Multimedia Tools and Applications (2024) 83:8359–8387
https://doi.org/10.1007/s11042-023-16008-2

DRLBTSA: Deep reinforcement learning based task-scheduling algorithm in cloud computing

Sudheer Mangalampalli[1] · Ganesh Reddy Karri[1] · Mohit Kumar[2] · Osama Ibrahim Khalaf[3] · Carlos Andres Tavera Romero[4] · GhaidaMuttashar Abdul Sahib[5]

Received: 30 August 2022 / Revised: 26 April 2023 / Accepted: 11 June 2023 /
Published online: 17 June 2023

Abstract

Task scheduling in cloud paradigm brought attention of all researchers as it is a challenging issue due to uncertainty, heterogeneity, and dynamic nature as they are varied in size, processing capacity and number of tasks to be scheduled. Therefore, ineffective scheduling technique may lead to increase of energy consumption SLA violations and makespan. Many of authors proposed heuristic approaches to solve task scheduling problem in cloud paradigm but it is fall behind to achieve goal effectively and need improvement especially while scheduling multimedia tasks as they consists of more heterogeneity, processing capacity. Therefore, to handle this dynamic nature of tasks in cloud paradigm, a scheduling mechanism, which automatically takes the decision based on the upcoming tasks onto cloud console and already running tasks in the underlying virtual resources. In this paper, we have used a Deep Q-learning network model to addressed the mentioned scheduling problem that search the optimal resource for the tasks. The entire extensive simulationsare performed usingCloudsim toolkit. It was carried out in two phases. Initially random generated workload is used for simulation. After that, HPC2N and NASA workload are used to measure performance of proposed algorithm. DRLBTSA is compared over baseline algorithms such as FCFS, RR, Earliest Deadline first approaches. From simulation results it is evident that our proposed scheduler DRLBTSA minimizes makespan over RR,FCFS, EDF, RATS-HM, MOABCQ by 29.76%, 41.03%, 27.4%, 33.97%, 33.57% respectively. SLA violation percentage for DRLBTSA minimized overRR,FCFS, EDF, RATS-HM, MOABCQ by48.12%, 41.57%, 37.57%, 36.36%, 30.59% respectively and energy consumption for DRLBTSA over RR,FCFS, EDF, RATS-HM, MOABCQ by36.58%,43.2%, 38.22%, 38.52%, 33.82%existing approaches.

Keywords Cloud Computing · Task Scheduling · Machine Learning · Deep Q- Learning · Makespan · Energy consumption · SLA violation

Extended author information available on the last page of the article

1 Introduction

Cloud Computing paradigm is a regime change in various industries, which changed utilization of computing, storage and network infrastructures and laid a platform to cope up with the evolvement of huge data in various industries especially to handle data intensive computations, large chunks of data storage. Therefore, this paradigm evolved as a utilization model through which all services i.e. computation, storage, network are given to consumers as services on demand. This model initially evolved as virtual infrastructure i.e., IaaS for various companies but later it was evolved as computing platform where we can develop our applications and install various software's by using different services provided by cloud platform. Now a days cloud computing is useful in various sectors and some of the domains are mentioned here but not limited i.e., healthcare, education, entertainment, Government organizations, multimedia, transport, IoT, AI and ML. In cloud paradigm, services related to IoT, AI, ML, image processing requires huge processing capacity infrastructure as all these services consists of multimedia data which need to be processed accurately and scheduling multimedia data is a challenge in cloud paradigm. All the above-mentioned domains use various service models based on SLA. SLA depends on user and organization to which services they are subscribed. It is the responsibility of cloud provider to render services based on agreement and violation of SLA should not be happened from the cloud provider. Many of users are accessing virtual resources in cloud simultaneously and it is difficult to handle all these requests and assign virtual resources according to SLA is a challenging task and cloud paradigm provision resources automatically to users based on SLA without human intervention and these provisioning of virtual resources to tasks are to be handled by a scheduler.

The effectiveness of cloud computing paradigm mainly depends on how scheduler effectively manages tasks and schedules tasks onto suitable virtual resources. It also effects various parameters i.e. energy consumption, SLA violation that leads to the issues related to both cloud provider, user. If scheduler is not suitably mapping tasks to virtual resources then it directly effects makespan, which takes high amount of execution time, which leads to decay in quality of service. If a task takes huge amount of execution time then it may also incurs high amount of energy consumption. This can be also one of the reason to effect quality of service. Finally, if a given task is not expected to complete within stipulated time or if a task is provisioned for certain amount of time to a virtual resource but if user is still accessing resource after the provisioned time then it is a violation of SLA. It will happen in this paradigm due to improper scheduling of tasks to virtual resources. Therefore, it will cause a problem to cloud provider in view of SLA violations. Many of authors used heuristic techniques [4, 12, 17, 22] and nature inspired approaches [1–3] for tackling task scheduling but still there is lacking of an effective scheduler which schedules these dynamic tasks onto virtual resources appropriately while minimizing metrics such as energy consumption, makespan and SLA violations. Therefore, we have used a Machine learning technique i.e. Q-learning based on reinforcement learning method to solve task scheduling problem focused on multimedia tasks which fed to task manager by calculating priority of tasks, those tasks are fed to Q-learning model which takes decision based on upcoming tasks and tasks running in virtual machines. Tasks already running in VMs will be consolidated or migrated based on upcoming tasks at cloud console and decision taken by the ML model employed in scheduling algorithm while minimizing metrics makespan, SLA violation, energy consumption.

1.1 Motivation and contributions

The Cloud paradigm emerged as a utility computing approach where all computing, storage, network infrastructure to be given as a utility to cloud user. When all these services are given as utility with ease and seamless access, many of users will be attracted towards this paradigm. Theend users around the world who are working in different sectors are using cloud services based on their requirement. Providing cloud services to all users without any interruption is a huge challenge in cloud computing because cloud resources are heterogenous nature and upcoming requests are diverse as well as uncertain. Therefore, for assigning virtual resources to user requests there is need of an efficient scheduling approach that handles requests and map them onto virtual resources while maintaining the quality of service and SLA violations. The above reason motivates us to do the research in this area of cloud computing. We have also evaluated very important and primary parameters, which influences the performance of cloud model i.e., makespan- which is time taking to execute a task on a VM, SLA Violation- agreement made by cloud user and provider for the services, Energy consumption- which is consumption of energy by VMS at computation and idle time. The objective of our research is to optimize all these parameters without violating the conditions.

The contributions of the article are given below here.

1. A Scheduling algorithm is proposed by employing a ML technique which dynamically takes decision according to upcoming and existing tasks.
2. Deep Q- Learning network model is used as a ML technique, which is based on Reinforcement learning, and it is integrated into scheduling module.
3. The Extensive simulations are carried on Cloudsim. Initially random workload have been considered and then we have tested efficacy of our algorithm using HPC2N and NASA parallel work logs for evaluation of parameters makespan, energy consumption and SLA Violation.
4. The experimental results show that proposed approach DRLBTSA is superior to existing round robin, FCFS, Earliest deadline first, RATS-HM, MOABCQalgorithms.

The remaining paper is organized as follows: Existing state of arts approaches are presented and compared in section 2, problem formulation and proposed methodology based upon ML approach are discussed in section 3 & 4. The computing simulation results are discussed in section 5 and conclusion is discussed in last section of the article.

2 Related works

The authors formulated a resource allocation and security mechanism [5], which used a hybrid ML, approach i.e., RATS-HM technique. Totally, this work was done in three stages. In the first stage, a Cat Swarm optimization technique was used to address makespan, throughput. In the second stage, a DNN was used to address metrics such as bandwidth, load on resources for efficient allocation of resources to tasks. Finally, in the third stage a security authentication scheme was implemented to provide security to data stored in cloud. The Cloudsim [7] is used as simulation toolkit to assess the FCFS, RR algorithms performance, results it was identified that proposed RATS-HM mechanism shown a great influenceand surpass existing techniques for mentioned parameters. Authors in [26] proposed task scheduling model for large-scale cloud computing systems to address parameters i.e. task execution delay, resource utilization. Methodology chosen for this approach is a ML approach based on reinforcement learning. Four techniques are used for developing of scheduling mechanism i.e. RL, DQN, RNN-LSTM and DRL-LSTM. Matlab was used for simulation purpose. A real time dataset was taken from Google cluster, it was given as an input to algorithm, and among all techniques, DRL-LSTM performs better than other algorithms when they are compared with RR, PSO and SJF for above mentioned metrics.

Authors in [14], devised a scheduling mechanism AIRL based on reinforcement learning technique to schedule time sensitive requests in cloud. Main objective of AIRL is to minimize request response time, maximize success rate of user requests. Finally AIRL was compared over different schedulers i.e. RR, earliest, random, DQN. From Simulation results,the proposed AIRL shows a great effect over baseline algorithms. In [8], authors proposed scheduling algorithm which addresses QoS, cost of VMs, success rate, response time for scheduling model. This framework uses a DQN model, which works based on reinforcement learning. Entire experimentation was done on a real time cloud and it was evaluated against random, RR and earliest schedulers and from simulation results it was identified that DQN overcomes mentioned algorithms for mentioned parameters. [32], scheduling framework formulated minimizes execution time, waiting time of tasks. Authors used a ML technique i.e. CDDQLS based on reinforcement learning. Entire simulation carried on Cloudsim, posed deadline and resource constraints. After simulation CDDQLS evaluated over Random, Time shared, Space shared algorithms and it shown a great impact for mentioned algorithms. [10] proposed task scheduling model formulated to minimize makespan. It uses a ML approach named as DQN which uses reinforcement learning strategy for scheduling tasks. Experimentations conducted on MATLAB and compared against HEFT, CPOP algorithms. From results, it revealed that makespan greatly minimized over baseline mechanisms.

In [33], scheduling scheme designed to minimize makespan. A machine learning model used as methodology. QL-HEFT i.e. a combination of Q-Learning and HEFT algorithms. This process was done in two stages. In First phase, tasks will be sorted to get effective task allocation by Q-Learning. In second phase, processor allocation was done based on HEFT. Entire scheme implemented over Cloudsim. It compared with existing HEFT, CPOP algorithms. Finally this scheme was shown impact over existing approaches with respect to makespan. In [9], Dynamic task scheduling model which aims minimization of energy consumption, utilization of CPU. It was modeled by using Q-learning technique which is a ML approach. This mechanism is totally lies in two phases. In first phase, all incoming tasks are assigned with a VM in cloud using M/M/S queuing mechanism. In Second phase, by using decisions of Q-Learning tasks are allocated to corresponding VMs in cloud. This approach was implemented on Cloudsim, evaluated over Random, Fair Schedulers. In [25], trust aware scheduling mechanism was developed to minimize makespan, to improve QoS and to address security challenges posed in cloud environment. This work was done in three phases i.e. computation of trust levels of VMs, computation of priorities of tasks and careful scheduling of tasks based on above mentioned conditions. It was implemented on a Hadoop cluster and data generated onto Hadoop clusters are collected from Google cloud platform real workload traces and evaluated over PSO,SJF, RR algorithms and finally from results trusted aware scheduling performs better than existing approaches.

In [34], schedulingtechnique is formulated to optimize the significant QoS parameters and modeled by DQTS i.e. a combination of Q-learning and deep neural network. It was implemented on Workflowsim. Initial workload was generated randomly and used synthetic datasets. It evaluated against existing models. From results, it shows impact over existing mechanisms for load balancing. In [28], edge computing-based task scheduling algorithm was developed to maximize task degree satisfaction and success ratios. It was modeled with DRL to solve task scheduling and resource allocation. It was implemented by using python language and compared with FCFS and SJF state of art algorithms. From results above-mentioned parameters were improved to a great extent. In [19], DeepJS, a job scheduling mechanism developed to improve makespan to address scheduling issues in cloud datacenters. It uses reinforcement learning integrated with bin packing algorithm. It simulated by cloudsim and workload taken from real world workload traces. It compared against existing models, which uses heuristics, finally from results DeepJS converges fast more and minimizes makespan compared with other approaches. In [36], authors formulated a QoS aware scheduler aims at response time, utilization of VMs and user request distributions among VMs. It was modeled by using Deep Reinforcement learning. It was implemented on a customized simulation environment. Real world traces of NASA workload were used for simulation and evaluated over RR, FF, random, earliest and best fit approaches. From results, it observed that average response time minimized by DRL approach by 40% and success rate was improved by 93% over compared mechanisms. Authors in [11] formulated an effective scheduling mechanism in fog environment, which aims to reduce delay of service and computational costs. It was modeled by combining Deep Q-Learning and double Q-Learning mechanisms. It was implemented on ifogsim and evaluated against FF, GS and RS algorithms, evaluated metrics energy, cost, and these parameters shown a huge impact against existing algorithms. In [35],proposed workflow scheduling technique to address makespan, cost. Technique used in this technique was DQN model, which is a multi-agent technique based on reinforcement learning which gives rewards as time and cost. It was implemented on real time cloud environment i.e. AWS and extensive simulations were carried out and it shown huge impact in above-mentioned parameters. Authors in [27] developed an energy efficient task scheduler, which uses RANN model. GA used to generate dataset, which is of 18 million instances. It implemented on MATLAB, evaluated over existing approaches, this proposed task scheduler overcomes existing models by makespan, energy consumption, required active racks, execution overhead.In [5], an efficient resource allocation with light weight authentication scheme developed by authors. A hybridized mechanism developed i.e. RATS-HM. It consists of three steps. In first step, they used ICS-TS which optimizes makespan of scheduling mechanism, In second step, they used GO-DDN which is a deep neural network mechanism for efficient allocation of resources and in final step a light weight authentication mechanism developed. Experimentations conducted extensively on Cloudsim. From results, RATS-HM allocated resources effectively to users while addressing deadline constraints. In [16], a workload balancing strategy proposed by authors by addressing parameters i.e. cost, degree of imbalance, resource utilization. MOABCQ i.e. Q-learning added to modified ABC approach to model scheduling strategy. Extensive simulations conducted on Cloudsim. MOABCQ approach evaluated using realtime workload datasets and synthetic workloads. It compared with existing approaches and from results MOABCQ shows significant impact on existing approaches. In [29], authors used deep reinforcement learning approach used to propose energy aware task scheduling model developed to minimize energy consumption, makespan, resource utilization. It compared over SOTA approaches and deep reinforcement approach outperformed for above specified parameters. [37] proposed a task scheduling approach addresses energy concerns in datacenters for realtime workloads. DRL methodology used for energy aware scheduling. Extensive simulations revealed energy aware scheduling mechanism tackled realtime jobs in datacenters by minimizing energy consumption while improving QoS services provided by cloud provider.

From the above Table 1, all existing scheduling algorithms uses different variations of reinforcement learning techniques and addressed metrics, which we have mentioned in above table. Despite usage of above metrics task scheduling is still ineffective and therefore we have used Deep Q-Learning network to schedule tasks effectively by considering priorities of tasks and schedule them by the decision of ML model i.e. DQN addressed metrics makespan, SLA Violation, Energy consumption.

In the below section, we have accurately defined problem, mentioned proposed system architecture in detailed manner.

3 Problem definition and proposed system architecture

In this section, problem definition is given below.

Definition Assume we have K tasks, which are indicated as $t_K = \{t_1, t_2, t_3, \ldots t_K\}$, n VMs which are indicated as $VM_n = \{VM_1, VM_2, VM_3, \ldots, VM_n\}$, $p$physical hosts indicated $H_p = \{H_1, H_2, H_3, \ldots, H_p\}$ and r datacenters indicated $DC_q = \{DC_1, DC_2, DC_3, \ldots.\ .\ DC_q\}$. Scheduling problem defined in such a way that these K tasks are scheduled on to n VMs sitting in p physical hosts in turn resided in q datacenters. Incoming task priorities to be considered before scheduling onto VMs priorities of tasks are considered and fed to DQN model, which takes scheduling decision based on upcoming and current running tasks in underlying resources, which minimizes makespan, SLA violation and Energy consumption. The below table represents notations of proposed architecture (Table 2).

The optimal task scheduling architecture is represented in Fig. 1, which considers diverse requests from different users simultaneously. After submission of tasks to cloud interface application task manager collects those requests and calculates priorities of all tasks based on length of task, processing capacities of tasks. Further, it will be fed to DQN model based on task priorities, which is integrated with scheduling model. From the recommendations of scheduler, which is integrated with DQN model, have to schedule tasks appropriately onto the VMs. Initially, scheduler need to send these prioritized tasks onto execution queue and send these tasks onto VMs. In this proposed architecture, after every certain time interval T our scheduler needs to keep track of upcoming requests and resource manager about virtual resources. For every time interval T scheduler, which consists of DQN, model keeps track of upcoming requests, executing requests in VMs and virtual resources in resource manager. Therefore, based on these conditions scheduler will take a decision dynamically i.e. mapping a task to new VM or mapping a task to an existing VM or migrating existing tasks to another VMs if a VM is sufficient storage and processing capacity to accommodate running tasks. We have used Deep Q-Learning model as a methodology to schedule tasks intelligently based on the above said conditions for every time interval$T$. It will update its decisions of scheduling to scheduler, takes care scheduling tasks appropriately onto VMs. Main aim of this scheduler to effectively task mapping to VMs based on their priorities minimizing parameters named as makespan, SLA Violation and Energy consumption. Initially evaluation of priorities of tasks need to check dependencies of task priorities of tasks. Therefore, evaluation of priorities of tasks entire load on VMs need to be calculated. The overall load on VMs can be identified by following eq. 1.

Table 1 Existing Task scheduling mechanisms using ML Techniques

Authors	Technique	Simulation Environment	Parameters
[5]	RATS-HM	Cloudsim	Makespan, throughput
[26]	RL, DQN, RNN-LSTM and DRL-LSTM.	Matlab	Task Execution Delay, resource utilization
[14]	AIRL	Customized simulator	Response time, Success rate
[8]	DQN	Real time cloud environment.	QoS, VM Cost, Success rate and response time.
[32]	CDDQLS	Cloudsim	Execution time, waiting time of tasks.
[10]	DQN	Matlab	Makespan.
[33]	QL-HEFT	Cloudsim	Makespan
[9]	Q-Learning	Cloudsim	Energy Consumption and CPU Utilization.
[25]	Reinforcement Learning	Hadoop cluster	Makespan, QoS
[34]	DQTS	Workflowsim	Makespan, QoS
[28]	DRL	Pytorch	Task degree satisfaction and success ratio
[19]	DeepJS	Cloudsim	Makespan
[36]	Deep Reinforcement learning	Customized simulation environment	Response time, VM utilization time
[11]	Deep Q-Learning and Double Queue Learning	Ifogsim	Delay of Service and Computational costs.
[35]	DQN	AWS cloud	Makespan and computational costs
[27]	RANN	MATLab	Makespan, energy consumption, execution overhead, required active racks.
[5]	RATS-HM	Cloudsim	Response time, Resource utilization, energy consumption
[16]	MOABCQ	Cloudsim	Makespan, cost, throughput, degree of imbalance
[29]	Deep Reinforcement approach	MATLAB	Makespan, throughput, resource utilization, energy consumption
[37]	ESRJS	Cloudsim	Success rate, response time, energy consumption

Table 2 Notations used in Proposed System Architecture

Notation	Meaning
t_K	Tasks
VM_n	Virtual Machines
H_p	Physical Hosts
DC_q	Datacenters
lo^{VM}	Current running load on virtual machines
lo^{H_p}	Current running load on Hosts
tr^p	Threshold value identification
pr_{ca}^{VM}	Processing capacity of a VM
t_k^l	Task length
t^{prio}	Task Priority
et_{t_k}	Execution time of a task
ft^{t_k}	Finishing time of a task
dl^t	Deadline constraint
m^k	Makespan of a task
e^{con}	Energy consumption
γ_n	Active state of VM
τ_n	Idle state of VM
res^{util}	Resource utilization
$SLAV$	SLA Violation
T_i^{over}	Total over loaded time of physical hosts
$T_i^{over-active}$	Total active time of physical hots
T_j^{pd}	Performance degradation of VM
T_j^{cpc}	Requested CPU capacity of VM for specific time
S^t	State space
A^t	Action space
σ	Rate of Learning
re^t	Reward function
$£$	Discount factor
S_t	state of a task t at time T
S_{Tt}^{VM}	State of a VM for a task t at time T
$M\omega$	Replay Memory
ε	Time for Learning agent
f	Frequency of learning

$$lo^{VM} = \sum lo^n \tag{1}$$

Where lo^n indicates current running n number of VMs.

After calculation of current load on VMs, as all VMs are running in p physical hosts. Therefore, overall load on hosts are calculated using eq. 2.

$$lo^{H_p} = lo^{VM} / \sum H_p \tag{2}$$

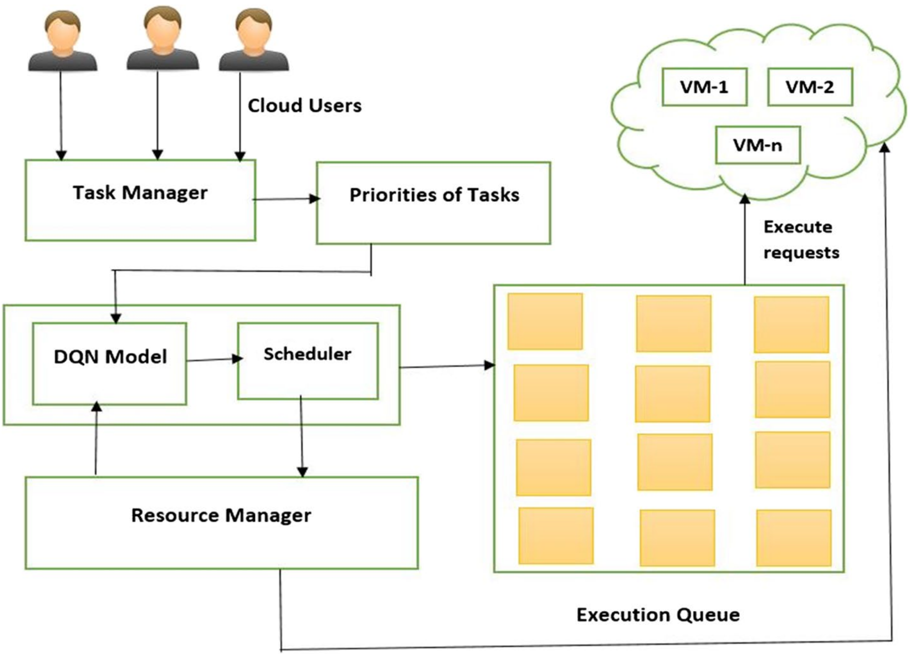


Fig. 1 Proposed optimal task scheduling Architecture

Where lo^{H_p} indicates load on p physical hosts, lo^{VM} indicates current load on all VMs, H_p indicates hosts.

After calculation of load on VMs and physical hosts, identified a threshold value as cloud computing paradigm is dynamic and to process huge number of requests by VMs in a balanced manner and this load balancer need to work according to the requests coming onto VMs. Therefore, to have a load balancer in our model, a threshold value was calculated. Threshold value should be dynamic as cloud workloads are not static and different parameters such as upcoming requests, existing resource capacity etc. Therefore, threshold value in this model can be calculated using following eq. 3.

$$tr^p = \frac{\sum_{i=1}^{p} lo_i^{H_p}}{p} \tag{3}$$

Where tr^p is a dynamic threshold value identified in our work, $lo_i^{H_p}$ is load on p physical hosts. This threshold value continuously changing as workload in cloud is dynamic and based on threshold value, utilization of hosts are calculated whether they are underutilized, balanced or over utilized. Utilization of hosts can be calculated using following eqs. 4, 5 and 6 respectively.

The below eq. 4 used to calculate over utilization of hosts.

$$VM_n > tr^{\,p} - \sum lo^{VM} \tag{4}$$

The below eq. 5 used to calculate underutilization of hosts.

$$VM_n < tr^p - \sum lo^{VM} \tag{5}$$

The below eq. 6 used to calculate balanced utilization of hosts

$$VM_n = tr^p - \sum lo^{VM} \tag{6}$$

From above equations, 4, 5 and 6 utilization of hosts through dynamic threshold value is calculated. Now, to schedule appropriate workload over cloud resources (VMs), need to calculate the processing power of resources as calculated in eq. 7. It is defined as multiplication of number of processing elements in VM, number of processing instructions per second in VM. It calculated by using following equn. Mentioned below.

$$pr_{ca}^{VM} = pr^{no} * pr^{mips} \tag{7}$$

The above equation shows that it is processing capacity of particular VM in n number of VMs considered in our architecture and after this entire processing capacity of VMs calculated as follows by using eq. 8.

$$ovr_{vm}^{pr} = \sum pr_{ca}^{VM} \tag{8}$$

Now after calculation of processing capacities of VMs, priorities of upcoming requests calculated based upon dependencies or inter-dependency, size of tasks, required resources by the request and many more parameters. Therefore, length of task is calculated using following eq. 9.

$$t_k^l = t_{mips} * t_{pr} \tag{9}$$

After calculation of length of task then priority of an incoming task onto scheduler can be calculated by using below equation.

$$t^{prio} = \frac{t_k^l}{pr_{ca}^{VM}} \tag{10}$$

Based on priorities of tasks these are moved onto execution queue and map those tasks by scheduler to appropriate VMs. For this scheduling model, we have also considered a dead line constraint in a way that task should complete its execution before deadline i.e. dl^t.

In this research work, our focus is to address parameters i.e. makespan, SLA Violation, Energy Consumption. Whenever makespan is evaluated, to calculate execution time because makespan is evaluated based on how much execution time it is taking for a task to run on a certain VM. Execution time of task for a certain task calculated by using following equation.

$$et_{t_k} = \frac{et_t}{pr_{ca}^{VM}} \tag{11}$$

For every task, which is scheduled into execution, queue gets a VM based on its availability in resources and it all depends on finishing time of a task. Therefore, finishing time of task calculated using below equation.

$$ft^{t_k} = \sum VM_n + et_{t_k} \tag{12}$$

In this model, we assumed that each task should complete its execution within specified deadline. Therefore, for every task we are scheduling in this model finish time should always be less than or equal to its deadline. It is indicated as below.

$$ft^{t_k} \leq dl^t \tag{13}$$

Here after mentioning deadline constraint, calculation of execution time, finish time then we have calculated makespan as in any scheduling model or mechanism makespan needs to be minimized. It is defined as execution time of tasks running over virtual resources. It is calculated as follows.

$$m^k = \max\left(ft^{VM_n}\right) \tag{14}$$

$$\min ft\left(t_k VM_n\right) = \sum_{i=1}^{k} \sum_{j=1}^{n} \delta_{ij} ft\left(t_k VM_n\right) \tag{15}$$

From above eq. 15 δ_{ij} is set to 1 if task t_k is assigned to VM i.e. VM_n otherwise set to 0.

Thus, from eqs. 14 and 15 makespan is calculated.

Our next focus to minimize the energy consumption in cloud computing paradigm. It is one of significant and impactful parameters in cloud paradigm. As for processing the huge workloads, need the large scale infrastructure or cloud resources, which leads to increase the energy consumption as well as large emissions of CO_2 [21, 38]and damages environment. Therefore, we are focusing on minimizing energy consumption in cloud paradigm. In Cloud model, energy consumption based on consumption for computing time, idle time. In this model for a VM energy consumption is calculated as follows. In cloud computing model any VM either should be in active state i.e. computing instructions or it should be in idle state represented in below eq. 16.

$$VM_n = \begin{pmatrix} \gamma_n \ \ Active\ State\ of\ VM \\ \tau_n \ \ Idle\ State\ of\ VM \end{pmatrix} \tag{16}$$

Energy consumption of all n VMs are calculated by using following equation

$$e_{VM_n}^{con} = ft_n * \gamma_n + \left(m^k - ft_n\right) * \tau_n \tag{17}$$

$$\min_{act}^{con} = \left(e^{mx} - e^{mn}\right) * res^{util} + e^{mn} \tag{18}$$

Energy consumption in datacenter calculated as follows

$$e^{con} = \sum e_{VM_n}^{con} + \min_{act}^{con} \tag{19}$$

Thus, from eqs. 16, 17, 18 and 19 Energy consumption in cloud computing calculated.

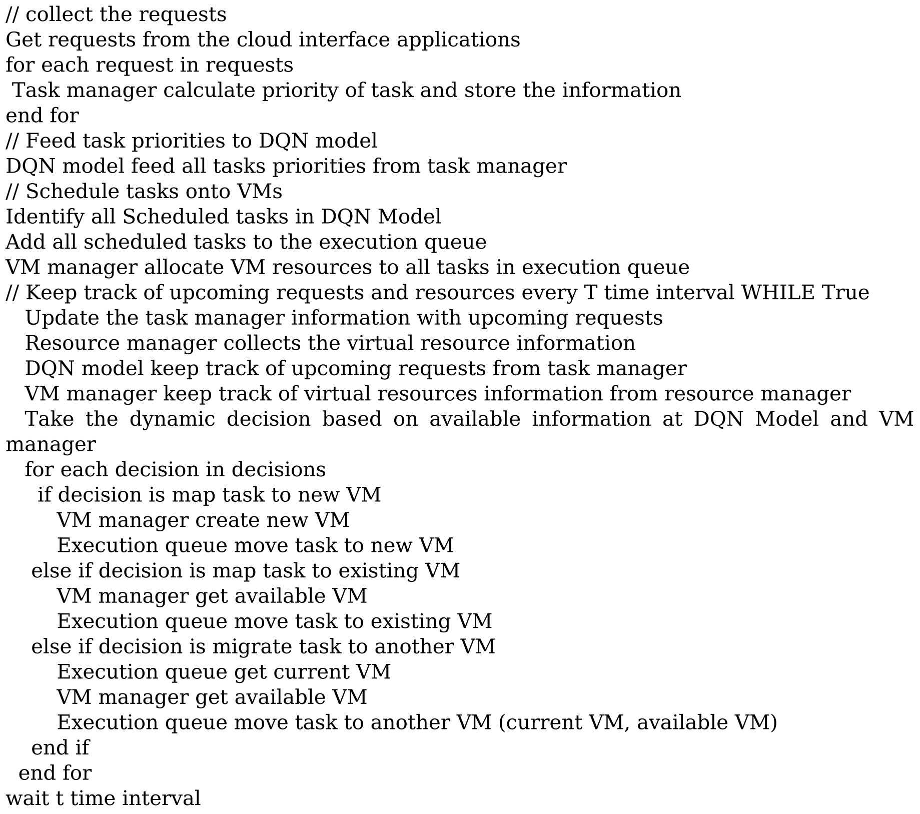
Our next focus is to minimize SLA Violation in cloud computing. Here, we need to discuss importance of SLA Violation. Service Level Agreement is one of the important perspective in terms of both user and cloud provider as if our system does not work according to SLA then problems will be persisted for both cloud provider and user. Therefore, it is important to design our scheduler, which should not violate SLA made between both user and cloud provider. Thus, we have defined SLA violation in cloud computing by using below equation.

$$SLAV = \frac{1}{p}\sum_{i=1}^{p}\frac{T_i^{over}}{T_i^{over-active}} * \frac{1}{n}\sum_{j=1}^{n}\frac{T_j^{pd}}{T_j^{cpc}} \tag{20}$$

Where p indicates number of physical hosts, T_i^{over} indicates total time for which host gets overloaded, $T_i^{over-active}$ indicates amount of time a host lies in active state.T_j^{pd} indicates estimation of performance degradation for a VM, T_j^{cpc} indicates requested CPU capacity of a VM during its specified time.

4 Methodology

This section precisely discusses about methodology used to design our scheduling algorithm. We have used a reinforcement learning approach [31], which takes adaptive decisions. It is a machine learning approach, which considers inputs and gives decisions based on the history of previous events. Over a period of time it learns from previous decisions and makes adaptive decisions. For any Reinforcement learning approach there are three basic parts. They are 1. Input and Output states- The data which we are given



```
// collect the requests
Get requests from the cloud interface applications
for each request in requests
 Task manager calculate priority of task and store the information
end for
// Feed task priorities to DQN model
DQN model feed all tasks priorities from task manager
// Schedule tasks onto VMs
Identify all Scheduled tasks in DQN Model
Add all scheduled tasks to the execution queue
VM manager allocate VM resources to all tasks in execution queue
// Keep track of upcoming requests and resources every T time interval WHILE True
   Update the task manager information with upcoming requests
   Resource manager collects the virtual resource information
   DQN model keep track of upcoming requests from task manager
   VM manager keep track of virtual resources information from resource manager
   Take the dynamic decision based on available information at DQN Model and VM
manager
   for each decision in decisions
     if decision is map task to new VM
        VM manager create new VM
        Execution queue move task to new VM
    else if decision is map task to existing VM
        VM manager get available VM
        Execution queue move task to existing VM
    else if decision is migrate task to another VM
        Execution queue get current VM
        VM manager get available VM
        Execution queue move task to another VM (current VM, available VM)
    end if
  end for
wait t time interval
```

Proposed Methodology Pseudo Code

as an input to the model is Input and the output state is to represent an outcome processed by algorithm based on data given as Input. 2. Rewards- This state is a representation of outcome by algorithm with positive or negative reward. 3. Artificial Intelligence framework- This can be used to take a decision based on input supplied to algorithm and gives outcomes through which rewards will be generated that may be good or bad. If those rewards generated at a time *T* are positive and framework captures those positive rewards and continue towards the next state for much more optimal decisions. If rewards are negative, it will learn from that experience and it will try to improve its decision making for the next state.

The complete functionalities of proposed methodology presented in the form of pseudo code.

Time complexity of the proposed Methodology:

The total time complexity of proposed methodology depends on time complexities of individual components of proposed methodology. Let's break down each component and analyze its time complexity:

1. Collect requests and calculate task priorities: Time complexity of this component is O(n), n indicates number of requests received from cloud interface application.
2. Feed task priorities to DQN model: The time complexity of this component depends on the implementation of the DQN model. In general, the time complexity of a single forward pass through a neural network with n layers is O(n), since each layer involves a matrix multiplication operation.
3. Schedule tasks onto VMs: The time complexity of this component depends on the implementation of the scheduler and the DQN model. The time complexity of scheduling tasks using a DQN model is typically higher than traditional scheduling algorithms like round robin, as it involves training a neural network on a large dataset. The time complexity of adding tasks to the execution queue and scheduling them onto VMs depends on the implementation of the execution queue and VM manager.
4. Keep track of upcoming requests and resources every T time interval: The time complexity of this component is O(1), since it involves fetching the upcoming requests and virtual resources from the resource manager.
5. Make dynamic decisions: The time complexity of this component depends on the number of decisions to be made, and the time complexity of each decision. In general, time complexity of component is O(m), where m is number of decisions to be made.

Overall, time complexity of algorithm can be approximated as O(n + f + s + m), where n is number of requests received, f is time complexity of feeding the task priorities to the DQN model, s is the time complexity of scheduling tasks onto VMs, and m is the time complexity of making dynamic decisions.

Reinforcement learning approach uses agents to take decisions based on inputs given to system. Generally, any machine-learning model consists of different states. Therefore, agents will take specific actions based on input, which generates rewards either good or bad. These rewards can be useful for the decision to be taken by the algorithm in the next state. Reinforcement learning approach works based on these rewards and agents will try to take actions in next state based on reward generate in this current state. This approach learns from its previous states whether it is a good or bad reward and will take its decisions over a period of time. This adaptive nature i.e. self-learning based on its previous states is one of the advantage of reinforcement learning approach.

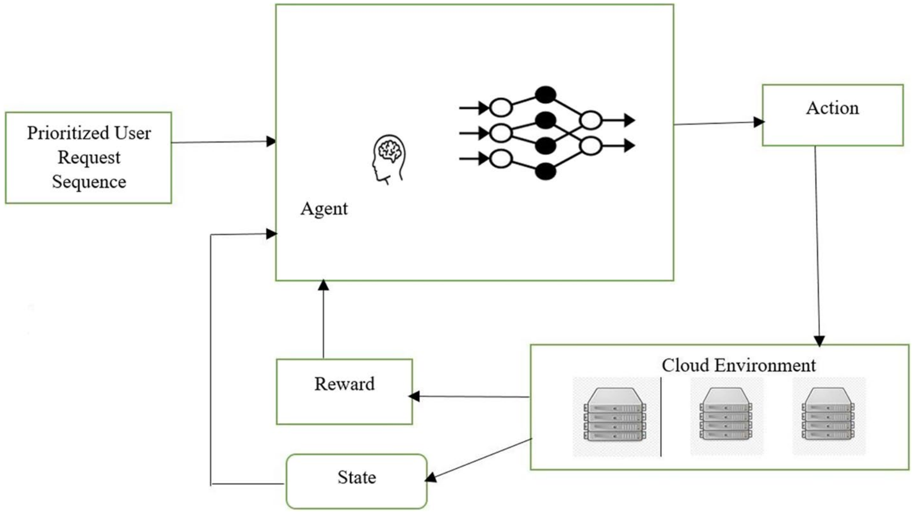


Fig. 2 Deep reinforcement learning technique for scheduling the tasks

The above Fig. 2 gives representation of task scheduling using deep reinforcement learning in which, agent will learn through history of incoming user request sequences i.e. agent will be trained through previous user requests or tasks which are coming onto cloud console. Initially, a Prioritized user request is to be given as input to agent and it should make a scheduling decision based on situation in cloud environment. Decision would be given as an output of executed task or user request, which is (e.g. makespan, energy consumption, SLA violation in our study) a reward for an agent. If the value of reward is bad then agent improve its decision by updating its parameters of model. If the value of the reward is good then it will be stored in the current state and it will be used for the next time when decision is to be made by the agent in the next state.

In reinforcement learning, we have used Q- learning for our scheduling model [30]. This Q- learning is one of the most powerful technique as it doesn't need any knowledge of current system. It will make decisions based on past actions stored as Q-function as a pair with two states indicated a q(S, A). It will updates its states by using below equation.

$$q\left(S^t, A^t\right) \leftarrow q\left(S^t, A^t\right) + \sigma * \left[re^t + \pounds * maximum^a q\left(S^{t+1}, A\right) - q\left(S^t, A^t\right)\right] \tag{21}$$

Where σ is rate of learning and value of it is in between (0,1). re^t is reward for taking action i.e. A^t for state S^t. $\pounds$ is discount factor and its value lies in between (0,1).

From above equation for every iteration, q-learning model needs to check for rewards and updates its decisions according to model by using above equation. In classical q-learning, all q values are stored in q-table but to apply these q-learning model to a problem such as scheduling in cloud computing it is difficult to adaptive and optimal decisions for classical model as number of states and actions are comparably high for task scheduling technique. Therefore, we want to use a deep neural network in combination with reinforcement learning which can be helpful for our scheduling problems in cloud computing. Therefore, we have used a Deep Reinforcement Learning model [39] to tackle the problem of scheduling in cloud computing. Moreover, that Deep

Reinforcement Learning approach already proved in various types of scheduling techniques in cloud computing as mentioned in [15, 26]. Therefore, combining deep neural network with reinforcement learning used for task scheduling in our model. The main reason to usethis scheduling model is to make it as a smart scheduler no prior knowledge will be given to agent and algorithm need to take a decision when real time data is given as input. It consists of different states as in q-learning, which consists of action space and state space.

4.1 Action space

In action space, as we have already mentioned *n* VMs which we have considered in this work. All incoming requests initially fed to task manager and after each task priority will be calculated and given to scheduler and it consists of DQN model which takes decision and sends tasks to a execution queue with respect to priorities of tasks. Then based on decision of Scheduler they need to execute on VMs according to the entry of tasks in queue according to their priorities. Thus, action space in our model is defined as follows.

$$A = \left[VM_1, VM_2, VM_3, \ldots, VM_n\right] \tag{22}$$

4.2 State space

In this subsection, we are defining a state space where it consists of state of a task at specific time and state of a VM at that time when task arrives.

Let us assume that a task *t* arrives at time *T* and it is to be represented as T_t. Then, state of this task can be represented as follows

$$S_{Tt} = S_t \cup S_{Tt}^{VM} \tag{23}$$

Where S_t is a state of a task *t* at time *T* and S_{Tt}^{VM} is a state of a VM when a task *t* comes on to a VM at time *T*.

$$S_{Tt} = \left[t_k^l, t^{\,prio}, et_{t_k}, ft^{t_k}, m^k, e_{VM_n}^{con}, SLAV\right] \tag{24}$$

Where t_k^l is length of*k* tasks, $t^{\,prio}$ is priority of *k*tasks, et_{t_k} is execution time of *k*tasks, ft^{t_k} is finish time of *k*tasks, m^k makespan of *k*tasks, $e_{VM_n}^{con}$ energy consumption of n VMs, *SLAV* is SLA Violation.

4.3 Reward function

Aim of this study is to find optimal mapping between cloud resources, mixed tasks with help of our DRLBTSA scheduler to optimize significant QoS parameters energy consumption, time, SLA violation. Thus, our reward function should be in terms of minimization of metrics mentioned in our work. It can be defined as follows.

$$re = \min\left(m^k, e_{VM_n}^{con}, SLAV\right) \tag{25}$$

Input: Number of tasks $t_K = \{t_1, t_2, t_3, \cdots t_K\}$, Number of VMs $VM_n = \{VM_1, VM_2, VM_3, \ldots, VM_n\}$, No.of Physical hosts $H_p = \{H_1, H_2, H_3, \cdots, H_p\}$, Number of Datacenters $DC_q = \{DC_1, DC_2, DC_3, \ldots\ldots DC_r\}$, parameters of DQN model.

Output: Mapping of tasks to virtual resources while minimizing Energy Consumption, SLA Violation, makespan.

```
Initialize ρ,σ,£,f, batch □ω, time for learning agent is ε.
Set$q(S^t, A^t)$ to 0.
for each event do
    choose $S^t$
    evaluate priority of tasks using eqn. 10
    for each event do
        check priority of a task to be scheduled and resources availability
        Choose action space i.e. VM with random probability ρ or choose $argmax_x^A q(S^t, A)$ .
        Evaluate reward by using equation 25.
        Check reward $re$ for corresponding action.
        Update equation 21 for the next action for corresponding state chosen.
        Update state from $S^t$ to $S^{t+1}$
        update probability value i.e. ρ
        repeat this until the last state
    end do
end do
```

Proposed DRLBTSA- Proposed Deep reinforcement learning based Task Scheduling Algorithm in Cloud Computing.

4.4 Training the agent

When incoming tasks arrived at scheduler DRLBTSA agent need to make decision in current state by considering priorities of tasks and resources available in Physical hosts and according to that it should map tasks to VMs. For this to happen, our DQN model need to be trained in such a way that initially it should map a task to a VM by considering above said condition with a probability ρ and its value decrease over to zero with respect to time. Therefore, initially DQN agent explores randomly and give its decision and later it gives its decisions by previous q-values stored in q-table. To make it happen, experience replay, fixed q-target values are to be used in algorithm. Whenever agent takes decisions over a period of time it will gain experience and it is to be represented as experience replay here in our work. Whenever a decision is taken by agent it will gives you a reward and it is represented as re^t and further state is to be represented as S^{t+1}. These experiences will be stored as values in a memory to be represented as replay memory indicated as ω. The values to be stored in this replay memory are (A^t, S^t, re^t, S^{t+1}) and capacity of a replay memory to be represented as Mω, and replay memory can be taken as batches and indicated as Gω. Whenever iterations are to be run and values in replay memory are to be updated and here iterations are to be called as batches. In our work, entire training was done in offline. After completion of training to our agent then it will become intelligent enough to take decisions in a smart way. In our work, 50 neurons were used for hidden layers in DQN model. We have kept scheduling time for taking decision for an agent is 10 ms, frequency of learning is $f = 1$, time for learning of agent taken as ε. The proposed DRLBTSA algorithm is shown below.

The above algorithm flow is discussed here in a detailed manner. Initially all parameters such as batch size, replay memory, learning frequency, learning rate and discount factor are initialized. In the next step, q-function consists of state space and action space are initialized to zero. In the next step, for each event i.e. for every incoming task comes to cloud platform priority of tasks are calculated. For every event, scheduler need to choose action space i.e. VMs for corresponding state space i.e. tasks based on priority of tasks, availability of resources in physical hosts. Based on this condition, every time scheduler need to make a decision. In the next step, tasks according to their priorities to be scheduled to corresponding VMs with a random probability if it is a first task which is to be scheduled for first time otherwise DRLBTSA need to take a decision from existing q-table available for agent. After this step when action space to be chosen for every state space a reward will be generated. Here in our work, it is minimization of parameters i.e. makespan, energy consumption, SLA Violation. Reward value can be calculated by using eq. 25. If it is positive reward, reward score will be improved and if it is negative then it have to improve i.e. scheduler need to improve based on its experience. If positive rewards are encountered i.e. makespan, energy consumption, SLA violation they will be updated as minimized values of the current state. Rewards either positive or negative they will be stored in replay memory. After evaluating rewards it should update its state space to the next state by using eq. 21. This process continues until last state space i.e. task is encountered (Fig. 3).

5 Simulation set up and experimental results

This section discusses simulation, results of our work. Entire simulation is conducted on cloudsim [7] simulator. We have identified real time parallel worklogs from HPC2N [13] and NASA [23] which are of high performance computing clusters, those workloads are given input to our algorithm. After that we have fabricated different datasets on our own with different distributions and those were explained here in a detailed manner.

5.1 Configuration settings for simulation

Our simulation was carried out on cloudsim [7]. We have done extensive simulations using cloudsim [7]. In our work, initially we fabricated datasets in such a way that tasks have to be with different distributions and these workload distributions are fed to scheduler. Then to test our work efficiency, we used HPC2N [13], NASA [23] worklogs from high performance computing clusters. The fabricated datasets distributions are considered as follows i.e. Uniform, Normal, left and right skewed distributions. We represent all these datasets with uniform, normal, left and right skewed distributions as d1, d2, d3, d4. Worklogs of HPC2N and NASA are represented as d5 and d6. Uniform distribution tasks means all types of tasks distributed in an equal manner. Normal distribution represents more number of medium size tasks, less number of small and large tasks. Left skewed distribution indicates more number of small tasks, less number of large tasks. Finally Right skewed distribution indicates less number of small tasks, more number of large tasks. We have intentionally fabricated these distributions as we need to verify how our algorithm behaves with different types of tasks. Finally we have given HPC2N and NASA parallel worklogs as d5 and d6 datasets as input to algorithm to check its efficiency through real-time workload. After giving workload as input to algorithm, we have evaluated our DRLBTSA against existing baseline algorithms RR, FCFS, Earliest deadline first, RATS-HM,

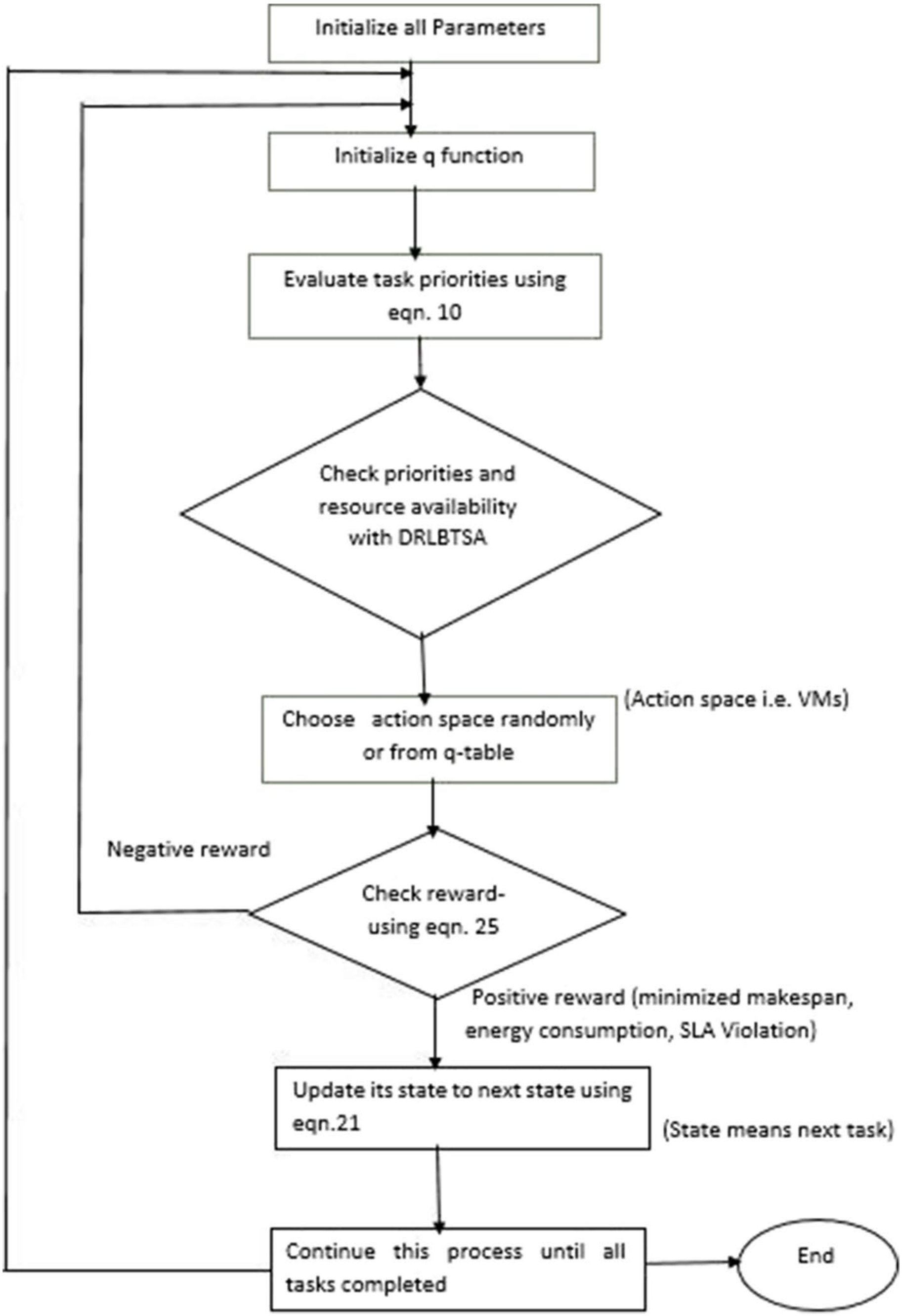

Initialize all Parameters
Initialize q function
Evaluate task priorities using eqn. 10
Check priorities and resource availability with DRLBTSA
Choose action space randomly or from q-table
(Action space i.e. VMs)
Negative reward
Check reward- using eqn. 25
Positive reward (minimized makespan, energy consumption, SLA Violation)
Update its state to next state using eqn.21
(State means next task)
Continue this process until all tasks completed
End

Fig. 3 Flowchart of proposed approach

Table 3 Configuration Settings in Simulation

Name of the entity	Quantity
No. of tasks	1000
Task length	700,000
Ram of Host	32 GB
Storage of Host	4 TB
Bandwidth	1000 MBPS
No. of VMs	35
Ram of VMs	4 GB
Bandwidth of VM	200 MBPS
VMM	Xen
Operating System	MAC
No. of Datacenters	10

Table 4 Parameter Settings for various algorithms

Name of the Algorithm	Parameter	Values
MOABCQ [16]	Size of Population	1000
	Number of VMs	100
	Maximum number of iterations	50
	Employed bees	200
	Onlooker bees	800
RATS-HM [5]	Cloudlets	1600
	Number of tasks	300
	Memory	500 MB
	Physical host capacity	500GB
	Number of VMs	4
RR [6]	Hypervisor	Xen
	Memory of VM	256 MB
	MIPS	1000
	Bandwidth	1000Mbps
	Number of Processing elements	1
	VM image size	10,000 MB
	Length of Cloudlets	1000
	Cloudlet allocation policy	Timeshared
FCFS [18]	Hypervisor	Xen
	Memory of VM	512 MB
	MIPS	256
	Bandwidth	1000Mbps
	Number of Pes	1
	Length of Cloudlets	1000
	Cloudlet allocation policy	Spaceshared
EDF [24]	No. of Cloudlets	100
	No. of VMs	150
	Memory of VM	2048 MB
	Bandwidth	1050Mbps
	Number of Pes	1

MOABCQ algorithms. We have taken standard configuration settings for our simulation from [20]. The following Table 3 clearly mention configuration settings required for simulation, Table 4 indicates various parameter settings of compared approaches with proposed DRLBTSA.

5.2 Calculation of makespan

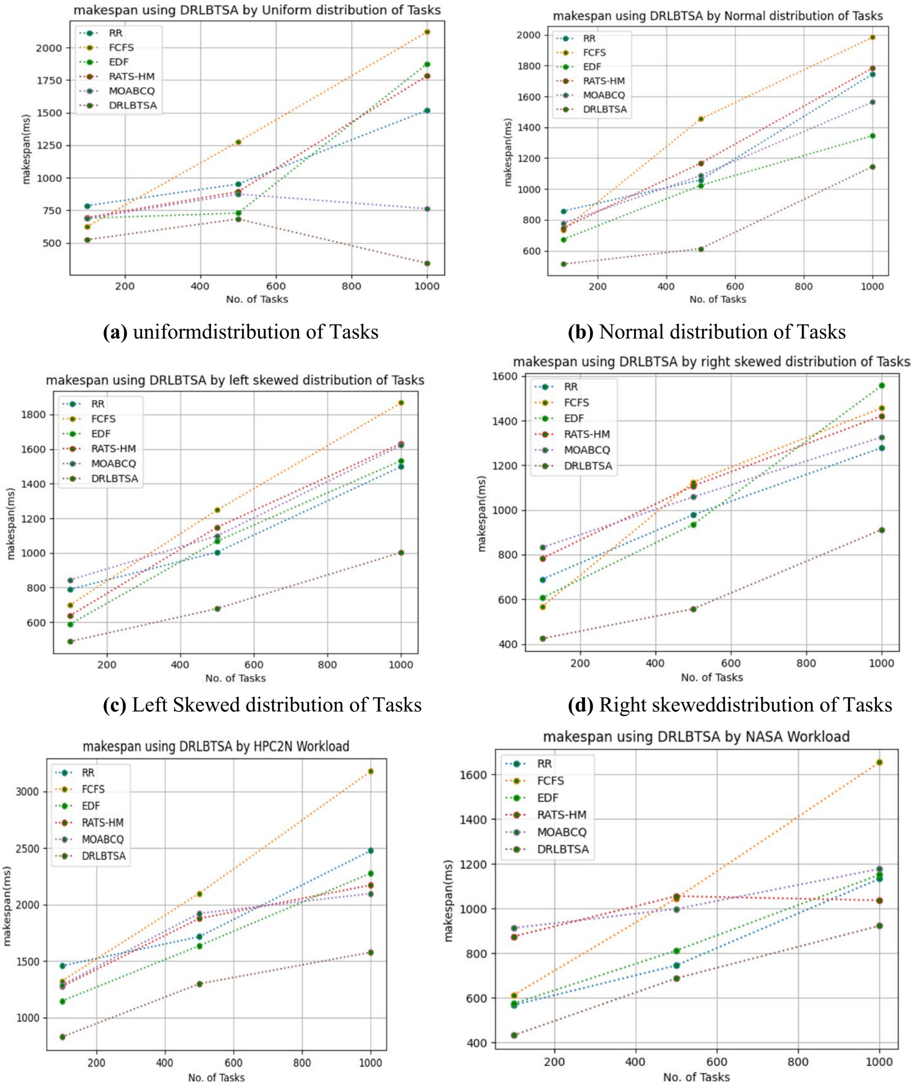
Makespan of tasks are evaluated using configuration settings available in Table 3 and different workloads are given to our DRLBTSA scheduler. Initially we have given workloads of d1, d2, d3, d4, d5 and d6 datasets and evaluated makespan using these datasets. We have evaluated our work against existing baseline algorithms RR, FCFS, Earliest deadline first. DRLBTSA ran for 100 iterations. Below Table 5 indicates calculation of makespan.

The above Table 5 shows makespan of different tasks with different fabricated datasets i.e. d1, d2, d3, d4 with different distributions and workloads from HPC2N [13], NASA [23]. From Table 4, it was clearly shows that our DRLBTSA algorithm minimized makespan over RR, FCFS, EDF,RATS-HM,MOABCQ algorithms.

The above Fig. 4 and Table 4 clearly shows that our proposed DRLBTSA approach evaluated over RR, FCFS, EDF, RATS-HM, MOABCQ algorithms and from simulation results makespan is minimized over the mentioned algorithms.

Table 5 Evaluation of makespan

No. of Tasks	RR	FCFS	EDF	RATS-HM	MOABCQ	DRLBTSA
d1						
100	785.67	623.8	689.8	694.7	687.35	523.9
500	951.9	1276.9	728.9	893.45	875.36	683.9
1000	1518.8	2120.9	1876.9	1784.34	1762.17	1342.6
d2						
100	856.43	734.28	672.99	746.78	778.35	512.89
500	1058.9	1453.9	1023.78	1167.24	1087.21	612.99
1000	1745.8	1983.78	1345.77	1784.24	1563.26	1145.9
d3						
100	789.34	698.53	587.87	636.72	843.21	487.98
500	1004.25	1247.54	1067.92	1147.89	1098.12	678.36
1000	1498.52	1867.78	1532.98	1632.99	1621.78	1002.78
d4						
100	689.67	567.38	606.19	783.32	832.18	424.43
500	978.54	1124.79	934.89	1107.78	1058.32	557.45
1000	1278.88	1457.26	1557.12	1421.53	1326.28	912.34
d5						
100	1456.78	1321.99	1145.34	1273.41	1287.21	825.67
500	1714.87	2098.78	1634.37	1876.23	1921.56	1297.32
1000	2478.99	3178.99	2279.56	2172.94	2098.32	1576.99
d6						
100	567.98	612.87	574.35	875.32	913.45	432.56
500	745.78	1045.89	812.47	1056.12	998.78	687.99
1000	1134.45	1654.35	1152.34	1037.45	1178.35	923.56



(a) uniformdistribution of Tasks

(b) Normal distribution of Tasks

(c) Left Skewed distribution of Tasks

(d) Right skeweddistribution of Tasks

(e) HPC2N Parallel worklogs

(f) NASA Parallel worklogs

Fig. 4 Calculation of makespan using DRLBTSA

5.3 Calculation of energy consumption

Energy consumption is evaluated using configuration settings available in Table 3 and different workloads are given to our DRLBTSA scheduler. Initially we have given workloads of d1, d2, d3, d4, d5 and d6 datasets and evaluated Energy Consumption using these datasets. We have evaluated our work against existing baseline algorithms RR, FCFS, EDF, RATS-HM, MOABCQ. DRLBTSA ran for 100 iterations. Below Table 6 indicates calculation of Energy Consumption.

Table 6 Evaluation of Energy Consumption

No. of Tasks	RR	FCFS	EDF	RATS-HM	MOABCQ	DRLBTSA
d1						
100	79.78	82.5	69.72	72.43	71.43	28.54
500	88.57	95.8	79.89	85.14	80.18	49.78
1000	121.8	113.9	131.25	123.12	114.76	98.23
d2						
100	80.71	95.47	77.57	83.57	69.38	34.25
500	92.45	72.35	87.88	79.38	75.48	41.98
1000	112.21	124.76	113.29	108.21	102.52	90.88
d3						
100	68.67	74.89	59.46	78.13	73.31	47.28
500	79.29	118.23	92.47	83.17	87.99	32.76
1000	103.25	134.98	121.09	109.21	105.21	100.78
d4						
100	74.89	89.32	72.78	68.43	64.80	30.59
500	65.09	109.87	87.23	94.21	82.17	41.28
1000	118.99	130.22	102.28	99.87	96.48	82.19
d5						
100	69.98	71.30	62.47	72.75	77.67	32.99
500	84.15	114.37	91.79	89.18	84.25	45.99
1000	103.36	129.46	105.91	112.75	107.21	88.74
d6						
100	77.99	62.43	83.57	87.46	69.13	37.92
500	81.73	93.98	92.89	95.37	74.23	62.88
1000	112.99	102.9	124.89	119.67	108.33	74.89

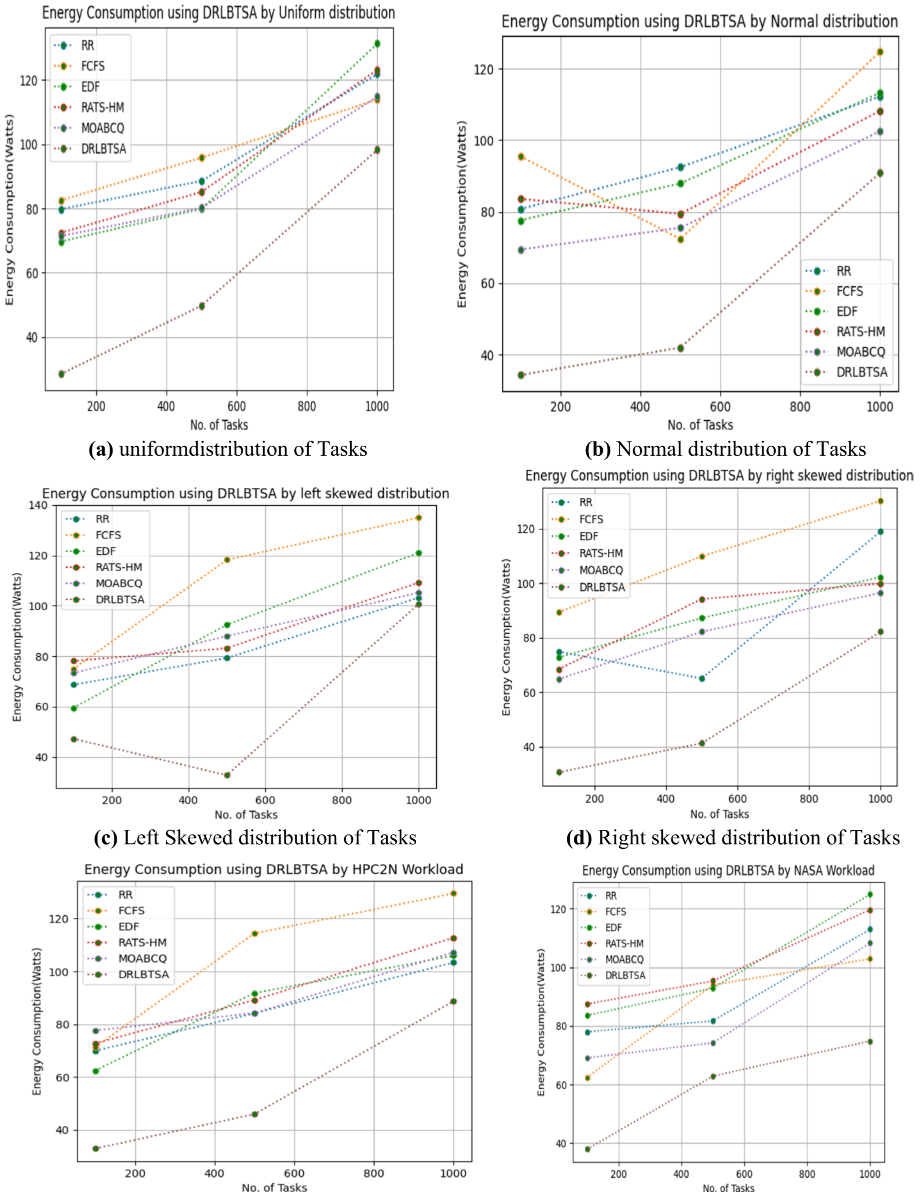
The above Table 6 shows Energy consumption of different tasks with different fabricated datasets i.e. d1, d2, d3, d4 with different distributions and workloads from HPC2N [13], NASA [23]. From Table 5, it was clearly shows that our DRLBTSA algorithm minimized energy consumption over RR, FCFS, EDF,RATS-HM, MOABCQ algorithms.

The above Fig. 5 and Table 5 clearly shows that our proposed DRLBTSA approach evaluated over RR, FCFS, EDF, RATS-HM, MOABCQ algorithms and from simulation results Energy Consumption is minimized over the mentioned algorithms.

5.4 Calculation of SLA violation

SLA Violation is evaluated using configuration settings available in Table 3 and different workloads are given to our DRLBTSA scheduler. Initially we have given workloads of d1, d2, d3, d4, d5 and d6 datasets and evaluated SLA Violation using these datasets. We have evaluated our work against existing baseline algorithms RR, FCFS, EDF, RATS-HM, MOABCQ. DRLBTSA ran for 100 iterations. Below Table 7 indicates calculation of Energy Consumption.

The above Table 7 shows SLA Violation of different tasks with different fabricated datasets i.e. d1, d2, d3, d4 with different distributions and workloads from HPC2N [13],



(a) uniformdistribution of Tasks

(b) Normal distribution of Tasks

(c) Left Skewed distribution of Tasks

(d) Right skewed distribution of Tasks

(e) HPC2N Parallel worklogs

(f) NASA Parallel worklogs

Fig. 5 Calculation of Energy Consumption using DRLBTSA

NASA [23]. From Table 6, it was clearly shows that our DRLBTSA algorithm minimized SLA violation over RR, FCFS, EDF,RATS-HM, MOABCQ algorithms.

The above Fig. 6 and Table 6 clearly shows that our proposed DRLBTSA approach evaluated over RR, FCFS, EDF,RATS-HM,MOABCQ algorithms and from simulation results SLA Violation is minimized over the mentioned algorithms.

Table 7 Evaluation of SLA Violation

No. of Tasks	RR	FCFS	EDF	RATS-HM	MOABCQ	DRLBTSA
d1						
100	23.87	19.24	21.23	26.17	21.21	16.56
500	18.92	16.74	17.8	18.34	16.11	12.29
1000	12.56	8.53	12.4	11.77	10.37	5.89
d2						
100	25.23	20.45	18.92	19.22	17.11	15.78
500	17.76	19.89	14.45	13.11	12.41	9.21
1000	11.24	8.22	12.36	10.21	11.33	7.78
d3						
100	28.67	22.86	24.45	17.36	16.19	14.32
500	19.78	17.34	18.72	19.67	18.11	10.12
1000	15.99	11.25	12.21	11.87	12.06	8.45
d4						
100	23.56	19.21	18.21	17.22	14.22	12.21
500	14.45	13.12	11.14	12.31	12.05	9.34
1000	11.24	10.9	9.5	11.12	10.33	5.7
d5						
100	32.34	34.23	26.24	19.13	18.87	17.23
500	24.45	26.89	19.25	17.28	16.43	8.78
1000	19.23	17.7	8.9	12.08	10.22	4.6
d6						
100	30.78	36.26	28.78	24.32	21.12	15.34
500	23.99	21.9	19.67	20.99	19.34	10.23
1000	15.8	16.56	10.45	18.42	17.21	3.56

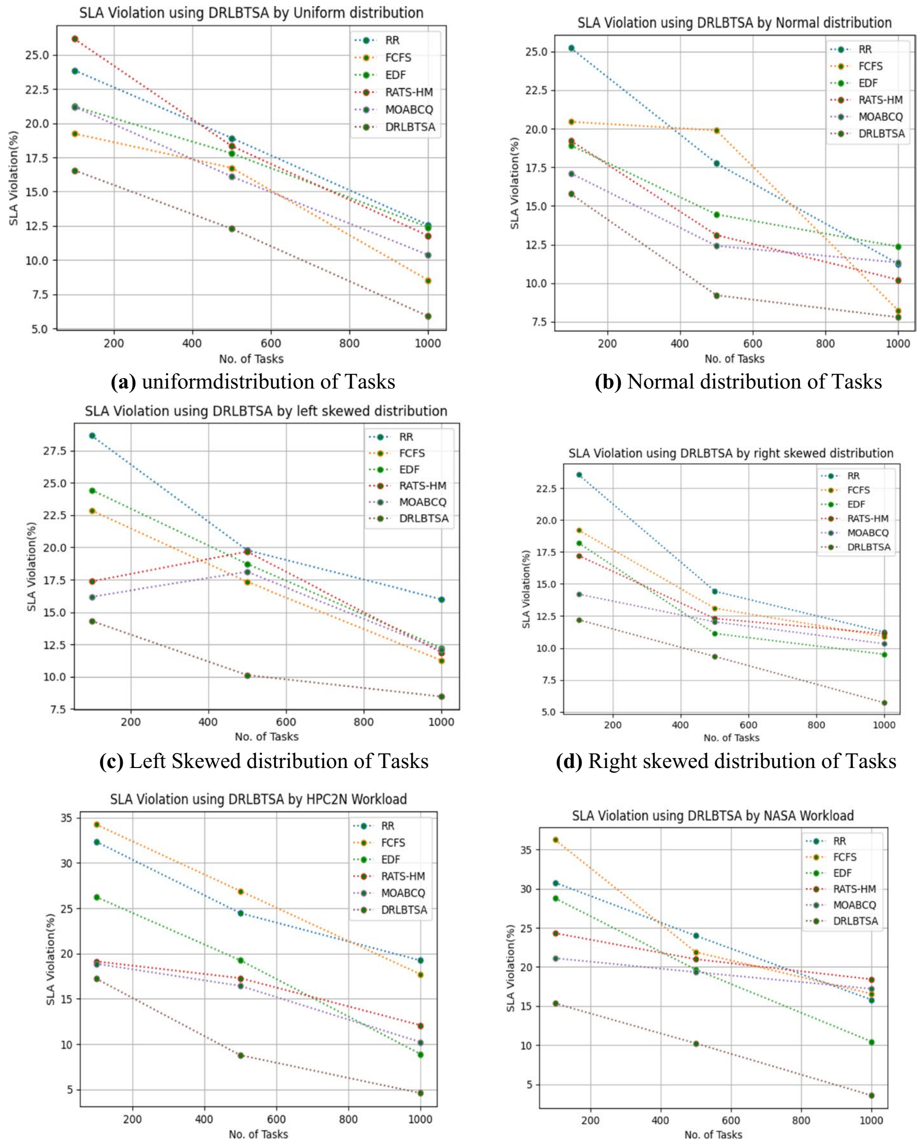
5.5 Results discussion

In this section, we discussed about results of evaluated parameters and their improvement over existing algorithms i.e. RR, FCFS, EDF, RATS-HM, MOABCQ. We have clearly mentioned about improvement of our DRLBTSA over existing baseline algorithms for parameters makespan, energy consumption, SLA Violation. The below Tables 8, 9 and 10 indicates improvement of makespan, energy consumption, SLA Violation.

Table 8 clearly represents our proposed DRLBTSA improves makespan over compared existing algorithms with different varying workloads.

Table 9 clearly represents our proposed DRLBTSA improves Energy Consumption over compared existing algorithms with different varying workloads.

Table 10 clearly represents our proposed DRLBTSA improves SLA Violation over compared existing algorithms with different varying workloads. In Tables 8, 9 and 10 improvement of results means it is minimization of parameters mentioned in our work.



(a) uniformdistribution of Tasks

(b) Normal distribution of Tasks

(c) Left Skewed distribution of Tasks

(d) Right skewed distribution of Tasks

(e) HPC2N Parallel worklogs

(f) NASA Parallel worklogs

Fig. 6 Calculation of Energy Consumption using DRLBTSA

6 Conclusion and future work

The scheduling of diverse workload over cloud paradigm is a challenge issue, due to dynamism and heterogeneity nature of cloud computing. It is very difficult to map tasks to precised VMs. Many existing authors proposed various scheduling mechanisms to map tasks to VMs but still there is a chance to do research in this area for mapping of tasks to appropriate VMs. The scheduling in cloud model is highly dynamic scenario

Table 8 Improvement of makespan over existing algorithms

Improvement of makespan for DRLBTSA over existing algorithms					
	RR	FCFS	EDF	RATS-HM	MOABCQ
d1	21.6%	36.5%	22.6%	24.37%	23.29%
d2	37%	45.5%	25.3%	38.5%	33.74%
d3	34%	43.1%	31.9%	36.5%	39.12%
d4	35%	39.8%	38.8%	42.8%	41.11%
d5	34.5%	43%	26.8%	30.48%	30.28%
d6	16.5%	38.3%	19%	31.14%	33.85%

Table 9 Improvement of Energy Consumption over existing algorithms

Improvement of Energy consunption for DRLBTSA over existing algorithms					
	RR	FCFS	EDF	RATS-HM	MOABCQ
d1	39.1%	39.5%	37.13%	37.09%	33.72%
d2	41.4%	42.8%	40.04%	38.37%	32.45%
d3	28.01%	44.8%	33.76%	33.15%	32.15%
d4	40.5%	53.2%	41.2%	41.31%	36.72%
d5	34.87%	46.7%	35.5%	39.27%	37.68%
d6	35.57%	32.2%	41.7%	41.92%	30.19%

Table 10 Improvement of SLA Violation over existing algorithms

Improvement of SLA Violation for DRLBTSA over existing algorithms					
	RR	FCFS	EDF	RATS-HM	MOABCQ
d1	37.2%	21.91%	32.4%	38.27%	27.01%
d2	39.5%	32.5%	28.3%	22.99%	19.76%
d3	48.9%	36%	40.6%	32.76%	29.06%
d4	44.66%	36.9%	29.88%	32.98%	25.57%
d5	59.74%	61.1%	43.73%	36.88%	32.76%
d6	58.7%	61%	50.5%	54.28%	49.4%

as many of miscellaneousworkload request the resources in multi-tenant environment to accomplished the demand based upon the processing capacities. To effectively map every task onto suitable VM, we have proposed DRLBTSA approach that find the optimal resources considering priority of taskswhile minimizing makespan, SLA Violation, Energy Consumption. We have used a machine-learning model i.e. DQN-model to solve task scheduling problem in our research. This DQN model is one of the variants of Deep Reinforcement learning. In this work, we have extensively done the simulations on cloudsim and input to algorithm is done through fabricated datasets different distributions, realtime parallel worklogs from HPC2N, NASA. We have evaluated our algorithm against existing baseline algorithms i.e. FCFS, RR, EDF, RATS-HM, MOABCQ. This simulation ran for 100 iterations. From results, it observed that our proposed approach i.e., DRLBTSA shown impact over baseline algorithms for above mentioned parameters. In future, we will test the efficacy of DRLBTSA by deploying in open stack and we want to generate realtime workloads in open stack environment and test the efficacy of our scheduler.

Data availability Authors not interested to disclose the availability of data.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abualigah L, Alkhrabsheh M (2022) Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing. J Supercomput 78(1):740–765
2. Adhikari M, Srirama SN, Amgoth T (2022) A comprehensive survey on nature-inspired algorithms and their applications in edge computing: Challenges and future directions. Softw Pract Exp 52(4):1004–1034
3. Agrawal K, Khetarpal P (2022) Computational intelligence in edge and cloud computing. J Inf Optim Sci 43:607–613
4. Amer DA et al (2022) Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing. J Supercomput 78(2):2793–2818
5. Bal PK et al (2022) A Joint Resource Allocation, Security with Efficient Task Scheduling in Cloud Computing Using Hybrid Machine Learning Techniques. Sensors 22(3):1242
6. Biswas D et al (n.d.) Optimized Round Robin Scheduling Algorithm Using Dynamic Time Quantum Approach in Cloud Computing Environment
7. Calheiros RN et al (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41(1):23–50
8. Cheng F et al (2022) Cost-aware job scheduling for cloud instances using deep reinforcement learning. Clust Comput 25(1):619–631
9. Ding D et al (2020) Q-learning based dynamic task scheduling for energy-efficient cloud computing. Futur Gener Comput Syst 108:361–371
10. Dong T et al (2020) Task scheduling based on deep reinforcement learning in a cloud manufacturing environment. Concurr Comput Pract Exp 32(11):e5654
11. Gazori P, Rahbari D, Nickray M (2020) Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach. Futur Gener Comput Syst 110:1098–1115
12. Ghafari R, HassaniKabutarkhani F, Mansouri N (2022) Task scheduling algorithms for energy optimization in cloud environment: a comprehensive review. Clust Comput 25:1035–1093
13. HPC2N: The HPC2N Seth log; 2016. http://www.cs.huji.ac.il/labs/parallel/workload/l_hpc2n/.0
14. Huang Y et al (2021) Deep adversarial imitation reinforcement learning for QoS-aware cloud job scheduling. IEEE Syst J 16:4232–4242
15. Karthiban K, Raj JS (2020) An efficient green computing fair resource allocation in cloud computing using modified deep reinforcement learning algorithm. Soft Comput 24(19):14933–14942
16. Kruekaew B, WarangkhanaKimpan. (2022) Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning. IEEE Access 10:17803–17818
17. Kumar R, Bhagwan J (2022) A comparative study of meta-heuristic-based task scheduling in cloud computing. In: Artificial Intelligence and Sustainable Computing. Springer, Singapore, pp 129–141
18. Lahande P, Kaveri P (2022) Implementing FCFS and SJF for finding the need of Reinforcement Learning in Cloud Environment. *ITM Web of Conferences*. Vol. 50. EDP Sciences
19. Li F, Bo H (2019) Deepjs: Job scheduling based on deep reinforcement learning in cloud data center. *Proceedings of the 2019 4th international conference on big data and computing*
20. Madni SHH et al (2019) Hybrid gradient descent cuckoo search (HGDCS) algorithm for resource scheduling in IaaS cloud computing environment. Clust Comput 22(1):301–334

21. Mohanapriya N et al (2018) Energy efficient workflow scheduling with virtual machine consolidation for green cloud computing. J Intell Fuzzy Syst 34(3):1561–1572
22. Nabi S et al (2022) AdPSO: adaptive PSO-based task scheduling approach for cloud computing. Sensors 22(3):920
23. NASA (n.d.): https://www.cse.huji.ac.il/labs/parallel/workload/l_nasa_ipsc/
24. Nayak SC et al (2022) An enhanced deadline constraint based task scheduling mechanism for cloud environment. J King Saud Univ Comput Inf Sci 34(2):282–294
25. Rjoub G, Bentahar J, Wahab OA (2020) BigTrustScheduling: Trust-aware big data task scheduling approach in cloud computing environments. Futur Gener Comput Syst 110:1079–1097
26. Rjoub G et al (2021) Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems. Concurr Comput Pract Exp 33(23):e5919
27. Sharma M, Garg R (2020) An artificial neural network based approach for energy efficient task scheduling in cloud data centers. Sustain Comput Inform Syst 26:100373
28. Sheng S et al (2021) Deep reinforcement learning-based task scheduling in iot edge computing. Sensors 21(5):1666
29. Siddesha K, Jayaramaiah GV, Singh C (2022) A novel deep reinforcement learning scheme for task scheduling in cloud computing. Clust Comput 25(6):4171–4188
30. Spano S et al (2019) An efficient hardware implementation of reinforcement learning: The q-learning algorithm. IEEE Access 7:186340–186351
31. Staddon JER (2020) The dynamics of behavior: Review of Sutton and Barto: Reinforcement learning: An introduction. J Exp Anal Behav 113(2):485–491
32. Swarup S, Shakshuki EM, Yasar A (2021) Task scheduling in cloud using deep reinforcement learning. Procedia Comput Sci 184:42–51
33. Tong Z et al (2020) QL-HEFT: a novel machine learning scheduling scheme base on cloud computing environment. Neural Comput & Applic 32(10):5553–5570
34. Tong Z et al (2020) A scheduling scheme in the cloud computing environment using deep Q-learning. Inf Sci 512:1170–1191
35. Wang Y et al (2019) Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning. IEEE Access 7:39974–39982
36. Wei Y et al (2018) DRL-scheduling: An intelligent QoS-aware job scheduling framework for applications in clouds. IEEE Access 6:55112–55125
37. Yan J et al (2022) Energy-aware systems for real-time job scheduling in cloud data centers: A deep reinforcement learning approach. Comput Electr Eng 99:107688
38. Zhang X et al (2019) Energy-aware virtual machine allocation for cloud with resource reservation. J Syst Softw 147:147–161
39. Zhou G, Tian W, Buyya R (2021) Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions. *arXiv preprint arXiv:2105.04086*

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Sudheer Mangalampalli[1] · Ganesh Reddy Karri[1] · Mohit Kumar[2] · Osama Ibrahim Khalaf[3] · Carlos Andres Tavera Romero[4] · GhaidaMuttashar Abdul Sahib[5]

✉ Sudheer Mangalampalli
sudheerkietmtech@gmail.com

[1] School of Computer Science and Engineering, VIT-AP University, Amaravati, AP, India

[2] Department of Information Technology, NIT Jalandhar, Jalandhar, India

[3] Al-NahrinNanorenewable Energy Research Center, Al-Nahrin University, Bhagdad, Iraq

[4] Universidad Santiago de Cali, Cali, Colombia

[5] Department of Computer Engineering, University of Technology, Bhagdad, Iraq