reinforcement based learning scheme of LA helps to get optimal action from a set of possible actions. This feature of LA makes it a suitable candidate for finding the best <task, VM > pair from a set of possible < task, VM > set for a task in the dynamic cloud environment.

The major contributions of this paper are as follows:

- We introduced an LA-based scheduling framework for deadline sensitive tasks in the cloud environment.
- The task scheduling problem is formulated as a bi-objective minimization problem which includes minimization of energy consumption and makespan. Later, we presented LA-based Scheduling (LAS) algorithm for finding the solution of the bi-objective minimization problem.
- An extensive set of experiments were performed to show the effectiveness of LAS over its peers. The comparison is evaluated in terms of energy consumption, makespan and success ratio.

The rest of the paper is organized as follows: Section 2 presents an overview of various works done by researchers, precisely focusing on deadline sensitive task scheduling and use of LA-based approach. Section 3 discusses the LA concept. Section 4 presents the cloud system framework which includes a VM and task model, energy model, LA model, and scheduling model. In Section 5, LAS algorithm is introduced based on the scheduling framework discussed in Section 4. Section 6 discusses the performance evaluation of the proposed approach with some existing approaches. Finally, the paper is ended with concluding remarks in Section 7.

2 Related Work

In this section, we present a review of existing studies on (i) performance metric based task scheduling, (ii) use of the LA-based approach.

2.1 Performance Metric based Task Scheduling

Scheduling plays a significant role in achieving high performance for applications running in clouds. Usually, scheduling algorithms are designed to optimize performance metrics like energy consumption, makespan, cost, etc. In this study, we mainly focus on energy-aware scheduling algorithms for deadline sensitive task in the cloud. Zhu *et al.* [1] developed an Energy-Aware Scheduling Algorithm (EARH) for the real-time independent task to optimize guarantee ratio, energy consumption, and resource utilization. Chen *et al.* [2] presented Energy-efficient Online Scheduling Algorithm (EONS) with the aim of achieving energy efficiency and better resource utilization for real-time workflows in the cloud data center. Mishra *et al.* [4] presented metaheuristic based service allocation framework to find the trade-off between energy consumption and makespan. Gao *et al.* [5] proposed "Guided Migrate and Pack" (GMaP) scheduling framework based on VM migration to maximize energy efficiency while reducing SLA violation due to deadline miss. Yassa *et al.* [6] used Dynamic Voltage and Frequency Scaling (DVFS) and Particle Swarm Optimization (PSO) techniques to optimize makespan, cost, and energy for workflow scheduling. In [17], a dynamic VM consolidation technique is used to realize energy efficiency in the cloud data center without committing SLA violations. To save energy, authors in [18] presented EEVS scheduling algorithm of VMs. First, they find the optimal performance-power ratio of Physical Machine (PM) and then assign VM to PM with the highest ratio.

Li *et al.* [19] proposed a failure aware energy efficient scheduling algorithm for the cloud data center considering computing and cooling energy, and the reliability of servers. Xing *et al.* [20] utilized VM migration and resource fairness concepts to optimize energy usage for IoT applications in the cloud environment. A greedy scheduling algorithm named Most Energy Efficient First (MESF) presented in [21] saves energy by assigning a task to the most efficient server based on energy profile. Li *et al.* [22] proposed an algorithm named CEAS to reduce execution cost and energy consumption of scientific workflows in the cloud. DVFS technique is incorporated in the algorithm to conserve energy. koodziej *et al.* [24] employed DVFS model and Genetic Algorithm (GA) for solving energy-aware scheduling problem in computational grid. In [25], authors have proposed HARMONY, a heterogeneity aware resource management system to decrease the total energy consumption and performance penalty. Whereas in [29], the scheduler allocates a real-time task to the processing node in a probabilistic way aiming at decreasing the average energy per successful task. DVFS technique is used in [33], [35] to find the tradeoff between performance and energy efficiency. The work presented in [37] minimizes energy consumption by turning off the most effective processor from energy saving perspective.

Panda *et al.* [23] proposed scheduling techniques based on min-max, and median max to reduce makespan while improving average cloud utilization in a multi-cloud environment. Stavrinides *et al.* [26] used the Earliest Deadline First (EDF) and best fit theory to guarantee execution of applications within deadline constraint while reducing makespan and cost charged to the user. Cai *et al.* [27] introduced a scheduling algorithm to minimize the resource renting cost while meeting workflow deadline. A regression model is used by researchers in [30] for provisioning cloud resources to find the tradeoff between cost saving and QoS requirement. Zhang *et al.* [28] used Bayes classifier to classify task based on historical scheduling data. They proposed a two-stage scheduling scheme whose performance is measured concerning makespan, waiting time and utilization of VM. Sahoo *et al.* [32] proposed EDF based algorithm

using best fit, worst fit and first fit concept to optimize guarantee ratio, VM utilization, and throughput while guaranteeing deadline constraint. Chen *et al.* [8] considered the uncertainty of task execution time and data transfer time in their scheduling algorithm for workflows in the cloud. Zhu *et al.* [7] developed a fault-tolerant task scheduling using primary-backup concept for real-time workflows in a virtualized cloud. Quang-Hung *et al.* [15] proposed a GA based virtual machine allocation scheme to minimize the total energy consumption of computing servers. From this survey, it is concluded that in the last few years a sizable amount of research has been conducted on task scheduling .using various performance metrics.

2.2 Use of LA-based Approach

LA theory is appropriate for the environment which is dynamic, complex, and there is a large number of uncertainties like cloud environment, computer networks, etc. Narendra *et al.* [9] presented a survey in the area of learning automata. Their study mainly focused on norms and behavior of learning automata, reinforcement/learning schemes, the convergence of learning algorithm, the interaction of several automata. Various applications of learning automata are also discussed like parameter optimization and decision making. Authors in [15] discussed how learning automata behave with changing the number of actions. Misra *et al.* [10] proposed an LA-based framework to improve the performance of QoS-enabled cloud services concerning response time and speed-up. Rezvanian *et al.* [11] used LA to find the solution of the minimum vertex-covering problem in a stochastic graph. Ranjbari *et al.* [12] proposed an algorithm based on LA to detect the overloaded PM. The prevention from PM overload reduces VM migration count and helps consolidation of VMs, which indeed minimizes energy consumption. In [13] authors have utilized LA theory to develop a prediction model for cloud resource usage. An LA-based ranking algorithm is introduced in [14], where a learning automaton ranks the search documents based on user feedback. Venkataramana *et al.* [16] proposed LA-based task assignment architecture for a heterogeneous computing system to achieve load balancing and minimum total execution time. Authors in [36] used LA concept to minimize the energy consumption in a heterogeneous cloud environment.

From the above study, we can infer that VM migration and DVFS technique for energy saving have been well studied separately by the researchers. Also, some researchers used VM consolidation, nature-inspired approach, and greedy approach for saving energy. The diversified approaches and need of the energy-efficient system allows one to explore different methods to achieve the desired goal. Further, a considerable amount of work has been done considering benefits of LA approach, but few of them deal with energy-aware scheduling of independent deadline sensitive tasks in the heterogeneous cloud environment. In this context, we employed LA concept to design a scheduling framework for deadline sensitive tasks in the heterogeneous cloud environment. Authors in [16] also presented a task scheduling framework, however, the main distinction between their work and ours is two-fold. First, the absence of the virtualization concept in their work, whereas in our work VM is the basic computation unit. Second, our work takes deadline of the task into account while scheduling, which was not considered in earlier work.

3 Learning Automata

A learning automaton act as an adaptive decision-making unit that learns to pick the best action from a set of allowed actions through repeated interaction with an uncertain environment $[9] - [16]$, [36]. Besides, learning automaton can cooperate to find the solution of many hard-to-solve problems like adaptive control, grid computing, combinatorial optimization problem, etc. The actions are chosen based on a probability distribution kept over the action set and served as the input to the random environment. The environment generates a reinforcement signal for each action. Based on this signal value, the action probability vector is updated. This process continues until the stopping criterion (i.e., maximum iteration count or probability value approach a threshold limit) is reached. Fig. 1 shows the relationship between environment and learning automata. Based on the reinforcement signal, an environment can be classified into P-model, Q-model, and S-model. In P-model environment, reinforcement signal can take two binary values 0 and 1, where 0 indicates favorable response and 1 shows an unfavorable response. On the other hand, Q-model allows reinforcement signal to have a finite number of values in the interval [0, 1]. In S-model environment, reinforcement signal lies in the interval [0,1] as given in [12], [14]. LA can be of two types: fixed structure and variable structure. In a variable structure LA (VLA), the number of actions available at each instant changes with time, which is constant for fixed structure LA (FLA). Since, in the cloud the number of requests for service changes over time, a VLA is appropriate to represent it.

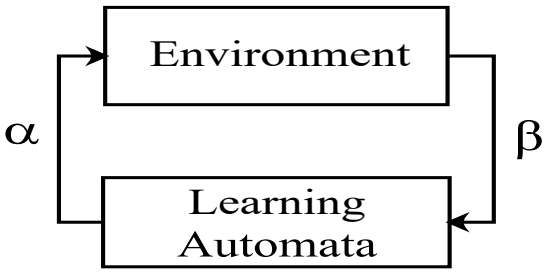


Fig. 1: Learning automata and Environment

A VLA can be defined by quadruple $< P, \alpha, \beta, L >$, where α is the action set, β is the set of inputs, P is the action probability set, and L is the learning or reinforcement algorithm. The learning algorithm is a recurrence relation adopted to revise the action probability vector. It can be described by a linear or non-linear or hybrid function. Let, r be the number of actions that can be taken by a LA. The action to be performed in the environment is denoted by α, where $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ and the reinforcement signal from the environment is indicated by β, where $\beta \equiv \{\beta_1, \beta_2, ..., \beta_r\}$. At instant k, $\alpha_i(k) \in \alpha$ denotes action taken by LA and $p_i(k) \in P$ is the probability value defined for action $\alpha_i(k)$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2906870, IEEE Transactions on Services Computing

The reward (ϕ) and penalty (φ) constants are used to manipulate action probabilities. The linear learning algorithm used to update p_i is as follows:

$$p_j(k+1) = \begin{cases} p_j(k) + \phi \times (1 - p_j(k)) & j = i \\ (1-\phi) \times p_j(k) & \forall j, j \neq i \end{cases} \quad (1)$$

$$p_j(k+1) = \begin{cases} (1-\varphi) \times p_j(k) & j = i \\ \frac{\varphi}{r-1} + (1-\varphi) \times p_j(k) & \forall j, j \neq i \end{cases} \quad (2)$$

If the selected action $\alpha_i(k)$ is favorable to the environment, then $\beta_i = 0$ and it is rewarded using Equation 1. Equation 1 states that increase the probability of $\alpha_i(k)$, whereas decrease the probability of other actions for the subsequent instant. Similarly, Equation 2 is used to penalize an action if it is unfavorable to the environment. It states that, decrease the probability of selected action while increasing the probability of other actions for the subsequent instant. Based on the values of ϕ and φ learning algorithm can be classified as follows: if $\phi = \varphi$, then it is called linear reward penalty (L_{R-P}) algorithm. When $\phi \gg \varphi$ it is reward-ϵ-penalty ($L_{R\epsilon P}$), and if $\varphi = 0$, it is called reward-inaction (L_{R-I}) algorithm. The primary goal of LA is to select the optimal action from a set of possible actions, similar to it, a task scheduling problem in cloud corresponds to pick the best mapping of the task to VM from a possible mapping set. Further, the reinforcement based learning scheme with very little historical information adds the suitability of LA for solving the task scheduling problem.

4 Cloud System Framework

In this section, we will discuss the models and terminologies used in this paper. The proposed scheduling framework as shown in Fig. 2 is designed for deadline sensitive task scheduling in the heterogeneous cloud environment. Various components of the framework are a task and VM model, energy model, LA model, and scheduling model. The cloud scheduler uses models mentioned above for allocation of tasks to different VMs. Various symbols used are listed in Table 1.

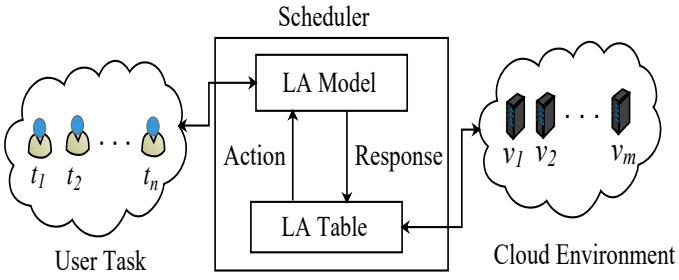


Fig. 2: Scheduling Framework

4.1 VM Model

The virtualized cloud environment is characterized by VM set $V = \{v_1, v_2, ..., v_m\}$ which provides computing infrastructure for processing user tasks (or requests). Each element v_j in V represents VM ID. The heterogeneous set of VM is realized in terms of its computing capability or speed, sp_j, $j = \{1, 2, 3, ..., m\}$. The widely-used metric, Million Instructions Per Second (MIPS) is used to measure the computing capability of the VM [1], [7], [27], [33], [34], [35].

TABLE 1: Symbols Used

Symbol	Description
t_i	Task ID of i^{th} task
v_j	VM ID of j^{th} VM
sp_j	Speed of v_j
etc_i^j	Expected execution time of t_i on v_j
st_i^j	Start time of t_i on v_j
ft_i^j	Finish time of t_i on v_j
tc_i^j	Binary variable indicating whether execution of t_i on v_j meet deadline or not
ξ_{a_j}	Energy consumption of v_j in active state
η_{l_j}	Energy consumption of v_j in idle state
ϱ_j	Total energy consumption of v_j
τ	Makespan
δ	Aggregated energy consumption of the cloud system
μ_j	Total execution time of tasks assigned to v_j
X_i^j	Binary variable indicating if t_i is assigned to v_j or not
A_i	Automaton associated with t_i
α_i	Action set of A_i
p_i	Action probability vector for α_i
β_i	Reinforcement signal for α_i
L	Learning algorithm
α_i^j	Action denoting t_i is assigned to VM v_j
$p_i^j(k)$	Probability value for α_i^j at iteration k
ϕ	Reward constant
φ	Penalty constant
$\omega(k)$	Cost metric at iteration k
θ_{mn}	Minimum value of p_i^j
θ_{mx}	Maximum value of p_i^j

4.2 Task Model

Let, $T = \{t_1, t_2, ..., t_n\}$ is the set of independent tasks that arrive dynamically. Each element t_i, $i = \{1, 2, ..., n\}$ in T represents task ID. A task t_i submitted by a user ($User_i$) is modeled by following parameters; $t_i = \{a_i, sz_i, dl_i\}$, where a_i, sz_i and dl_i are the arrival time, task size (in terms of Million Instruction (MI)), and deadline of task t_i respectively. The heterogeneity of task and VMs are modeled as the estimate of the expected execution time for each task on each VM, which is known before the execution. Let etc_i^j be

the expected execution time of t_i on VM v_j. It is computed as:

$$etc_i^j = \frac{sz_i}{sp_j} \tag{3}$$

etc_i^j is contained within a $n \times m$ expected time to compute (ETC) matrix, where n is the number of tasks, and m is the number of VMs. One row of ETC matrix contains the estimated execution time of a task on each VM [4]. We assume that a new task is affixed to the end of previously allocated tasks on a VM. Let st_i^j and ft_i^j are the start and finish time of task t_i on VM v_j respectively. The start time st_i^j can be computed as below:

$$st_i^j = max\{ft_p^j, a_i\} \tag{4}$$

Equation 4 says that a new task t_i can start its execution on v_j, either after the completion of the previously assigned task t_p on it or just after the arrival, whichever is earlier. Finish time of a task on a particular VM is calculated by adding start time and expected execution time on that VM. It is calculated as follows:

$$ft_i^j = st_i^j + etc_i^j \tag{5}$$

The finish time is employed to ascertain whether the task's timing constraint (dl_i) can be guaranteed or not. Let, binary variable tc_i^j indicate whether a task t_i executing on v_j met its timing constraint or not. If t_i met the constraint then tc_i^j takes value "1" and is "0" otherwise. Mathematically, it is shown as:

$$tc_i^j = \begin{cases} 0 & if \;\; ft_i^j > dl_i \\ 1 & if \;\; ft_i^j \leq dl_i \end{cases} \tag{6}$$

4.3 Energy Model

The energy consumed in the cloud system is mostly from the execution environment, cooling system, and power conditioning [1], [4]. The execution environment consists of VM and is the basis of our energy model. Generally, energy consumption of a VM depends on it's state. A VM can be in an active or idle state. We assume that a VM is said to be active when it is executing a task otherwise it is idle. Usually, energy consumed by a VM in idle state is 60-70% of the active state [1], [4], [5]. Let, ξ_{a_j} and η_{l_j} denotes the energy consumption of VM v_j in the active and idle state respectively. Total energy consumption (ϱ_j) of VM v_j is estimated considering both active and idle state energy usage. It's calculation is shown in Equation 7.

$$\varrho_j = (\xi_{a_j} + \eta_{l_j}) \times sp_j \tag{7}$$

Total execution time (μ_j) of all the tasks assigned to v_j is calculated as follows:

$$\mu_j = \sum_{i=1}^{n} X_i^j \times etc_i^j \tag{8}$$

where binary variable $X_i^j = 1$, if t_i is assigned to v_j. If t_i is not assigned to v_j then $X_i^j = 0$. Makespan (τ) is defined as the maximum execution time among all the VMs. It is computed as:

$$\tau = Max(\mu_j) \tag{9}$$

The active and idle state energy consumption of v_j is computed as follows:

$$\xi_{a_j} = \mu_j \times \sigma_j \tag{10}$$

$$\eta_{l_j} = (\tau - \mu_j) \times 0.6 \times \sigma_j \tag{11}$$

where $\sigma_j = 10^{-8} \times (sp_j)^2$ Joules/MI [4], [29], [31]. The aggregate energy consumption (δ) of the cloud system is evaluated as:

$$\delta = \sum_{j=1}^{m} \varrho_j \tag{12}$$

4.4 LA Model

In this work, LA is used to generate a scheduling decision that will optimize the specified objectives. We assume that there are n number of tasks and m number of VMs in the cloud system, whose values may vary over the time. The proposed VLA model is presented in Fig. 3. Each incoming task is associated with an automaton, and a set of VMs realizes the random environment. The VLA is represented by a sextuple $< t_i, A_i, \alpha_i, p_i, \beta_i, L >$, where t_i indicates i^{th} task, A_i is the learning automaton associated with t_i, α_i is the action set of A_i, p_i is the action probability vector corresponding to α_i, β_i is the reinforcement signal from the environment for action α_i and L is the learning algorithm. The action set of an automaton is represented by the probable set of VM assignment possible for execution of a task. Hence, $\alpha_i = \{\alpha_i^j | 1 \leq j \leq m\}$ is the action set of A_i, where α_i^1 means task t_i is assigned to VM v_1, α_i^2 means t_i is assigned to v_2 and so on. The action probability vector

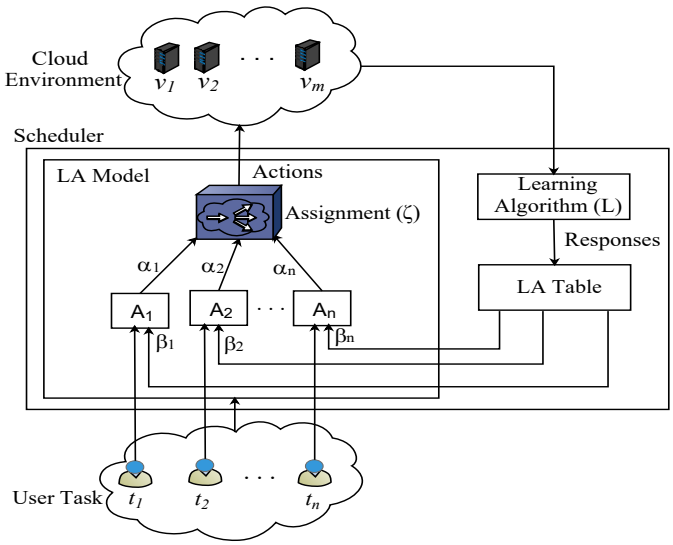


Fig. 3: Proposed VLA Model

$p_i = \{p_i^j | \alpha_i^j \in \alpha_i\}$, where p_i^1 is the probability value of action α_i^1, p_i^2 is the probability of action α_i^2 and so on. At any instant, action with highest probability is chosen. The expression for p_i can be rewritten as $p_i = max\{p_i^1, p_i^2, ..., p_i^m\}$. Hence, $\alpha_i = \{\alpha_i^j | p_i^j = max(p_i)\}$. The reinforcement signal β_i can have binary values 0 and 1. Let $\omega(k)$ represents our bi-objective minimization equation at iteration k.

The learning algorithm is defined as follows: If $\omega(k) \leq \omega(k-1)$, then check whether action α_i taken by automaton A_i meet the deadline constraint or not. If condition holds

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2906870, IEEE Transactions on Services Computing

then set $\beta_i = 0$, otherwise set $\beta_i = 1$. This process avoids the possibility of misinterpretation of favorable output response from the environment as the favorable input to a particular automaton. The reward and penalty calculation defined in Equation 1 and Equation 2 are rewritten as:

$$p_i^j(k+1) = \begin{cases} p_i^j(k) + \phi \times (1 - p_i^j(k)) & j = i \\ (1-\phi) \times p_i^j(k) & \forall j, j \neq i \end{cases} \quad (13)$$

$$p_i^j(k+1) = \begin{cases} 1 - \varphi \times p_i^j(k) & j = i \\ \frac{\varphi}{r-1} + (1-\varphi) \times p_i^j(k) & \forall j, j \neq i \end{cases} \quad (14)$$

If the reinforcement signal $\beta_i = 0$, action taken by A_i is rewarded using Equation 13 and if $\beta_i = 1$, action of A_i is penalized using Equation 14. The outcome of the scheduling decision at each iteration is stored in LA table. The LA table consists of fields, task ID (t_i), VM ID (v_j), α_i and p_i. We assume that p_i^j shows the goodness value of a VM. Since, the goodness value is an indication of the performance of the VM, a higher number for it is preferable. The proposed LA model optimizes the specified objectives based on a given T. Hence, the scheduling framework is adaptive to the changing cloud environment.

4.5 Scheduling Model

The primary goal of the scheduler is to map a task set to a VM set such that the specified objectives (in this case minimization of energy consumption and makespan) can be achieved. A scheduling process is triggered at each time interval, and all the tasks arrived until that time gets scheduled. The scheduler first invokes LA model to generate the best scheduling decision possible through the reinforcement learning process. LA model is executed for a fixed number of iteration and outcome of each iteration is stored in LA table. In each iteration, the scheduler selects a VM based on its goodness value. The scheduler uses the outcome of LA model after the final iteration for actual execution on the cloud system. The novel feature of the proposed scheduling framework is that:

- A reinforcement based learning process is used to select an appropriate VM for a task. The reinforcement scheme penalizes the bad selection and rewards the good selection. Further, it can be tuned to improve the performance of the system.
- The history information about the responses from the environment is modeled in the action probability matrix.
- The automaton decides without performing any highly time-consuming calculations, which makes it suitable for the real-time system.
- Since, the task scheduling decision is made considering the models of the application's task and the VM of the cloud system; the scheduling framework is well suited for the dynamic cloud environment.

5 LAS Scheduling Technique

In this section, we first manifest the steps of the proposed scheduling technique and then present LA-based scheduling (LAS) algorithm. The steps taken for the proposed approach are as follows:

- *Initialization:* In the beginning, a task can be assigned to any of the available VMs in set V. So, the initial probability of every action α_i^j in α_i for automaton A_i is set to $\frac{1}{m}$. Mathematically, it is represented as: $p_i^j(0) = \frac{1}{m}$, for $i = \{1, 2, ..., n\}$ and $j = \{1, 2, ..., m\}$. Let, ζ is an assignment vector of size $1 \times n$. The assignment vector represents the action chosen by each automaton at a particular iteration. The initial value of ω set to INF, i.e., $\omega(0) = INF$.
- *Construction of Matrices:* Each row of ETC matrix imitates the action set of automaton A_i, $i = \{1, 2, ..., n\}$. For instance, row 1 shows the set of actions (i.e., α_1^j, $j = \{1, 2, ..., m\}$) possible for automaton A_1. Let, the action probability of each automaton formed a $n \times m$ matrix, namely P. Each element of the matrix corresponds to p_i^j.

Algorithm 1 : Pseudo code of $MatrixGeneration()$

```
for each task t_i in T do
    for each VM v_j in V do
        Calculate etc_i^j using Equation 3;
        Set p_i^j = 1/m;
    end for
end for
```

In the first iteration, assignment vector ζ is formed by randomly picking an element from each row of the ETC matrix as all the actions are equiprobable. In subsequent iterations, for each row in P, an element having the highest value is marked. Then ζ is constructed by picking the corresponding entry in ETC. Pseudo code to generate ETC and P matrices is shown in Algorithm 1.

- *Calculation of Cost Metric (ω):* The elements of the proposed bi-objective minimization problem have distinct measurement units. So, to eliminate the computational problem caused by this, each element is normalized. Let τ_{max} be the maximum allowable makespan, and δ_{max} be the maximum possible energy consumption of the system. Then the bi-objective minimization problem can be represented as:

$$\omega = \frac{\delta}{\delta_{max}} \times x + \frac{\tau}{\tau_{max}} \times y \quad (15)$$

Algorithm 2 : Pseudo code of $CostMetricCalculation()$

```
for each (t_i, v_j) pair in the system do
    Calculate start time st_i^j by Equation 4 and pick etc_i^j from ETC matrix;
    Compute ft_i^j of each task t_i on VM v_j using Equation 5;
    if ft_i^j ≤ dl_i then
        Set tc_i^j = 1;
        Calculate ϱ_j and τ using Equations 7, 8, 9, 10, 11;
        Calculate δ using Equation 12;
        Obtain ω using Equation 15;
    else
        Set tc_i^j = 0;
    end if
end for
```

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2906870, IEEE Transactions on Services Computing

where x, and y are weight constant associated with δ and τ respectively.

Detailed computation of cost metric is shown in Algorithm 2. The algorithm first calculates the start time and execution time of task t_i on VM v_j (see line 2). If t_i's deadline can be met on v_j, then set $tc_i^j = 1$ and estimate cost metric (see lines 6-8). If $tc_i^j = 0$, i.e., there is no VM v_j which can execute t_i within deadline, then add a new VM.

- *Updation of Action Probabilities:* Due to the heterogeneity of task and VM, the action performed by automaton can be rewarded sometimes, while, other times, the very same action will be penalized. An action taken by A_i is either rewarded or penalized based on the learning algorithm L. The probability updating formula is presented in Equations 13 and 14. Let θ_{mn} indicate minimum value and θ_{mx} indicate maximum value for action probability, i.e., p_i^j. If $p_i^j \leq \theta_{mn}$, the action α_i^j is deactivated for the task t_i. If $p_i^j = \theta_{mx}$, all the actions except α_i^j are deactivated and t_i is assigned to v_j. This process avoids unnecessary computation and helps faster computation. The deactivated actions are not considered for scheduling decision. Pseudocode for probability updation is shown in Algorithm 3.

Algorithm 3 : Pseudo code of $ProbUpdation()$

```
1: for each task t_i in T do
2:     for each VM v_j in V do
3:         if p_i^j ≤ θ_mn then
4:             Deactivate α_i^j;
5:         end if
6:         if p_i^j == θ_mx then
7:             Assign t_i to v_j;
8:         end if
9:     end for
10: end for
```

- *Calculation of Constants:* The weight constants x and y are assigned different values based on the importance of each metrics. Still, the law of probability measure must be satisfied, i.e., $\sum(x + y) = 1$. Here, we set $x, y = 0.5$, as equal importance is given to both the metrics. The reward (ϕ) and penalty (φ) parameters are given the same value, as we considered a reward-penalty learning algorithm.
- *Stopping Criteria:* The learning process stops when the number of iteration k exceeds maximum iteration count k_{max} or p_i of all automaton reaches its threshold value, whichever is earlier.

The overall working of the proposed algorithm as shown in Algorithm 4 is as follows: first, the ETC and probability matrix P is generated for a given set of VMs and tasks. Then, for each iteration, assignment vector ζ is generated by previously mentioned methods. For each assignment, cost metric is calculated using Algorithm 2. If the cost metric at iteration k is better than the value at iteration $k-1$, then check whether task execution is completed within deadline constraint or not. The reinforcement signal for task's meeting deadline constraint is set to "0," i.e., $\beta_i = 0$, whereas

Algorithm 4 : Pseudo code of LA-based Scheduling (LAS)

```
1: Set k = 1;
2: Generate ET matrix and probability matrix P using MatrixGeneration();
3: while k ≤ k_max do
4:     if k == 1 then
5:         Find assignment vector ζ by random selection;
6:     else
7:         Find assignment vector ζ by choosing actions with highest probabilities in P;
8:     end if
9:     Calculate cost metric using CostMetricCalculation();
10:    if ω(k) ≤ ω(k − 1) then
11:        for each (t_i, v_j) pair in the system do
12:            if tc_i^j == 1 then
13:                Set β_i = 0;
14:            else
15:                Set β_i = 1;
16:            end if
17:        end for
18:    end if
19:    for each task t_i in the system do
20:        if β_i == 0 then
21:            Reward the action taken by A_i using Equation 13;
22:        else if β_i == 1 then
23:            Penalize the action chosen by A_i using Equation 14;
24:        end if
25:    end for
26:    Update P using ProbUpdation();
27:    Increment k;
28: end while
```

for other tasks $\beta_i = 1$. Based on the reinforcement signal, an automaton is either rewarded or penalized. This process continues until the stopping criterion is not satisfied.

Theorem 1. The time complexity of LAS algorithm is $O(nm)$.

Proof: The time complexity of generating ETC and P matrix is $O(nm)$ (Algorithm 1). Generation of assignment vector is $O(nm)$. In Algorithm 2, calculation of task's start time and finish time, and picking an element from ETC matrix is $O(nm)$. Checking if a task meet it's deadline or not takes $O(nm)$ time. In Algorithm 4, the complexity of reward and penalty calculation is $O(n)$ and for setting reinforcement signal value the time complexity is $O(nm)$. In Algorithm 3 the time complexity of probability updation is $O(nm)$. For other lines, the time complexity is $O(1)$. Hence, the time complexity of LAS algorithm is $O(nm)+k_{max}(O(nm)+O(nm)+O(nm)+O(n)+O(nm)+O(nm)) = O(nm)$. □

The proposed scheduling scheme is explained with an example. Let, there are 5 tasks (n=5) and 3 VMs (m=3) in the cloud system. An automaton $A_i,\ i = \{1, 2, 3, 4, 5\}$ can choose three actions $\alpha_i^j,\ j = \{1, 2, 3\}$. We set $k_{max} = 3$.

- *Iteration 1:* Initial configuration of matrices P and ζ are shown in Fig.4a. We assume the action set

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2906870, IEEE Transactions on Services Computing

$P =$

Task \VM	v_1	v_2	v_3
t_1	0.33	0.33	0.33
t_2	0.33	0.33	0.33
t_3	0.33	0.33	0.33
t_4	0.33	0.33	0.33
t_5	0.33	0.33	0.33

$\zeta =$

1	2	3	4	5
v_2	v_3	v_1	v_2	v_2

(a) Matrices at iteration 1

$P =$

Task \VM	v_1	v_2	v_3
t_1	0.30	0.40	0.30
t_2	0.30	0.30	0.40
t_3	0.30	0.35	0.35
t_4	0.30	0.40	0.30
t_5	0.35	0.30	0.30

$\zeta =$

1	2	3	4	5
v_2	v_3	v_1	v_2	v_2

(b) Matrices at iteration 2

$P =$

Task \VM	v_1	v_2	v_3
t_1	0.27	0.46	0.27
t_2	0.27	0.27	0.46
t_3	0.27	0.415	0.315
t_4	0.27	0.46	0.27
t_5	0.315	0.32	0.365

$\zeta =$

1	2	3	4	5
v_2	v_3	v_2	v_2	v_3

(c) Matrices at iteration 3

$P =$

Task \VM	v_1	v_2	v_3
t_1	0.243	0.514	0.243
t_2	0.243	0.243	0.514
t_3	0.243	0.4735	0.2835
t_4	0.243	0.514	0.243
t_5	0.2835	0.288	0.4285

$\zeta =$

1	2	3	4	5
v_2	v_3	v_2	v_2	v_3

(d) Final Scheduling Decision

Fig. 4: An Example for LAS

$\alpha = \{\alpha_1^2, \alpha_2^3, \alpha_3^1, \alpha_4^2, \alpha_5^2\}$. Let, actions α_1^2, α_2^3, and α_4^2 meet the deadline constraint of respective tasks. But, actions α_3^1 and α_5^2 fail to do so. Let, the calculated cost metric value, $\omega(1) = 20$. As $\omega(1) \leq \omega(0)$, the reinforcement signal for each action is $\{\beta_1 = 0, \beta_2 = 0, \beta_3 = 1, \beta_4 = 0, \beta_5 = 1\}$. The actions α_1^2, α_2^3, and α_4^2 are rewarded. The action probability of A_1 with $\phi = \varphi = 0.1$, is calculated as follows:

$$\begin{cases} p_1^1(2) = (1-\phi) \times p_1^1(1) = 0.9 \times 0.33 = 0.297 \simeq 0.3 \\ p_1^2(2) = 0.33 + .1 \times 0.67 = 0.397 \simeq 0.4 \\ p_1^3(2) = (1-\phi) \times p_1^3(1) = 0.3 \end{cases} \tag{16}$$

Similarly, probability value for A_2 and A_4 are computed. The action probability of A_3 is updated as follows:

$$\begin{cases} p_3^1(2) = (1-\varphi) \times p_3^1(1) = 0.9 \times 0.33 = 0.297 \simeq 0.3 \\ p_3^2(2) = \frac{0.1}{2} + .9 \times p_3^2(1) = 0.347 \simeq 0.35 \\ p_3^3(2) = 0.35 \end{cases} \tag{17}$$

Likewise, probability of A_5 is computed. The updated values are shown in Fig. 4b.

- *Iteration 2:* We assume the action set $\alpha = \{\alpha_1^2, \alpha_2^3, \alpha_3^2, \alpha_4^2, \alpha_5^1\}$. Let, actions α_1^2, α_2^3, α_3^2 and α_4^2 meet the deadline constraint of respective tasks. But, action α_5^1 fail to do so. Let, the calculated cost metric value, $\omega(2) = 18$. As $\omega(2) \leq \omega(1)$, the reinforcement signal for each action is $\{\beta_1 = 0, \beta_2 = 0, \beta_3 = 0, \beta_4 = 0, \beta_5 = 1\}$. The actions having reinforcement signal value "0" are rewarded whereas actions with value "1" are penalized. The action with reward is updated as follows:

$$\begin{cases} p_1^1(3) = (1-\phi) \times p_1^1(2) = 0.27 \\ p_1^2(3) = 0.4 + .1 \times 0.6 = 0.46 \\ p_1^3(3) = 0.27 \end{cases} \tag{18}$$

Similarly, probability value for A_2, A_3 and A_4 are computed. The action probability of A_3 is updated as follows:

$$\begin{cases} p_5^1(3) = 0.9 \times 0.35 = 0.315 \\ p_5^2(3) = \frac{0.1}{2} + .9 \times p_3^2(2) = 0.32 \\ p_5^3(3) = \frac{0.1}{2} + .9 \times 0.35 = 0.365 \end{cases} \tag{19}$$

The updated values are shown in Fig. 4c.

- *Iteration 3:* Let actions $\alpha = \{\alpha_1^2, \alpha_2^3, \alpha_3^2, \alpha_4^2, \alpha_5^3\}$ meet both cost metric and deadline constraint. The reinforcement signal for each action is "0", i.e., $\{\beta_1 = 0, \beta_2 = 0, \beta_3 = 0, \beta_4 = 0, \beta_5 = 0\}$. The updated P matrix is shown in Fig. 4d. Hence, the final scheduling decision is $\zeta = \{\alpha_1^2, \alpha_2^3, \alpha_3^2, \alpha_4^2, \alpha_5^3\}$.

Lemma 1: To preserve the law of probability measure, the sum of probabilities of actions possible by an automaton A_i is one. Mathematically, it is expressed as:

$$\sum_{j=1}^{m} p_i^j = 1 \tag{20}$$

where m is the number of actions possible by A_i.

Proof: Let, A_1 is the automaton for task t_1 and can choose five actions $\alpha_1^1, \alpha_1^2, \alpha_1^3, \alpha_1^4, \alpha_1^5$ with probabilities $p_1^1, p_1^2, p_1^3, p_1^4, p_1^5$ respectively. α_1^1 means task t_1 is assigned to VM v_1, α_1^2 means task t_1 is assigned to VM v_2 and so on. p_1^1 is the probability of choosing action α_1^1, p_1^2 is the probability of choosing action α_1^2 and so on. Let, task t_1 is assigned to VM v_1 and based on response from the environment this action is either rewarded or penalized. If the action is rewarded, then the action probability is updated according to Equations 21, and 22.

$$p_1^1(k+1) = p_1^1(k) + \phi \times (1 - p_1^1(k)) \tag{21}$$

$$p_1^{2,3,4,5}(k+1) = (1-\phi) \times p_1^{2,3,4,5}(k) \tag{22}$$

$p_1^{2,3,4,5}$ is used instead of individual probability $p_1^2, p_1^3, p_1^4, p_1^5$. Similarly, Equations 23, and 24 are used to update action probability for penalty.

$$p_1^1(k+1) = p_1^1(k) - \varphi \times (1 - p_1^1(k)) \tag{23}$$

$$p_1^{2,3,4,5}(k+1) = (1-\varphi) \times p_1^{2,3,4,5}(k) + (\frac{\varphi}{4}) \tag{24}$$

By reframing Equation 21, we get,

$$p_1^1(k+1) = (1-\phi) \times p_1^1(k) + \phi \tag{25}$$

By putting different k values in Equation 25, we get,

$$\begin{cases} k = 1, \;\; p_1^1(2) = (1-\phi) \times p_1^1(1) + \phi \\ k = 2, \;\; p_1^1(3) = (1-\phi) \times p_1^1(2) + \phi \\ k = 3, \;\; p_1^1(4) = (1-\phi) \times p_1^1(3) + \phi \end{cases} \tag{26}$$

By solving Equation 26 with substitution method, we get,

$$p_1^1(4) = (1-\phi)^3 \times p_1^1(1) + \phi \times [1 + (1-\phi) + (1-\phi)^2] \quad (27)$$

Generalizing Equation 27, we get,

$$p_1^1(k+1) = (1-\phi)^{k+1} \times p_1^1(0) + \phi \times [1 + (1-\phi) + ... + (1-\phi)^k] = (1-\phi)^{k+1} \times p_1^1(0) + \phi \times \left[\frac{1-(1-\phi)^k}{1-(1-\phi)}\right] \quad (28)$$

For large value of k, $(1-\phi)^{k+1}$ and $(1-\phi)^k$ tends to zero. So, we can rewrite Equation 28 as:

$$p_1^1(k+1) = \phi \times \left[\frac{1}{1-(1-\phi)}\right] = 1 \quad (29)$$

Now, putting different k values in Equation 22, we get,

$$\begin{cases} k=1, \;\; p_1^{2,3,4,5}(2) = (1-\phi) \times p_1^{2,3,4,5}(1) \\ k=2, \;\; p_1^{2,3,4,5}(3) = (1-\phi) \times p_1^{2,3,4,5}(2) \\ k=3, \;\; p_1^{2,3,4,5}(4) = (1-\phi) \times p_1^{2,3,4,5}(3) \end{cases} \quad (30)$$

By solving Equation 30 with substitution method, we get,

$$p_1^{2,3,4,5}(4) = (1-\phi)^3 \times p_1^{2,3,4,5}(1) \quad (31)$$

Generalizing Equation 31, we get,

$$p_1^{2,3,4,5}(k+1) = (1-\phi)^{k+1} \times p_1^{2,3,4,5}(1) = 0 \quad (32)$$

For large value of k, $(1-\phi)^{k+1}$ tends to 0. Sum of all probabilities is:

$$\sum_{j=1}^{5} p_1^j = p_1^1(k+1) + p_1^{2,3,4,5}(k+1) = 1 + 0 = 1 \quad (33)$$

Similarly, we can prove for penalty equations.

6 Performance Evaluation

To show the effectiveness of LAS, we compare it with the following algorithms, Greedy-R and Greedy-P [1], EEVS [18], Dynamic Task Scheduling (DTS) [28] and Random. All the algorithms are modified to suit the heterogenous tasks and VMs here. The algorithms for comparisons are summarized as follows:

Greedy-R [1]: It allocates tasks with the quickest execution time first to the highest speed available VM so that task deadline constraint is met.

Greedy-P [1]: It allocates tasks with the quickest execution time first to the slowest speed available VM.

EEVS [18]: An optimal frequency for VM is obtained to minimize the energy consumption. Based on the optimal frequency a performance-power ratio is calculated, then PM with the highest ratio is used to process VM.

DTS [28]: First, the tasks are classified based on historical scheduling information. VM of various types is accordingly created. Then, matching of the task with VM is performed dynamically, which is a time-consuming process.

Random: It assigns task randomly amongst available VMs under the deadline constraint.

The metrics used to evaluate the performance of the algorithm include the success ratio, makespan, and total energy consumption. The success ratio is defined as the ratio between the number of tasks meet their deadline and the total number of tasks.

6.1 Simulation Settings

The detailed settings and parameters for simulation are given as follows:

- Each VM is modeled to have computing capacity in the range [3000-6000] MIPS.
- A task generator is developed to quantify real-time task arrival to the cloud system. Task arrival is modeled using the Poisson distribution where the inter-arrival time is exponentially distributed. The task's deadline is set as: $dl_i = a_i + baseD$, where $baseD$ is in uniform distribution $U(5, 10)$. Task size is set in the range [1000-10000] MI. MIPS and MI are assumed based on the study found in [39] and [38] respectively.
- Each experiment is repeated 20 times, and averaging is done to avoid the influence of uncertainty factors on the experimental outcomes.
- The reward and penalty constant is set to $\phi = \varphi = 0.1$.

Fig. 5a shows the variation of cost metric value using LAS over the number of iterations. The experiment is carried out with task count=1000 and varying VM count between [40-100] in the step of 20. It can be seen from Fig. 5a that, after 500 iterations there is minimal variation in the cost metric value. This indicates automata is starting to converge to a solution. Also, we performed experiments with different values of reward and penalty constant. The experiment is carried out with task count=1000, number of iteration=500, number of VM=40 and 60 respectively. From Fig. 5b and Fig. 5c we can infer that the cost metric value goes up with a rise in task count. The nature of the graph is similar for different values of ϕ and φ. Thus, we set the ϕ and φ value to 0.1.

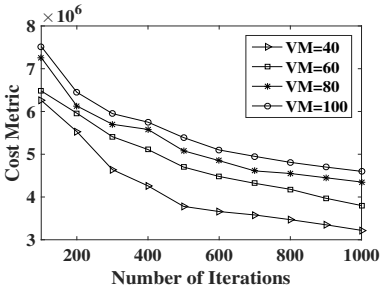
6.2 Performance based on the task count

Here, we display a group of experimental outcomes to perceive the performance comparison of algorithms based on task count. We performed experiments by setting VM count to 40, 60, 80 and 100. We vary the task count in the range [200-1000] in the interval of 200. In this paper, we exhibit the experimental results with VM=40 and VM=60 in Fig. 6 and Fig. 7 respectively. It can be perceived from Fig. 6a and Fig. 7a that the makespan of algorithms increases with the rise in task count. This is because, as there is an addition to tasks the completion time of tasks will also grow with fixed VM count. Moreover, we can see that LAS has minimum makespan as compared to others and the trend is similar for all the task count.

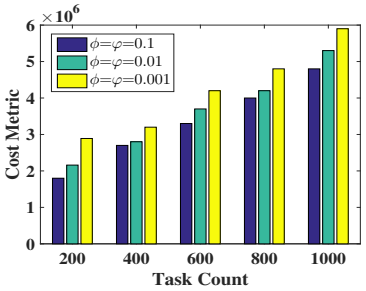
From Fig. 6b and Fig. 7b, it can be concluded that LAS has better energy conservation as compared to others and the trend becomes apparent with the rise in task count. This experimental outcome indicates that LA theory helps in the efficient utilization of VMs which helps to save more energy.

Fig. 6c and Fig. 7c demonstrates that for all algorithms there is negligible change in the success ratio regardless of task count. The increase in the number of VMs causes slight variations in the success ratio. This is because an efficient task scheduling can lead to executing as many real-time tasks as possible within the deadline. Besides, it can be found that LAS has a higher success ratio than other

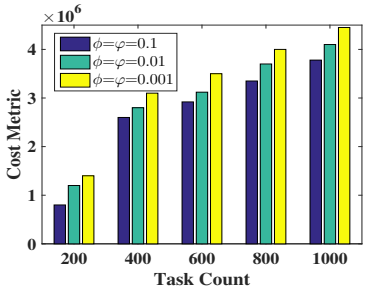
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2906870, IEEE Transactions on Services Computing



(a) Number of Iterations v/s Cost Metric

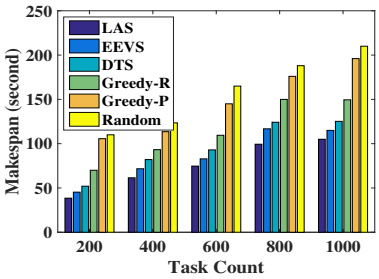


(b) Task count v/s Cost Metric (Number of VM=40)

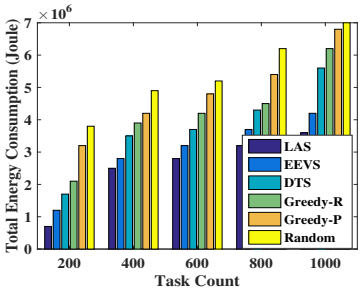


(c) Task count v/s Cost Metric (Number of VM=60)

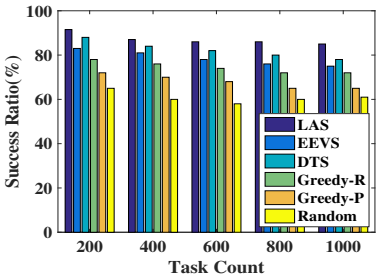
Fig. 5: Cost Metric variation



(a) Makespan

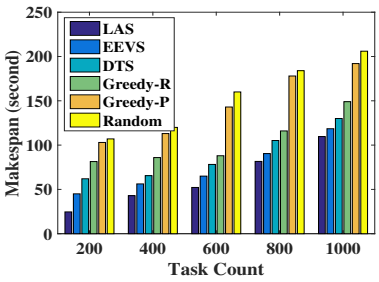


(b) Energy Consumption

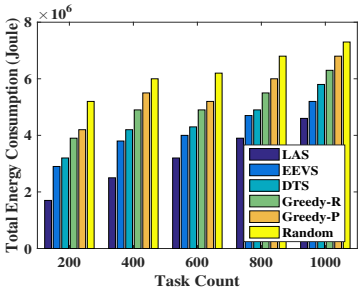


(c) Success Ratio

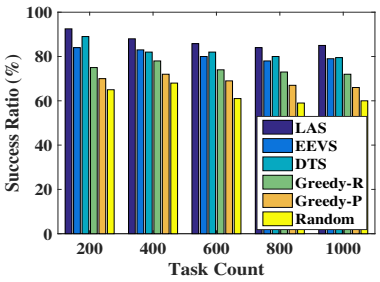
Fig. 6: Number of VM=40 with variation in task count



(a) Makespan



(b) Energy Consumption



(c) Success Ratio

Fig. 7: Number of VM=60 with variation in task count

algorithms. This can be attributed that LAS employs LA optimization policy that helps to improve task's schedulability.

6.3 Performance based on Deadline Variation

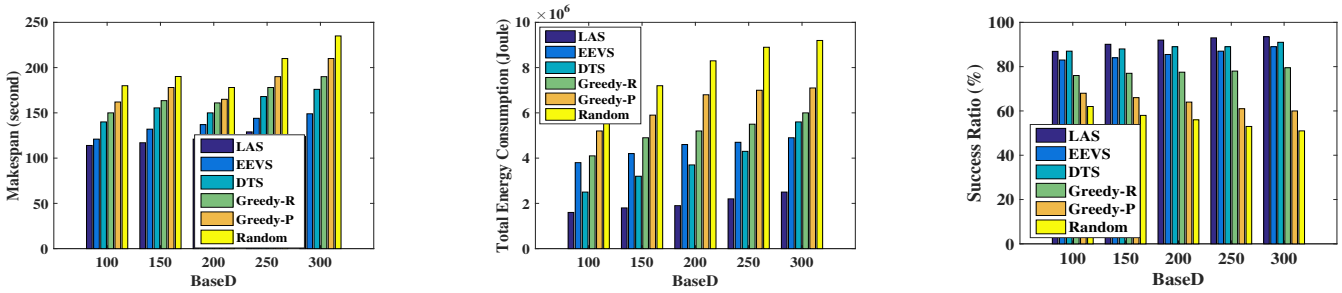
Here, we present a set of experiments to show the impact of task deadline on the performance of different algorithms. The number of VMs is set in the range [40-100] in the interval of 20, and task count is fixed to 1000. In this paper, we display the experimental results with VM=40 and VM=60 in Fig. 8 and Fig. 9. Parameter baseD varied from 100 to 300 with the step of 50.

From Fig. 8a and Fig. 9a it can be inferred that with the rise in baseD, the makespan of the algorithms increases correspondingly. This is because, with the addition of long deadline, tasks will get more time to complete their execution within the deadline. Ultimately, it will add up to the makespan. Besides, it can be seen that LAS outperforms other algorithms.

Fig. 8b and 9b show that as the baseD widens, the total energy consumption by the algorithms grows correspondingly. This outcome indicates that due to the rise in deadline, a more significant number of tasks get the chance to complete their execution before the deadline. Thus, there is an increase in energy consumption. Additionally, LAS consumes less energy as compared to other algorithms. This is because of the use of LA theory, which helps in efficient resource utilization.

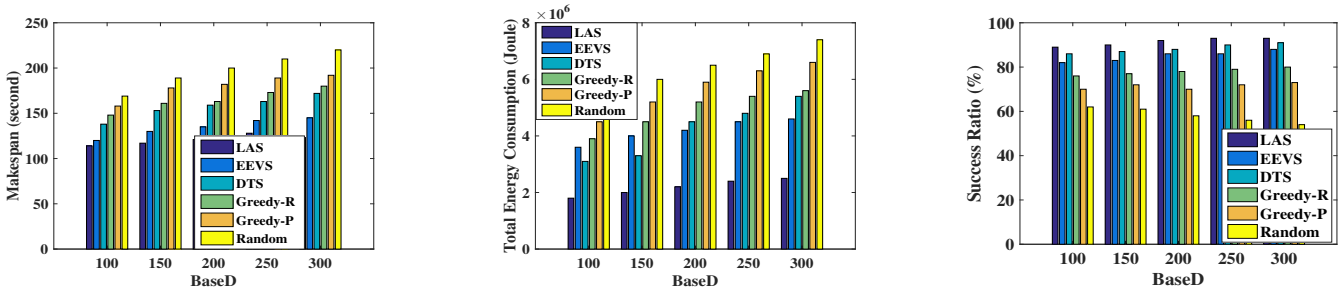
Fig. 8c and 9c exhibits that the increase in baseD improves the success ratio of the algorithms. This can be justified by the fact that, the prolonged deadlines give tasks extra time to finish their execution within time constraints. Also, Fig. 8c and Fig. 9c shows that LAS has higher success ratio

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2906870, IEEE Transactions on Services Computing



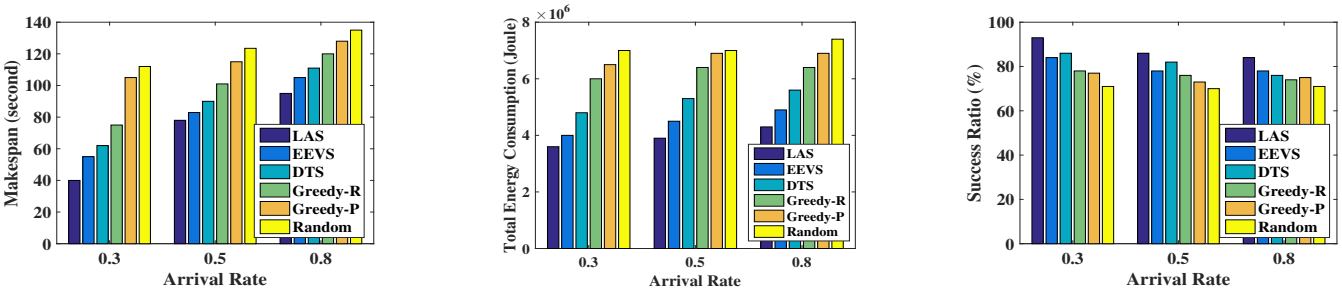
(a) Makespan (b) Energy Consumption (c) Success Ratio

Fig. 8: Number of VM=40 with variation in $baseD$



(a) Makespan (b) Energy Consumption (c) Success Ratio

Fig. 9: Number of VM=60 with variation in $baseD$



(a) Makespan (b) Energy Consumption (c) Success Ratio

Fig. 10: Performance with respect to arrival rate

as compared to its counterparts. LA theory implemented in LAS cause a significant number of tasks to finish their execution before the deadline and hence improves success ratio.

6.4 Performance based on Arrival Rate

Fig. 10 demonstrates the performance of the six algorithms concerning arrival rate. We set the arrival rate as 0.3 (low), 0.5 (moderate) and 0.9 (high). Task count is set to 200, and the number of VMs is 40. Fig. 10a shows that makespan of the system increases with an increase in task arrival rate. As the arrival rate rises the number of tasks executed in the system increases which contributes to the makespan of the system. However, LAS has minimum makespan as compared to other algorithms. Fig. 10b shows the impact of task arrival rate on energy consumption. From the figure, we can infer that high task arrival rate causes high energy consumption. Besides, LAS exhibits better result compared to other algorithms. The reason can be explained like that in Fig. 8b. Fig. 10c shows the effect of the task arrival rate on the success ratio. LAS algorithm performs better as compared to other algorithms, which can be justified as that in Fig. 8c.

An unpaired t-test is conducted with the following null hypothesis: the success ratio is 75 % for all the algorithms. The t-test is applied to 200 tasks with the number of VMs 40 (Experiment-1) and 60 (Experiment-2). For Experiment-1, the calculated mean value= 79.583, and standard deviation=9.972 and N=6. For Experiment-2, the calculated mean value= 79.250, and standard deviation=10.953 and N=6. The calculated value of t is 0.0551 and degree of freedom (df)=10. Tabulated value of t at 10 df is 2.23 at 5 % level of significance (Value of $t_6(0.05)$ for 10 df in two-tailed Table). Since the calculated t value is less than 2.23 the null

hypothesis may be accepted at 5 % level of significance.

7 Conclusions

Energy conservation in the cloud system has significant interest among researchers. It becomes challenging for a cloud system to save energy along with timing constraint of deadline sensitive task. In this paper, we investigated a bi-objective scheduling problem for deadline sensitive tasks in the heterogeneous cloud environment. The scheduling objectives are to minimize energy consumption and makespan simultaneously. To realize the objectives, we employed LA theory, which works on the principle of reinforcement learning. First, we proposed an LA-based scheduling framework and then presented a scheduling algorithm, LAS, for deadline sensitive task in the cloud. The proposed scheduling technique is better explained with a suitable example. The experimental outcomes implied that LAS significantly performs better compared to some existing algorithms. In the future, we plan to implement the Pareto-optimality concept to solve the bi-objective scheduling problem.

References

[1] X. Zhu, L. T. Yang, H. Chen, J. Wang, S. Yin, & X. Liu, "Real-time tasks oriented energy-aware scheduling in virtualized clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 168-180, 2014.

[2] H. Chen, X. Zhu, D. Qiu, H. Guo, L. T. Yang, & P. Lu," EONS: minimizing energy consumption for executing real-time workflows in virtualized cloud data centers," *In 45th International Conference on Parallel Processing Workshops (ICPPW)*, pp. 385-392, 2016.

[3] J. Koomey, "Growth in data center electricity use 2005 to 2010," *A report by Analytical Press, completed at the request of The New York Times*, 2011.

[4] S. K. Mishra, D. Puthal, J. J. Rodrigues, B. Sahoo, & E. Dutkiewicz, "Sustainable Service Allocation using Metaheuristic Technique in Fog Server for Industrial Applications," *IEEE Transactions on Industrial Informatics*, 2018.

[5] Y. Gao, Y. Wang, S. K. Gupta, & M. Pedram, "An energy and deadline aware resource provisioning, scheduling and optimization framework for cloud systems," *In International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pp. 1-10, 2013.

[6] S. Yassa, R. Chelouah, H. Kadima, & B. Granado, "Multi-objective approach for energy-aware workflow scheduling in cloud computing environments," *The Scientific World Journal*, vol. 2013, 2013.

[7] X. Zhu, J. Wang, H. Guo, D. Zhu, L. T. Yang and L. Liu, "Fault-Tolerant Scheduling for Real-Time Scientific Workflows with Elastic Resource Provisioning in Virtualized Clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 12, pp. 3501-3517, 2016.

[8] H. Chen, X. Zhu, G. Liu and W. Pedrycz, "Uncertainty-Aware Online Scheduling for Real-Time Workflows in Cloud Service Environment," *IEEE Transactions on Services Computing*, pp. 1-1, 2018.

[9] K. S. Narendra, & M. A. Thathachar, "Learning automata-a survey," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 4, pp. 323-334, 1974.

[10] S. Misra, P. V. Krishna, K. Kalaiselvan, V. , Saritha, & M. S. Obaidat, "Learning automata-based QoS framework for cloud IaaS," *IEEE Transactions on Network and Service Management*, vol. 11, no. 1, pp. 15-24, 2014.

[11] A. Rezvanian, & M. R. Meybodi, "Finding minimum vertex covering in stochastic graphs: a learning automata approach," *Cybernetics and Systems*, vol. 46, no. 8, pp. 698-727, 2015.

[12] M. Ranjbari, & J. A. Torkestani, "A learning automata-based algorithm for energy and SLA efficient consolidation of virtual machines in cloud data centers," *Journal of Parallel and Distributed Computing*, vol. 113, pp. 55-62, 2018.

[13] A. A. Rahmanian, M. Ghobaei-Arani, & S. Tofighy, "A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment," *Future Generation Computer Systems*, vol. 79, pp. 54-71, 2018.

[14] J. A. Torkestani,"An adaptive learning to rank algorithm: Learning automata approach," *Decision Support Systems*, vol. 54, no. 1, pp. 574-583, 2012.

[15] M. A. L. Thathachar and B. R. Harita, "Learning automata with changing number of actions," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 17, no. 6, pp. 1095-1100, 1987.

[16] R. D. Venkataramana, & N. Ranganathan, "A learning automata based framework for task assignment in heterogeneous computing systems," *In Proceedings of the ACM symposium on Applied computing*, pp. 541-547, 1999.

[17] A. Beloglazov, J. Abawajy, & R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755-768, 2012.

[18] Y. Ding, X. Qin, L. Liu, & T. Wang, "Energy efficient scheduling of virtual machines in cloud with deadline constraint," *Future Generation Computer Systems*, vol. 50, pp. 62-74, 2015.

[19] X. Li, X. Jiang, P. Garraghan, & Z. Wu, "Holistic energy and failure aware workload scheduling in Cloud datacenters," *Future Generation Computer Systems*, vol. 78, pp. 887-900, 2018.

[20] G. Xing, X. Xu, H. Xiang, S. Xue, S. Ji, & J. Yang, "Fair energy-efficient virtual machine scheduling for Internet of Things applications in cloud environment," *International Journal of Distributed Sensor Networks*, vol. 13, no. 2, 2017.

[21] Z. Dong, N. Liu, & R. Rojas-Cessa, "Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers," *Journal of Cloud Computing*, vol. 4, no. 1, pp. 5, 2015.

[22] Z. Li, J. Ge, H. Hu, W. Song, H. Hu, & B. Luo, "Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds," *IEEE Transactions on Services Computing*, 2015.

[23] S. K. Panda, & P. K. Jana, "Efficient task scheduling algorithms for heterogeneous multi-cloud environment," *The Journal of Supercomputing*, vol. 71, no. 4, pp. 1505-1533, 2015.

[24] J. Koodziej, S. U. Khan, L. Wang, & A. Y. Zomaya, "Energy efficient geneticbased schedulers in computational grids," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 4, pp. 809-829, 2015.

[25] Q. Zhang, M. F. Zhani, R. Boutaba, & J. L. Hellerstein, "Harmony: Dynamic heterogeneity-aware resource provisioning in the cloud," *In IEEE 33rd International Conference on Distributed Computing Systems (ICDCS)*, pp. 510-519, 2013.

[26] G. L. Stavrinides, & H. D. Karatza, "A cost-effective and qos-aware approach to scheduling real-time workflow applications in paas and saas clouds," *In 3rd International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 231-239, 2015.

[27] Z. Cai, X. Li, R. Ruiz, & Q. Li, "A delay-based dynamic scheduling algorithm for bag-of-task workflows with stochastic task execution times in clouds," *Future Generation Computer Systems*, vol. 71, pp. 57-72, 2017.

[28] P. Zhang, & M. Zhou, "Dynamic cloud task scheduling based on a two-stage strategy," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 772-783, 2018.

[29] T. Shi, M. Yang, X. Li, Q. Lei, & Y. Jiang, "An energy-efficient scheduling scheme for time-constrained tasks in local mobile clouds," *Pervasive and Mobile Computing*, vol. 27, pp. 90-105, 2016.

[30] Y. Ran, J. Yang, S. Zhang, & H. Xi, "Dynamic IaaS computing resource provisioning strategy with QoS constraint," *IEEE Transactions on Services Computing*, vol. 10, no. 2, pp. 190-202, 2017.

[31] E. Grochowski, and M. Annavaram, "Energy per instruction trends in intel microprocessors" *Technology @Intel Magazine*, vol. 4, no. 3, pp. 1-8, 2006.

[32] S. Sahoo, S. Nawaz, S. K. Mishra, & B. Sahoo, "Execution of real time task on cloud environment," *In Annual IEEE India Conference (INDICON)*, pp. 1-5, 2015.

[33] K. H. Kim, A. Beloglazov, & R. Buyya, "Poweraware provisioning of virtual machines for realtime Cloud services," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 13, pp. 1491-1505, 2011.

[34] N. Quang-Hung, P. D. Nien, N. H. Nam, N. H. Tuong, & N. Thoai, "A genetic algorithm for power-aware virtual machine allocation in private cloud," *In Information and Communication Technology-EurAsia Conference*, pp. 183-191, 2013.

[35] Y. Chen, C. Lin, J. Huang, X. Xiang, & X. Shen, "Energy efficient scheduling and management for large-scale services computing systems," *IEEE Transactions on Services Computing*, no. 1, pp. 1-1, 2017.

[36] S. Sahoo, B. Sahoo, & A. K. Turuk, "An Energy-Efficient Scheduling Framework for Cloud Using Learning Automata," *In 9th In-*

ternational Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1-5, 2018.

[37] G. Xie, G. Zeng, R. Li & K. Li, "Energy-Aware Processor Merging Algorithms for Deadline Constrained Parallel Applications in Heterogeneous Cloud Computing," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 62-75, 2017.

[38] E. Hwang, & K. H. Kim, " Minimizing cost of virtual machines for deadline-constrained mapreduce applications in the cloud," *In Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing*, pp. 130-138, 2012.

[39] $https://www.cisco.com/c/dam/global/da\_dk/assets/docs/presentations/vBootcamp_Performance_Benchmark.pdf$. [$Onine$].

Sampa Sahoo Sampa Sahoo is pursuing a Ph.D. in Department of Computer Science and Engineering at National Institute of Technology, Rourkela, India. Her area of research is Cloud Computing, Parallel, and Distributed computing System, Real-time task scheduling. She is a member of IEEE computer society.

Bibhudatta Sahoo Bibhudatta Sahoo received the M.Tech. and Ph.D. degrees in computer engineering from the National Institute of Technology (NIT), Rourkela, India. He is currently an Associate Professor with the Department of Computer Science and Engineering, NIT Rourkela. He has 24 years of teaching experience in undergraduate and graduate level in the field of computer science and engineering. His technical interests include data structures and algorithm design, parallel and distributed systems, and networks.

Ashok Kumar Turuk Ashok Kumar Turuk received the Ph.D. degree in computer science and engineering from the Indian Institute of Technology, Kharagpur, India. He is a Professor with the Department of Computer Science and Engineering, National Institute of Technology, Rourkela, India. He has 18 years of teaching experience in undergraduate and graduate level in the field of computer science and engineering. He has handled several technical projects.