IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Multi-objective Prioritized Task Scheduler using improved Asynchronous advantage actor critic (a3c) algorithm in multi cloud environment

SUDHEER MANGALAMPALLI[1], GANESH R. KARRI[1], (Member, IEEE), SACHI N. MOHANTY[1], (Senior Member, IEEE), SHAHID ALI[2], M. IJAZ KHAN[3,4], SHERZOD ABDULLAEV[5,6], SALMAN A. ALQAHTANI[7]

[1]School of Computer Science and Engineering, VIT-AP University, Amaravati, India-522237 (e-mail: sudheerkietmtech@gmail.com, e-mail: guncity11@gmail.com, e-mail: sachinandan09@gmail.com)
[2]School of Computer Electronics Engineering, Peking University, Beijing, P.R. China (e-mail; alikhan@pku.edu.cn)
[3]Department of Mechanical Engineering, Lebanese American University, Beirut, Lebanon (e-mail; scientificresearchglobe@gmail.com)
[4]Department of Mathematics and Statistics, Riphah International University, Islamabad 44000, Pakistan (e-mail; scientificresearchglobe@gmail.com)
[5]Engineering School, Central Asian University, Tashkent, Uzbekistan (e-mail; sherzodbek.abdullaev.1001@gmail.com)
[6]Scientific and Innovation Department, Tashkent State Pedagogical University named after Nizami, Tashkent, Uzbekistan
[7]Computer Engineering Department, College of Computer and Information Sciences, King Saud University, Riyadh , Saudi Arabia (e-mail; salmanq@ksu.edu.sa)

Corresponding author: Shahid Ali (e-mail: alikhan@pku.edu.cn), Sudheer Mangalampalli (e-mail: sudheerkietmtech@gmail.com), Salman A. Alqahtani (e-mail: salmanq@ksu.edu.sa).

ABSTRACT Task scheduling is a crucial challenge in cloud computing paradigm as variety of tasks with different runtime processing capacities generated from various heterogeneous devices are coming up to cloud application console which effects system performance in terms of makespan, resource utilization, resource cost. Therefore, traditional scheduling algorithms may not adapt to this paradigm efficiently. Many existing authors developed various task schedulers by using metaheuristic approaches to solve Task scheduling problem(TSP) to get near optimal solutions but still TSP is a highly dynamic challenging scenario as it is a NP hard problem. To tackle this challenge, this paper introduces a multi objective prioritized task scheduler using improved asynchronous advantage actor critic(a3c) algorithm which uses priorities of tasks based on length of tasks, runtime processing capacities and priorities of VMs based on electricity unit cost using multi cloud environment. Scheduling process carried out in two stages. In the first stage, all incoming tasks, VM priorities are calculated at the task manager level and in the second stage, Priorities are fed to (MOPTSA3C) scheduler to generate scheduling decisions to map tasks effectively onto VMs by considering priorities and schedule tasks based on cost, resource utilization, makespan in the available multi cloud environment. Extensive simulations are conducted on Cloudsim toolkit by giving input trace different fabricated data distributions and real time worklogs of HPC2N, NASA datasets to the scheduler. For evaluating the efficacy of proposed MOPTSA3C, it compared against existing techniques i.e. DQN, A2C, MOABCQ. From the results, it is evident that proposed MOPTSA3C outperforms existing algorithms for makespan, resource utilization, resource cost, reliability.

INDEX TERMS Cloud Computing; makespan; Resource utilization; Resource cost; DQN; A2C;MOABCQ

I. INTRODUCTION

Cloud Computing paradigm gives seamless access to compute, storage, network access in terms of various services to all the users around the world by accessing them from their web browser with any type of device [1]. These services provided by cloud service provider(CSP) through this paradigm mainly categorized as Infrastructure as a service in which virtual infrastructure to the user to be provided to deploy their applications directly on cloud environment and access it from anywhere in the world. Platform as a service in which CSP provides a platform to users to develop their applications by providing necessary software, run time, development environment as a service. This service gives a great relaxation to the users as they don't need to worry about setting the development environment and software licenses, patching of software and they can focus on development of the application by saving time and investment in infrastructure. Software as a service provides readymade software services provided to cloud users on demand based on the requirement [2], [3]. All these services are to be provided to cloud users around the world on demand based on user requirement i.e. Service Level Agreement (SLA). These resources are to be made available to users with a technique known as virtualization. All these

virtual resources should be made available to users around the clock without having down time. It is possible only when these virtual resources are properly managed by the CSP. Therefore, it is important to employ an efficient task scheduler which schedules variety of tasks onto existing resources provided by CSP. It plays a major role in cloud paradigm from both the facets of cloud provider, user. It will be helpful for CSP in a way that it schedules all the tasks/jobs from various users around the world to the available virtual resources in the cloud paradigm automatically but this is a difficult challenge for a CSP to choose an algorithm which automatically manages and schedules all tasks onto virtual resources because the generated tasks are of different in size, runtime processing capacities and moreover that all tasks from users cannot be processed on a same type of a virtual resource. Therefore, choosing a proper virtual resource for a task is a main challenge. Employing an efficient task scheduler helps user to execute their tasks on an appropriate virtual resource and thereby helps user to provide quality of service and not violating SLA. The importance of the task scheduler in cloud paradigm is it effects various parameters directly or indirectly and it effects both CSP and cloud users. Resource utilization is one of the important parameter to be effected in cloud paradigm if a scheduler is not properly employed by the CSP. It results directly either into overutilization or underutilization of resources. It directly effects both CSP and users. From the facet of cloud user, it will be a direct effect if the resource utilization is very much high and if tasks are not accommodatable in the existing infrastructure, CSP would require more number of virtual resources which results in increase of resource costs and it will also impact on availability of a virtual resource to the user. Therefore, it is very important to choose and employ a scheduling algorithm which should carefully checks type of tasks, run time capacity and accordingly it should map tasks to suitable virtual resources. All the types of tasks cannot be mapped to same type of virtual resources. Therefore, it is the responsibility of CSP to carefully employ a scheduling algorithm to get balance between CSP and user to compute and facilitate all the requests of users in an efficient way which gives benefits to both users and CSP. Many existing task scheduling algorithms are proposed using various metaheuristic approaches i.e. GA [4], PSO[5], ACO[6], HEFT[7] etc. These metaheuristic approaches generates near optimal solutions as the scheduling problem in cloud computing is NP-Hard. Existing authors also used various Machine learning and Deep learning techniques i.e. DRBTSA[8], MOABCQ[9], RATS-HM[10] and few authors used hybridized approaches combining AI and ML algorithms with metaheuristic approaches to tackle task scheduling i.e. AINN-BPSO[11], [12] but still all these generates near optimal solutions in their perspective and addressed parameters makespan, energy consumption, resource utilization but these algorithms still suffers from adopting to heterogeneous tasks as it is a dynamic environment and scheduling these variety of tasks to appropriate precise VM is a challenging scenario while balancing the resource utilization and resource cost in multi cloud environment. Therefore, to tackle this issue, in this paper, we formulated a multi objective task scheduling approach which considers priorities of tasks based on their size, runtime capacity and priorities of VMs based on unit electricity cost. Schedules will be generated by using a deep reinforcement learning technique asynchronous advantage actor critic (a3c) algorithm in multi cloud environment which minimizes makespan, resource cost and improves resource utilization. The reason to choose a multi cloud environment is that while scheduling tasks to virtual resources there may be a chance of unavailability of resources in cloud environment or there may be a chance of increase in cost of resources in the cloud environment. Therefore, to minimize resource cost and improve resource utilization while scheduling the task our proposed MOPTSA3C scheduler checks for the pricing of requested resource and availability in multiple cloud environments and schedule tasks into that respective cloud environment while minimizing resource cost.

A. MOTIVATIONS AND CONTRIBUTIONS

Task Scheduling problem (TSP) plays a major role in cloud computing paradigm as it effects quality of service renders to customers and CSP while improving resource utilization, minimizing makespan, resource cost. It is important to employ an efficient task scheduler in this environment as if an incoming task is not scheduled to a suitable/precise virtual resource without considering size of tasks, runtime capacity then that task scheduling algorithm generates schedules which results in increase in makespan, improper utilization of resources. Therefore, it causes a serious problem to CSP by not utilizing virtual resources thereby effecting the makespan which results in increase of resource cost which is a serious concern for the cloud users. Therefore, this motivates us to tackle this problem using a reinforcement learning approach (a3c) which takes priorities of tasks, VMs based on unit electricity cost and checks resource availability and the cost of virtual resource in multiple cloud environments and it generates schedules while addressing makespan, resource utilization and resource cost. The main objectives and highlights of this manuscript are presented below.

1. A multi objective prioritized task scheduling algorithm is formulated using reinforcement learning strategy.
2. For effective scheduling process, we have incorporated priorities of tasks, VMs based on unit electricity cost to schedule tasks in multi cloud environment.
3. Improved Asynchronous advantage actor critic(a3c) algorithm is used as methodology in this research to tackle task scheduling problem in cloud computing.

4. Simulations are conducted on Cloudsim to generate schedules and it is compared against existing DRLBTSA, MOABCQ, RATSHM approaches.
5. Fabricated data distributions, HPC2N, NASA worklogs are used as input to this approach to evaluate its efficacy.
6. Finally, we evaluated parameters makespan, resource utilization, resource cost, reliability by using MOPTSA3C.

Rest of the manuscript is organized as follows. Section II discusses related works, Section III discusses System architecture, Section IV discusses asynchronous advantage actor critic algorithm which is the methodology used in this research, Section V discusses results, Section VI discusses Conclusion & future works.

II. RELATED WORKS

This section clearly presents existing algorithms formulated by various authors to tackle task scheduling in cloud computing. For minimization of total cost, energy consumption, authors in [13] proposed a task scheduling algorithm based on bi directional GGCN to choose precise VMs to deploy jobs or requests from various users. Authors used a randomized dataset to evaluate scheduler capability. It was implemented on COSCO framework in which they used Defog benchmark for scheduling in this approach. Hunter plus model is evaluated with different variations of CNN and results shown huge impact over other variations by minimizing energy consumption, job completion rate. Authors in [14], [15] proposed a task scheduling algorithm in multi cloud environment to tackle trust based parameters by using a hybrid approach FTTHDRL which is a combination of Harris hawk optimization and DQN model which is a reinforcement learning based approach. In this process, scheduling performed in two stages. In the first stage, task selection and mapping to the VMs are performed using Harris Hawk algorithm. In the second stage, scheduling optimization is performed by DQN model to adapt to dynamic nature of cloud paradigm as it is difficult to identify and schedule tasks precisely. It was implemented on Cloudsim and conducted rigorous simulations are done by using realtime worklog traces. Finally, it was evaluated over state of art approaches to check the efficacy of approach. Results proved that FTTHDRL improves trust on cloud provider through SLA based parameters. Authors in [16] proposed a hybrid task scheduling mechanism which addressed makespan, resource utilization, processing cost. They used three algorithms in total to perform scheduling process. Initially task collection and prioritization performed using HEFT, initial solution generated using GRASP, schedules generated using BABC algorithm with pareto front technique. It was implemented using workflowsim. It was evaluated over state of art algorithms and results of EBABC-PF shown dominance over them for the above specified parameters.

In [17], a hybrid workflow scheduling algorithm proposed using HEFT, BAT approaches. It was implemented on workflowsim by using random workload but authors considered various realtime scientific workflows to evaluate MOHBA. This approach was compared against contemporary approaches and results shown impact over existing algorithms for improvement of makespan, resource utilization. Minimizing energy consumption in datacenter by making green computing environment is the target of authors in [18,19] and this aim made them to develop a VM placement algorithm by taking the constraints VM dependency, type of topology. This VMP algorithm chooses the place of VM based on above said constraints by making the unused switches to become idle and by reducing resource waste by improving resource utilization. Modified discrete Jaya optimization was used as methodology in this approach. A customized simulation environment was developed by authors considering various scenarios by varying different number of VMs to evaluate energy consumption, total task time, makespan.

Authors in [20] proposed a hybrid task scheduling approach which combines wild horse optimization, levy flight operator. In the first stage of scheduling, task distribution model developed based on schedule length, time, cost. In the second stage, generated schedules will be optimized by levy flight operator to improve local search process and to avoid premature convergence. Cloudsim tool used to implement this approach. It compared over existing state of art approaches WOA, MSA,ALO, MALO for evaluating parameters makespan, energy consumption. Simulation results proved that IWHOLF-TSC dominated all existing approaches for above mentioned parameter.

In [21], authors proposed a hybrid approach HWACO which works based on weights imposed to converge towards solutions easily compared with conventional approaches. Cloudsim used as simulation toolkit in this research. It compared against conventional approaches ACO, QANA, BPSO, FCFS. Randomized workloads are given as input to the HWACO. Analysis of results shown that HWACO outperformed conventional approaches in view of cost, efficiency, makespan. A trust based task scheduling algorithm developed using firefly algorithm in [22], [23] to address makespan, availability, turnaround efficiency which effects trust on CSP. They used prioritized scheduling in which they considered task priorities to carefully schedule tasks to map virtual resources. It was implemented on Cloudsim toolkit. TAFFA took an input trace from HPC2N, NASA worklogs. It evaluated over state of art approaches and observed that above mentioned parameters are greatly minimized. A fault tolerant aware scheduler with multiple objectives while considering QoS constraints is developed using GBFD which minimizes expenditure cost for users and success rate for CSP was proposed in [24]. Simulation with real world cluster taken as input and performed on Cloudsim. It evaluated against existing task scheduling mechanisms i.e. FCFS, CGDPS, MBFD. Results proved that GBFD outperforms other algorithms in various scenarios for improvement of fault tolerance, user satisfaction.

Minimization of task execution time by assigning a suitable task to an appropriate virtual resource is discussed in [25]. For this to happen, GA combined with map reduce architecture is proposed in [25], [26]. This scheduler works in two stages. In first stage, tasks are assigned to processor by scheduling it with GA. In the second stage, GA with map reduce is combined and assigns heterogeneous tasks to processors in parallel with the help of priority queues. Simulations conducted using MATLAB software by using random task generation. From results it proved that GA combined with MapReduce greatly minimizes task execution time over PSO, GA, IWD, MFO, GA algorithms.

In [27], a three layered task scheduling model which minimizes makespan in Cloud paradigm. In first layer, a model that uses opposition based learning technique which uses adaptive mobility factor to expand search strategy. In the second layer, a whale optimization based gaussian approach formulates multi objective task scheduling model which minimizes task completion time. Finally, in the third layer GCWOA strategy implemented to optimize scheduling process. This model was implemented on MATLAB software by using random workload. Finally, GCWOAS2 improves resource utilization, makespan over ACO, WOA, PSO algorithms. Scheduling cost, time plays a major role in task scheduling cloud paradigm from both facets of cloud user and CSP. These issues addressed by authors in [28,29] by developing a scheduling algorithm using improved whale optimization. Initially a task scheduling, distribution model was developed by considering scheduling time, cost constraints. After this phase, by using inertia weight strategy whale optimization algorithm applied on this model to choose best whale i.e. in this case it is best possible task to map on to a VM. MATLAB tool was used as simulation tool for simulation. Results of IWC greatly minimizes scheduling cost, time when it was compared over PSO, ACO, WOA. Energy consumption in datacenters is a crucial part as number of users are getting increased in this model, thereby difficulty arises in efficient distribution of tasks, balancing the load among different VMs. This problem was tackled by authors in [30] by using a hybrid approach by combining squirrel search with improved GA. Proposed hybrid method improves makespan, execution time and energy consumption when it was compared ACO, PSO, GA algorithms for the above specified parameters.

In [31], [31], authors developed sub models to improve performance of task scheduler. They used reinforcement learning and queuing models to formulate sub models i.e. Task scheduling model, execution model, transmission model to identify repetitive processes to optimize performance of the scheduler by using aggregators. Experimentation conducted using MATLAB software. With this approach efficiency of task scheduling improved by taking server rate, arrival rate of tasks as constraints when it is compared against state of art approaches. Scheduling analytics jobs in cloud paradigm is difficult as those jobs confined with different computing characteristics. Therefore authors in [33] proposed a RL based framework spark deployed cloud cluster which consists of two RL based frameworks to schedule jobs which tackles multiple objectives VM usage cost, job duration. Results shown that there is a huge impact on improvement of VM usage cost, job duration on existing frameworks which are configured with conventional algorithms.

Resource utilization, task execution time gained importance in task scheduling in cloud computing as workloads in this model drastically increased and to automate this scheduling process is a must in cloud model. To handle this situation, authors in [34] proposed a task scheduler with different RL approaches i.e. RL, RLL-LSTM, DQN, DRL-LSTM. Out of these four DRL-LSTM improves memory usage, CPU usage, task execution time over SJF, RR, IPSO.

In[35], [36], an energy efficient based task scheduling algorithm developed using Deep reinforcement learning. This scheduler was implemented using Cloudsim toolkit and assumed that entire architecture as public cloud because users in public cloud can generate any number of tasks at any time. It was compared over conventional heuristic approaches in view of energy consumption, response time. Results proved that DRL based scheduler improves above said parameters while scheduling jobs efficiently when it is evaluated over conventional approaches. Streaming applications causes lot of challenges in cloud paradigm as they need to be scheduled with specific virtual resources configured with streaming set of configuration of resources. This problem solved by authors in [37] by deriving a dynamic online task scheduler which need to schedule huge processing capacity tasks to limited virtual resources which need to render good QoS services. This scheduler modeled by using DDQN model which have adaptive learning especially required in cloud model. DDQN-TS evaluated over conventional metaheuristics with random, google workload traces, Alibaba benchmarks and observed the improvement in evaluated parameters task completion rate, average response time over state of art approaches.

In [38], a bi objective task scheduling algorithm developed using DQL. Initially Q-learning was combined with Deep neural network to gain advantages of Q-learning. Primary concerns in formulation of this scheduler is to improve resource utilization, makespan. DQL implemented on workflowsim and compared over MIN-MIN, FCFS, MAX-MIN, RR algorithms. Results revealed that DQL based scheduler improves resource utilization, makespan over existing algorithms. In [39], energy aware task scheduler concerned with addressing multiple objectives formulated by using an AINN model. Initially all tasks from various resources are scheduled accurately using AINN model by predicting suitable VM for all incoming tasks. Input dataset generated by using GA algorithm which consists of 18 million instances. MATLAB tool used to simulate this model. It evaluated over MIN-MIN, GA, Linear regression models and AINN scheduler revealed that improvement in average makespan, energy consumption, execution

overhead, active racks by 59%, 45%, 88%, 70% respectively over compared approaches.

In [31], authors formulated a task scheduling mechanism which focused on energy consumption, SLA violations. Methodology chosen by authors is a Deep Reinforcement Learning model and it consists of two stages. In first stage, Deep learning model is deployed in which QoS features were extracted using autoencoders. In second stage, a reinforcement learning approach which uses collaborative learning through which characteristics of a task can be easily deduced and scheduled onto a virtual resource. Extensive set of simulations were conducted using MATLAB. From results, it shown that proposed approach outperformed over state of art algorithms in view of SLA violation, energy consumption, QoS.

Authors in [40], [41] proposed a cost aware task scheduler to handle realtime workloads to be scheduled onto VMs which should minimize cost to run on VMs. DRL model i.e. DQN is used as methodology to implement this cost model. Pytorch is used as tool to train and evaluate parameters. DQN was compared over conventional mechanisms i.e. RR, Random, Earliest schedulers as they are used as conventional mechanisms to process batch workloads. Finally, results proved that DQN surpassed existing algorithms over parameters Success rate, Average Response time, Cost of execution of tasks on VMs. A Three level scheduling policy is designed in [42] by authors to address parameters makespan, cost. A Deep Q-network model is enhanced to adopt this procedure. In the first level, a dynamic adaptive coefficient procedure is adapted to precisely estimate target value among all diversified values i.e. in this case they need to estimate the precise VM for set of tasks. In second level, a pointer based agent network is deployed which selects set of tasks to identify and send them onto respective VMs for processing. In third level, a sensing mechanism was deployed to identify objectives of each task set and preserve the QoS in the environment. TensorFlow framework used as simulation tool. WDQDN-RL compared over existing approaches NSGA-II, MOPSO, DQN-RL. Finally, results revealed that WDQN-RL outperforms above algorithms in view of makespan, cost. Energy consumption, makespan, Migration time are measured in [43] by formulating a task scheduling algorithm by using a hybrid approach. This hybrid approach uses capuchin search as local search process and inverted ACO as global search process. Simulations are conducted using Cloudsim with input of realtime supercomputing worklogs. From the outcomes of CapSA, it was proved that above mentioned parameters are improved in a drastic manner when it was compared over CSO, PATS, FHCS approaches.

In [44], authors formulated a hybrid workflow scheduling algorithm (PCP-ACO) which is a combination of partial critical path and Ant colony optimization algorithms. In the initial stage, PCP heuristic calculates priorities based on sub tasks and deadlines involved in workflow. In the final stage, metaheuristic will select tasks based on priorities generated by heuristic in the initial stage. Simulations conducted on workflowsim and evaluated over state of art algorithms. Execution cost of PCP-ACO improved by 19%, 17%, 21% over L-ACO, HP-GA, IC-PCP approaches.

In [45], a multi objective task scheduling model formulated by authors using an improved a3c algorithm by incorporating RCNN which consists of multiple threaded training models which helps in assigning tasks to VMs in dynamic environment. All the experimentation conducted on edge-cloud-co simulator. It was evaluated over A3C, A3C+LSTM,GOBI algorithms and evaluated parameters Average response time, Energy consumption outperformed over existing approaches. Authors in [46] proposed an adaptive multi objective scheduling strategy proposed using a metaheuristic approach. PSO is the metaheuristic used in the algorithm which uses adaptive acceleration coefficient to explore diversity of search solution space and allot tasks to appropriate VMs based on generated solutions. Cloudsim toolkit used as simulation platform and evaluated over different metaheuristic approaches. Finally generated schedules using AMTS improves resource utilization, energy consumption over existing algorithms.

In [47], a two layered scheduling strategy developed using EDA, GA metaheuristic approaches. In the initial stage, task selection, assignment are done by using EDA and expandability of search process is enhanced by GA. Finally, optimization of scheduling process carried out by combining of both approaches. It simulated on Cloudsim and evaluated over classical GA, EDA algorithms. Results proved that task completion time greatly minimized, load balancing is improved over above classical approaches. Authors in [39] proposed a multi objective scheduling mechanism which preserves QoS while allocating tasks to suitable VMs. A hybrid metaheuristic approach HGA-ACO was used to formulate scheduling mechanism in which operators of GA are enhanced using ACO and initialization of ACO performed using GA approach. All simulations are conducted using Cloudsim toolkit and compared against classical GA, ACO algorithms. From results, it proved that HGA-ACO minimizes response time, task completion time over conventional mechanisms. Energy consumption is a crucial aspect for both CSP and user in cloud paradigm. Authors in [48], [49] aimed at minimization of energy by formulation of a task scheduler using NSGA-II, AINN techniques. In this approach, initially characteristics of tasks and selection of tasks are identified using NSGA-II approach by incorporating DVFS technique into NSGA-II. For generated tasks, an AINN technique used to predict VM for selected tasks in cloud paradigm. Simulation results shown huge impact over existing approaches by minimizing energy consumption.

In [50], a task scheduling mechanism formulated using two folded biological heuristic approaches. These algorithms are GA, BF algorithms in which initial stage formulated using GA using different operators to explore search space and generated solutions are scheduled using BF approach. These generated solutions are compared over conventional algorithms. Results shown huge impact over these

algorithms with respect to reduction of makespan, energy consumption in a huge manner.

In [51], a task scheduling strategy formulated using an improved ACO which considers constraints i.e. makespan, user budget and these two constraints are used as feedback mechanism in this approach. Improved ACO considers feedback of these constraints for every iteration and evaluates makespan, cost, deadline violations, utilization of resources. Simulation results shown IACO outperformed classical metaheuristics in terms of above specified parameters. Makespan is one of the primary concern in task scheduling as it effects QoS of CSP. Therefore, authors in [52] proposed three scheduling approaches, which considers various tasks from heterogeneous resources and schedules tasks based on peak load at that respective CSP. Three scheduling approaches are MCC, MEMAX, CMMNN where it considers makespan as primary criteria and thereby resource utilization by all these approaches. Finally these are evaluated using various synthetic datasets for checking efficacy of formulated approaches over conventional approaches. All the proposed formulated approaches outperforms makespan, utilization of resources over classical algorithms.

Authors in [53] designed a task scheduling approach which addresses datacenter infrastructure efficiency, utilization of CPU, SLA violations. This model formulated by using RL which looks at rewards for every iteration and make a decision based on corresponding rewards. It was implemented using Cloudsim. With the observations of results generated by RL-EERA approach surpassed on conventional approaches in view of above mentioned parameters by effectively allocating resources to heterogeneous tasks. In [54], a task scheduler is formulated in two stages using a queuing mechanism and Q-learning which is a reinforcement learning. In the first stage, a task dispatcher uses M/M/S queuing mechanism to assign tasks to virtual resources in cloud paradigm. In second stage, for generated assignment of tasks, Q-learning mechanism is applied which gives optimized schedules for each task assigned to appropriate cloud resources. It was implemented using Cloudsim and evaluated over classical approaches to minimize energy consumption.

Authors in [55] designed a scheduling algorithm which considers multiple objectives makespan, cost to be addressed. These issues are addressed by authors in [56] using Markov gaming model which is an AI approach. It takes number of requests from different workflows, available VMs in cloud model. Extensive simulations are conducted by taking AWS EC2 instances. From observing results of Markov model based algorithm makespan, cost are greatly reduced over conventional algorithms.

In [57], a workflow scheduling mechanism developed using DRL. It developed in two stages. In the first stage, task selection and assignment operations are performed using Markov decision model. In second stage, all these schedules generated are given as an input to DDQN model to predict failures. It was simulated using Workflowsim. It compared over classical approaches and observed that improvement in makespan, utilization of resources, fault tolerance. For achieving optimized makespan results in cloud paradigm, authors in [59] developed an ML-based task scheduling algorithm which uses Q-Learning and HEFT algorithms. This scheduling is divided into two phases. In first phase, using a HEFT approach with the help of upward rank task sorting phase performed and generates schedules according to the ranks. In Second phase, Q-learning applied on generated schedules to check whether they achieved better optimized results or not. Generated schedules may vary in this paradigm as Q-table updated with different values based on obtained rewards in previous iterations. QL-HEFT compared over HEFT_U, HEFT_D,CPOP approaches. From results, it proved that QL-HEFT minimizes makespan and improves speedup ratio of tasks in huge manner.

Authors in [59], [60] proposed a multi objective task scheduling algorithm which uses enhanced version of multiverse optimizer which is a metaheuristic approach. Main aim of authors is to address execution time, cost, resource utilization. Adaptive coefficient used to explore search space. EMVO compares over MVO, PSO approaches. Results shown that EMVO minimizes cost, execution time while resource utilization improved significantly when it is evaluated against classical approaches.

Authors in [61] designed a task scheduling algorithm focuses on addressing task processing time, makespan. This framework designed based on Q-learning which is a RL based approach. In first phase, tasks are allocated to virtual servers based on server type. In second phase, Q-learning based scheduling performed based on past history and interactions of tasks with VMs by using a parameter upper confidence bound. It works based on RL mechanism which is totally reward based. It compared against classical PSO, RR algorithms. Upon observing results of QMTSF, above said parameters are significantly improved over classical approaches. A resource scheduling framework developed by authors in [62] using Q-learning which is a RL based approach. It was implemented in workflowsim. This approach mainly addresses time, cost, deadline analysis, load balance in scheduling. When it compared over PSO and CSO, resource utilization improved by 63%, rate of task acceptance is increased by 54% when it was compared over crow search mechanism.

In [63], [67], a DRL based scheduling approach was developed to address makespan, energy consumption, throughput resource utilization. It was implemented using Cloudsim toolkit and compared with PSO, MVO, EMVO algorithms. DRL based approach takes the input of google cloud job traces and outperformed over all approaches for mentioned parameters. Authors in [64], [68] proposed a container based task scheduling algorithm using two folded approach. In the first phase, to choose a virtual container, MMCO used as methodology for preserving SLA. For proper CPU allocation, MPIO approach used for task clustering and for allocating tasks accurately to suitable virtual server DCNN is used. Finally it was implemented

using Kubernetes container to perform containerization and when compared over classical approaches DSTS shown improvement of makespan and efficient allocation of tasks to suitable virtual servers. In [65], [69], an adaptive task scheduling algorithm based on Reinforcement learning proposed using gradient updation for different cloud environments to accelerate and quickly adapt to that respective environment. MRLCC compared over existing baseline algorithms and proved that resource utilization rate is improved in results of MRLCC.

TABLE I
TASK SCHEDULING ALGORITHMS PROPOSED BY EXISTING AUTHORS

Authors	Technique used	Addressed Parameters
[11]	Hunter Plus model	Total Cost, makespan, Energy Consumption
[12]	FTTHDRL	makespan, Total cost.
[13]	EBABC-PF	Makespan, processing cost, resource utilization.
[14]	MOHBA	Resource utilization, Makespan, energy consumption.
[15]	MOD-JAYA	Total task time, makespan, cost, energy consumption.
[16]	IWHOLF-TSC	Power consumption, Task execution time.
[17]	HWACOA	execution time, Makespan, Cost.
[18]	TAFFA	success rate, availability, makespan and turnaround efficiency.
[19]	GBFD	Fault tolerance, user satisfaction
[20]	Parallel GA with a MapReduce	total execution time, cost, Makespan.
[21]	GCWOAS2	Resource utilization, Cost, throughput, Degree of Imbalance
[22]	IWC	scheduling cost, task scheduling time.
[23]	CTSS	Makespan, energy consumption and Total power cost.
[24]	Random TSRL	Server rate, arrival rate of tasks
[25]	DRL-based scheduling	Gain cost, resource utilization, and makespan.
[26]	DRL-LSTM	task waiting time and resource consumption.
[27]	DRL	job success rate, average response time, energy consumption.
[28]	DDQN-TS	Estimated completion time, Task transfer time.
[29]	DQTS	Load balance, makespan
[30]	Artificial Neural Network-based scheduling	energy consumption, execution overhead, makespan
[31]	Collaborative VM scheduling	energy, cost, resource utilization, makespan, SLA
[32]	Deep Q-learning network model	energy efficiency, load balancing
[33]	WDDQN-RL	Total power cost in datacentres, Makespan, migration time, energy consumption.
[34]	CAPSA & IACO	Load balancing, Execution time.
[35]	PCP–ACO	Average Execution Cost, makespan.
[36]	A3C	energy consumption, task response time
[37]	AMTS	Resource utilization, Task Completion time, energy consumption.
[38]	EDA-GA	task completion time, load balancing, cost
[39]	HGA–ACO	throughput, completion time and response time.
[40]	NSGA	Energy consumption and makespan
[41]	GA- BF	energy consumption, response time, makespan
[42]	Improved ACO	resource utilization, cost, deadline violation rate, makespan.
[43]	MCC algorithm	makespan and average cloud utilization
[44]	RL-EERA	accuracy, CPU Utilization, Response time.
[45]	QEEC	task response time, energy consumption and CPU utilization
[46]	DQN Based Multi agent RL	Task completion time and cost.
[47]	RLFTWS	makespan, resource usage rate
[48]	QL-HEFT	Makespan, total cost, response time.
[49]	EMVO	resource utilization, Throughput, execution time
[50]	QMTSF	Average Processing time, makespan.
[51]	DR Q-learning	Load balancing, energy consumption, deadline violation, makespan.
[52]	Adaptive DRL	throughput, makespan, energy consumption, resource utilization.
[53]	DSTS	throughput, resource residual degree, response time, resource imbalance degree.
[54]	MRLCC	average utilization rate, makespan.
[55]	GAGELS	Execution time, Resource utilization
[56]	SGO based SJF	Makespan, throughput
[57]	SG-PBFS	Makespan
[58]	MTD-DHJS	Makespan

From above Table I it is clearly observed that many authors used various metaheuristic, ML, DL based approaches in order to solve task scheduling problem in Cloud Computing. While addressing task scheduling in Cloud paradigm, authors addressed makespan, execution cost, task waiting time, energy consumption, power cost, fault tolerance, resource utilization and generated near optimal solutions but in cloud model still the problem of resource utilization i.e. over utilization and underutilization problem persists as it is an NP-hard problem. Many authors used various approaches to address utilization of resources and failed to get balance in between CSP and user as if overutilization occurs resource cost will be increased drastically. If resources are underutilized configured virtual resources will be wasted which incurs huge power consumption. This will create a burden on the CSP as well as on user. There may be chances that a virtual resource may not be available in cloud environment for a specific task or the cost of virtual resource service is high in cloud environment. Therefore, to tackle this

situation and to address parameters initially, (MOPTSA3C) task scheduler by carefully calculates priorities of both tasks, VMs which are coming onto cloud application console and these tasks are sorted in task manager according to task priorities. Prioritized tasks are to be mapped to prioritized VMs i.e. in this case VM priorities are evaluated using highest electricity unit cost among datacenters to electricity unit cost at that respective datacenter. In the second stage, all these priorities are fed to scheduler which uses improved A3C mechanism which is a RL based approach generates schedules for the collected prioritized tasks. The main aspect we used in this research is that we have simulated our proposed MOPTSA3C in multi cloud environment to minimize resource cost and migrate tasks to respective VMs where that respective service cost is low.

III. MATHEMATICAL MODELING & SYSTEM ARCHITECTURE

This section discusses mathematical modeling and System architecture of proposed MOPTSA3C. Initially, for mathematical formulation of task scheduler, we consider $\boldsymbol{k1}$ number of tasks indicated as $\boldsymbol{t_{k1}} = \{\boldsymbol{t_1}, \boldsymbol{t_2}, \boldsymbol{t_3}, \ldots \boldsymbol{t_{k1}}\}$. $\boldsymbol{n1}$ number of VMs indicated as $\boldsymbol{v_{n1}} = \{\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_3} \ldots \boldsymbol{v_{n1}}\}$, $\boldsymbol{i1}$ number of physical machines indicated as $\boldsymbol{PM_i} = \{\boldsymbol{PM_1}, \boldsymbol{PM_2}, \boldsymbol{PM_3} \ldots \boldsymbol{PM_{i1}}\}$, $\boldsymbol{j1}$ number of datacenters indicated as $\boldsymbol{DC_{j1}} = \{\boldsymbol{DC_1}, \boldsymbol{DC_2}, \boldsymbol{DC_3}, \ldots . \boldsymbol{DC_{j1}}\}$.In this research, we formulated problem statement as $\boldsymbol{t_k}$ tasks should be mapped to $\boldsymbol{v_n}$ VMs which resided in $\boldsymbol{PM_i}$ physical machines which placed in $\boldsymbol{DC_j}$ datacenters and assumed it as a multi cloud environment while minimizing makespan, resource cost, improves resource utilization. The below Fig.1. indicates proposed system architecture of MOPTSA3C. Initially, various tasks are generated from heterogeneous resources and coming to cloud application console. These tasks are captured by brokers on behalf of CSP which is a software agent employed in cloud architecture. Brokers will submit all these tasks to task manager. We have induced a process in task manager to calculate priorities of tasks based on size of tasks and to which VM it need to be assigned. Therefore, VM priorities also to be calculated based on unit electricity cost of VMs. These two priorities are fed together to MOPTSA3C which is a Deep Reinforcement learning based scheduler captures these priorities and generates schedules according to resources available in multiple cloud environments. In this approach, if one task arrived at scheduler with certain priority i.e. if it is highest priority it should be mapped to a VM which is having highest priority i.e. VM with low electricity cost at respective datacenter in multi cloud environment. Initially scheduler looks for prioritized VM availability at the datacenter and if it is not available it looks for the same prioritized VM in datacenter in other cloud environment and it also looks for the pricing of the services requested by the user in both cloud environments and migrates tasks wherever the resource cost is less. If there is a case that if none of the datacenters are available with required prioritized VM then scheduler will assign a VM with next priority in the cloud model with least cost pertaining to that service. While scheduling tasks according to the procedure adapted by MOPTSA3C we are addressing parameters makespan, resource utilization and resource costs. The below Table II indicates all notations we have used in mathematical modeling.

TABLE II
NOTATIONS USED IN MATHEMATICAL MODELING OF MOPTSA3C

Notation	Meaning
wl_{n1}^{v}	Current Workload running on $n1$ VMs
wl_{i1}^{PM}	Current Workload running on $i1$ Physical Machines
$prca_{n1}^{v}$	processing capacity of $n1$ VMs
$totprca_{n1}^{v}$	Total capacity of all $n1$ VMs
t_{k1}^{l}	size of $k1$ Tasks
t_{k1}^{pri}	Task Priorities of all $k1$ tasks
v_{n1}^{pri}	VM Priorities of all $n1$ VMs
mp^{k1}	Makespan of $k1$ Tasks.
$ex_{t_{k1}}$	Execution time of 1 Tasks.
$fini_{time}^{t_{k1}}$	Finish time of $k1$ Tasks.
$dl_{t_{k1}}$	Deadline constraint of $k1$ Tasks.
Re^{cost}	Resource cost in multiple Cloud environments.
$load_{cpu_{i_1}}$	Load on CPU on considered $i1$ Physical Machines

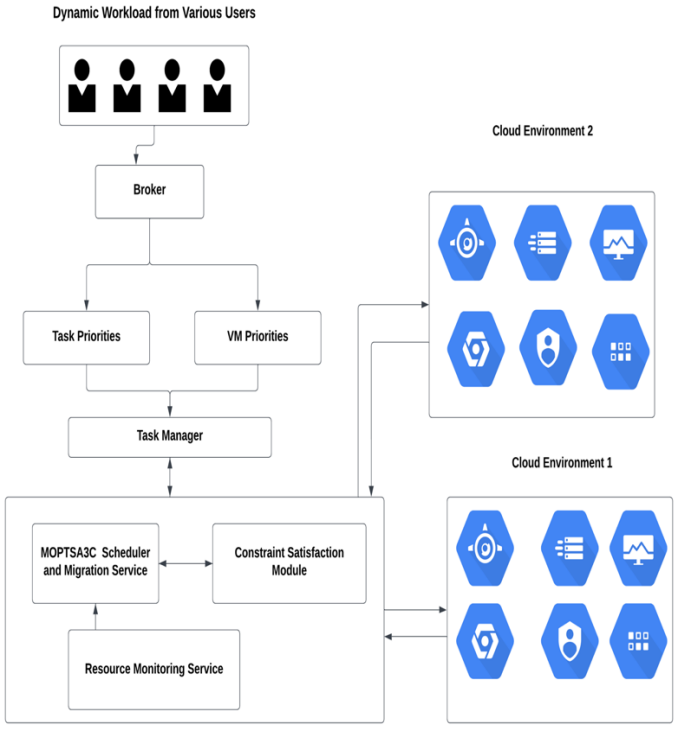

FIGURE 1. System Architecture for MOPTSA3C

In this mathematical modeling, we assumed all these resources are in multiple cloud environments. Initially in this mathematical modeling, to calculate priorities of tasks, it is important to know how much workload currently being run on the VMs. Therefore, current workload on VMs in multiple cloud environments to be calculated using below equation(1).

$$wl_{n1}^{v} = \sum wl_{n1} \quad (1)$$

After calculation of current running load on VMs, as these VMs are placed in Physical machines. Therefore, we also need to calculate current running workload on physical machines in all considered $i1$ physical machines in multiple cloud environments. It is calculated using below equation(2).

$$wl_{i1}^{PM} = \frac{wl_{n1}^{v}}{PM_{i1}} \quad (2)$$

It is necessary to know about processing capacities of VMs considered in different multiple cloud environments as priorities of tasks will be depends on VM capacities and it is calculated using equation(3).

$$prca_{n1}^{v} = pr^{no} * pr^{mips} \quad (3)$$

After calculating each capacity of a VM, total processing capacities of $n1$ VMs considered in multiple cloud environments calculated using equation(4).

$$totprca_{n1}^{v} = \sum prca_{n1}^{v} \quad (4)$$

In our research, it is important to calculate priorities of tasks as we choose a specific VM for prioritized tasks. To evaluate priority, we need to know the size of tasks coming to cloud application console. It is evaluated using equation(5).

$$t_{k1}^{l} = t_{mips}^{l} * t_{k1}^{pr} \quad (5)$$

After calculation of size of tasks from eqn.5. priorities of tasks are calculated using equation(6).

$$t_{k1}^{pri} = \frac{t_{k1}^{l}}{prca_{n1}^{v}} \quad (6)$$

In our research, we are calculating priorities of VMs based on unit electricity cost which helps scheduler for the efficient mapping of tasks to VMs. It is calculated using below equation(7).

$$v_{n1}^{pri} = \frac{\left(el_{high}^{cost}{}_{DC_{j1}}\right) * load_{cpu_{i_1}}}{el_{DC_{j1}}^{cost}} \quad (7)$$

From equation (6) we calculated priorities of tasks, equation(7) gives priorities of VMs using electricity cost at datacenters. These both priorities are fed to MOPTSA3C scheduler by task manager to generate schedules for incoming tasks. Our scheduler generates schedules with consideration of priorities by using A3C while minimization of parameters makespan, resource cost, utilization of resources. Before calculation of makespan, we are interested in identifying execution time of tasks as makespan depends on execution time. Execution time of tasks calculated using equation(8).

$$ex_{t_{k1}} = \frac{ex_t}{prca_{n1}^{v}} \quad (8)$$

Every task will have finish time and we have posed a deadline constraint in our work through which every task should complete its execution before the deadline is completed. Therefore, finish time of a task always should be less than deadline. Initially, finish time of $k1$ tasks are calculated using equation(9). We have mentioned that finish time should always be less than deadline of considered $k1$ tasks and it is mentioned in equation(10).

$$fini_{time}^{t_{k1}} = \sum v_{n1} + ex_{t_{k1}} \quad (9)$$

$$fini_{time}^{t_{k1}} < dl_{t_{k1}} \quad (10)$$

After mathematical formulation of task, VM priorities makespan is formulated in the equation(11). Makespan is a primary concern of any task scheduler as it depends on execution time of all considered tasks. If value of makespan increases, it effects performance of task scheduler directly. Therefore it is considered as one of the parameter to be addressed in our research. It is calculated using equation(11).

$$mp^{k1} = min(fini_{time}^{t_{k1}}) \quad (11)$$

$$min\ fini_{time}^{t_{k1}v_{n1}} = \sum_{i=1,j=1}^{k1,n1} \Psi_{i,j}(fini_{time}^{t_{k1}v_{n1}}) \quad (12)$$

In equation(12), where, $\Psi_{i,j}$ is a parameter which indicates when a task t_{k1} is assigned to a VM v_{n1} and it will be set to 1 otherwise it will be set to 0. After formulation of makespan, next parameter we considered is resource cost. The main reason to choose resource cost as a parameter is most of the cloud users facing issues with high resource cost which incurs high billing for the services consumed by cloud user. This is mainly due to the inefficient mapping of tasks/jobs requested by user in the cloud environment. To tackle this and to get benefit for both customer and CSP, we formulated a prioritized task mapping procedure which maps high prioritized tasks to a high prioritized VM by checking availability of that resource availability in multiple cloud environments where the resource cost is low. If that corresponding resource is not available and then it can be

assigned to the next prioritized resource in cloud environment where the resource cost is less. It is calculated using below equation(13).

$$\boldsymbol{Re^{cost}} = \sum_{n1=1}^{v_{n1}} \frac{\boldsymbol{running\ cost\ of\ t_{k1} * memory\ for\ t_{k1}}}{\boldsymbol{v_{n1} * PM_{i1}}} \quad (13)$$

After evaluating resource cost carefully, we are interested in evaluating utilization of resources in the cloud environment as if tasks are suitably mapped to an efficient VM then makespan is minimized which also effects parameters resource cost, utilization of resources in cloud paradigm. This motivates us to formulate utilization of resources using equation(14). Resources in cloud are of two types i.e. CPU, I/O and bandwidth. In this research, we are focussed on load of CPU in considered $\boldsymbol{i_1}$Physical machines in cloud environment. Therefore, load on CPU in $\boldsymbol{i_1}$physical machines are calculated using below equation(14).

$$\boldsymbol{load_{cpu_{i_1}}} = \sum_{k1=1}^{s} \frac{\boldsymbol{usage(k1)}}{\boldsymbol{cpu_{i_1}}} \quad (14)$$

Where, $\boldsymbol{s}$ indicates number of active tasks on a Physical machine. $\boldsymbol{cpu_{i_1}}$is the capacity of $\boldsymbol{cpu}$, $\boldsymbol{usage(k1)}$ is the usage of $\boldsymbol{cpu_{i_1}}$of $\boldsymbol{i_1}$physical machine. Reliability of the scheduler depends on decrease in number of faults. Generally, in any cloud model, there may be a chance to occur short term faults like system crash, bugs in software. These are common faults occurred in system. Probability of occurrence for transient failures is likely to be followed by Poisson distribution. We haven't focused on transient faults memory, network interfaces in this research. we have concentrated mainly on fault rate($\boldsymbol{\tau}$) which depends on computing node operational frequency $\boldsymbol{freq_{op}}$. The relation between operational frequency and fault rate is given in equation (15).

$$\boldsymbol{\tau(freq_{op}) = \tau_o . F(freq_{op}) = \tau_o . 10^{\frac{d(1-freq_{op})}{1-freq_{op min}}}} \quad (15)$$

Where $\boldsymbol{freq_{op}}$ indicates operational frequency, $\boldsymbol{\tau_o}$ is initial fault rate, $\boldsymbol{F(freq_{op})}$ is a decreasing function, where $\boldsymbol{d > 0}$ is constant. Reliability of the system is defined in equation (16)

$$\boldsymbol{Re_{t_{k1}}(freq_{op}) = e^{-\tau(freq_{op}).ex_{tk1}/freq_{op}}} \quad (16)$$

IV. METHODOLOGY USED IN PROPOSED MOPTSA3C

This section discusses methodology used in proposed MOPTSA3C which is a reinforcement learning approach i.e. improved Asynchronous Advantage Actor Critic (A3C) algorithm . It is composed with two components i.e. Actor network which is used to map your incoming state of tasks to action space where tasks need to be mapped and executed whereas on the other hand critic network evaluates action which is performed by actor network. It is an asynchronous approach in which each actor network evaluated parallelly on different threads and each thread after completion of running it evaluates loss in actor network and interacts with global network by accumulating gradients. In this research an improved A3C approach used because conventional A3C suffers with learning features in a dynamic policy based complex environments. Therefore, improved A3C which uses residual convolutional neural network which can helpful to draw complex relationship between set of tasks and hosts which improves acceleration of training to make appropriate decisions in scheduling environment. In this approach, initially all the data which is two dimensional folded is to be fed to actor critic network and it is flattened as one dimensional form and in turn which should be passed to hidden layer which is fully connected and hidden layer neurons are set to 256, kernel size is set to 2, step size is set as 1 .All the data passed through hidden layers and output of that network is connected to a SoftMax activation function to keep the range of values are between 0,1. In Asynchrous advantage actor critic each of the agent runs with different threads as it is a multi-threading network where each agent employs a thread independently and based on the outcome evaluated at each node subsequently submits outcome to a global network which gives the reward. When multi thread agents are running in parallel, training speed of algorithm improves as data given as state space into every actor network. In our research, for scheduling interval at time $\boldsymbol{T}$ is represented as $\boldsymbol{i^T}$, state space is represented as $\boldsymbol{s_T}$, action space is represented as $\boldsymbol{a_T}$. Next sequence of state space is represented as $\boldsymbol{s_{T+1}}$, after evaluation of input state sequences on it generates a reward which is represented as $\boldsymbol{Rew_T}$. The reward function should give either it give positive or negative results. Therefore, a policy μ should observe the results and guide it and adjust the reward to be maximized. The reward should be maximized and learned on its own by repetitive process of iterations in the model. It should be expressed as $< \boldsymbol{s}, \boldsymbol{a}, \mu, \mathbf{Rew}, \boldsymbol{val_{fn}} >$.

A. STATE SPACE

In the above tuple $\boldsymbol{s}$ represents state space which consists of set of states named as $\boldsymbol{s} = \{\boldsymbol{s_1}, \boldsymbol{s_2}, \ldots \boldsymbol{s_T}\}$which consists of different tasks . Assume $\boldsymbol{stat^T} = \{(\boldsymbol{ftin^T_{PM_{i1}}}, \boldsymbol{ftin^T_{t_{k1}}}\}$ in which $\boldsymbol{ftin^T_{PM_{i1}}}$ indicates feature information of Physical hosts . It is represented as a matrix. $\boldsymbol{ftin^T_{t_{k1}}}$ indicates feature information of tasks computed on physical machines which is also represented as a matrix.

B. ACTION SPACE

In the action space, we represent all the actions to be done for all the possible states which map tasks to the concerned virtual resources. It is represented as $\boldsymbol{a} = \{\boldsymbol{a_0}, \boldsymbol{a_1}, \boldsymbol{a_2} \ldots \boldsymbol{a_T}\}$. In $\boldsymbol{a_T} = \{\boldsymbol{d_{ij}}\}$ where $\boldsymbol{d_{ij}}$ is mapping action or decision variable in time interval $\boldsymbol{T}$. Entire task mapping process depends on decision variable.

C. POLICY

In the improved a3c approach, the policy μ have to control the results obtained from reward function to adjust the result in maximum optimized manner. It is represented using $\mu(\boldsymbol{a_T}|\boldsymbol{s_T})$ [66]. This is characterized as policy function by neural network which is represented as $\mu(\boldsymbol{a_T}|\boldsymbol{s_T};\pounds_{\boldsymbol{a}})$.

D. REWARD FUNCTION

This reward function is most important aspect in this reinforcement learning approach as it will give the outcome of mapping of tasks. It is indicated as $\boldsymbol{Rew_T}(\boldsymbol{s_T},\boldsymbol{a_T})$. It should be calculated using equation(17).

$$\boldsymbol{Rew_T} = \boldsymbol{min}\left(\boldsymbol{mp^{k1}}, \boldsymbol{Re^{cost}}\right), \boldsymbol{max}(\boldsymbol{load_{cpu_{i_1}}}) \quad (17)$$

Reward function should give an outcome and if reward is negative then it need to give cumulative discount reward and indicated as $\boldsymbol{g_T}$. It is calculated using equation (18).

$$\boldsymbol{g_T} = \boldsymbol{Rew_{T+1}} + ¥.\boldsymbol{Rew_{T+2}} + \cdots + ¥^{\boldsymbol{t-T-1}}.\boldsymbol{Rew_T} \quad (18)$$

E. VALUE FUNCTION

Value function represents expectation of state and action sequences performed by action, state spaces. State value function is represented as $\boldsymbol{V^{\mu}}(\boldsymbol{s_T})$ and it is calculated using equation(19). State-action value function is indicated as $\boldsymbol{q^{\mu}}(\boldsymbol{s_T},\boldsymbol{a_T})$ and it is calculated using equation(20). Expectation from these two functions is indicated using $\boldsymbol{expect}\{\}$.

$$\boldsymbol{V^{\mu}}(\boldsymbol{s_T}) = \boldsymbol{expect}_{\mu}[\boldsymbol{Rew_T} + ¥.\boldsymbol{Rew_{T+1}} + ¥^{\boldsymbol{t-T-1}}.\boldsymbol{Rew_t}|\boldsymbol{s_T}] \quad (19)$$

$$\boldsymbol{q^{\mu}}(\boldsymbol{s_T},\boldsymbol{a_T}) = \boldsymbol{expect}_{\mu}[\boldsymbol{g_T}|\boldsymbol{s_T},\boldsymbol{a_T}] = \boldsymbol{expect}_{\mu}[\boldsymbol{Rew_T} + ¥.\boldsymbol{q^{\mu}}(\boldsymbol{s_{T+1}},\boldsymbol{a_{T+1}}) + \cdots + |\boldsymbol{s_T},\boldsymbol{a_T}] \quad (20)$$

Value function can also be calculated using neural network with the help of network parameter $\boldsymbol{\theta_b}$.It is calculated using equation(21).

$$\boldsymbol{V^{\mu}}(\boldsymbol{s_T}) \approx \boldsymbol{V}(\boldsymbol{s_T};\boldsymbol{\theta_b}), \boldsymbol{q^{\mu}}(\boldsymbol{s_T},\boldsymbol{a_T}) \approx \boldsymbol{q}(\boldsymbol{s_T},\boldsymbol{a_T};\boldsymbol{\theta_b}) \quad (21)$$

F. TRAINING NETWORK

In this improved A3C approach, all threads runs simultaneously which consists of different agents and updates their decisions to global network. This process continuous until all iterations are completed and returns maximum reward value. Initially agents in multiple thread networks run with their sample data and observe the rewards and cumulative gradient will be collected and submitted to global network which checks the expected and actual values and guides each agent in runs in different threads to take a good decision for scheduling process. In the training process, for every iteration policy function to be given to the thread as $\mu(\boldsymbol{a_T}|\boldsymbol{s_T};\pounds_{\boldsymbol{a}}{}^{\sim})$, and value function as $\boldsymbol{q}(\boldsymbol{s_T},\boldsymbol{a_T};\boldsymbol{\theta_b}{}^{\sim})$. $\pounds_{\boldsymbol{a}}{}^{\sim}$ and $\boldsymbol{\theta_b}{}^{\sim}$ are control parameters in network and after every iteration for state $\boldsymbol{s_T}$, an action to be done with $\boldsymbol{a_T}$ and a reward should be generated as $\boldsymbol{Rew_T}$ which should be maximum. For every iteration, if the same policy function if applied there may be a chance of getting different gradient values. Therefore, gradient ascent method is used to get cumulative gradient and it is calculated using equation(22).

$$\boldsymbol{\nabla_{\pounds_{a}{}^{\sim}}}\boldsymbol{expect}_{\mu}[\boldsymbol{g_T}] = \boldsymbol{\nabla_{\pounds_{a}{}^{\sim}}}\boldsymbol{log}\,\mu(\boldsymbol{a_T}|\boldsymbol{s_T};\pounds_{\boldsymbol{a}}{}^{\sim})\boldsymbol{g_T} \quad (22)$$

After calculation of cumulative gradient using gradient ascent, there may be a chance that more action causes increase in gradient value. For every iteration, the probability of gradient value should be greater than equal to zero. Increase in gradient value slow down the learning rate. It should guide actor network to make optimized schedules but not to slow down the learning process. This is the reason A3C uses advantage function which is indicated as $\boldsymbol{adv}(\boldsymbol{s_T},\boldsymbol{a_T})$ improves calculation of gradient by subtracting from baseline function $\boldsymbol{base}(\boldsymbol{t})$. It helps algorithm to maintain unbiasedness in the process and helps it to converge in an efficient manner. It is calculated using equation(23).

$$\boldsymbol{d\pounds_a} = \boldsymbol{d\pounds_a} + \boldsymbol{\nabla_{\pounds_{a}{}^{\sim}}}\boldsymbol{log}\,\mu\boldsymbol{log}\,\mu(\boldsymbol{a_T}|\boldsymbol{s_T};\pounds_{\boldsymbol{a}}{}^{\sim})\boldsymbol{adv}(\boldsymbol{s_T},\boldsymbol{a_T})$$

$$= \boldsymbol{d\pounds_a} + \boldsymbol{\nabla_{\pounds_{a}{}^{\sim}}}\boldsymbol{log}\,\mu\boldsymbol{log}\,\mu(\boldsymbol{a_T}|\boldsymbol{s_T};\pounds_{\boldsymbol{a}}{}^{\sim})(\boldsymbol{q}(\boldsymbol{s_T},\boldsymbol{a_T};\boldsymbol{\theta_b}{}^{\sim}) - \boldsymbol{V}(\boldsymbol{s_T};\boldsymbol{\theta_b}{}^{\sim})) \quad (23)$$

In the process for every agent at state $\boldsymbol{s_T}$, calculates reward with $\boldsymbol{Rew_T}$ and value function is calculated as $\boldsymbol{V}(\boldsymbol{s_T};\boldsymbol{\theta_b}{}^{\sim})$, for the next state $\boldsymbol{s_{T+1}}$, value function is updated as $\boldsymbol{V}(\boldsymbol{s_{T+1}};\boldsymbol{\theta_b}{}^{\sim})$.

$$\boldsymbol{V}(\boldsymbol{s_T};\boldsymbol{\theta_b}{}^{\sim}) = \boldsymbol{V}(\boldsymbol{s_T};\boldsymbol{\theta_b}{}^{\sim}) + \boldsymbol{\beta}(\boldsymbol{Rew_T} + ¥.\boldsymbol{V}(\boldsymbol{s_{T+1}};\boldsymbol{\theta_b}{}^{\sim}) - \boldsymbol{V}(\boldsymbol{s_T};\boldsymbol{\theta_b}{}^{\sim})) \quad (24)$$

After calculation of value function updated for every iteration temporal check point error is calculated using eqn.23.

$$\boldsymbol{d\theta_b} = \frac{\boldsymbol{\delta}[\boldsymbol{Rew_T} + ¥\boldsymbol{V}(\boldsymbol{s_{T+1}};\boldsymbol{\theta_b}{}^{\sim}) - \boldsymbol{V}(\boldsymbol{s_T};\boldsymbol{\theta_b}{}^{\sim})]^{\boldsymbol{2}}}{\boldsymbol{\delta\theta_b}{}^{\sim}} \quad (25)$$

G. UPDATING PARAMETERS

This process have to be repeated by collecting data and map the tasks to suitable VMs by using improved A3C and to get maximum reward. After collecting all the gradients , it need to be submitted to global network by updating parameters. It is calculated using equation(26).

$$£_a = £_a + \beta . d£_a, \theta_b = \theta_b + \alpha\, d\theta_b \quad (26)$$

H. PROPOSED MULTI OBJECTIVE PRIORITIZED TASK SCHEDULER BY USING IMPROVED A3C

Input: Number of considered tasks $t_{k1} = \{t_1, t_2, t_3, \ldots t_{k1}\}$,Number of Considered VMs $v_{n1} = \{v_1, v_2, v_3 \ldots v_{n1}\}$, Number of Physical machines$\{PM_1, PM_2, PM_3 \ldots PM_{i1}\}$,Number of considered Datacenters $DC_{j1} = \{DC_1, DC_2, DC_3, \ldots . DC_{j1}\}$, $£_a, \theta_b, £_{a^{\sim}}, \theta_{b^{\sim}}, T, t$

Output: Scheduling decision $\mu^*(a_T|s_T; £_a)$

Initialize $£_a, \theta_b, t$ and set $t \leftarrow 1$.
Initialize $£_{a^{\sim}}, \theta_{b^{\sim}}$.
Calculate t_{k1}^{pri} using eqn.6.
Calculate v_{n1}^{pri} using eqn.7.
repeat
set gradient values $d£_a \leftarrow 0, d\theta_b \leftarrow 0$
set network specific parameters as $£_{a^{\sim}} \leftarrow £_a, \theta_{b^{\sim}} \leftarrow \theta_b$.
$T^{strt} = T$
repeat
input status information to state space as s_T, action space a_T. Apply policy $\mu(a_T|s_T; £_{a^{\sim}})$
Get reward as Rew_T and move to next state s_{T+1}
Increment global shared counter, step counter.
until $t - t^{strt} == t^{max}$ or $t == t^{end}$
Evaluate value function $V^{\mu}(s_T)$ using eqn.19.
for $i = t - 1, \ldots, t^{strt}$ do
calculate value function using eqn.24.
calculate $d\theta_b$ using eqn.25.
calculate $d£_a$ using eqn.23.
check the parameters mp^{k1}, Re^{cost}, $load_{cpu_{i_1}}$
end for
update $£_a$by $d£_a$, θ_b by $d\theta_b$.
until $T > T^{max}$
return $\mu^*(a_T|s_T; £_a)$

The below Fig.2. indicates flow of proposed MOPTSA3C. Initially, it starts with initialization of Global network and network specific actor-critic parameters. After initialization, priorities of task, VMs are evaluated using eqns.6,7. Input state space, action space values, apply the policy and observe the reward using value function using eqn.17. After observing reward, check how far the values of parameters are optimized and if they produce scheduling decisions according to the expectation in training update them as best optimized values and update global and local network parameters. If not calculate the accumulated or cumulative gradient value and suggest the better scheduling decision to the policy function we used in the approach. Repeat this process until all the iterations are completed.

V. SIMULATION AND RESULTS

This section discusses about Simulation and results of proposed MOPTSA3C(Multi Objective Prioritized Task scheduler using improved A3C) algorithm. Entire simulation of the proposed approach using Cloudsim toolkit. This proposed approach uses various data distributions of fabricated datasets represented as u01, n02, l03, r04 i.e. uniform, normal, left, right skewed distributions and realtime supercomputing worklogs which are represented as h05 for HPC2N , na06 for NASA respectively. Subsection A discusses Simulation and configuration settings, Subsection B discusses calculation of makespan using MOPTSA3C, Subsection C discusses calculation of Resource cost using MOPTSA3C, Subsection D discusses calculation of Resource utilization using MOPTSA3C, Subsection E discusses calculation of Reliability using MOPTSA3C, Subsection F discusses Analysis of results and discussion. Entire simulation ran for 100 iterations. Finally proposed approach evaluated over existing approaches DQN, MOABCQ, A2C algorithms for evaluating parameters makespan, resource cost, resource utilization.

A. SIMULATION SETTINGS USED IN MOPTSA3C

The below subsection discusses simulation and configuration settings used in proposed MOPTSA3C. This below Table III indicates simulation settings used in our simulation.

TABLE III
CONFIGURATION SETTINGS FOR SIMULATION

Entity	Quantity
Tasks	1000
VMs	100
Tasks length	900,000
Memory of PM	64GB
PM Bandwidth	1200 MBPS
Memory of VM	4 GB
Storage of PM	4TB
Storage of VM	64GB
VM Bandwidth	10 MBPS
PM Operating System	MAC
Datacenters	70

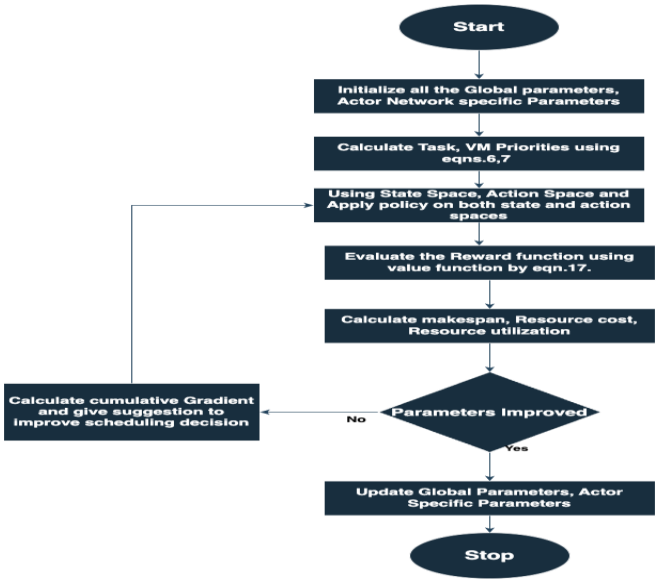

FIGURE 2. Flow of MOPTSA3C algorithm

The MOPTSA3C algorithm's total time complexity encompasses several components: first, the task priority calculation with a complexity of O(k1 log k1) for k1 tasks. Second, the computation of VM priorities among n1 VMs, taking O(n1 log n1) time. Third, establishing the mapping between k1 tasks and prioritized VMs, which demands O(k1 * n1) due to reward calculation. Finally, executing k1 tasks incurs a complexity of O(k1). In the context of the MOPTSA3C algorithm, the dominating factor for the total time complexity emerges from the mapping step (O(k1 * n1)), which significantly influences the algorithm's computational load. Consequently, while individual operations like task and VM priority calculations (O(k1 log k1) and O(n1 log n1) respectively) and task execution (O(k1)) are relevant, the algorithm's overall complexity predominantly aligns with the mapping process, specifically due to its dependence on both the number of tasks and virtual machines.

The below Table IV indicates parameter settings for MOPTSA3C which is used for training.

TABLE IV
PARAMETER SETTINGS FOR MOPTSA3C

Name	Value
Rate of Learning (β)	0.00001
Rate of Learning(α)	0.001
Decay factor(¥)	0.8
Activation Functions	SoftMax, RELU
Number of Threads	15
Global Shared counter T^{max}	2200
Local Thread Counter t^{local}	220

B. MAKESPAN EVALUATION BY MOPTSA3C

This subsection discusses evaluation of makespan for MOPTSA3C. The reason to evaluate makespan is that it directly affect scheduling process in cloud paradigm. An inefficient task scheduler increases makespan and thereby effects QoS of cloud service provider. This motivates us to evaluate makespan of MOPTSA3C scheduler in multi cloud environment by using different statistical distributions and realtime worklogs. The below Fig.3 and Table V represents evaluated makespan for MOPTSA3C using uniform distribution.

TABLE V
EVALUATION OF MAKESPAN USING UNIFORM DISTRIBUTION

Tasks(u01)	DQN	MOABCQ	A2C	MOPTSA3C
100	735.21	802.66	712.08	688.18
500	828.57	836.75	809.26	709.27
1000	912.35	926.77	887.12	723.38

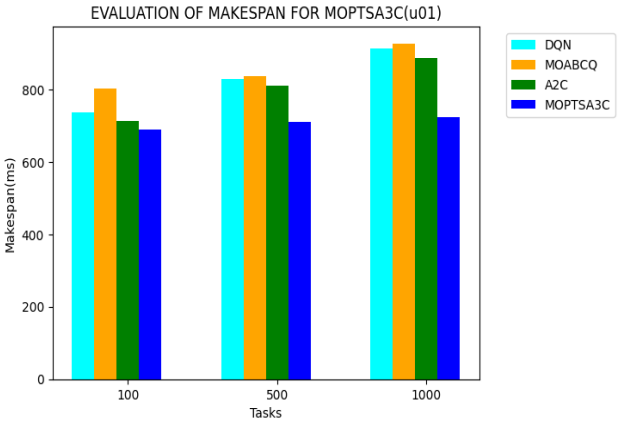

FIGURE 3. Evaluation of Makespan using u01

Initially our proposed MOPTSA3C evaluated over baseline approaches DQN, MOABCQ, A2C algorithms to check efficacy of MOPTSA3C in view of makespan. We considered 100-1000 tasks for evaluating makespan with fabricated uniform distribution of tasks(u01). Generated makespan for DQN for 100,500, 1000 tasks is 735.21, 828.57, 912.35 respectively. Generated makespan for MOABCQ with 100,500, 1000 tasks is 802.66, 836.75, 926.77 respectively. Makespan generated for A2C with 100,500,1000 tasks is 712.08, 809.26, 887.12 respectively. Makespan generated for MOPTSA3C with 100,500,1000 tasks is 688.18, 709.27, 723.38 respectively. From the above Fig.3 and Table V it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by minimizing makespan for uniform distribution of tasks.

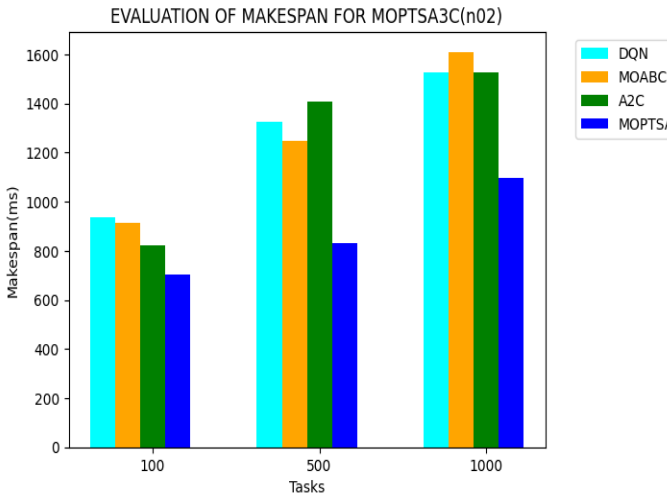

FIGURE 4. Evaluation of Makespan using n02

TABLE VI
EVALUATION OF MAKESPAN USING NORMAL DISTRIBUTION

Tasks(n02)	DQN	MOABCQ	A2C	MOPTSA3C
100	935.78	912.67	824.18	705.26
500	1326.77	1245.71	1408.36	832.11
1000	1524.17	1609.87	1527.15	1096.36

The above Table VI and Fig. 4 indicates evaluated makespan using normal distribution. Generated makespan for DQN for 100,500, 1000 tasks is 935.78, 1326.77, 1524.17 respectively. Generated makespan for MOABCQ with 100,500, 1000 tasks is 912.67, 1245.71, 1609.87 respectively. Makespan generated for A2C with 100,500,1000 tasks is 824.18, 1408.36, 1527.15 respectively. Makespan generated for MOPTSA3C with 100,500,1000 tasks is 705.26, 832.11, 1096.36 respectively. From the above Figure 4 and Table VI it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by minimizing makespan for Normal distribution of tasks.

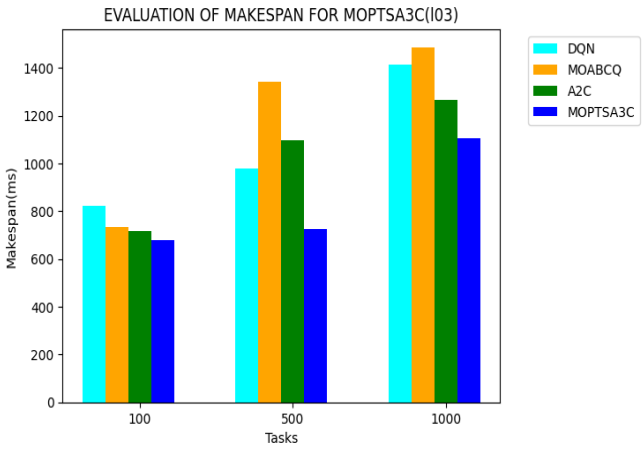

FIGURE 5. Evaluation of Makespan using l03

TABLE VII
EVALUATION OF MAKESPAN USING LEFT SKEWED DISTRIBUTION

Tasks(l03)	DQN	MOABCQ	A2C	MOPTSA3C
100	824.56	736.06	718.66	678.19
500	978.16	1343.22	1098.43	725.32
1000	1413.22	1487.35	1267.18	1104.36

The above Table VII and Fig. 5 indicates evaluated makespan using left skewed distribution. Generated makespan for DQN for 100,500, 1000 tasks is 824.56, 978.16, 1413.22 respectively. Generated makespan for MOABCQ with 100,500, 1000 tasks is 736.06, 1343.22, 1487.35 respectively. Makespan generated for A2C with 100,500,1000 tasks is 718.66, 1098.43, 1267.18 respectively. Makespan generated for MOPTSA3C with 100,500,1000 tasks is 678.19, 725.32, 1104.36 respectively. From the above Figure 5 and Table VII it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by minimizing makespan for left skewed distribution of tasks.

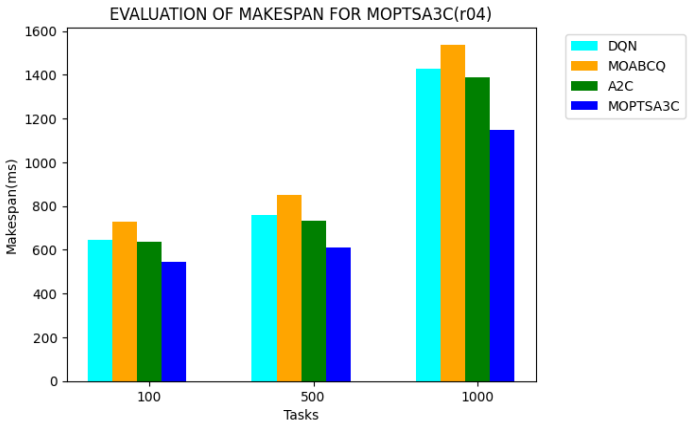

FIGURE 6. Evaluation of Makespan using r04

TABLE VIII
EVALUATION OF MAKESPAN USING RIGHT SKEWED DISTRIBUTION

Tasks(r04)	DQN	MOABCQ	A2C	MOPTSA3C
100	643.39	728.34	638.18	544.38
500	757.68	851.18	732.07	612.21
1000	1426.18	1538.17	1387.19	1146.09

Generated makespan for DQN for 100,500, 1000 tasks is 643.39, 757.68, 1426.18 respectively. Generated makespan for MOABCQ with 100,500, 1000 tasks is 728.34, 851.18, 1538.17 respectively. Makespan generated for A2C with 100,500,1000 tasks is 638.18, 732.07, 1387.19 respectively. Makespan generated for MOPTSA3C with 100,500,1000 tasks is 544.38, 612.21, 1146.09 respectively. From the above Fig.6 and Table VIII it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by minimizing makespan for right skewed distribution of tasks.

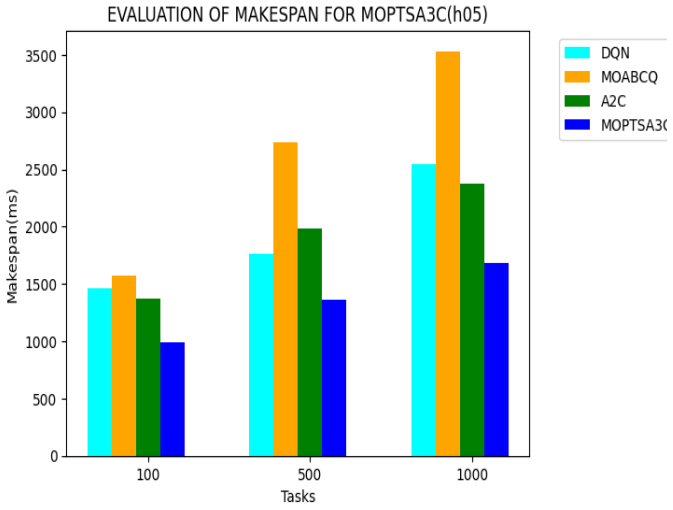

FIGURE 7. Evaluation of Makespan using h05

TABLE IX
EVALUATION OF MAKESPAN USING HPC2N WORKLOGS

Tasks(h05)	DQN	MOABCQ	A2C	MOPTSA3C
100	1464.73	1567.21	1372.17	989.45
500	1762.18	2732.44	1983.45	1357.81
1000	2542.17	3531.19	2372.09	1678.19

Generated makespan for DQN for 100,500, 1000 tasks is 1464.73, 1762.18, 2542.17 respectively. Generated makespan for MOABCQ with 100,500, 1000 tasks is 1567.21, 2732.44, 3531.19 respectively. Makespan generated for A2C with 100,500,1000 tasks is 1372.17, 1983.45, 2372.09 respectively. Makespan generated for MOPTSA3C with 100,500,1000 tasks is 989.45, 1357.81, 1678.19 respectively. From the above Fig.7 and Table IX it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by minimizing makespan for HPC2N worklogs.

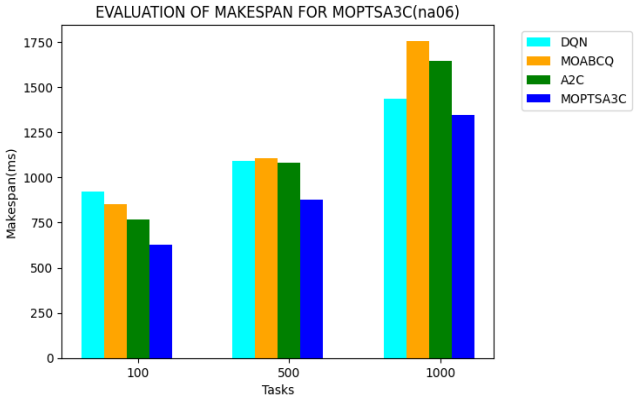

FIGURE 8. Evaluation of Makespan using na06

TABLE X
EVALUATION OF MAKESPAN USING NASA WORKLOGS

Tasks(na06)	DQN	MOABCQ	A2C	MOPTSA3C
100	924.14	853.07	765.64	627.09
500	1089.26	1107.26	1082.15	876.33
1000	1437.58	1756.93	1643.62	1347.22

Generated makespan for DQN for 100,500, 1000 tasks is 924.14, 1089.26, 1437.58 respectively. Generated makespan for MOABCQ with 100,500, 1000 tasks is 853.07, 1107.26, 1756.93 respectively. Makespan generated for A2C with 100,500,1000 tasks is 765.64, 1082.15, 1643.62 respectively. Makespan generated for MOPTSA3C with 100,500,1000 tasks is 627.09, 876.33, 1347.22 respectively. From the above Fig.8 and Table X it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by minimizing makespan for NASA worklogs.

C. *RESOURCE COST EVALUATION BY MOPTSA3C*

This subsection discusses clearly about evaluation of Resource cost using our proposed MOPTSA3C. The reason for evaluating resource cost in scheduling in multi cloud environment is an effective scheduler chooses precise VM to generate optimize schedules while effecting resource cost. Ineffective scheduling leads to increase in resource cost which causes a burden to CSP and as well as to cloud users. This motivates us to evaluate resource cost using MOPTSA3C in multi cloud environment. It is evaluated over existing baseline approaches DQN, MOABCQ, A2C algorithms using different statistical distributions and realtime worklogs. The below Fig.9 and Table XI shows evaluated resource cost using uniform distribution for MOPTSA3C.

TABLE XI
EVALUATION OF RESOURCE COST USING UNIFORM DISTRIBUTION

Tasks(u01)	DQN	MOABCQ	A2C	MOPTSA3C
100	5.27	6.12	4.98	4.41
500	7.08	7.26	5.87	5.25
1000	8.14	8.28	6.22	6.72

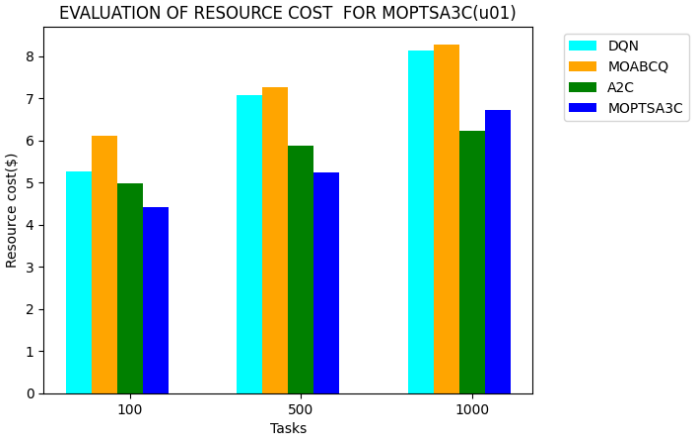

FIGURE 9. Evaluation of Resource Cost using u01

Generated Resource cost for DQN for 100,500, 1000 tasks is 5.27, 7.08, 8.14 respectively. Generated Resource cost for MOABCQ with 100,500, 1000 tasks is 6.12, 7.26, 8.28 respectively. Resource cost generated for A2C with 100,500,1000 tasks is 4.98, 5.87, 6.22 respectively. Resource cost generated for MOPTSA3C with 100,500,1000 tasks is

4.41, 5.25, 6.72 respectively. From the above Fig.9 and Table XI it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by minimizing resource cost for uniform distribution.

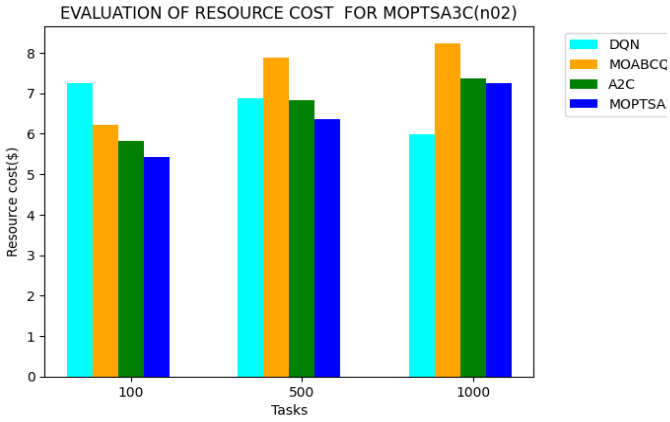

FIGURE 10.Evaluation of Resource Cost using n02

TABLE XII
EVALUATION OF RESOURCE COST USING NORMAL DISTRIBUTION

Tasks(n02)	DQN	MOABCQ	A2C	MOPTSA3C
100	7.26	6.23	5.83	5.43
500	6.87	7.88	6.84	6.37
1000	5.98	8.24	7.36	7.25

Generated Resource cost for DQN for 100,500, 1000 tasks is 7.26, 6.87, 5.98 respectively. Generated Resource cost for MOABCQ with 100,500, 1000 tasks is 6.23, 7.88, 8.24 respectively. Resource cost generated for A2C with 100,500,1000 tasks is 5.83, 6.84, 7.36 respectively. Resource cost generated for MOPTSA3C with 100,500,1000 tasks is 5.43, 6.37, 7.25 respectively. From the above Fig.10 and Table XII it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by minimizing resource cost for Normal distribution.

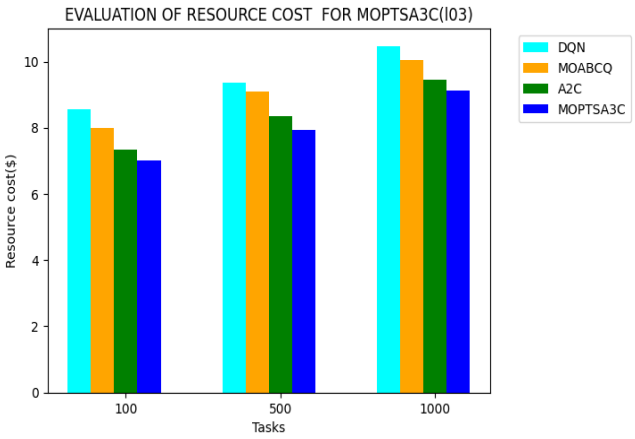

FIGURE 11.Evaluation of Resource Cost using l03

TABLE XIII
EVALUATION OF RESOURCE COST USING LEFT SKEWED DISTRIBUTION

Tasks(l03)	DQN	MOABCQ	A2C	MOPTSA3C
100	8.56	7.98	7.34	7.02
500	9.35	9.08	8.36	7.94
1000	10.47	10.06	9.46	9.12

Generated Resource cost for DQN for 100,500, 1000 tasks is 8.56, 9.35, 10.47 respectively. Generated Resource cost for MOABCQ with 100,500, 1000 tasks is 7.98, 9.08, 10.06 respectively. Resource cost generated for A2C with 100,500,1000 tasks is 7.34, 8.36, 9.46 respectively. Resource cost generated for MOPTSA3C with 100,500,1000 tasks is 7.02, 7.94, 9.12 respectively. From the above Fig.11 and Table XIII it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by minimizing resource cost for left skewed distribution.

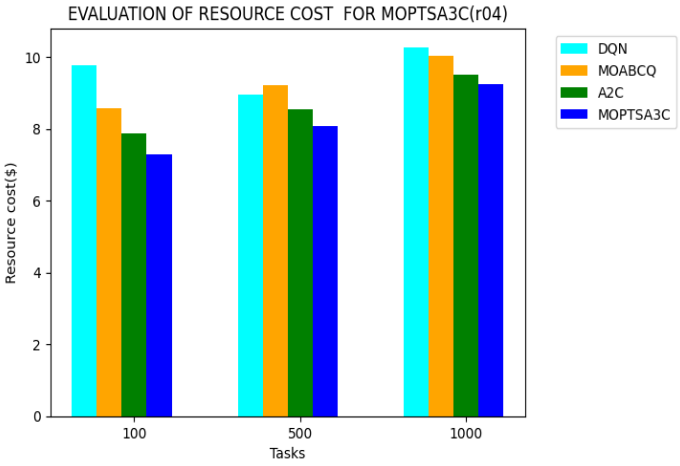

FIGURE 12.Evaluation of Resource cost using r04

TABLE XIV
EVALUATION OF RESOURCE COST USING RIGHT SKEWED DISTRIBUTION

Tasks(r04)	DQN	MOABCQ	A2C	MOPTSA3C
100	9.78	8.57	7.87	7.29
500	8.94	9.22	8.54	8.07
1000	10.27	10.02	9.51	9.23

Generated Resource cost for DQN for 100,500, 1000 tasks is 9.78, 8.94, 10.27 respectively. Generated Resource cost for MOABCQ with 100,500, 1000 tasks is 8.57, 9.22, 10.02 respectively. Resource cost generated for A2C with 100,500,1000 tasks is 7.87, 8.54, 9.51 respectively. Resource cost generated for MOPTSA3C with 100,500,1000 tasks is 7.29, 8.07, 9.23 respectively. From the above Fig.12 and Table XIV it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by minimizing resource cost for right skewed distribution.

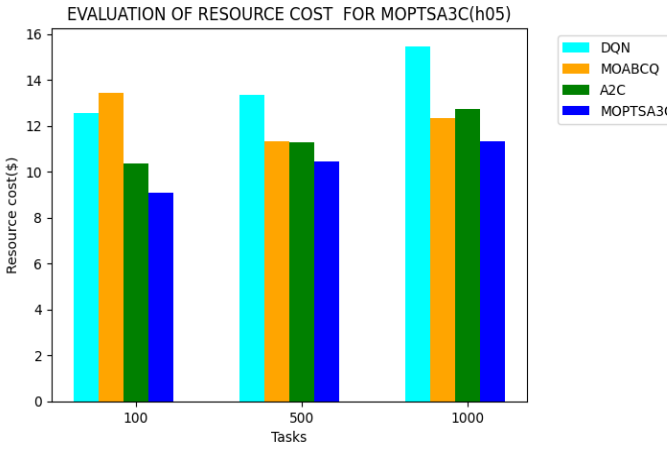

FIGURE 13.Evaluation of Resource cost using h05

TABLE XV
EVALUATION OF RESOURCE COST USING HPC2N WORKLOAD

Tasks(h05)	DQN	MOABCQ	A2C	MOPTSA3C
100	12.57	13.45	10.37	9.07
500	13.36	11.32	11.27	10.46
1000	15.47	12.35	12.74	11.33

Generated Resource cost for DQN for 100,500, 1000 tasks is 12.57, 13.36, 15.47 respectively. Generated Resource cost for MOABCQ with 100,500, 1000 tasks is 13.45, 11.32, 12.35 respectively. Resource cost generated for A2C with 100,500,1000 tasks is 10.37, 11.27, 12.74 respectively. Resource cost generated for MOPTSA3C with 100,500,1000 tasks is 9.07, 10.46, 11.33 respectively. From the above Fig.13 and Table XV it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by minimizing resource cost for HPC2N Workload.

TABLE XVI
EVALUATION OF RESOURCE COST USING NASA WORKLOAD

Tasks(na06)	DQN	MOABCQ	A2C	MOPTSA3C
100	14.22	12.86	10.26	9.57
500	12.87	11.53	11.08	10.09
1000	13.21	10.67	12.25	11.29

Generated Resource cost for DQN for 100,500, 1000 tasks is 14.22, 12.87, 13.21 respectively. Generated Resource cost for MOABCQ with 100,500, 1000 tasks is 12.86, 11.53, 10.67 respectively. Resource cost generated for A2C with 100,500,1000 tasks is 10.26, 11.08, 12.25 respectively. Resource cost generated for MOPTSA3C with 100,500,1000 tasks is 9.57, 10.09, 11.29 respectively. From the above Fig.14 and Table XVI it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by minimizing resource cost for HPC2N Workload.

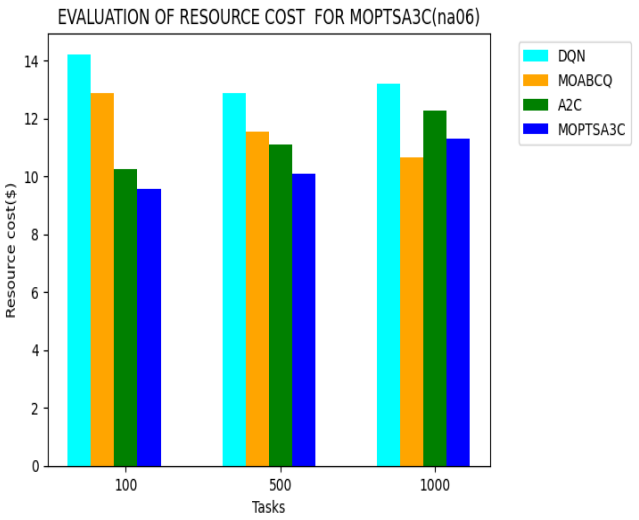

FIGURE 14.Evaluation of Resource cost using na06

D. RESOURCE UTILIZATION EVALUATION BY MOPTSA3C

This subsection discusses clearly about evaluation of Resource utilization using our proposed MOPTSA3C. The reason for evaluating resource utilization because improper assignment of tasks to VMs in cloud paradigm leads to over utilization or underutilization. It mainly effects CSP adversely which leads to high energy consumption and power cost. Therefore, in this proposed MOPTSA3C scheduler we evaluated utilization of resources over DQN, MOABCQ, A2C algorithms using different statistical distributions and realtime worklogs. The below Fig.15 and Table XVII shows evaluated resource utilization using uniform distribution for MOPTSA3C.

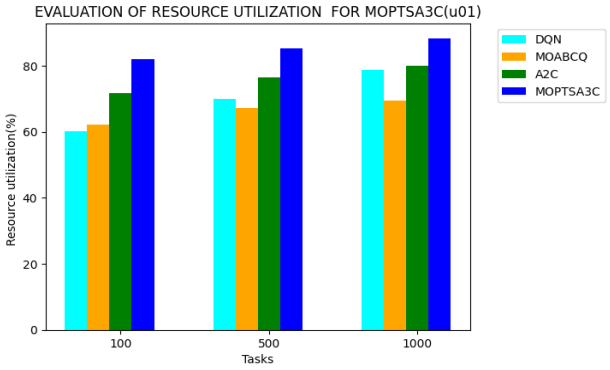

FIGURE 15.Evaluation of Resource utilization using u01

TABLE XVII
EVALUATION OF RESOURCE UTILIZATION USING UNIFORM DISTRIBUTION

Tasks(u01)	DQN	MOABCQ	A2C	MOPTSA3C
100	60.07	62.12	71.64	82.09
500	70.09	67.28	76.38	85.36
1000	78.74	69.44	80.12	88.47

Generated Resource utilization for DQN for 100,500, 1000 tasks is 60.07, 70.09, 78.74 respectively. Generated Resource utilization for MOABCQ with 100,500, 1000

tasks is 62.12, 67.28, 69.44 respectively. Resource utilization generated for A2C with 100,500,1000 tasks is 71.64, 76.38, 80.12 respectively. Resource utilization generated for MOPTSA3C with 100,500,1000 tasks is 82.09, 85.36, 88.47 respectively. From the above Fig.15 and Table XVII it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by improving resource utilization for uniform distribution.

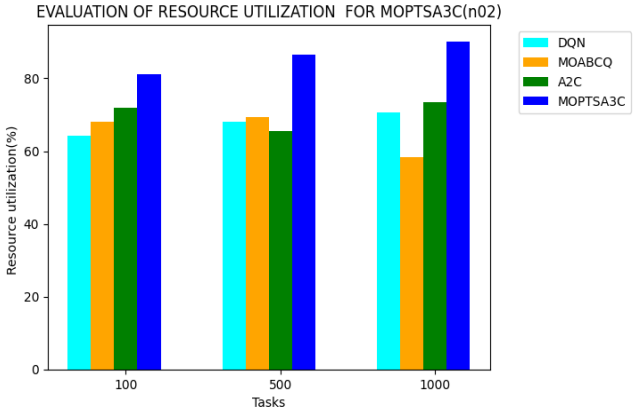

FIGURE 16.Evaluation of Resource utilization using n02

TABLE XVIII
EVALUATION OF RESOURCE UTILIZATION USING NORMAL DISTRIBUTION

Tasks(n02)	DQN	MOABCQ	A2C	MOPTSA3C
100	64.28	68.16	72.04	81.26
500	68.17	69.35	65.58	86.48
1000	70.62	58.22	73.44	90.18

Generated Resource utilization for DQN for 100,500, 1000 tasks is 64.28, 68.17, 70.62 respectively. Generated Resource utilization for MOABCQ with 100,500, 1000 tasks is 68.16, 69.35, 58.22 respectively. Resource utilization generated for A2C with 100,500,1000 tasks is 72.04, 65.58, 73.44 respectively. Resource utilization generated for MOPTSA3C with 100,500,1000 tasks is 81.26, 86.48, 90.18 respectively. From the above Fig.16 and Table XVIII it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by improving resource utilization for Normal distribution.

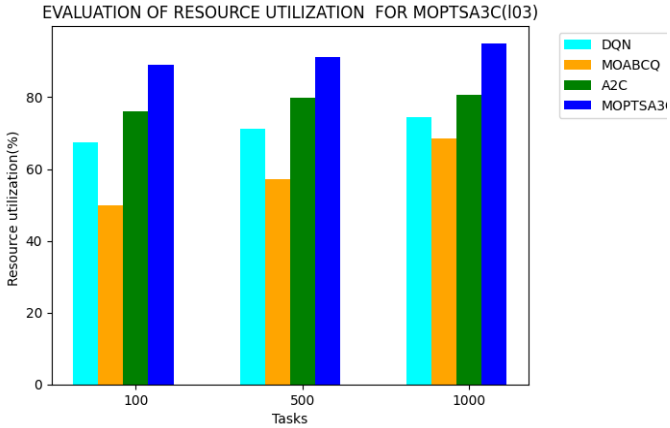

FIGURE 17.Evaluation of Resource utilization using l03

TABLE XIX
EVALUATION OF RESOURCE UTILIZATION USING LEFT SKEWED DISTRIBUTION

Tasks(l03)	DQN	MOABCQ	A2C	MOPTSA3C
100	67.35	49.77	76.09	88.98
500	71.28	57.26	79.78	91.26
1000	74.36	68.46	80.56	95.06

Generated Resource utilization for DQN for 100,500, 1000 tasks is 67.35, 71.28, 74.36 respectively. Generated Resource utilization for MOABCQ with 100,500, 1000 tasks is 49.77, 57.26, 68.46 respectively. Resource utilization generated for A2C with 100,500,1000 tasks is 76.09, 79.78, 80.56 respectively. Resource utilization generated for MOPTSA3C with 100,500,1000 tasks is 88.98, 91.26, 95.06 respectively. From the above Fig.17 and Table XIX it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by improving resource utilization for Left Skewed distribution.

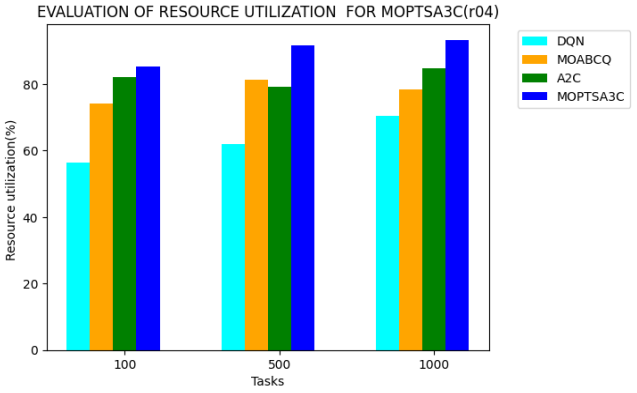

FIGURE 18.Evaluation of Resource utilization using r04

TABLE XX
EVALUATION OF RESOURCE UTILIZATION USING RIGHT SKEWED DISTRIBUTION

Tasks(r04)	DQN	MOABCQ	A2C	MOPTSA3C
100	56.37	74.03	82.17	85.27
500	62.02	81.26	79.09	91.64
1000	70.43	78.52	84.67	93.32

Generated Resource utilization for DQN for 100,500, 1000 tasks is 56.37, 62.02, 70.43 respectively. Generated Resource utilization for MOABCQ with 100,500, 1000 tasks is 74.03, 81.26, 78.52 respectively. Resource utilization generated for A2C with 100,500,1000 tasks is 82.17, 79.09, 84.67 respectively. Resource utilization generated for MOPTSA3C with 100,500,1000 tasks is 85.27, 91.64, 93.32 respectively. From the above Fig.18 and Table XX it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by improving resource utilization for Right Skewed distribution.

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3355092

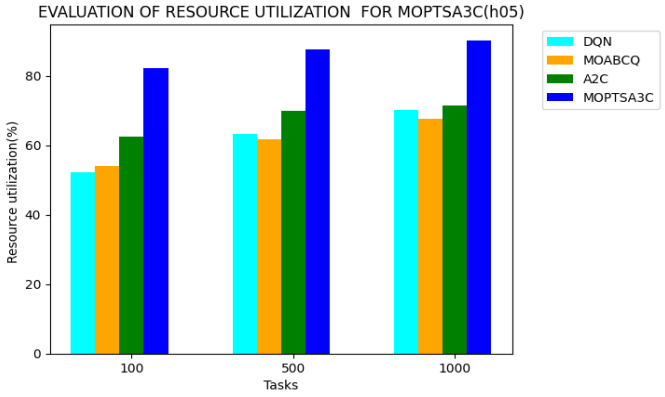

FIGURE 19.Evaluation of Resource utilization using h05

TABLE XXI
EVALUATION OF RESOURCE UTILIZATION USING HPC2N WORKLOAD

Tasks(h05)	DQN	MOABCQ	A2C	MOPTSA3C
100	52.26	54.06	62.40	82.32
500	63.18	61.76	69.98	87.71
1000	70.23	67.65	71.43	90.26

Generated Resource utilization for DQN for 100,500, 1000 tasks is 52.26, 63.18, 70.23 respectively. Generated Resource utilization for MOABCQ with 100,500, 1000 tasks is 54.06, 61.76, 67.65 respectively. Resource utilization generated for A2C with 100,500,1000 tasks is 62.4, 69.98, 71.43 respectively. Resource utilization generated for MOPTSA3C with 100,500,1000 tasks is 82.32, 87.71, 90.26 respectively. From the above Fig.19 and Table XXI it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by improving resource utilization for HPC2N Workload.

TABLE XXII
EVALUATION OF RESOURCE UTILIZATION USING NASA WORKLOAD

Tasks(na06)	DQN	MOABCQ	A2C	MOPTSA3C
100	49.58	58.43	74.62	88.45
500	62.97	67.48	79.85	90.36
1000	69.48	71.33	80.25	96.28

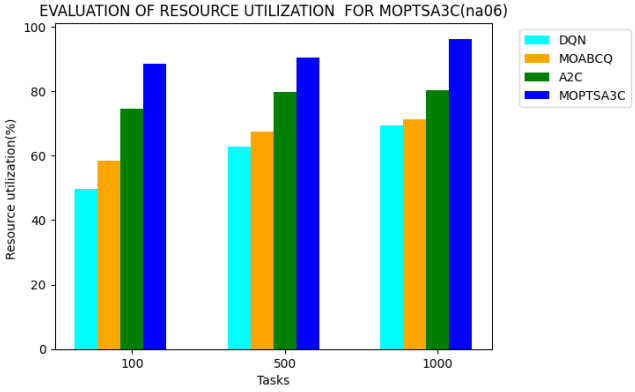

FIGURE 20.Evaluation of Resource utilization using na06

Generated Resource utilization for DQN for 100,500, 1000 tasks is 49.58, 62.97, 69.47 respectively. Generated Resource cost for MOABCQ with 100,500, 1000 tasks is 58.43, 67.48, 71.33 respectively. Resource utilization generated for A2C with 100,500,1000 tasks is 74.62, 79.85, 80.25 respectively. Resource utilization generated for MOPTSA3C with 100,500,1000 tasks is 88.45, 90.36, 96.28 respectively. From the above Fig.20 and Table XXII it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by improving resource utilization for NASA Workload.

E. RELIABILITY EVALUATION BY MOPTSA3C

This subsection discusses evaluation of Reliability of scheduler using MOPTSA3C. The main reason to evaluate Reliability of the scheduler is it will directly impacts QoS of Cloud Service Provider through which users are choosing the services of that vendor. Reliability directly depends on fault rate of system i.e. in this case for Scheduler, it will be depends on fault rate of tasks which are not executed properly in the model. With this reason we have calculated Reliability using MOPTSA3C. We ran simulation of MOPTSA3C with 100, 500, 1000 tasks. Proposed Scheduler is evaluated over existing DQN, MOABCQ, A2C algorithms with both fabricated workloads and realtime supercomputing worklogs.

Initially, we evaluated Reliability of MOPTSA3C using uniform workload distribution. Generated Reliability for DQN for 100,500, 1000 tasks is 0.2,0.15,0.23 respectively. Generated Reliability for MOABCQ with 100,500, 1000 tasks is 0.134, 0.08, 0.136 respectively. Reliability generated for A2C with 100,500,1000 tasks is 0.52,0.27,0.38 respectively. Resource cost generated for MOPTSA3C with 100,500,1000 tasks is 0.89, 0.91, 0.92 respectively. From the below Fig.21 and Table XXIII it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by improving reliability for uniform Workload.

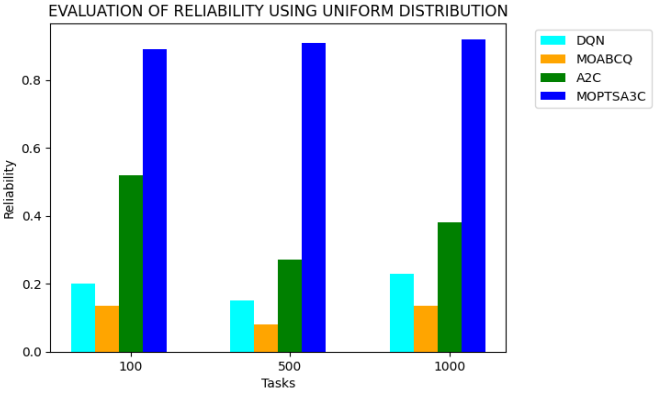

FIGURE 21.Evaluation of Reliability using u01.

TABLE XXIII
EVALUATION OF RELIABILITY USING UNIFORM DISTRIBUTION

Tasks(u01)	DQN	MOABCQ	A2C	MOPTSA3C
100	0.2	0.134	0.52	0.89
500	0.15	0.08	0.27	0.91
1000	0.23	0.136	0.38	0.92

Reliability of MOPTSA3C using Normal workload distribution is calculated below. Generated Reliability for DQN for 100,500, 1000 tasks is 0.52,0.36,0.49 respectively. Generated Reliability for MOABCQ with 100,500, 1000 tasks is 0.167, 0.135, 0.179 respectively. Reliability generated for A2C with 100,500,1000 tasks is 0.64, 0.48, 0.73 respectively. Resource cost generated for MOPTSA3C with 100,500,1000 tasks is 0.85, 0.96, 0.97 respectively. From the below Fig.22 and Table XXIV it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by improving reliability for Normal Workload.

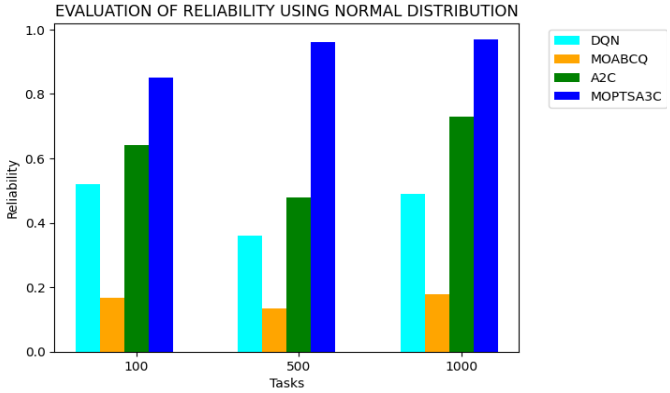

FIGURE 22.Evaluation of Reliability using n02.

TABLE XXIV
EVALUATION OF RELIABILITY USING NORMAL DISTRIBUTION

Tasks(n02)	DQN	MOABCQ	A2C	MOPTSA3C
100	0.52	0.167	0.64	0.85
500	0.36	0.135	0.48	0.96
1000	0.49	0.179	0.73	0.97

Reliability of MOPTSA3C using left skewed workload distribution is calculated below. Generated Reliability for DQN for 100,500, 1000 tasks is 0.35,0.78,0.21 respectively. Generated Reliability for MOABCQ with 100,500, 1000 tasks is 0.154, 0.178, 0.127 respectively. Reliability generated for A2C with 100,500,1000 tasks is 0.73, 0.56, 0.81 respectively. Resource cost generated for MOPTSA3C with 100,500,1000 tasks is 0.91, 0.94, 0.98 respectively. From the below Fig.23 and Table XXV it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by improving reliability for left skewed Workload.

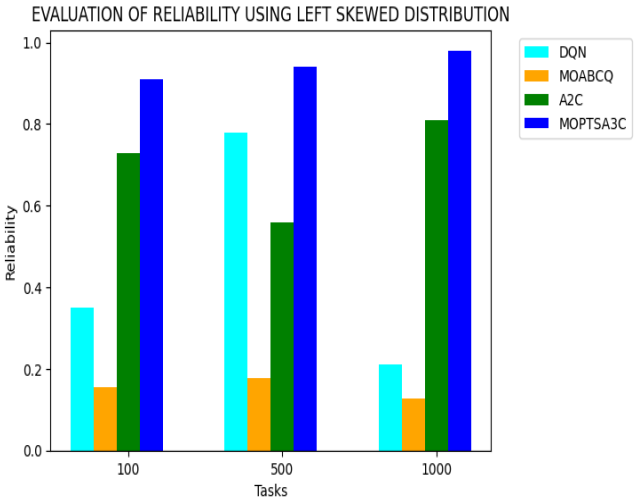

FIGURE 23.Evaluation of Reliability using l03.

TABLE XXV
EVALUATION OF RELIABILITY USING LEFT SKEWED DISTRIBUTION

Tasks(l03)	DQN	MOABCQ	A2C	MOPTSA3C
100	0.35	0.154	0.73	0.91
500	0.78	0.178	0.56	0.94
1000	0.21	0.127	0.81	0.98

Reliability of MOPTSA3C using right skewed workload distribution is calculated below. Generated Reliability for DQN for 100,500, 1000 tasks is 0.47, 0.81, 0.38 respectively. Generated Reliability for MOABCQ with 100,500, 1000 tasks is 0.127, 0.142, 0.156 respectively. Reliability generated for A2C with 100,500,1000 tasks is 0.68, 0.54, 0.73 respectively. Resource cost generated for MOPTSA3C with 100,500,1000 tasks is 0.913, 0.925, 0.978 respectively. From the below Fig.24 and Table XXVI it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by improving reliability for Right skewed Workload.

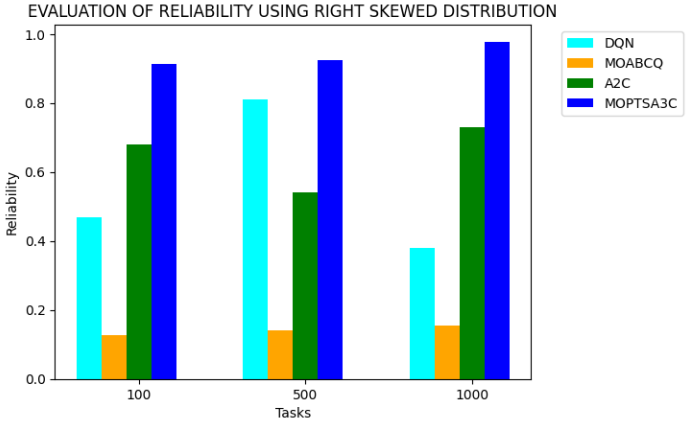

FIGURE 24.Evaluation of Reliability using r04.

TABLE XXVI
EVALUATION OF RELIABILITY USING RIGHT SKEWED DISTRIBUTION

Tasks(r04)	DQN	MOABCQ	A2C	MOPTSA3C
100	0.47	0.127	0.68	0.913

500	0.81	0.142	0.54	0.925
1000	0.38	0.156	0.73	0.978

Reliability of MOPTSA3C using parallel computing workload (HPC2N) is calculated below. Generated Reliability for DQN for 100,500, 1000 tasks is 0.43, 0.52, 0.73 respectively. Generated Reliability for MOABCQ with 100,500, 1000 tasks is 0.163, 0.157, 0.131 respectively. Reliability generated for A2C with 100,500,1000 tasks is 0.87, 0.78, 0.79 respectively. Resource cost generated for MOPTSA3C with 100,500,1000 tasks is 0.926, 0.941, 0.987 respectively. From the below Fig.25 and Table XXVII it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by improving reliability for HPC2N Workload.

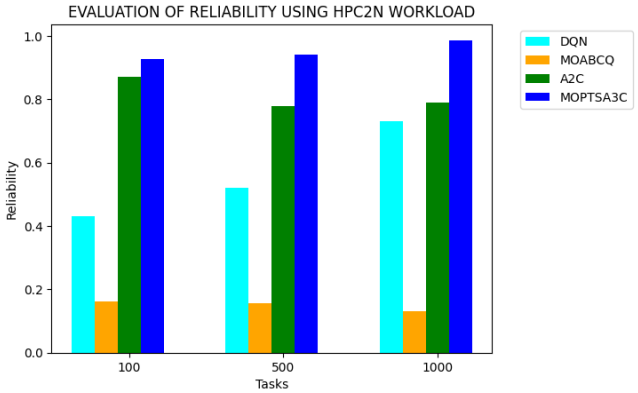

FIGURE 25.Evaluation of Reliability using h05.

TABLE XXVII
EVALUATION OF RELIABILITY USING HPC2N WORKLOAD

Tasks(h05)	DQN	MOABCQ	A2C	MOPTSA3C
100	0.43	0.163	0.87	0.926
500	0.52	0.157	0.78	0.941
1000	0.73	0.131	0.79	0.987

Reliability of MOPTSA3C using parallel computing workload (NASA) is calculated below. Generated Reliability for DQN for 100,500, 1000 tasks is 0.81, 0.59, 0.79 respectively. Generated Reliability for MOABCQ with 100,500, 1000 tasks is 0.168, 0.149, 0.182 respectively. Reliability generated for A2C with 100,500,1000 tasks is 0.88, 0.91, 0.93 respectively. Resource cost generated for MOPTSA3C with 100,500,1000 tasks is 0.946, 0.972, 0.99 respectively. From the below Fig.26 and Table XXVIII it is clearly shown that when tasks are increased from 100 to 1000 still MOPTSA3C learns the policies posed in scheduler and outperforms all existing approaches by improving reliability for HPC2N Workload.

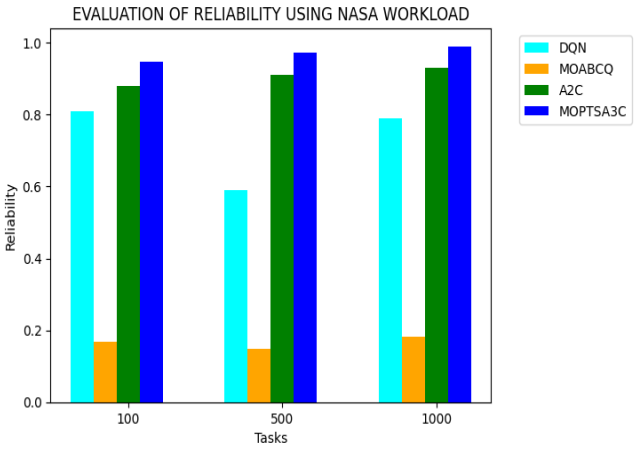

FIGURE 26.Evaluation of Reliability using na06.

TABLE XXVIII
EVALUATION OF RELIABILITY USING NASA WORKLOAD

Tasks(na06)	DQN	MOABCQ	A2C	MOPTSA3C
100	0.81	0.168	0.88	0.946
500	0.59	0.149	0.91	0.972
1000	0.79	0.182	0.93	0.99

F. ANALYSIS OF SIMULATION RESULTS

This subsection discusses about analysis of simulation results of MOPTSA3C. Extensive simulations are conducted on Cloudsim toolkit and evaluated proposed approach using state of art algorithms by DQN, MOABCQ, A2C algorithms with different fabricated workload distributions and HPC2N, NASA realtime worklogs. In the above results mentioned in subsections of V all the parameters evaluated are outperformed over existing approaches. In this subsection, detailed analysis performed in view of different parameters. The below Tables XXIX, XXX, XXXI indicates improvement of makespan, resource cost, resource utilization respectively for proposed MOPTSA3C approach over state of art algorithms.

TABLE XXIX
IMPROVEMENT OF MAKESPAN(%) OVER EXISTING ALGORITHMS

Tasks(u01)	DQN	MOABCQ	A2C
100	6.40%	14.26%	3.36%
500	14.40%	15.24%	12.36%
1000	20.71%	21.95%	18.46%
Tasks(n02)	**DQN**	**MOABCQ**	**A2C**
100	24.63%	22.73%	14.43%
500	37.28%	33.20%	40.92%
1000	28.07%	31.90%	28.21%
Tasks(l03)	**DQN**	**MOABCQ**	**A2C**
100	17.75%	7.86%	5.63%
500	25.85%	46.00%	33.97%
1000	21.86%	25.75%	12.85%
Tasks(r04)	**DQN**	**MOABCQ**	**A2C**
100	15.39%	25.26%	14.70%
500	19.20%	28.08%	16.37%
1000	19.64%	25.49%	17.38%
Tasks(h05)	**DQN**	**MOABCQ**	**A2C**
100	32.45%	36.87%	27.89%
500	22.95%	50.31%	31.54%
1000	33.99%	52.48%	29.25%

Tasks(na06)	DQN	MOABCQ	A2C
100	32.14%	26.49%	18.10%
500	19.55%	20.86%	19.02%
1000	6.29%	23.32%	18.03%

From the above Table XXIX, it is clearly observed that proposed MOPTSA3C approach clearly improved makespan over existing algorithms.

TABLE XXX
IMPROVEMENT OF RESOURCE COST (%) OVER EXISTING ALGORITHMS

Tasks(u01)	DQN	MOABCQ	A2C
100	16.32%	27.94%	11.45%
500	25.85%	27.69%	10.56%
1000	17.44%	18.84%	8.04%
Tasks(n02)	**DQN**	**MOABCQ**	**A2C**
100	25.21%	12.84%	6.86%
500	7.28%	19.16%	6.87%
1000	21.24%	12.01%	1.49%
Tasks(l03)	**DQN**	**MOABCQ**	**A2C**
100	17.99%	12.03%	4.36%
500	15.08%	12.56%	5.02%
1000	12.89%	9.34%	3.59%
Tasks(r04)	**DQN**	**MOABCQ**	**A2C**
100	25.46%	14.94%	7.37%
500	9.73%	12.47%	5.50%
1000	10.13%	7.88%	2.94%
Tasks(h05)	**DQN**	**MOABCQ**	**A2C**
100	27.84%	32.57%	12.54%
500	21.71%	7.60%	7.19%
1000	26.76%	8.26%	11.07%
Tasks(na06)	**DQN**	**MOABCQ**	**A2C**
100	32.70%	25.58%	6.73%
500	21.60%	12.49%	8.94%
1000	14.53%	5.81%	7.84%

From the above Table XXX, it is clearly observed that proposed MOPTSA3C approach clearly improved resource cost over existing algorithms.

TABLE XXXI
IMPROVEMENT OF RESOURCE UTILIZATION (%) OVER EXISTING ALGORITHMS

Tasks(u01)	DQN	MOABCQ	A2C
100	16.32%	27.94%	11.45%
500	25.85%	27.69%	10.56%
1000	17.44%	18.84%	8.04%
Tasks(n02)	**DQN**	**MOABCQ**	**A2C**
100	25.21%	12.84%	6.86%
500	7.28%	19.16%	6.87%
1000	21.24%	12.01%	1.49%
Tasks(l03)	**DQN**	**MOABCQ**	**A2C**
100	17.99%	12.03%	4.36%
500	15.08%	12.56%	5.02%
1000	12.89%	9.34%	3.59%
Tasks(r04)	**DQN**	**MOABCQ**	**A2C**
100	25.46%	14.94%	7.37%
500	9.73%	12.47%	5.50%
1000	10.13%	7.88%	2.94%
Tasks(h05)	**DQN**	**MOABCQ**	**A2C**
100	27.84%	32.57%	12.54%
500	21.71%	7.60%	7.19%
1000	26.76%	8.26%	11.07%
Tasks(na06)	**DQN**	**MOABCQ**	**A2C**
100	32.70%	25.58%	6.73%
500	21.60%	12.49%	8.94%
1000	14.53%	5.81%	7.84%

From the above Table XXXI, it is clearly observed that proposed MOPTSA3C approach clearly improved resource utilization over existing algorithms. From the above section in result analysis, we have observed that improved A3C in multi cloud environment learns features very fast even tasks are drastically increased or decreased. We evaluated MOPTSA3C with different fabricated statistical distributions and realtime workloads HPC2N, NASA.

TABLE XXXII
IMPROVEMENT OF RELIABILITY (%) OVER EXISTING ALGORITHMS

Tasks(u01)	DQN	MOABCQ	A2C
100	86.00%	54.18%	71.15%
500	58.67%	37.50%	46.04%
1000	85.00%	58.47%	58.11%
Tasks(n02)	**DQN**	**MOABCQ**	**A2C**
100	63.46%	58.98%	65.81%
500	66.67%	63.21%	82.11%
1000	97.96%	49.90%	74.1%
Tasks(l03)	**DQN**	**MOABCQ**	**A2C**
100	60.00%	49.91%	43.66%
500	28.51%	48.09%	61.86%
1000	36.67%	67.65%	68.99%
Tasks(r04)	**DQN**	**MOABCQ**	**A2C**
100	82.26%	68.90%	54.26%
500	37.54%	55.41%	70.30%
1000	57.37%	56.92%	68.97%
Tasks(h05)	**DQN**	**MOABCQ**	**A2C**
100	88.46%	48.10%	6.44%
500	81.96%	49.36%	20.64%
1000	57.21%	65.44%	24.94%
Tasks(na06)	**DQN**	**MOABCQ**	**A2C**
100	16.79%	43.10%	7.50%
500	64.75%	52.35%	6.81%
1000	25.32%	43.96%	6.45%

From the above Table XXXII, it is clearly observed that proposed MOPTSA3C approach clearly improved reliability over existing algorithms. From the above section in result analysis, we have observed that improved A3C in multi cloud environment learns features very fast even tasks are drastically increased or decreased. We evaluated MOPTSA3C with different fabricated statistical distributions and realtime workloads HPC2N, NASA. All the evaluated results of MOPTSA3C outperformed DQN, MOABCQ, A2C approaches in view of makespan, Resource cost, utilization of Resources, Reliability.

VI. CONCLUSION AND FUTURE WORK

Task scheduling problem (TSP) is a prodigious challenge in cloud computing due to variable incoming tasks comes up to cloud application console. It is an important concern for CSP to employ a dynamic and effective task scheduler which take care of suitability of tasks mapped to VMs in cloud environment. An ineffective task scheduler in cloud paradigm effects various parameters i.e. makespan, resource cost, resource utilization. Many existing authors used

metaheuristic approaches and developed task schedulers through which they got near optimal approximated scheduling decisions which may not always fit for all the conditions as it is a dynamic environment. Therefore, to tackle this situation, In this research, we used a reinforcement learning technique named as Improved Asynchronous Advantage Actor critic (A3C) algorithm to model MOPTSA3C scheduler in multicloud environment There are two phases in this scheduling approach.
In the First stage, all tasks coming to cloud application console are captured and their priorities are evaluated and VM priorities also evaluated based on unit electricity cost at datacenters. In the second stage, these priorities are fed to scheduler which integrated with Reinforcement learning model, generates scheduling decisions and generates reward based on multiple thread workers running on actor networks. After that critic network evaluates generated rewards and evaluate cumulative gradient based on applied policy in the network and guides it to move towards a better scheduling decisions according to the training given for the agent. Finally, we compared MOPTSA3C with existing state of art algorithms DQN, MOABCQ, A2C approaches by variating tasks from 100 t0 1000. In all the cases, MOPTSA3C minimizes makespan, resource cost, improved utilization of resources and reliability over existing approaches. In future, we are planning to deploy this scheduler in realtime cloud environment such as OpenStack to check the efficacy of the scheduler.

Acknowledgments: This work was supported by Research Supporting Project Number (RSP2024R421), King Saud University, Riyadh, Saudi Arabia.

REFERENCES

1. Sunyaev, Ali, and Ali Sunyaev. "Cloud computing." *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies* (2020): 195-236.
2. Mangalampalli, Sudheer, et al. "Cloud Computing and Virtualization." *Convergence of Cloud with AI for Big Data Analytics: Foundations and Innovation* (2023): 13-40.
3. W. Zheng et al., "A Deep Fusion Matching Network Semantic Reasoning Model," Applied Sciences, vol. 12, no. 7, p. 3416, 2022.
4. Fu, Xueliang, et al. "Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm." *Cluster Computing* 26.5 (2023): 2479-2488.
5. Pirozmand, Poria, et al. "An improved particle swarm optimization algorithm for task scheduling in cloud computing." *Journal of Ambient Intelligence and Humanized Computing* 14.4 (2023): 4313-4327.
6. Elcock, Jeffrey, and Nekiesha Edward. "An efficient ACO-based algorithm for task scheduling in heterogeneous multiprocessing environments." *Array* 17 (2023): 100280.
7. Mikram, Hind, Said El Kafhali, and Youssef Saadi. "HEPGA: a new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment." *Simulation Modelling Practice and Theory* (2023): 102864.
8. Mangalampalli, Sudheer, et al. "DRLBTSA: Deep reinforcement learning based task-scheduling algorithm in cloud computing." *Multimedia Tools and Applications* (2023): 1-29.
9. Kruekaew, Boonhatai, and Warangkhana Kimpan. "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning." *IEEE Access* 10 (2022): 17803-17818.
10. Bal, Prasanta Kumar, et al. "A joint resource allocation, security with efficient task scheduling in cloud computing using hybrid machine learning techniques." *Sensors* 22.3 (2022): 1242.
11. Alghamdi, Mohammed I. "Optimization of Load Balancing and Task Scheduling in Cloud Computing Environments Using Artificial Neural Networks-Based Binary Particle Swarm Optimization (BPSO)." *Sustainability* 14.19 (2022): 11982.
12. W. Zheng and L. Yin, "Characterization inference based on joint-optimization of multi-layer semantics and deep fusion matching network," PeerJ Computer Science, 2022.
13. Iftikhar, Sundas, et al. "HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments." Internet of Things 21 (2023): 100667.
14. Mangalampalli, Sudheer, et al. "Fault tolerant trust based task scheduler using Harris Hawks optimization and deep reinforcement learning in multi cloud environment." *Scientific Reports* 13.1 (2023): 19179.
15. Y. Mao et al., "New time-differenced carrier phase approach to GNSS/INS integration," GPS Solutions, vol. 26, no. 4, pp. 122, 2022.
16. Zeedan, Maha, Gamal Attiya, and Nawal El-Fishawy. "Enhanced hybrid multi-objective workflow scheduling approach based artificial bee colony in cloud computing." *Computing* 105.1 (2023): 217-247.
17. Sobhanayak, Srichandan. "MOHBA: multi-objective workflow scheduling in cloud computing using hybrid BAT algorithm." *Computing* (2023): 1-24.
18. Shirvani, Mirsaeid Hosseini. "An energy-efficient topology-aware virtual machine placement in Cloud Datacenters: A multi-objective discrete JAYA optimization." Sustainable Computing: Informatics and Systems 38 (2023): 100856.
19. Saravanan, G., et al. "Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing." Journal of Cloud Computing 12.1 (2023): 24.
20. Chandrashekar, Chirag, et al. "HWACOA scheduler: Hybrid weighted ant colony optimization algorithm for task scheduling in cloud computing." Applied Sciences 13.6 (2023): 3433.
21. Mangalampalli, Sudheer, Ganesh Reddy Karri, and Ahmed A. Elngar. "An Efficient Trust-Aware Task Scheduling Algorithm in Cloud Computing Using Firefly Optimization." *Sensors* 23.3 (2023): 1384.
22. G. Sun et al., "Live Migration for Multiple Correlated Virtual Machines in Cloud-Based Data Centers," IEEE Transactions on Services Computing, vol. 11, no. 2, pp. 279-291, 2018.
23. Xu, Heyang, et al. "Fault tolerance and quality of service aware virtual machine scheduling algorithm in cloud data centers." The Journal of Supercomputing 79.3 (2023): 2603-2625.
24. Peng, Zhihao, et al. "Genetic Algorithm-Based Task Scheduling in Cloud Computing Using MapReduce Framework." Mathematical Problems in Engineering 2022 (2022).
25. G. Sun et al., "Dynamic Network Function Provisioning to Enable Network in Box for Industrial Applications," IEEE Transactions on Industrial Informatics, vol. 17, no. 10, pp. 7155-7164, 2021.
26. Ni, Lina, et al. "GCWOAS2: multiobjective task scheduling strategy based on Gaussian cloud-whale optimization in cloud computing." Computational Intelligence and Neuroscience 2021 (2021): 1-17.
27. Jia, LiWei, Kun Li, and Xiaoming Shi. "Cloud computing task scheduling model based on improved whale optimization algorithm." Wireless Communications and Mobile Computing 2021 (2021): 1-13.
28. Qiuju, D. E. N. G., W. A. N. G. Ning, and L. U. Yang. "Cloud Task Scheduling using the Squirrel Search Algorithm and Improved Genetic Algorithm." International Journal of Advanced Computer Science and Applications 14.3 (2023).

29. Peng, Zhiping, et al. "Random task scheduling scheme based on reinforcement learning in cloud computing." Cluster computing 18 (2015): 1595-1607.
30. T. Li et al., "Understanding the Long-Term Evolution of Mobile App Usage," IEEE Transactions on Mobile Computing, vol. 22, no. 2, pp. 1213-1230, 2023,
31. Islam, Muhammed Tawfiqul, Shanika Karunasekera, and Rajkumar Buyya. "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 (2021): 1695-1710.
32. Rjoub, Gaith, et al. "Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems." *Concurrency and Computation: Practice and Experience* 33.23 (2021): e5919.
33. Yan, Jingchen, et al. "Energy-aware systems for real-time job scheduling in cloud data centers: A deep reinforcement learning approach." Computers and Electrical Engineering 99 (2022): 107688.
34. X. Xiao, J. Shu, H. Jiang, J. C. S. Lui, G. Min, J. Liu, et al., "Multi-Objective Parallel Task Offloading and Content Caching in D2D-aided MEC Networks," *IEEE Trans. Mobile Comput.*, (2022).
35. Tong, Zhao, et al. "DDQN-TS: A novel bi-objective intelligent scheduling algorithm in the cloud environment." Neurocomputing 455 (2021): 419-430.
36. Tong, Zhao, et al. "A scheduling scheme in the cloud computing environment using deep Q-learning." Information Sciences 512 (2020): 1170-1191.
37. Sharma, Mohan, and Ritu Garg. "An artificial neural network based approach for energy efficient task scheduling in cloud data centers." Sustainable Computing: Informatics and Systems 26 (2020): 100373.
38. Wang, Bin, Fagui Liu, and Weiwei Lin. "Energy-efficient VM scheduling based on deep reinforcement learning." Future Generation Computer Systems 125 (2021): 616-628.
39. Cheng, Feng, et al. "Cost-aware job scheduling for cloud instances using deep reinforcement learning." *Cluster Computing* (2022): 1-13.
40. P. Li, J. Hu, L. Qiu, Y. Zhao, and B. K. Ghosh, "A Distributed Economic Dispatch Strategy for Power–Water Networks," *IEEE Trans. Control Netw. Syst.*, vol. 9, no. 1, pp. 356-366, 2022.
41. Li, Huifang, et al. "Weighted double deep Q-network based reinforcement learning for bi-objective multi-workflow scheduling in the cloud." Cluster Computing (2022): 1-18.
42. Rostami, Safdar, Ali Broumandnia, and Ahmad Khademzadeh. "An energy-efficient task scheduling method for heterogeneous cloud computing systems using capuchin search and inverted ant colony optimization algorithm." *The Journal of Supercomputing* (2023): 1-37.
43. Shobeiri, Peyman, et al. "PCP–ACO: a hybrid deadline-constrained workflow scheduling algorithm for cloud environment." The Journal of Supercomputing (2023): 1-31.
44. Cheng, Yuqing, et al. "Multi objective dynamic task scheduling optimization algorithm based on deep reinforcement learning." The Journal of Supercomputing (2023): 1-29.
45. He, Hua, et al. "AMTS: Adaptive multi-objective task scheduling strategy in cloud computing." China Communications 13.4 (2016): 162-171.
46. Pang, Shanchen, et al. "An EDA-GA hybrid algorithm for multi-objective task scheduling in cloud computing." IEEE Access 7 (2019): 146379-146389.
47. Senthil Kumar, A. M., and M. Venkatesan. "Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment." *Wireless Personal Communications* 107 (2019): 1835-1848
48. Sathya Sofia, A., and P. GaneshKumar. "Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II." *Journal of Network and Systems Management* 26 (2018): 463-485.
49. J. Zhang, Y. Liu, Z. Li, and Y. Lu, "Forecast-Assisted Service Function Chain Dynamic Deployment for SDN/NFV-Enabled Cloud Management Systems," *IEEE Syst. J.*, 2023.
50. Srichandan, Sobhanayak, Turuk Ashok Kumar, and Sahoo Bibhudatta. "Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm." *Future Computing and Informatics Journal* 3.2 (2018): 210-230.
51. Zuo, Liyun, et al. "A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing." Ieee Access 3 (2015): 2687-2699.
52. Panda, Sanjaya K., and Prasanta K. Jana. "Efficient task scheduling algorithms for heterogeneous multi-cloud environment." The Journal of Supercomputing 71 (2015): 1505-1533.
53. Thein, Thandar, et al. "Reinforcement learning based methodology for energy-efficient resource allocation in cloud data centers." Journal of King Saud University-Computer and Information Sciences 32.10 (2020): 1127-1139.
54. Ding, Ding, et al. "Q-learning based dynamic task scheduling for energy-efficient cloud computing." *Future Generation Computer Systems* 108 (2020): 361-371.
55. Wang, Yuandou, et al. "multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning." *IEEE access* 7 (2019): 39974-39982.
56. Dong, Tingting, et al. "Deep reinforcement learning for fault-tolerant workflow scheduling in cloud environment." *Applied Intelligence* 53.9 (2023): 9916-9932.
57. Tong, Zhao, et al. "QL-HEFT: a novel machine learning scheduling scheme base on cloud computing environment." Neural Computing and Applications 32 (2020): 5553-5570.
58. Shukri, Sarah E., et al. "Enhanced multi-verse optimizer for task scheduling in cloud computing environments." *Expert Systems with Applications* 168 (2021): 114230.
59. X. Zhao, Y. Fang, H. Min, X. Wu, W. Wang, et al., "Potential sources of sensor data anomalies for autonomous vehicles: An overview from road vehicle safety perspective," *Expert Syst. Appl.*, vol. 236, p. 121358, 2024.
60. Wang, Yugui, Shizhong Dong, and Weibei Fan. "Task Scheduling Mechanism Based on Reinforcement Learning in Cloud Computing." *Mathematics* 11.15 (2023): 3364.
61. Uma, J., P. Vivekanandan, and S. Shankar. "Optimized intellectual resource scheduling using deep reinforcement Q-learning in cloud computing." Transactions on Emerging Telecommunications Technologies 33.5 (2022): e4463.
62. Siddesha, K., G. V. Jayaramaiah, and Chandrapal Singh. "A novel deep reinforcement learning scheme for task scheduling in cloud computing." Cluster Computing 25.6 (2022): 4171-4188.
63. Muniswamy, Saravanan, and Radhakrishnan Vignesh. "DSTS: A hybrid optimal and deep learning for dynamic scalable task scheduling on container cloud environment." *Journal of Cloud Computing* 11.1 (2022): 33.
64. Xiu, Xi, et al. "MRLCC: an adaptive cloud task scheduling method based on meta reinforcement learning." Journal of Cloud Computing 12.1 (2023): 1-12.
65. Praveen, S. Phani, et al. "A hybrid gravitational emulation local search-based algorithm for task scheduling in cloud computing." *Mathematical Problems in Engineering* 2023 (2023).
66. Praveen, S. Phani, K. Thirupathi Rao, and B. Janakiramaiah. "Effective allocation of resources and task scheduling in cloud environment using social group optimization." *Arabian Journal for Science and Engineering* 43 (2018): 4265-4272.
67. Murad, Saydul Akbar, et al. "SG-PBFS: Shortest Gap-Priority Based Fair Scheduling technique for job scheduling in cloud environment." *Future Generation Computer Systems* 150 (2024): 232-242.
68. Banerjee, Pallab, et al. "MTD-DHJS: Makespan-Optimized Task Scheduling Algorithm for Cloud Computing With Dynamic Computational Time Prediction." *IEEE Access* (2023).

69. Mnih, Volodymyr, et al. "Asynchronous methods for deep reinforcement learning." *International conference on machine learning*. PMLR, 2016.

Dr. S Sudheer Mangalampalli, Sudheer Mangalampalli working as an Assistant Professor senior Grade-1 in School of Computer Science and Engineering in VIT AP University. He is a Passionate researcher and his research interests include Cloud Computing, Edge Computing, Fog Computing and Machine Learning. Towards his profile, he is having 12 Publications, which are indexed in Scopus and SCI databases. He is currently guiding 3 Full time Ph.D. Scholar in VIT-AP university. He is a member of IEEE. He is a reviewer for various SCI indexed journals.

Ganesh Reddy Karri is working as an Associate Professor Senior Grade-2 in School of Computer Science and Engineering in VIT-AP University. He is a Passionate researcher and his research interests include Cloud Computing, Network Security, Fog Computing and Edge Computing. Towards his Profile he is having 15 Publications indexed in Scopus and SCI indexing. He is a certified trainer for Security Analyst profile by NAASCOM. He is currently guiding 6 Full time Ph.D. Scholars in VIT-AP university. He is a member of IEEE.

SACHI NANDAN MOHANTY (Senior Member, IEEE) received a Ph.D. degree from IIT Kharagpur, India, in 2015, with MHRD scholarship from the Government of India. He is a Post-Doctoral from IIT Kanpur in 2019. He has authored/edited 32 books, published by IEEE-Wiley, Springer, Wiley, CRC Press, NOVA, and DeGruyter. His research interests include data mining, big data analysis, cognitive science, fuzzy decision-making, brain-computer interface, cognition, and computational intelligence. He has received four Best Paper Awards during his Ph.D. at IIT Kharagpur, International Conference at Benjing, China, and the others at International Conference on Soft Computing Applications organized by IIT Rookee in 2013. He has awarded the Best Thesis award first prize by the Computer Society of India in 2015. He has guided nine Ph.D. Scholar. He has published 120 International Journals of International repute and he has been elected as a Fellow of the Institute of Engineers, European Alliance Innovation (EAI) Springer. He is a senior member of the Computer Society Hyderabad chapter. He is also the reviewer of the Journal of Robotics and Autonomous Systems (Elsevier), Computational and structural Biotechnology Journal (Elsevier), Artificial Intelligence Review (Springer), and Spatial Information Research (Springer).

SHAHID ALI, PhD degree from Peking university, in signal and information processing with the School of Electronics, Peking University, MS degree in Aerospace Engineering from Beijing Institute of Technology, Beijing, China, and the BS degree in Communication Engineering from University of Engineering and Technology Peshawar, KPK, Pakistan, in 2015. His research interests include channel estimation, MIMO, OFDM, and channel capacity in wireless communications.

MUHAMMAD IJAZ KHAN, received his Master, Mpill and PhD degree from Quaid-eAzam University Islamabad in 2014, 2016 and 2019 respectively. Worked as academic researcher in Quaid-i-Azam University. The author has contributed to research in topics: Nusselt number and Nanoflid. The author has an hindex of 36, co-authored 87 publications receiving 4704 citations**.**

Sherzod Abdullaev, Prof at Engineering School, Central Asian University, Tashkent, Uzbekistan, I am doing scientific work on methods of obtaining biofuel by processing household waste. I like to study mathematics and physics in depth. I enjoy reading art books and articles published in scientific journals in my spare time.

SALMAN A. ALQAHTANI, Salman A. AL Qahtani current research interests are in the area of 5G networks, broadband wireless communications, radio resource management for 4G and beyond networks (call admission control, packet scheduling and radio resource sharing techniques), cognitive and cooperative wireless networking, small cell and heterogeneous networks, self-organizing networks, smart grid, intelligent IoT solutions for smart cities, dynamic spectrum access, coexistence issues on heterogeneous networks in 5G, industry 4.0 issues, Internet of Everything and mobile cloud computing. In addition, his interests also include performance evaluation and analysis of high-speed packet switched networks, system model and simulations and integration of heterogeneous wireless networks. Mainly his focus is on the design and optimization of 5G MAC layers, closed-form mathematical performance analysis, energy-efficiency, and resource allocation and sharing strategies. He has authored two scientific books and authored/co-authored around 76 journal and conference papers in the topic of his research interests since 2004