AI Enabled Resources Scheduling in Cloud Paradigm

Sudheer Mangalampalli, Ganesh Reddy Karri, and Prabha Selvaraj

Abstract Cloud Computing was evolved as one of the paradigm, which gives services to users in a utility-based manner. Services of cloud computing were extended to various fields and applications. Due to the enormous number of users, flexibility and easy to use nature of cloud paradigm, many of companies are trying to migrate towards cloud paradigm but from a cloud provider perspective it is a difficult to job to handle or schedule these heterogeneous workloads, which are coming onto cloud console. Therefore, it is important for a cloud provider to employ a task scheduling mechanism, which should be more proactive based on the nature of workloads coming onto cloud interface and how effectively they are scheduled onto suitable virtual resources. Many of existing scheduling algorithms used nature or bio inspired techniques to model schedulers as scheduling problem in cloud paradigm is a classical NP-Hard problem but still to make a schedule for a task onto a suitable VM based on its processing capacity while minimizing its makespan, energy consumption and other operational costs is still a tedious job as incoming user requests are highly dynamic in nature. In this paper, we have used a deep reinforcement learning technique i.e. DDQN model to make decisions of scheduling in cloud paradigm while checking incoming requests and underlying resources for every task. Task priorities are evaluated for all incoming tasks and prioritized tasks are fed to our scheduler and based on imposed conditions our scheduler will make decisions effectively. This entire research implemented on cloudsim. Extensive simulations are conducted by generating workload randomly and from realtime workload traces. Finally, our proposed scheduler is evaluated against existing baseline approaches i.e. Round Robin, FCFS, and Earliest Deadline first. From Simulation results, our proposed approach shown a huge impact over existing baseline approaches in terms of makespan, Energy consumption.

S. Mangalampalli (✉) · G. R. Karri · P. Selvaraj
School of Computer Science and Engineering, VIT-AP University, Amaravati, India
e-mail: sudheer.mangalampalli@vitap.ac.in; ganesh.reddy@vitap.ac.in; prabha.s@vitap.ac.in

M. Kumar et al. (eds.), *6G Enabled Fog Computing in IoT*,
https://doi.org/10.1007/978-3-031-30101-8_1

Keywords Task scheduling · Cloud computing · Artificial intelligence · Deep reinforcement learning · Double deep Q-network model · Round Robin · FCFS · Earliest dead line first · Makespan · Energy consumption

1 Introduction to Cloud Computing

Cloud Computing is a distributed model, which gives on demand ubiquitous services from virtual resources as a service needed for cloud consumers based upon service level agreements. With the advent of huge generation of data from various resources, there is a huge pressure on IT firms to maintain applications with commodity hardware and trying to adopt cloud environment [1]. In on premises environment, organizations follows cluster or grid computing approaches where cluster computing follows a centralized computing architecture [2] and Grid Computing follows either centralized or distributed computing approach [3] which renders their services to their corresponding customers. In Cluster and Grid Computing environments, computing resources are fixed and they cannot scale automatically as per the on demand requirements of users. Cloud Computing paradigm gives users scaling facility to increase or decrease virtual resources as per the requirements of users. This entire paradigm based on service oriented architecture [4] through which virtual resources will be given to all users as services as per SLA made between cloud user and provider. The main characteristics of cloud computing paradigm are mentioned below [5].

1.1 Characteristics of Cloud Computing

Resource Pooling It gives cloud users a virtual resource from pool of resources as per SLA of customer. This is an important characteristic in this paradigm as many of users will access virtual resources and these resources should be provisioned automatically from cloud provider from pool of resources running at cloud provider and allocation of resources to users should not affect other users while provisioning resources.

On Demand Service It provides self-control for cloud customer/user for the application running in cloud environment. He/She can provision or deprovision resources based on need of the application.

Ease of Maintenance This paradigm provides a huge flexibility to their users as they don't need to maintain their applications as they do in on premises environment. Therefore, users can focus on their development of application improvement and business objectives as cloud provider will take care of maintenance in terms of updates, patching etc.

Scalability This is a key characteristic in cloud paradigm as in now a days many of applications need to increase or decrease their resources instantly. To do so, we need a special paradigm, which adapts to environment and provision or deprovision resources according to situation. Coming to scalability it is of two types in cloud paradigm i.e. Horizontal and Vertical scaling. Horizontal scaling is to increase or decrease entire virtual machine if workload of the application cannot be handled by existing infrastructure. Vertical scaling is about to increase or decrease a specific resource.

No Upfront Investment It is economical as cloud user need not invest money on resources unlike in on premises environment. Zero upfront investment is needed in this paradigm, as users don't need to pay for their operational and administrational costs as in on premises environment.

Measured Service It is to be used for both cloud user and provider as for all services which were provisioned from cloud provider end need to compute pricing and from cloud user end it is to be used for them to know what are different services they have used and amount of consumption for corresponding services. This will be mainly useful for generating bill for cloud user automatically.

Security It is one of crucial characteristic of cloud paradigm as users are deploying their applications in third party environment not at their on premise environment. Therefore, cloud providers will follow high standards of security at their end and they will create a replica of each data point in the cloud environment as an end point through which data can be restored if any file gets corrupted or any crash happens in cloud environment. Cloud paradigm uses IAM i.e. Identity and access management service as their primary security service, which can give users a high standard of security at different levels, based on the need of the application.

Automation This is an essential characteristic for cloud paradigm as from around the globe many of users requests wide variety of resources according to their application. All these requests need to be fulfilled by the cloud provider as per the demand of cloud user according to SLA. Therefore, to handle these kind of heterogeneous and diversified requests from various users for different types of needs and provisioning those resources from cloud provider on demand needs automation. All provisioning and deprovisioning should be done automatically. If it should be done automatically, underlying policies need to be defined by cloud provider for different resources in the cloud environment.

Resiliency It is an important characteristic for cloud computing paradigm as if any data, file or computing server goes down or any crash happens it will be recovered quickly with little downtime unlike on premise environment.

Availability In this model, all virtual resources to cloud user will be highly available as many users accessing their resources on demand without any haseleness. To accommodate resources to all users seamless access to be provided by cloud provider and infrastructure to be resilient enough to handle requests from cloud

users. Cloud paradigm is having enough strength of resilient network and highly available and scalable resources as they are using virtualization.

Remote Access In this environment all resources to cloud users will be given as services on demand and these resources can be accessible from anywhere around the world. Therefore, it uses Internet to access resources in cloud paradigm. It allows users to work from anywhere to access their applications in cloud environment, which gives high availability, resilient network and seamless access to users.

Multitenant Architecture It is a primary characteristic in cloud paradigm through which, a resource is shared among several users as per SLA. When a single resource is shared among several users, there will be no conflicts among provision of resources as it is having an underlying software program managed by cloud provider and it is automated.

The below Fig. 1 represents characteristics of Cloud Computing.

1.2 Deployment Models of Cloud Computing

To adopt to cloud environment into on premises environment either they need to migrate their existing environment onto cloud computing infrastructure or they need to build their new applications on cloud environment. IT firms need to use certain deployment models for their application deployment in cloud environment. There are certain deployment models [6] named as Private, Public, Hybrid and Community cloud models. The below are primary deployment models of cloud paradigm which are represented in Fig. 2.

Public Cloud renders services to all cloud users who subscribed to services of a corresponding cloud provider on a paid basis. This deployment model can render services to all of its users around the world and users can access cloud services at any time around the clock in a seamless manner.

Private Cloud renders services to cloud users to a specific organization based on the subscription of services made with cloud provider. This deployment model helps cloud users to access their application with a secured channel and no other users out of that organization can access the services.

Hybrid Cloud It is a combination of both public and private cloud services rendered by a cloud provider to an organization in which some of the resources can be accessed publicly and some other resources have to be restricted to part of users in organization. Therefore, to handle this situation, organizations will subscribe to a model named as Hybrid cloud model, which gives resource access to users based on the restrictions mentioned by their organization.

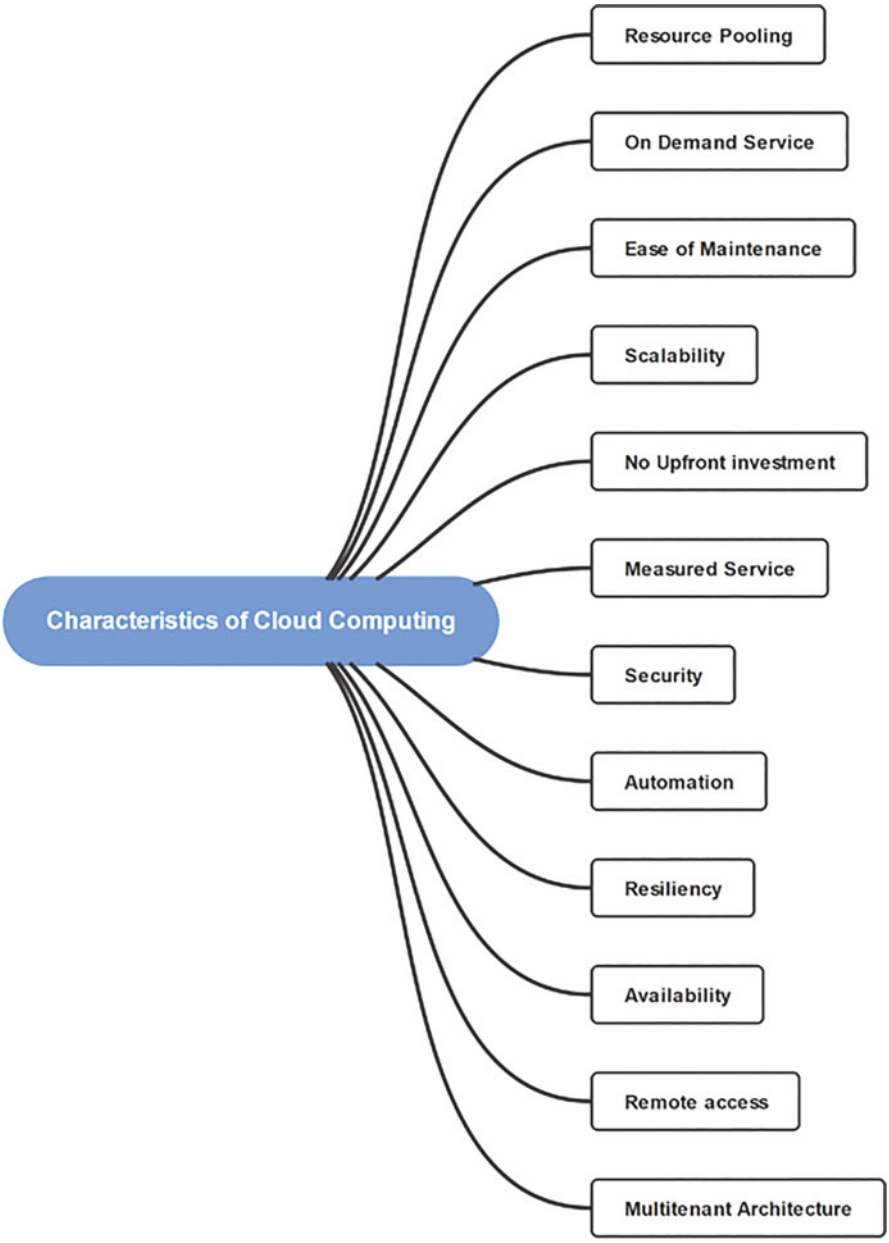

Fig. 1 Characteristics of Cloud Computing

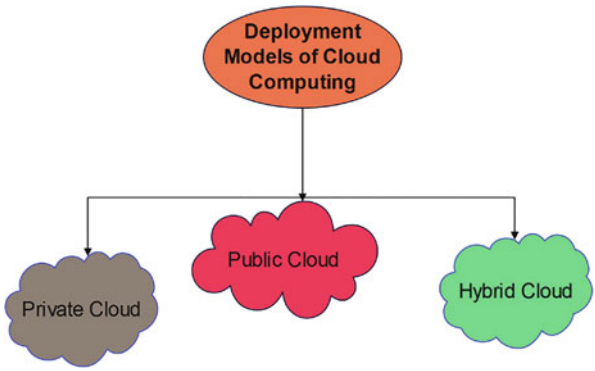

Fig. 2 Deployment models of Cloud Computing

1.3 Service Models of Cloud Computing

Cloud Computing paradigm provides resources through service models as this paradigm renders resources to users based on SLA between cloud users and providers. There are several services available in these days with different cloud providers.

Infrastructure-as-a-Service This service model provides virtual infrastructure to cloud user, which can replace hardware infrastructure in on premise environment. The advantage of this service is to scale infrastructure to an extent based on the requirement of users. Entire infrastructure of user application can be accessed via user interface.

The below Fig. 3 represents basic service models [7] in cloud computing.

Platform-as-a-Service This model is used to provide virtual development opportunities for cloud user based on SLA between cloud user and provider. This model can be helpful to users to develop cloud naïve applications. It provides virtual development platform for users by integrating it with different environments with use of RESTFUL API service. It provides support of various languages and APIs to develop cloud naïve applications.

Software-as-a-Service This service model renders service to cloud users for applications hosted in cloud environment and developed by cloud provider or applications hosted in cloud environment and developed by other users. This model is not used to develop applications but to render services to users for applications already hosted in cloud environment.

Main highlights of this chapter are presented below.

- Main objective of this research is to design a task-scheduling algorithm using DDQN model i.e. reinforcement learning based approach.
- Energy consumption, makespan are considered as parameters in this approach.
- Workload considered from fabricated workloads and another real world dataset i.e. BigDataBench [36].

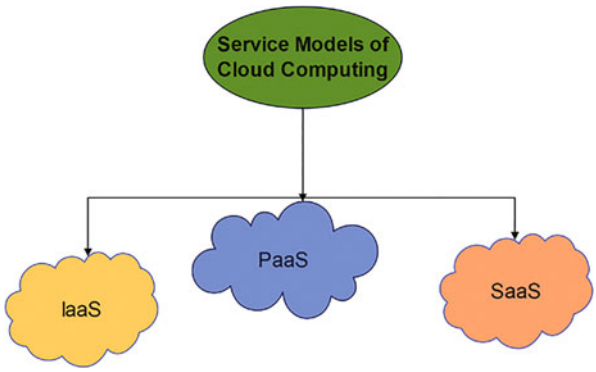

Fig. 3 Service models of Cloud Computing

2 Resource Scheduling in Cloud Computing

After discussion of basic service, deployment models we need to discuss about Provisioning of virtual resources to users by considering underlying resources in cloud infrastructure. This process is known as Resource scheduling [8]. In Cloud Computing, assignment of user requests to appropriate virtual resources is a highly dynamic scenario as many users are accessing cloud resources concurrently. It is a class of NP – hard problem and assignment of requests to virtual resources in cloud paradigm is a challenging issue as incoming requests coming onto cloud console varies with respect to time, size, processing capacity. Cloud paradigm needs an effective scheduler as provisioning and deprovisioning of resources to user requests is depends only on scheduler. Therefore, scheduler will impacts various parameters when assigning virtual resources to user requests. It will directly impacts the performance of cloud environment which impacts directly both cloud provider and users. Many of resource scheduling algorithms in cloud computing were modeled by nature inspired and metaheuristic algorithms. There are many existing algorithms i.e. PSO [9], GA [10], ACO [11], CSA [12], CSO [13] are used to model scheduling algorithms in cloud computing. There are many nature inspired and metaheuristic algorithms used to develop resource scheduling mechanisms in cloud paradigm but still it is a challenge in cloud paradigm to suitably map incoming tasks to appropriate virtual machines. Therefore, an artificial intelligence mechanism is needed to schedule incoming tasks to appropriate virtual resources. In this chapter, we used a DDQN model i.e. deep reinforcement learning technique to map incoming requests to VMs carefully based on incoming requests and checking underlying resources. Initially in this chapter, we carefully studied about various resource scheduling algorithms modeled by metaheuristic algorithms and their impact on cloud computing.

In the coming section, we mentioned various metaheuristic algorithms used to solve resource scheduling problems in cloud computing.

2.1 Resource Scheduling Algorithms Modeled by Metaheuristic Approaches in Cloud Computing

This section clearly discusses about various resource scheduling algorithms modeled by various metaheuristic approaches.

In [9], a scheduling framework designed by authors which focuses on minimization of task processing time. This algorithm modeled by using MPSO, which used over diversified population. It was simulated on Cloudsim. Main intention of this algorithm to minimize processing time of task with in deadline. It was evaluated against existing PSO, APSO, artificial bee colony, BAT, min-min algorithms. From simulation results it was proved that it was dominant over existing approaches for mentioned parameter. In [15], authors proposed a hybrid resource scheduling model was developed to address a single objective i.e. makespan. It was modeled by hybridization of whale algorithm by tuning parameters for both exploration, exploitation and to avoid premature convergence. It implemented on cloudsim. Workload taken from real world and synthetic datasets. It was compared over existing Whale optimization algorithm and finally evaluated makespan. From simulation results, it was proved that hybrid approach improves makespan over existing baseline algorithm for mentioned parameter. In [16], authors focused on development of a resource-scheduling model, which minimizes energy consumption, execution cost. Methodology used in this approach CSSA algorithm. Cloudsim was used as simulation tool. Real time and synthetic workloads given as input to algorithm. It compared over existing baseline models i.e. Hybrid GA-PSO, PSO-BAT, SSA algorithms. Results demonstrated that proposed model minimizes specified parameters. In [17], a resource scheduling model was developed to focus on parameters i.e. makespan, throughput, degree of imbalance. Hybrid Gradient was added to cuckoo search to solve resource-scheduling problem. Cloudsim was used as a simulation tool for experimentation. Real time work log traces from HPC2N [13], NASA [13] were used in simulation. It was compared over existing approaches i.e. ACO, CS. From results, it proved that HGDCS outperforms over existing algorithms for mentioned parameters. In [18], a resource-scheduling algorithm developed for vehicular cloud, which addressed makespan, energy consumption. It was mainly developed to schedule tasks properly onto vehicular clouds to avoid latency from centralized cloud architecture. It was modeled into three layers of scheduling and schedules on demand requests of road side users successfully based on usage of HAPSO by combining genetic, PSO algorithms. Sumo, NS2, MATLAB were used as simulation environments for resource scheduling. It was evaluated against PSO, GA algorithms. From simulation results it was greatly improved makespan, energy consumption by 34%, 32.5% respectively. In [19], task-scheduling mechanism was developed to focus on makespan. It works with crow search which is a nature inspired algorithm based on food habits of the crow. It was simulated on cloudsim. Heterogeneous random workloads were given as input to algorithm. It was evaluated over existing ACO, Min-Min algorithms. Simulation results revealed that existing works were outperformed by this approach

for mentioned parameter. In [20], authors formulated a task scheduling approach focused on parameters i.e. makespan, resource utilization, cost. LOA was used as methodology to solve scheduling problem. Cloudsim was used for entire simulation. Random generated workload used in simulation. It was evaluated over existing GA, PSO algorithms. Simulation results proved this approach was dominant over existing mechanisms for specified parameters. In [21], scheduling algorithm formulated to address computational cost, makespan, resource utilization, degree of imbalance. This mechanism modeled based on CSSA algorithm, which selects search space for optimization by using randomized inertia weights. It helps to converge swarm towards solution quickly. It implemented on cloudsim. It compared over GA, PSO, ABC approaches. It outperformed over existing algorithms for mentioned parameters. In [22], authors designed two scheduling algorithms i.e. LJFP, MCT based on existing PSO algorithm. It developed based on PSO in which modification was done at initialization of population done by these two-mentioned LJFP, MCT. It implemented on MATLAB. It was evaluated against baseline approach i.e. PSO. From results, it proved that these two approaches dominant over classical PSO in terms of execution time, energy consumption, degree of imbalance. In [23], a scheduling framework designed in two folds aimed at minimization of makespan, energy consumption. This approach based on scheduling tasks in compute clouds. Hybrid methodology used to solve task scheduling problem by combining GA, BFA. In first fold, this approach was addressed makespan. In second fold, it was addressed energy consumption. Entire simulations conducted on MATLAB. Metrics addressed in this approach evaluated with different workload heterogeneities. Initially simulation carried out with low heterogeneity workload and later it carried out with high heterogeneous and diversified workload. It was evaluated over existing GA, PSO, BFA algorithms. Simulation results revealed that it outperforms all existing approaches for specified parameters. In [24], SACO was developed to address how makespan, processing time of tasks effects scheduling in cloud computing. This algorithm uses diversification, reinforcement approach to avoid long path to capture their food shown by leader ants. Simulation carried out on Cloudsim. It was compared over variants of ACO. From results, it proved that SACO outperformed other variants for mentioned parameters. In [25], BMDA was proposed by authors to solve scheduling in cloud computing. Methodology used in this algorithm is a combination of BBO and dragon fly algorithms. BBO used as a technique to avoid premature convergence, which combined with dragon, fly algorithm to give optimal solutions. It implemented on Cloudsim. Workload given to this algorithm is from NASA [13] parallel work log archives and from CEC 2017 [26] benchmark functions. It compared over DA, PSO, BAT, GWO, RRO, Adaptive DA algorithms. From results, BMDA outperforms over existing algorithms for metrics i.e. response time, execution time, SLA violation. In [27], authors developed a task-scheduling algorithm focuses on minimization of makespan, maximization of resource utilization. EMVO developed by combining MVO, PSO algorithms that addresses local optimization problem in PSO. In EMVO, experiments conducted by using fixed and variable number of VMs. Entire experiments conducted on MATLAB. It evaluated over MVO, PSO algorithms and results revealed that it

shows huge impact over existing approaches for specified parameters. Authors in [28] proposed a task scheduling algorithm i.e. IE-ABC a hybrid metaheuristic approach addressed parameters i.e. Security, QoS. It was modeled by classical ABC approach which improved by adding a dedicated employee bee which keeps track of VM and datacenter status. Therefore, it is easy for a scheduler to look at VM and datacenter status to map its tasks easily and precisely. Simulations conducted on cloudsim. It was compared against classical ABC algorithm with respect to makespan, cost, Number of tasks migrated. Finally, from simulation results, there is a huge impact over existing ABC for specified parameters. In [29], scheduling algorithm formulated to schedule tasks onto virtual machines. CRO and ACO combined to solve scheduling problem. It implemented on Cloudsim. Random workload and Amazon EC2 instances workload given input to algorithm. It evaluated against CRO, ACO, PSO, CEGA algorithms and results revealed that it shows improvement in makespan, cost. In [30], FA-SA algorithm proposed by authors to introduce a new local search to optimize solution. This algorithm initializes a new population strategy to converge towards near optimal solutions. Cloudsim. used for simulation. Workload given to algorithm from real time datasets and synthetic workloads. It compared against existing firefly, SA, min-min, max-min algorithms. Results revealed that it shown huge impact over existing approaches for specified parameters makespan for different workloads with different datasets. In [31], a hybrid algorithm proposed by authors to schedule tasks effectively by addressing energy consumption, SLA violation. Methodology used in this algorithm is BMW-TOPSIS to map tasks to VMs. Entire simulations conducted on Cloudsim. It compared over existing BMW, TOPSIS algorithms and performed ANOVA test to evaluate statistics from results. From simulation results, it outperforms existing approaches for energy consumption, makespan, resource utilization.

Table 1, it clearly shown that many of metaheuristic algorithms addressed baseline parameters but scheduling in cloud computing environment is highly dynamic and to map tasks effectively and these metaheuristic approaches still facing challenges to get optimal solutions in terms of metrics addressed in cloud environment. Therefore, it is necessary to employ a machine-learning model in scheduling architecture through which decision need to be taken for mapping of requests with resources.

In the next section, we mentioned various ML based scheduling algorithms to solve resource scheduling problems in cloud computing.

2.2 Resource Scheduling Algorithms Modeled by Metaheuristic Approaches in Cloud Computing

This section clearly discusses about various resource scheduling algorithms in cloud computing modeled with various machine-learning techniques.

Table 1 Summary of metaheuristic resource scheduling algorithms in Cloud Computing

References	Methodology	Objectives of resource scheduling algorithms modeled by metaheuristic approaches
[9]	MPSO	Task processing time
[10]	Improved GA	Execution time
[11]	MORA-ACS	Energy consumption, load balancing
[12]	MOCSO	Completion time, execution cost
[13]	CSO	Energy consumption, makespan, total power cost, migration time
[15]	Hybrid Whale	Makespan
[16]	CSSA	Energy consumption, execution cost
[17]	HGDCS	Makespan, throughput, degree of imbalance
[18]	HAPSO	Makespan, energy consumption
[19]	Crow Search	Makespan
[20]	LOA	Makespan, resource utilization, cost
[21]	CSSA	Computational cost, makespan, resource utilization, degree of imbalance
[22]	LJFP, MCT	Execution time, energy consumption, degree of imbalance
[23]	GA-BFA	Makespan, energy consumption
[24]	SACO	Makespan, processing time of tasks
[25]	BMDA	Response time, execution time, SLA violation
[27]	EMVO	Makespan, resource utilization
[28]	IE-ABC	Security, QoS
[29]	CR-ACO	Makespan, cost
[30]	FA-SA	Makespan
[31]	BMW-TOPSIS	Energy consumption, makespan, resource utilization

In [32], authors proposed an automation approach for scheduling workloads in cloud paradigm. Initially authors used three ML models to develop this scheduling algorithm i.e. RL, DQN, RNN-LSTM, DRL-LSTM. From all these approaches, DRL-LSTM works well in minimization of CPU usage cost, Memory usage cost. It was implemented using Pytorch framework. It evaluated against existing RR, SJF, IPSO algorithms. From results, DRL-LSTM shows a huge improvement in minimization of CPU usage cost 67% to SJF, 35% to RR, IPSO respectively and memory usage cost minimized by 72% for SJF, 65% for RR, 31.25% for IPSO approaches. In [33], a scheduling model designed by authors which uses deep reinforcement learning approach to effectively schedule tasks coming onto cloud console to Cloud nodes or edge nodes. This scheduling process follows precedence constraints in their tasks, which are incoming to cloud console. It gives a clear distinct mechanism to identify which tasks need to be scheduled to a VM or edge nodes at deployment locations or cache locations of applications. This approach implemented using cloudsim. It compared over several baseline approaches and identified that this approach minimizes 56% of energy consumption, 46% of execution time compared with baseline approaches. In [34], authors focused on

development of a deadline aware scheduling model in fog cloud environment to deal with delicate time sensitive applications. These time sensitive and on demand requirement applications, found more often in IOT environment which may deal smart city applications. These applications changes their behavior according to time and heterogeneity of tasks are also is an important aspect in dealing these kind of applications. Therefore, authors come up with hybridizing MTOA with DQN machine learning model to solve scheduling problem. iFogsim used as a simulation tool for this entire experimentation. It compared over CAG, DNGSA, policy learning approaches. From results, it proved that MTOA-DQN approach shows huge impact over existing policies for makespan, energy consumption. In [35], authors developed a scheduling mechanism, which works with spark jobs in their customized clusters. They developed this customized cluster to check the behavior of spark jobs running in the cluster while maintain SLA objectives. They used DRL based mechanisms for scheduling and workload used by them were real-time AWS instances according to pricing models in Australia. They have used another workload from BigDataBench [36] which consists of heterogeneous jobs i.e. IO sensitive, Network sensitive, Computational sensitive. This entire experimentation conducted on AWS cloud. They Compared this work with existing algorithms i.e. RR, RRC, FF, ILP mechanisms. From experimental results, it proved that DRL based mechanism gain success in minimization of VM cost by 30%. In [37], a computational sensitive based scheduler formulated by authors to effectively schedule tasks among VMs with the use of multi tenancy. A RL based technique used to effectively map tasks to VMs. Simulations carried out on green cloud simulator and evaluated against existing RR, FCFS approaches. From simulation results, it proved that it outperforms existing approaches by minimizing operational costs and maximizing resource utilization. In [38], authors formulated a scheduling algorithm based on RL focuses on improvement of system performance. This algorithm takes heterogeneous requests as input and fed to RL based scheduler to make a decision to schedule tasks in cloud computing. This entire experimentation carried out on Cloudsim. It evaluated against existing algorithms i.e. RR, Max, FIFO, Greedy, Q-Scheduling algorithms. From Simulation results, response time greatly minimized over existing algorithms by 49%, 46%, 44%, 43%, 38% respectively for above mentioned existing algorithms. In [39], authors focused on development of a green fair scheduler in cloud computing which minimizes energy consumption in datacenters. This algorithm uses a DL approach to schedule tasks in this complex system. Simulation carried out on cloudsim. It evaluated against existing conventional migration approach with variable request sizes ranging from 50 to 500 and identified that energy consumption greatly minimized over existing approaches. In [40], authors formulated a scheduling mechanism, which used in edge computing environment. Edge computing suffers from high task failure rate, high service time, high mobility of devices. DRL used as methodology in this algorithm. Edge Cloudsim [41] used as simulation tool for experimentation. It evaluated against existing DQN, PPO approaches. Simulation results revealed that it greatly minimizes service time, task failure rate over DQN, PPO for various heterogeneous workloads. In [42], a multi workflow scheduling mechanism developed for IaaS

clouds to minimize makespan, cost. Multi agent DQN model used for developing this approach, which takes input as multiple workflows with variable number of VMs. Experimentation carried out on real time AWS environment. It compared over existing NSGA-II, MPSO, GTBGA approaches. From simulation results, it proved that multi agent DQN model which takes scheduling decisions based on no prior knowledge outperforms existing algorithms for specified parameters. In [43], Reliability taken as primary objective for design of scheduler in cloud environment. Authors identified a multi agent approach, which takes your task to global queue, and then it will schedule based on buffer capacity and consumed resource usage. For learning purpose, this algorithm uses neural network and it combined with RL approach thereby achieving rewards based on metrics addressed by authors. It implemented on customized simulation environment and it compared against greedy, FIFS, Random approaches. Simulation results shown that makespan minimized to great extent while success rate of tasks, VM utilization rate increased to a good extent. In [44], a dynamic scheduler for cloud environment designed based on Sched RL. This approach transforms existing multi-NUMA scheduler used in existing approaches. Sched RL used 1500 epochs to run entire simulation and gives delta rewards for corresponding parameters i.e. allocation rate, fulfill number of tasks. Authors also mentioned that Sched RL have two limitations i.e. scalability, generalization. It implemented on a real time Azure cloud environment with variable workloads. It compared over First fit, best fit heuristics, and from results, it proved that proposed approach shown huge impact over existing approaches for mentioned parameters. In [45], workflow scheduling formulated for multiple workflows, which designed for prioritizing tasks based on its type and quality of service need to be delivered to customer. This algorithm mainly deals with task ordering into execution mode based on their priorities to get load balance among all nodes. To achieve their goal, authors used RL model, which takes decisions, based on input of tasks, type of tasks and priorities. Simulation carried out on Cloudsim and it compared over Q- learning, Random, Mixed scheduling techniques. Results revealed that RL model outperforms existing approaches by minimizing SLA Violations and maximizing resource utilization. In [46], authors proposed an energy efficient VM scheduling technique, which minimizes energy consumption, SLA violations while maintaining QoS. This work mainly focuses on extracting QoS information from datacenters by making it to learn by using DRL model. It compared with different existing resource allocation mechanisms and extensive simulations conducted on Cloudsim. From simulation results, it shown a huge impact over existing allocation mechanisms for above-mentioned parameters. In [47], a cost based scheduling algorithm formulated by authors to schedule VM instances in Cloud Computing. DRL used as methodology to schedule instances in an effective way. Cloudsim used as simulation tool for simulation. It compared over existing algorithms i.e. Random, RR, Earliest approaches. From simulation results it proved that it shown huge impact over existing algorithms for parameters i.e. Response time, cost, success rate of tasks.

Table 2, clearly shown that many of ML approaches used for resource scheduling algorithms addressed baseline parameters but scheduling in cloud computing

Table 2 Summary of ML approaches for resource scheduling algorithms in Cloud Computing

References	Methodology	Objectives of resource scheduling algorithms modeled by ML approaches
[32]	DRL-LSTM	CPU usage cost, memory usage cost
[33]	DRL	Execution time, energy consumption
[34]	MTOA-DQN	Makespan, energy consumption
[35]	DRL	VM cost
[37]	RL	Operational costs, resource utilization
[38]	RL	Response time
[39]	DL	Energy consumption
[40]	DRL	Service time, task failure rate
[42]	DQN	Makespan, cost
[43]	RL	Makespan, VM utilization rate
[44]	SchedRL	Allocation rate, fulfill number of tasks
[45]	RL	SLA violation, resource utilization
[46]	DRL	Energy consumption, SLA violation, QoS
[47]	DRL	Response time, cost, success rate

environment is highly dynamic and more over that many of researchers used RL and DRL approaches to address problems in resource scheduling. To make decisions more accurate and precisely with heterogeneous workloads. In this chapter, we employed a Deep reinforcement learning approach i.e. DDQN model which is based on RL.

From the extensive literature reviewed in Sect. 2, we identified that many of existing scheduling algorithms formulated based on nature-inspired approaches, which schedules tasks with near optimal solutions. Therefore still there is a challenge exists for researchers to map upcoming dynamic workloads onto suitable virtual resources. Therefore a prominent scheduling approach is needed which should dynamically behaves and allocate requests based on upcoming workloads by considering underlying virtual resources. Therefore, we thought that a machine learning mechanism need to be employed which should consider upcoming workloads and considering underlying resources, which also need to minimize energy consumption, makespan.

3 Double Deep Q-Networks

When using DQN, there is a chance that the Q values will be overestimated, which can result in the underutilization of resources, an increase in the makespan, and the need to wait for the tasks. We use double deep Q networks (DDQN) to get around the problems with the DQN when it comes to scheduling tasks in cloud-based environments.

To begin, we divide the available resources into three distinct categories. The first one is the bandwidth of the network in relation to the links that are established between the switches and routers. Second, the processing power of VMs in terms of CPU and Memory. The third issue is the accessibility of the data in relation to its storage when it is spread out across multiple locations. An environment that contains all three of these types of resources is known as a reinforcement learning environment. We consider Q1 and Q2 to be the queuing models that define the agents, with the Q1 agent being determined by the resources needed to carry out the tasks in the queue and the Q2 agent being determined by the resources that are readily available in the datacenter. The random weights for Q1 and Q2 are the first things that we look at. These values are updated as Algorithm 2 performs its processing of the input. In the system we suggested, we started by setting up the cloud environment and giving each agent its starting weights, as described in Algorithm 1.

Algorithm 1: Configuring Cloud Environment and Setting Up Agent

```
  Input: CPU resources, Memory resources network resources,
                storage resources
      for i=1 to l   //where l is the number of CPU and Memory
                resources of VMs in the cloud
           T^i_cm= T^i_c + T^i_m + T^(i-1)_cm
       for i=1 to n   //where n number of available network
links to VMs in the cloud
              L^i_b=  L^i_b +L^(i-1)_b
           for i=1 to m   //where m number storage components
associated with the VMs in the cloud
              St^i_v=  St^i_v +St^(i-1)_v
        //Creating  Q1 agent
        for i=1 to t_k   //where t_k is the number of tasks
on queue
                  t_w  = w_i    //initial task weights to
random weights w_i
       //Creating  Q2 agent
        for j=1 to r_k    //where t_k is the number of
available VMs on queue
                  t_w  = w_j     //initial  available VM
resource weights to random weights r_w
```

After setting up all resource configurations and Q1 and Q2 agents, Algorithm 2 starts executing.

Algorithm 2: DDQL Task Scheduling

```
  Input: total number of network resources,  vm resources, and
storage rescues, Q1, Q2 agents
   Output: updating the Q1, Q2 agents rewards, select action
agent
   for i=1 to a_j    //where a_j number of agents
      task_rewards=0
          for j=0 to  n     //where n number of tasks
               Q1(s,ETET_j)= t_end_time- t_start_time   //where
estimated task execution time
```

```
                Q2(s,AVMR_i)= val(T^i_cm,, L^i_b, St^i_v)
          If UPDATE(ETET_j) then
                Define a_j = arg_max_a Q1(s',a) // where a is ETET_j
                Q1(s,a)←Q1(s,a) + α (s,a)(r+ γ Q2(s',a_j)-Q1(s,a))
        else If UPDATE(AVMR_j) then      //where AVMR_i
is available VM resources
                Define b_j = arg_max_b Q1(s',b) // where a is AVMR_i
                Q2(s,b)←Q2(s,b) + α (s,b)(r+ γ Q1(s',b_j)-Q2(s,b))
          S←s'
```

The initial implementation of the Double Q-learning algorithm makes use of two independent estimates, which are denoted by Q1 and Q2. We use estimate Q1 to determine the action that will maximize profit when the probability is 0.5, but we also use it to update Q1. In Q1 calculations we consider the estimated task execution time, will update the reward of task in Q1, similarly for Q2 calculations we consider the available VM resources, based on the resource utilization of VMs the VM reward are updated. If any update in Q1 or Q2 the state parameters will be updated(s').

By carrying out this procedure, we are able to obtain an unbiased estimator Q1(state, argmax Qnext state, action) for the expected value of Q and inhibit bias.

In our approach we use the present best for selection of the action agent.

$$Q^* (s_t, a_t) = \mathbb{E}_s{}' \left[R_{t+1} \gamma \max_a Q^* \left(s't, a'_t \right) | s, a \right] \quad (1)$$

With $\mathbb{E}_s$ equal to the Q-value of the state-action pair plus the learning rate. The algorithm's reliability on the objective reward value is a function of its learning rate. A discount factor, regulates the relative value of present and future benefits. From Algorithm 2, the time required to select the best agent to execute the task on available resources O(m*n), where m is number of agents and n in number of tasks in cloud environment.

4 Simulation and Results

This section clearly discusses about entire simulation and results of our work. Our simulation carried out on cloudsim [14] simulator which is a discrete event simulator written in Java. It was developed at university of Melbourne. It simulates cloud environment with different policies mentioned by developers. Users can customize and add their policies to evaluate different parameters over this simulator. Therefore, we have chosen this simulator to implement our scheduling model. In this work, simulation carried out by using two different types of workloads. Initially we have fabricated our datasets with different workload distributions and given as input to algorithm and later we are enthusiastic to evaluate efficiency of our approach using a heterogeneous workload real time benchmark dataset mentioned in [36] i.e. BigdataBench which consists of different types of tasks with different heterogeneities. Fabrication of dataset done in 4 types i.e. Uniform distribution-

Table 3 Configuration settings for simulation

Name of the entity	Quantity
No. of tasks	100–1000
Length of tasks	100,000
Computational capacity of physical host	32 GB
Storage capacity of physical host	5 TB
Network bandwidth	1500 mbps
No. of VMs	40
Computational capacity of a VM	2GB
Network bandwidth of a VM	150 mbps
Hypervisor	Xen
Operating system	Linux
No. of Datacenters	8

which consists all types of equally distributed tasks. Normal distribution-which consists of more medium distribution of tasks and less number of small, large tasks. Left Skewed distribution-which consists of more small tasks and less large tasks. Right Skewed distribution-which consists of more large tasks and less small tasks. All these distributions represented as r1, r2, r3, r4 respectively. After fabrication of these dataset distributions. BigdataBench [36] represented as r5. Our proposed DDQN model evaluated against existing RR [48], FCFS [49], EDF [50] algorithms. Table 3 represents configuration settings for our simulation.

5 Evaluation of Makespan

In this section, we clearly presents evaluation of makespan, as it is a primary influential parameter for cloud computing paradigm. This parameter evaluated using above configuration settings mentioned in Table 3. We have given r1, r2, r3, r4, r5 workloads as input to algorithm as mentioned above with different distributions. We evaluated DDQN approach against existing RR, FCFS, EDF algorithms. DDQN run for 50 iterations. Table 4 represents evaluation of makespan for 100 to 1000 tasks. For considered workload r1, when DDQN used makespan generated for 100, 500, 1000 tasks 587.34, 624.99, 1458.37 respectively. For considered workload r2, when DDQN used makespan generated for 100, 500, 1000 tasks 512.89, 945.89, 1034.36 respectively. For considered workload r3, when DDQN used makespan generated for 100, 500, 1000 tasks 543.92, 987.23, 1327.9 respectively. For considered workload r4, when DDQN used makespan generated for 100, 500, 1000 tasks 412.89, 523.78, 866.24 respectively. For considered workload r5, when DDQN used makespan generated for 100, 500, 1000 tasks 769.35, 1023.56, 1356.78 respectively.

Table 4 represents evaluation of makespan of DDQN algorithm over existing algorithms i.e. RR, FCFS, EDF respectively by using fabricated dataset distributions and a real-time benchmark dataset used to test makespan of our approach. From

Table 4 Evaluation of makespan

Algorithms				
No. of tasks	RR	FCFS	EDF	DDQN
r1				
100	793.98	654.76	773.78	587.34
500	967.45	1146.98	876.29	624.99
1000	1562.89	2376.98	2178.34	1458.37
r2				
100	905.76	856.32	798.88	512.89
500	1124.78	1267.90	1078.45	945.89
1000	1892.67	2035.78	1224.23	1034.36
r3				
100	867.23	747.89	623.87	543.92
500	1227.98	1367.65	1164.24	987.23
1000	1562.79	1923.65	1743.87	1327.9
r4				
100	785.73	689.99	534.7	412.89
500	1124.99	1038.78	865.45	523.78
1000	1323.92	1534.78	1425.79	866.24
r5				
100	1867.56	1745.34	1265.23	769.35
500	2034.78	1672.23	1429.9	1023.56
1000	2989.21	3126.78	2578.89	1356.78

Table 4, it is evident that for all distributions and benchmark dataset makespan of DDQN is greatly minimized over existing approaches.

The above Fig. 4 represents evaluation of makespan using DDQN over existing approaches i.e. RR, FCFS, EDF by using various distribution of workloads and real time benchmark dataset i.e. r1, r2, r3, r4, r5 respectively. It is evident that in all the cases our evaluated makespan is outperformed over existing approaches as mentioned in above Fig. 4.

5.1 Evaluation of Energy Consumption

In this section, we clearly presents evaluation of energy consumption, as it is an important parameter for both cloud provider and user. This parameter evaluated using above configuration settings mentioned in Table 3. We have given r1, r2, r3, r4, r5 workloads as input to algorithm as mentioned above with different distributions. We evaluated our proposed DDQN approach against existing RR, FCFS, EDF algorithms. Proposed DDQN run for 50 iterations. For considered workload r1, when DDQN used Energy Consumption generated for 100, 500, 1000 tasks 34.67, 51.45, 89.27 respectively. For considered workload r2, when DDQN used Energy Consumption generated for 100, 500, 1000 tasks 43.24, 59.23, 78.45 respectively.

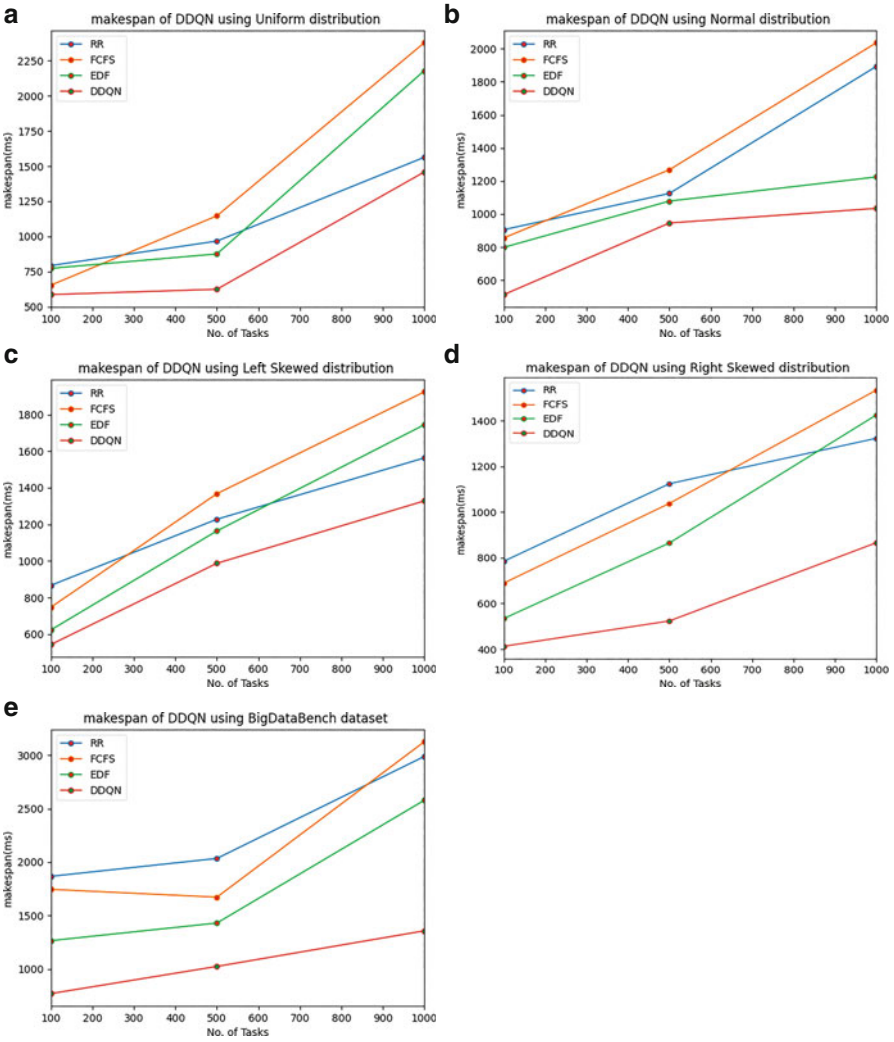

Fig. 4 Evaluation of makespan using DDQN (**a**) Uniform Distribution of Tasks. (**b**) Normal Distribution of Tasks. (**c**) Left Skewed Distribution of Tasks. (**d**) Right Skewed Distribution of Tasks. (**e**) BigDataBench worklogs

For considered workload r3, when DDQN used Energy Consumption generated for 100, 500, 1000 tasks 49.23, 56.45, 90.35 respectively. For considered workload r4, when DDQN used Energy Consumption generated for 100, 500, 1000 tasks 38.26, 45.78, 75.38 respectively. For considered workload r5, when DDQN used Energy Consumption generated for 100, 500, 1000 tasks 74.29, 81.56, 90.22 respectively. Table 5 represents evaluation of energy consumption for 100 to 1000 tasks.

Table 5 Evaluation of Energy Consumption

Algorithms				
No. of tasks	RR	FCFS	EDF	DDQN
r1				
100	88.99	94.35	72.86	34.67
500	100.36	106.88	92.45	51.45
1000	143.25	123.99	108.76	89.27
r2				
100	92.67	103.56	84.34	43.24
500	104.65	89.46	92.22	59.23
1000	124.24	135.89	120.56	78.45
r3				
100	87.57	91.45	79.23	49.23
500	93.99	100.56	94.67	56.45
1000	114.2	124.78	105.22	90.35
r4				
100	89.34	92.11	87.56	38.26
500	73.78	98.21	99.14	45.78
1000	103.24	121.67	114.67	75.38
r5				
100	108.56	94.67	98.22	74.29
500	124.79	114.56	106.89	81.56
1000	137.35	121.78	114.26	90.22

Table 5 represents evaluation of energy consumption of DDQN algorithm over existing algorithms i.e. RR, FCFS, EDF respectively by using fabricated dataset distributions and a real-time benchmark dataset used to test energy consumption of our approach. From Table 4, it is evident that for all distributions and benchmark dataset energy consumption of DDQN is greatly minimized over existing approaches.

The above Fig. 5 represents evaluation of Energy Consumption using DDQN over existing approaches i.e. RR, FCFS, EDF by using various distribution of workloads and real time benchmark dataset i.e. r1, r2, r3, r4, r5 respectively. It is evident that in all the cases our evaluated energy consumption is outperformed over existing approaches.

6 Conclusions and Future Research Directions

Resource scheduling in cloud computing paradigm is a huge challenge because incoming requests onto cloud console varies in terms of processing capacities. Therefore scheduling these wide varieties of requests onto virtual resources in cloud is a challenge for cloud provider. Improper mapping of requests to virtual resources leads to decay in system performance i.e. increase in makespan and consumption of energy can be increased which affects both cloud user and provider. Existing

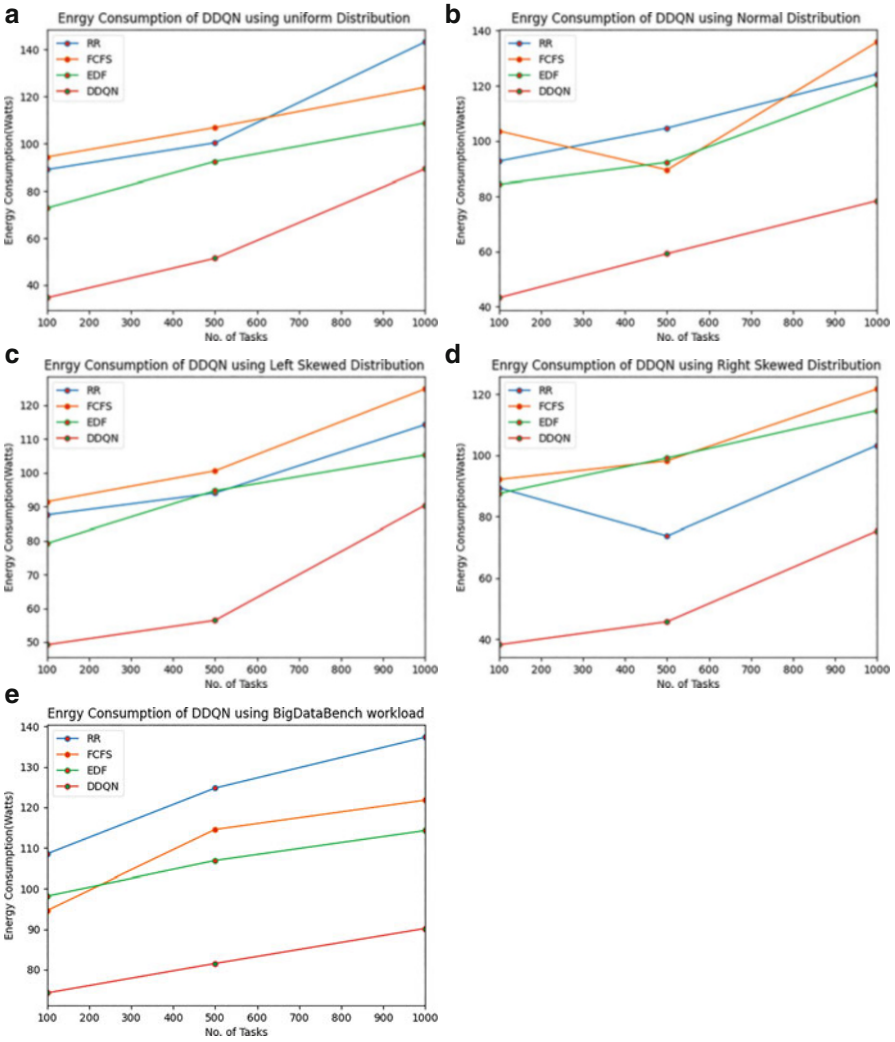

Fig. 5 Evaluation of Energy Consumption using DDQN. (**a**) Uniform Distribution of Tasks. (**b**) Normal Distribution of Tasks. (**c**) Left Skewed Distribution of Tasks. (**d**) Right Skewed Distribution of Tasks. (**e**) BigDataBench worklogs

authors used metaheuristic approaches to design schedulers and solve scheduling problems by taking it to near optimal solutions but cloud paradigm is dynamic in terms of requests heterogeneity and to make appropriate decision making out of incoming requests onto virtual resources. In this chapter, we have used DDQN model, which is a reinforcement learning approach fed to the scheduler module helps to take decisions considers task priorities and underlying resource capacity. Entire simulations carried out on cloudsim. Workload for algorithm considered from

a real-time benchmark dataset and datasets fabricated using different distributions. Our proposed approach compared over existing RR, FCFS, EDF algorithms and simulation results revealed that proposed approach using DDQN model shown great impact in makespan, Energy Consumption.

7 Future Research Directions

To manage dynamic workloads, SLA guarantee services, and resource opinions, AI-based algorithms are crucial. However, the various cloud scheduling algorithms, including AI-based algorithms, are constrained by limited(multi-objective) parameters such as task execution time, available resources, etc., to schedule the tasks, which makes the existing solutions considered near-optimal solutions. Future AI algorithms must take into account the following factors in order to achieve optimal solutions in a cloud environment.

Although virtual machines are capable of handling a variety of workloads, the current scheduling algorithms require users to deploy their virtual machines in order to run a single application. If users run many apps, these existing methods deliver average performance. Thus, dynamic workloads on each VM must be taken into account by future AI-based algorithms.

Although virtual and physical resource clustering improves QoS services, the current VM and physical clusters remain essentially static until an auto-scaling event occurs, with overutilization and underutilization of resources as a result. In order to overcome this, AI algorithms must take into account the dynamic clusters in cloud environments, where VMs must cooperate with one another.

Study the static workloads in the cloud that can provide better performance using static schedulers because AI solutions in the cloud environment do not come with free computing and storage.

The risk associated with cloud scheduling algorithms grows with the use of AI-based algorithms. Attackers in this case create dynamic traffic using bots, which can bypass the cloud security measures and result in denial of service attacks. To lower risk in the cloud, AI scheduling algorithms need to be able to find malicious payloads ahead of time.

References

1. Low, C., Chen, Y., & Mingchang, W. (2011). Understanding the determinants of cloud computing adoption. *Industrial Management & Data Systems, 111*(1006).
2. Khallouli, W., & Huang, J. (2021). Cluster resource scheduling in cloud computing: Literature review and research challenges. *The Journal of Supercomputing, 78*, 1–46.
3. Pires, A., Simão, J., & Veiga, L. (2021). Distributed and decentralized orchestration of containers on edge clouds. *Journal of Grid Computing, 19*(3), 1–20.

4. Hustad, E., & Olsen, D. H. (2021). Creating a sustainable digital infrastructure: The role of service-oriented architecture. *Procedia Computer Science, 181*, 597–604.
5. Rashid, A., & Chaturvedi, A. (2019). Cloud computing characteristics and services: A brief review. *International Journal of Computer Sciences and Engineering, 7*(2), 421–426.
6. Diaby, T., & Rad, B. B. (2017). Cloud computing: A review of the concepts and deployment models. *International Journal of Information Technology and Computer Science, 9*(6), 50–58.
7. Tadapaneni, N.R. (2017). *Different types of cloud service models*.
8. Singh, S., & Chana, I. (2016). A survey on resource scheduling in cloud computing: Issues and challenges. *Journal of grid computing, 14*(2), 217–264.
9. Kumar, M., & Sharma, S. C. (2020). PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing. *Neural Computing and Applications, 32*(16), 12103–12126.
10. Ma, J., et al. (2016). A novel dynamic task scheduling algorithm based on improved genetic algorithm in cloud computing. In *Wireless communications, networking and applications* (pp. 829–835). Springer.
11. Pham, N. M., & Nhut, and Van Son Le. (2017). Applying Ant Colony System algorithm in multi-objective resource allocation for virtual services. *Journal of Information and Telecommunication, 1*(4), 319–333.
12. Madni, S. H. H., et al. (2019). Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds. *Arabian Journal for Science and Engineering, 44*(4), 3585–3602.
13. Mangalampalli, S., Swain, S. K., & Mangalampalli, V. K. (2022). Multi objective task scheduling in cloud computing using cat swarm optimization algorithm. *Arabian Journal for Science and Engineering, 47*(2), 1821–1830.
14. Calheiros, R. N., et al. (2011). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience, 41*(1), 23–50.
15. Strumberger, I., et al. (2019). Resource scheduling in cloud computing based on a hybridized whale optimization algorithm. *Applied Sciences, 9*(22), 4893.
16. Sanaj, M. S., Joe, P. M., & Prathap. (2020). Nature inspired chaotic squirrel search algorithm (CSSA) for multi objective task scheduling in an IAAS cloud computing atmosphere. *Engineering Science and Technology, an International Journal, 23*(4), 891–902.
17. Madni, S. H. H., et al. (2019). Hybrid gradient descent cuckoo search (HGDCS) algorithm for resource scheduling in IaaS cloud computing environment. *Cluster Computing, 22*(1), 301–334.
18. Midya, S., et al. (2018). Multi-objective optimization technique for resource allocation and task scheduling in vehicular cloud architecture: A hybrid adaptive nature inspired approach. *Journal of Network and Computer Applications, 103*, 58–84.
19. Prasanna Kumar, K. R., & Kousalya, K. (2020). Amelioration of task scheduling in cloud computing using crow search algorithm. *Neural Computing and Applications, 32*(10), 5901–5907.
20. Almezeini, N., & Hafez, A. (2017). Task scheduling in cloud computing using lion optimization algorithm. *International Journal of Advanced Computer Science and Applications, 8*, 11.
21. Arul Xavier, V. M., & Annadurai, S. (2019). Chaotic social spider algorithm for load balance aware task scheduling in cloud computing. *Cluster Computing, 22*(1), 287–297.
22. Alsaidy, S. A., Abbood, A. D., & Sahib, M. A. (2020). Heuristic initialization of PSO task scheduling algorithm in cloud computing. *Journal of King Saud University-Computer and Information Sciences, 34*(6), 2370–2382.
23. Srichandan, S., Kumar, T. A., & Bibhudatta, S. (2018). Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm. *Future Computing and Informatics Journal, 3*(2), 210–230.
24. Moon, Y. J., et al. (2017). A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments. *Human-centric Computing and Information Sciences, 7*(1), 1–10.

25. Shirani, M. R., & Safi-Esfahani, F. (2021). Dynamic scheduling of tasks in cloud computing applying dragonfly algorithm, biogeography-based optimization algorithm and Mexican hat wavelet. *The Journal of Supercomputing, 77*(2), 1214–1272.
26. Awad, N., Mz, A., Liang, J. (2016). *Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective bound constrained real-parameter numerical optimization*. Technical report, Nanyang Technology University, Singapore
27. Shukri, S. E., et al. (2021). Enhanced multi-verse optimizer for task scheduling in cloud computing environments. *Expert Systems with Applications, 168*, 114230.
28. Thanka, M., Roshni, P. U., & Maheswari, and E. Bijolin Edwin. (2019). An improved efficient: Artificial Bee Colony algorithm for security and QoS aware scheduling in cloud computing environment. *Cluster Computing, 22*(5), 10905–10913.
29. Nasr, A. A., et al. (2019). Cost-effective algorithm for workflow scheduling in cloud computing under deadline constraint. *Arabian Journal for Science and Engineering, 44*(4), 3765–3780.
30. Fanian, F., Bardsiri, V. K., & Shokouhifar, M. (2018). A new task scheduling algorithm using firefly and simulated annealing algorithms in cloud computing. *International Journal of Advanced Computer Science and Applications, 9*, 2.
31. Khorsand, R., & Ramezanpour, M. (2020). An energy-efficient task-scheduling algorithm based on a multi-criteria decision-making method in cloud computing. *International Journal of Communication Systems, 33*(9), e4379.
32. Rjoub, G., et al. (2021). Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems. *Concurrency and Computation: Practice and Experience, 33*(23), e5919.
33. Jayanetti, A., Halgamuge, S., & Buyya, R. (2022). Deep reinforcement learning for energy and time optimized scheduling of precedence-constrained tasks in edge–cloud computing environments. *Future Generation Computer Systems, 137*, 14–30.
34. Shruthi, G., et al. (2022). Mayfly Taylor optimisation-based scheduling algorithm with deep reinforcement learning for dynamic scheduling in fog-cloud computing. In *Applied computational intelligence and soft computing*. Hindawi Limited.
35. Islam, M. T., Karunasekera, S., & Buyya, R. (2021). Performance and cost-efficient spark job scheduling based on deep reinforcement learning in cloud computing environments. *IEEE Transactions on Parallel and Distributed Systems, 33*(7), 1695–1710.
36. Wang, L., et al. (2014). Bigdatabench: A big data benchmark suite from internet services. In *2014 IEEE 20th international symposium on high performance computer architecture (HPCA)*. IEEE.
37. Suresh Kumar, D., & Jagadeesh Kannan, R. (2020). Reinforcement learning-based controller for adaptive workflow scheduling in multi-tenant cloud computing. *Journal of Electrical Engineering & Education*, 0020720919894199.
38. Mostafavi, S., Fatemeh, A., & Sarram, M. A. (2020). *Reinforcement-learning-based foresighted task scheduling in cloud computing* (pp. 387–401)
39. Karthiban, K., & Raj, J. S. (2020). An efficient green computing fair resource allocation in cloud computing using modified deep reinforcement learning algorithm. *Soft Computing, 24*(19), 14933–14942.
40. Zheng, T., et al. (2022). Deep reinforcement learning-based workload scheduling for edge computing. *Journal of Cloud Computing, 11*(1), 1–13.
41. Sonmez, C., Ozgovde, A., & Ersoy, C. (2018). Edgecloudsim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies, 29*(11), e3493.
42. Wang, Y., et al. (2019). Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning. *IEEE Access, 7*, 39974–39982.
43. Balla, H. A. M., Sheng, C. G., & Jing, W. (2021). Reliability-aware: Task scheduling in cloud computing using multi-agent reinforcement learning algorithm and neural fitted Q. *International Arab Journal of Information Technology, 18*(1), 36–47.
44. Sheng, J., et al. (2022). Learning to schedule multi-NUMA virtual machines via reinforcement learning. *Pattern Recognition, 121*, 108254.

45. Zhong, J. H., et al. (2019). Multi workflow fair scheduling scheme research based on reinforcement learning. *Procedia Computer Science, 154*, 117–123.
46. Wang, B., Liu, F., & Lin, W. (2021). Energy-efficient VM scheduling based on deep reinforcement learning. *Future Generation Computer Systems, 125*, 616–628.
47. Cheng, F., et al. (2022). Cost-aware job scheduling for cloud instances using deep reinforcement learning. *Cluster Computing, 25*(1), 619–631.
48. Alhaidari, F., & Balharith, T. Z. (2021). Enhanced round-robin algorithm in the cloud computing environment for optimal task scheduling. *Computers, 10*(5), 63.
49. Hamid, L., Jadoon, A., & Asghar, H. (2022). Comparative analysis of task level heuristic scheduling algorithms in cloud computing. *The Journal of Supercomputing, 78*, 1–19.
50. Neciu, L.-F., et al. (2021). Efficient real-time earliest deadline first based scheduling for apache spark. In *2021 20th International Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE.