

Received 10 July 2024, accepted 26 July 2024, date of publication 30 July 2024, date of current version 16 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3435914

RESEARCH ARTICLE

Efficient Hybrid DDPG Task Scheduler for HPC and HTC in Cloud Environment

S. SUDHEER MANGALAMPALLI¹, (Member, IEEE), GANESH REDDY KARRI², (Member, IEEE), SACHI NANDAN MOHANTY², (Senior Member, IEEE), SHAHID ALI³, ATIF M. ALAMRI⁴, AND SALMAN A. ALQAHTANI⁵

¹Department of Computer Science and Engineering, Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Manipal 576104, India

²School of Computer Science and Engineering, VIT-AP University, Amaravati 522237, India

³School of Electronics, Peking University, Beijing 100871, China

⁴Software Engineering Department, College of Computer and Information Sciences, King Saud University, Riyadh 11421, Saudi Arabia

⁵Computer Engineering Department, College of Computer and Information Sciences, King Saud University, Riyadh 11421, Saudi Arabia

Corresponding authors: Shahid Ali (alikh@pku.edu.cn) and S. Sudheer Mangalampalli (ms.sudheer@manipal.edu)


This work was supported by the King Saud University, Riyadh, Saudi Arabia, under Project RSP2024R421.

ABSTRACT Task Scheduling is a crucial challenge in cloud computing as diversified tasks come rapidly onto cloud console dynamically from heterogeneous resources which consists of different task lengths, processing capacities. Generating schedules for these type of tasks is a challenge for Cloud Service Provider(CSP). Therefore, to generate task schedules in cloud paradigm effectively by considering type of task arising to cloud console and match it with respective Virtual Machine (VM), a task scheduler is formulated by using Deep Deterministic Policy Gradient (DDPG) algorithm which is used as methodology to design scheduler. This scheduler works in three stages. In the initial stage, tasks are classified based on length and processing capacity to identify them whether they are High Performance Computing (HPC) tasks or High Throughput Computing (HTC) tasks. After classification, in the second stage, resources are to be tracked which matches the corresponding nature of tasks. Finally, in the third stage, according to the VM priorities calculated based on electricity unit cost and tasks are mapped according to the priorities to the corresponding VMs. Simulations are conducted using Cloudsim with fabricated workload distributions and realtime worklogs. Finally, our proposed Hybrid workload Deep Deterministic Policy Gradient Task scheduler(HDDPGTS) evaluated over DQN, A2C algorithms. From results, it proved that our proposed HDDPGTS significantly improved makespan, Energy consumption, scheduling overhead, scalability over baseline approaches.

INDEX TERMS Task scheduling, cloud computing, makespan, energy consumption, DQN, DDPG, A2C.

I. INTRODUCTION

Cloud Computing paradigm renders various on demand services (computing, network, storage) to its customers around the world seamlessly by using a simple application without having any on premises infrastructure at the customer end. All these services can be rendered to various customers around the world with help of virtualization technology which helps to auto scale the resources in cloud environment based on the demand of customers. Facilitating all these services on demand by Cloud Service Provider(CSP) with the help of task

The associate editor coordinating the review of this manuscript and approving it for publication was Xiong Luo .

scheduling algorithm employed by CSP. Handling enormous number of tasks and assign them to appropriate computation resource in cloud paradigm is a challenging task [1]. Task Scheduling Problem (TSP) is a critical issue to tackle in cloud computing model as variety of tasks from heterogeneous resources with diversified run time capacities are arriving at cloud application console. Therefore, to tackle these variety of tasks and effectively to be mapped onto appropriate VMs is challenge in this paradigm as it is NP-hard problem [2]. Ineffective task scheduling in other words improper mapping of tasks to VMs by not considering type of task, runtime leads to decrease in quality of service of CSP which can effect makespan, energy consumption. Earlier authors

proposed various schedulers using metaheuristic approaches and solved TSP in their own perception [3] but still there is a challenge in cloud paradigm as TSP is a NP-hard problem which cannot render solutions with in definite amount of time. Many earlier authors modeled task scheduling algorithms using nature inspired/metaheuristic approaches to optimize the scheduling process but still challenges persists in cloud paradigm as metaheuristic approaches are not able to render optimized solutions when heavy and diversified workloads comes to cloud platform [1] because of exploration of search space is not easy when huge workloads comes to cloud platform. Therefore, our research is started to formulate task scheduler using a deep reinforcement learning technique known as Deep Deterministic Policy Gradient algorithm(DDPG) which is a reinforcement learning technique. It is more adaptable, accelerable when compared with other approaches. Therefore, by using the above approach a task scheduler is formulated which initially classifies type of task in cloud paradigm. The main objective of our research is to classify HPC, HTC (hybrid) types of workloads and based on their type, task scheduler which is proposed by using DDPG need to schedule to the corresponding VMs based on priorities while minimizing parameters makespan, energy consumption, scheduling overhead, scalability, utilization of resources.

A. MOTIVATION AND CONTRIBUTIONS

In Cloud Computing, Task scheduling is a crucial challenge which was addressed by many earlier authors but still this research has certain potential problems to be solved while scheduling tasks to virtual resources in cloud datacenters. Main challenge in Task scheduling is assigning upcoming heterogeneous tasks to the precise VMs in cloud datacenters because it is highly dynamic in arrival patterns of tasks and there are higher chances that at all times same requests may not comes to cloud platform. Therefore, it is difficult for CSP to identify and schedule these tasks precisely to VMs in cloud datacenters. In the scheduling process, assigning a task to suitable resource in cloud platform is highly desirable as ineffective assignment of tasks to VMs without consideration of their type, length, run time capacities may lead to increase in execution time, overhead of scheduling in terms of memory, energy consumption. Therefore, these reasons motivated us take up this research to formulate a task scheduling technique which classifies hybrid workloads i.e. either HPC or HTC and then assign those tasks to VMs resided in prioritized datacenters with low electricity cost. Our proposed approach is modeled using Deep Deterministic policy gradient technique which can accelerate fast and learn and extract feature information from the posed data into the scheduler module to evaluate makespan, scheduling overhead, resource utilization, scalability, energy consumption.

Highlights of manuscript is presented below.

1. A hybrid workload aware Deep Deterministic policy gradient based Task scheduler is formulated in this manuscript.

2. Initially, in the first stage, a task classification mechanism to detect whether it is HPC or HTC traffic based on task length, processing capacity of tasks.
3. In second stage, resource manager tracks resources for the match of corresponding workload.
4. In the third stage, VM priorities calculated based on electricity cost. After calculating priorities, based on processing capacities of VMs, type of tasks are matched to those VMs using HDDPGTS.
5. Simulations conducted on Cloudsim with the input of fabricated workload and realtime supercomputing worklogs.
6. Finally, HDDPGTS evaluated over DQN, A2C to evaluate parameters makespan, energy consumption, scheduling overhead, scalability, utilization of resources.

Structure of the remaining manuscript is presented here. Section II presents related works, Section III presents Mathematical Modeling & System architecture of HDDPGTS, Section IV presents Methodology of Proposed HDDPGTS, Section V presents Simulation & Results, Section VI presents Conclusion & Future Works.

II. RELATED WORKS

In [1] author formulated scheduling model uses integer programming for minimization of task latency, acknowledging NP-hard nature of the scheduling problem, which makes exact optimization solution for larger instances. As a heuristic approach a customized Genetic Algorithm (GA) is introduced to achieve the objectives. The performance of this GA-based approach is evaluated in Cloudsim environment which accounts for dynamic nature of IoT environment. It was compared over baseline algorithms. In [2] author design the task scheduling approach edge computing which brings the cloud computing facilities closer to mobiles users, addressing needs of computation intensive, time sensitive tasks. The nature of edge networks characterized using user mobility, intermittent traffic, often leads to imbalanced resource allocation and performance issues. To overcome this challenge a novel deep q network algorithm is employed to handle complexity, high dimensionality of the scheduling problem. The simulation carried out the edge Cloudsim. The proposed method minimized service time, failed task rate. Task scheduling challenges [3] in fog environments due to inherent heterogeneity among fog devices. To address this issue, they proposed multi criteria decision making methods AHP, TOPSIS. The primary objective is model for mapping tasks to resource nodes in fog computing environments with focus on performance optimization. Authors in [3] studied various performance characteristics including memory, storage, latency, bandwidth, trust, cost factors are considered for tasks resources mapping. Two distant approaches are explored: one utilizing AHP for priority weight calculation, fog device ranking and the other employing AHP for priority weight determination followed by TOPSIS for fog devices raking. The simulation carried out the IFogsim to achieve the cost, latency. It was compared against baseline models RT-SANE, PORA. In [4] author formulated DFTLA task

scheduling approach. It leverages the variable structure learning automata to determine efficient assignment of tasks to fog nodes. It provides computing, storage, communication resources in proximity to network edge, enabling efficient processing of compute intensive tasks. Managing task execution for fog platforms poses significant challenges due to dynamic, loosely interconnected fog nodes, along with their failures. It ensures reliable task execution while optimizing response time, energy consumption. It's simulated in MATLAB 2017. The experimental results are compared against the baseline algorithms. Transmission of vast data volumes to cloud and subsequent responses can lead to significant latency and impose high demands on network bandwidth. The energy sources for fog computing servers presents a notable challenge for service providers in IIOT application, with task scheduling identified as a crucial challenge for energy consumption on fog servers. In [5] author address these challenges and formulated the task scheduling approach called HHOLS. It tackles discrete natured task scheduling. It additionally integrates swap mutation mechanism to improve solution quality for workload balance across VMs. It was simulated by Cloudsim. The proposed algorithm was compared against the FALS, GWOLA, BALS, PSOLS, MVALS and WALs achieved the makespan, cost, flow time, energy consumption.

In [6] author formulated task scheduling by establishing a quantitative energy aware model for load balancing, scheduling optimization in smart factories. To tackle complex energy consumption issues within manufacturing clusters author presents ELBS method. It begins by constructing energy model linked to workload on fog node. Experiments are conducted using a candy packing line. Results demonstrated that ELBS successfully achieves the optimal scheduling, load balancing for mixing work robots, showcasing its ability to address intricate energy consumption these results compared against conventional central station control scheduling. In [7] author introduces the utility computing where clients are charged based on their service consumption, posing challenges i.e. VM placement in data centers. This mapping the VMs to hosts to achieve objectives like load balancing, energy efficiency, resource utilization, response time optimization. To address these challenges author formulated ML based strategy for VM placement aimed at load balancing across host machines (HMs). In this approach, a learning agent selects actions in each learning episode and receives rewards based on the effectiveness of the chosen action. Through this iterative learning episodes, the agent updates its action value table, enhancing decision making for future actions and ultimately improving performance. The proposed algorithm was simulated by the cloudsim environmental and compared against the MOVMrB strategy across various scenarios demonstrates of our approach showcasing superior load balancing with reduced runtime and fewer active HMs. In [8] authors proposed workflow scheduling in fog-cloud environments with aim of achieving energy aware scheduling

within task completion times. Author proposed scheduling algorithm which operates in two stages. It assigns computational intensive tasks to cloud on the other hand tasks with low computation time are assigned to fog nodes. Author proposed Deadline aware frequency scaling approach for reducing energy consumption. It was simulated by the MATLAB and proposed model compared against the baseline algorithms EM-MOO, EM-MCC. In [9] author formulated a two-level scheduling model using fog nodes within same fog cluster, leveraging an improved variant NSGA-II that accounts for diversity among different devices. Simulations conducted using MATLAB validate the effectiveness of scheme in minimizing service latency, enhancing task execution stability. cloud computing offers seemingly limitless computational capabilities, it falls short in meeting the real-time demands of these applications. cloud computing lies in the inherent delays incurred during data transmission between edge IoT devices, centralized cloud data centers.

In [10] author address this challenge, fog computing has emerged, extending computational resources closer to edge devices to better fulfill requirements of time-sensitive applications. It proposes scheduling solution that leverages three-tier fog computing architecture to effectively manage a maximal number of requests while meeting their specified deadline constraints. It utilizes mixed integer programming to minimize instances of missed deadlines, with validation conducted through an exact solution method. Given NP-hard nature of scheduling problem inherent in fog computing environments, exact optimization techniques may prove inadequate for typical-sized problem instances. Hence, heuristic approach by employing genetic algorithm (GA). Performance of proposed GA is evaluated, compared against traditional RR, priority scheduling methodologies. It indicates a significant reduction in deadline misses, ranging from 20% to 55%, when utilizing the proposed approach compared to alternative techniques. It was simulated by the Cloudsim. In [11] author formulated a novel approach that combines GA with GELS to tackle task scheduling problem in cloud computing. To evaluate its effectiveness, it compared against traditional algorithms GA, PSO. It was simulated by the Cloudsim environmental and achieved minimizing makespan, maximizing resource utilization. Cloud computing provides users with a shared and customizable pool of computing resources accessible online, catering to their specific needs. Efficient job scheduling within the cloud is vital to uphold service quality and enhance overall system efficiency.

In [12] authors designed scheduling technique based on MPSO, aimed at addressing the challenges associated with lengthy scheduling times, high computation costs prevalent in cloud environment job scheduling. MPSO algorithm allocates jobs to VMs with objective of tackle cost, makespan. It integrates biological principles into the algorithm to mitigate premature convergence, enhance local search capability. It compared against baseline algorithm standard PSO. In [13] author formulated a workload-based approach for

executing interdependent tasks using heterogeneous computing resources. Primary objective of approach is to allocate tasks which minimizes total time required for completion. It builds upon max-min algorithm applied to DAG which represents tasks. It was simulated in Cloudsim and conducted tests in standard scientific workflows and the results indicates proposed approach significantly improved makespan, improvement in task allocation. In [14] author designed a task scheduling algorithm for increasing demand for higher processing power coupled with the exponential growth of data presents a significant challenge for efficient task scheduling within the cloud environment. It utilizes SSV, LSV, CSV to optimize task dispatching. Through experimentation, our results demonstrate that the BCSV algorithm achieves superior load balance and makespan compared to existing algorithms in heterogeneous network environments.

In [15] author formulated the task scheduling algorithm for multiprocessor task scheduling involves executing multiple tasks simulated in system crucial operation in fog-cloud multiprocessor. Author addresses these issues effectively and minimize energy consumption. It integrates genetic algorithm with an energy aware scheduling model. It was simulated by the MATLAB. Results are compared against the baseline algorithms GA, GSA, ACO, RR for scheduling efficiency. In [16] author designed scheduling mechanism with the use of cooperation search mechanism to solve dynamic nature of user service demand task scheduling. It allocate tasks efficiently to fastest executing processor. Effective task assignments, schedules significant impact system operation, especially in heterogeneous multiprocessor systems. It was simulated by the Cloudsim and results are compared against the baseline algorithms NGA, GA, WOA, GSA, HHG to tackle makespan, speedup, throughput. It optimizes allocation of tasks to VMs to minimize execution time and maximize efficiency.

Authors in [17] propose an ideal optimal task scheduling HWACO algorithm. It builds upon ACO Algorithm, offering enhancements in performance improvement. Through experimentation, the HWACO algorithm demonstrates superior performance compared to traditional algorithms ACO, QANA, MTF-BPSO, MIN-MIN, FCFS. Simulation analysis is done Cloudsim. In [18] author formulated a task scheduling mechanism for maximizing resource utilization, minimize execution time. Authors proposed a CNN-MBO for scheduling to maximize throughput, minimize makespan. Additionally, modified RSA employed to encrypt data for secure transmission. It was evaluated using a cloudlet simulator. Comparative analysis made with other task scheduling approaches to tackle response time, energy consumption. In [19] author introduces DAMPA for scheduling in cloud computing. Two sets of experiments conducted to evaluate DAMPA's performance. In the first case, DAMPA demonstrated significant reductions in both makespan, energy consumption, achieving reductions of up to 21.06% and 23.47%, respectively, compared the IMMPA, HHO, WOA,

MRFOSSA algorithm. In the second case, average reductions of 34.35% in makespan and 38.60% in energy consumption were observed. Additionally, DAMPA consistently achieved higher throughput in both case experiments.

In [20] author formulated task scheduling hybrid approach that combines PSO with GA. Initially, particle swarm is divided into sub-populations in each generation. These sub-populations undergo adjustments using a phagocytosis mechanism and crossover mutation from genetic algorithms, thereby broadening search space of solution and exploration. It was simulated by the Cloudsim and experimental results are compared against the baseline algorithm are PSO scheduling, GA scheduling, EIGA scheduling and proposed scheduling achieved minimizing the task completion time. In [21], author formulated Q – learning based task scheduling in cloud computing with centralized task dispatcher implements it which prioritizes to minimize response time, CPU utilization, energy consumption. Simulation carried with Cloudsim and results are compared against the two approaches MMS-O0Q and MMS-O1Q. In [22] author address the demand for efficient information processing and networking services continues to rise, the complexity of managing tasks dynamically. The complexity of this tasks classified as NP-hard challenges in achieving optimal balance across various factors of cloud computing. Author proposed a novel algorithm named DQTS. It integrates strengths of Q-learning, deep neural network to tackle task scheduling problem, particularly concerning DAG tasks in cloud computing environments. Experimental results are carried out the workflowsim compared against the baseline algorithms FCFS, MAXMIN, MINMIN and RR. The proposed model achieved to minimized the makespan and load balance. In [23] cloud computing, distributed and parallel computing challenge in task scheduling due to inherent complexity of cloud systems. To tackle this issue author formulated the two approaches first approach, Efficient K-means (Ekmeans), and the second, K-means HEFT (KmeanH), incorporating HEFT to minimize processing time, enhance efficiency for given set of tasks. Simulation carried out the MATLAB 2018b results are compared with variations of HEFT.

In [24] author formulated a task scheduling clipped double deep q learning algorithm to solve cloud datacenters is no longer feasible due to concerns such as bandwidth constraints, latency issues, cost implications and energy consumption. The proposed approach target network and experience relay to optimize task scheduling efficiency to minimizing cost, energy consumption. Simulation carried out with simpy. In [25] author formulated energy efficient scheduler in cloud data centers is crucial not only for cost reduction but also for aligning with green computing principles. Recent research efforts have been directed towards addressing challenges such as execution overhead and scalability in energy-efficient task scheduling. Author proposed a machine learning based ANN scheduler to minimizing the makespan, energy consumption, execution overhead.

Simulation was carried out the Cloudsim and MATLAB 2018b results are compared against the baseline algorithms GA, linear regression based efficient task schedulers. In [26] author formulated the task scheduler for task need to be scheduled effectively to ensure optimal resource utilization and meet performance requirements. Author proposed a framework integrates the six different scheduling approaches including MET, Suffrage, Min-mean, Min-var. Each of these algorithms is considered for its suitability in different scenarios and workload characteristics. To enable accurate prediction, employed four separate neural networks, each dedicated to predicting the best algorithm for a specific scheduling parameter. Additionally, PCA used to extract relevant features from dataset, enhancing prediction accuracy. It was minimizing the cost, throughput, makespan and degree of imbalance and its was simulated by the Cloudsim. Recognizing the criticality of reliability enhancement in dynamic environments like the cloud.

In [27] Author developed multi-agent scheduler powered by RL, NFQ algorithms to focus on efficiently managing user requests by taking queue buffer size each resource. Proposed framework aims to optimize resource allocation and scheduling decision thereby enhancing system reliability and service quality, makespan. It was simulated by the python. The results are compared against the baseline algorithms FIFS, greedy, random scheduling. In [28] author introduced QMTSF designed to dynamically allocate tasks to precise servers for cloud environment. Initially, tasks are allocated to precise servers, later an enhanced Q-learning employed on each server to assign tasks to VMs. In proposed framework, agent makes decision based on past experience, interactions with environment, learning from rewards to formulate an optimal task allocation while meeting task deadlines. Performance of proposed mechanism compared with PSO, random assignment, RR. In [29] author formulated the MOABCQ method is to tackle task scheduling, resource utilization while maximizing Virtual Machine (VM) throughput. It was achieved by considering parameters makespan, cost, resource utilization. Through performance evaluations conducted using Cloudsim. It was compared against FCFS, Q-learning, MOPSO, MOCS, across various datasets such as Random, Google Cloud Jobs, Synthetic workload. In [30] author formulated job scheduling problem specific to cloud-deployed Spark cluster, proposed an innovative RL model capable of accommodating SLA effectively. To implement this model, developed an RL-based cluster environment, deployed two DRL based schedulers using the TF-Agents. These DRL scheduling agents operate at fine-grained level, strategically allocating job executors while taking into considering pricing model of VM instances. It was simulated by the MATLAB 2018b reduce the cost efficiency performance improvement. Vehicular networks can also be used as an application of cloud paradigm and the importance of reputation, trustworthiness in vehicular networks when cloud paradigm is used for all transactions is mentioned in [51]. Authors in [51] formulated a privacy preserving

trustworthiness updation mechanism developed based on updates given by CSP to trusted authorities about the movement of vehicles and their interactions discussed. It was compared over existing approaches and results proved significant improvement by improving robustness and decreasing communication overhead.

From the Table 1, it was observed that many authors in developed different scheduling algorithms using metaheuristic, machine learning techniques while evaluating various parameters but still as TSP is a dynamic problem and a scheduler is needed which initially classifies the hybrid workload comes from heterogeneous devices onto cloud console and later calculating priorities of VMs based on electricity cost while giving these classified tasks and priorities to DDPG based scheduler to generate schedules while evaluating parameters named as energy consumption, utilization of resources, scheduling overhead, scalability, makespan.

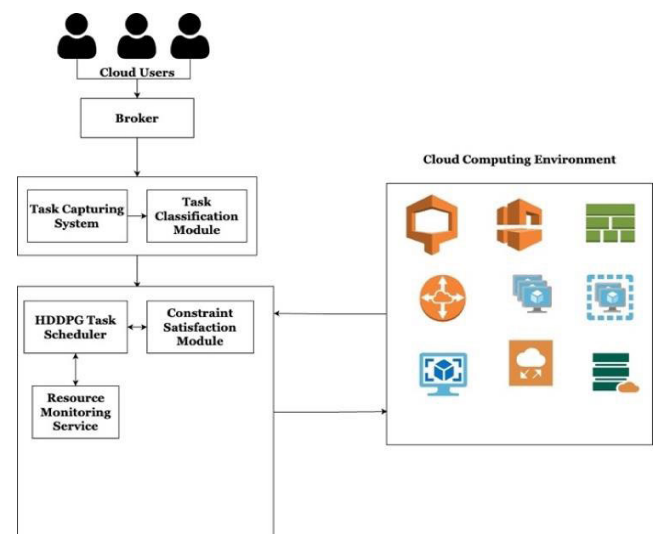


FIGURE 1. Proposed system architecture of HDDPGTS.

III. SYSTEM ARCHITECTURE AND MATHEMATICAL MODELLING OF HDDPGTS

This section discusses System Architecture of proposed Hybrid DDPG based Task Scheduler (HDDPGTS). Therefore, proposed HDDPGTS is formulated by considering set of tasks indicated as $t_{n1} = \{t_1, t_2, \dots, t_{n1}\}$ to be scheduled on to considered set of VMs $vm_{k1} = \{vm_1, vm_2, \dots, vm_{k1}\}$ which are placed in physical machine nodes Indicated as $hn_{i1} = \{hn_1, hn_2, \dots, hn_{i1}\}$ and all these physical machine nodes are placed in datacenters indicated as $dtc_{j1} = \{dtc_1, dtc_2, dtc_3 \dots, dtc_{j1}\}$. Problem formulation for task scheduler is done in such a way that considered number of t_{n1} tasks are to be precisely schedule HPC, HTC tasks to considered number of vm_{k1} VMs while evaluating above mentioned parameters.

In this architecture in fig.1, all incoming tasks submitted onto user cloud application and there is a broker which is a software agent on behalf of cloud service provider captures

TABLE 1. Earlier task, workflow scheduling algorithms proposed by various authors using ML techniques.

Authors	Technique	Parameters
[1]	Integer programming and GA	Average delay, processing power
[2]	Deep Q Network	service time , task rate
[3]	AHP Technique for order preference	Cost, latency, power consumption
[4]	Dynamic fault tolerant learning automata	Response time
[5]	HHOLS	Makespan, cost, flow time
[6]	ELBS	Energy consumption
[7]	RLVMrB	Reducing energy consumption, response time
[8]	Deadline aware stepwise frequency scaling	Makespan, energy consumption
[9]	(NSGA - II)	Service latency, task execution
[10]	Genetic algorithm	Deadline aware approach
[11]	Genetic algorithm and gravitational emulation local search	Makespan
[12]	PSO	Cost
[13]	Max-min	Makespan
[14]	BCSV	Resource utilization
[15]	HGAECS	Makespan
[16]	ECSA	Speedup, throughput
[17]	HWACO	Makespan and cost
[18]	CNN-MBO	Response time
[19]	DAMP Algorithm	Energy consumption
[20]	PSOGA phagocytosis	Task completion time
[21]	Q-learning based tasks scheduling	Task response time, CPU utilization
[22]	Deep Q – learning	Makespan, load balance
[23]	K-means	Makespan, speedup efficiency
[24]	Clipped double DQN	Cost, energy consumption
[25]	ANN – based scheduler	Makespan, execution overhead
[26]	PCA	Cost, throughput, makespan
[27]	Muti-agent NFQ	CPU utilization, reliability, makespan
[28]	QMTSF	Makespan, average completed time
[29]	MOTSABC	Makespan, cost, resource utilization
[30]	DRLTFS	Cost efficiency, performance improvements
[31]	HPSOGWO	QOS, task execution time, makespan, waiting time
[32]	Q-learning HEFT	Makespan, speed up efficiency
[33]	DRL	Makespan
[34]	DRL	Makespan, throughput.
[35]	Quantum Salp Swarm	Makespan, SLA Violation, throughput
[36]	Symbiotic Organism Search	Makespan, cost, hypervolume
[37]	MAPSO	Makespan, load balancing, stability, efficiency
[38]	ACO, PSO	Makespan, cost
[39]	Ordinal optimization	Makespan

TABLE 1. (Continued.) Earlier task, workflow scheduling algorithms proposed by various authors using ML techniques.

[40]	Fine-grained ACO	Task delay
[41]	DRL	Response time, power consumption
[42]	Multi-objective grey wolf optimizer	QoS, task delay
[43]	Multi agent system	Makespan, cost
[44]	Smart PSO	Makespan, cost
[45]	IDGA with penalty	Task execution time, Resource allocation.
[46]	Bi-directional gated recurrent unit	Job completion time
[47]	AIOSSA	Makespan, throughput
[48]	Markov decision process	Average end to end delay, load balancing
[49]	Marine predator algorithm	Makespan, energy consumption
[50]	PPRU	Robustness, Computation overhead
[56]	SG-PBFS	Makespan
[57]	MT-DHJS	Makespan
[58]	PBFS	Makespan, Flow time, overall delay
[59]	Resource management survey	Resource allocation, workload provisioning for IOT tasks
[60]	LBT	Surveyed Load balancing techniques
[61]	SDLS	Makespan
[62]	SWA	Makespan
[63]	LMWS-TM	QoS

these tasks, submit to task manager. Inside the task manager, a classification mechanism is induced to identify HPC, HTC tasks to precisely schedule onto appropriate VMs by calculating the priorities of VMs by considering electricity unit cost per unit. After classifying tasks whether they belong to HPC or HTC then according to their computing capacity scheduler need to send those tasks onto appropriate VMs by checking their priorities based on electricity unit cost of VMs at that datacenter. In other words, after classifying tasks in task manager, HPC tasks need to be scheduled onto VMs with high priority that means VMs with low electricity price at datacenters as these tasks need more computing capacity and our target is to consume less energy for these type of tasks and remaining tasks which are classified as HTC are scheduled onto VMs with priorities next to the HPC tasks based on their availability. This task scheduler modeled using Deep Deterministic Policy based Gradient(DDPG) which is a policy based Deep reinforcement learning technique induced into scheduler to generate optimized scheduling decisions by minimizing makespan, energy consumption while tracking the available resources from Resource manager which continuously in touch with Scheduler and VMs running in datacenters.

Deep Deterministic Policy Gradient (DDPG) used for modeling the task scheduler ensure that resource tracking is for efficient scheduling. The policy-based deep reinforcement learning DDPG reduces the makespan and power utilization

through training and improving its observation of the environment. These include the Resource Manager which controls the real time interactions with the Scheduler and the Virtual Machines (VMs) takes place in data centers. By constantly monitoring the resources' usage and availability, it gives the Scheduler all the relevant information that is needed for decision-making.

Due to dynamic tasks in cloud environment over resource demands and variability in resource availability, the Scheduler use adaptive scheduling mechanisms issued by the Resource Manager. Using real time data the Scheduler is also able to identify when a particular task requires more cloud resources as it reallocated task across the available VMs in order to ensure that the resources available are fully utilized without having to overload a particular VM. This continuous feedback loop and learning capability of the DDPG make it possible for the Scheduler to well control resource contention by timely providing the best VM allocation for tasks based on the current and future availability.

TABLE 2. Notations in modelling HDPGTS.

Notation	Meaning
t_{n1}^{iden}	Classification of disbursed $n1$ tasks
t_{n1}^{len}	Length of considered $n1$ tasks
t_{n1}^{mips}	$n1$ tasks runtime capacity
vm_{k1}^{prio}	Priorities of considered $k1$ VMs
$work_{vm_{k1}}^{load}$	Current workload on considered $k1$ VMs.
$work_{hn_{i1}}^{load}$	Current workload on considered hn_{i1} Physical host nodes
$proc_{k1}^{vm}$	Processing capacity of considered $k1$ VMs.
$exe_{t_{n1}}$	Execution time of considered $n1$ tasks.
$f_{t_{n1}}^{time}$	Finish time of considered $n1$ tasks
$ddl_{t_{n1}}$	Deadline of considered $n1$ tasks
$mkip_{n1}$	Makespan of considered $n1$ tasks
$ene_{vm_{k1}}^{cons}$	Energy consumption of considered $k1$ VMs
st_T	State space
$feainfo_T^{hn_{i1}}$	Feature information of considered physical host nodes
$feainfo_T^{t_{n1}}$	Feature information of considered tasks.
act_T	Action space for a time interval of T .
$\lambda(act_T st_T)$	Policy function used in DDPG
rew_T	Reward function

A. TASK CLASSIFICATION

Initially, as mentioned previously in the above architecture in Figure 1, after submitting tasks to broker task manager in which there are two components named as task capturing system in which all the submitted tasks captured by it and later there is a task classification module in which based on its length and runtime capacity of task it will be identified as

either HPC or HTC task based on the time interval duration. Initial task classification calculated using (1).

$$t_{n1}^{iden} = t_{n1}^{len} * t_{n1}^{mips} \quad (1)$$

From eqn.1. t_{n1}^{iden} represents either it is a HPC or HTC type of task. t_{n1}^{len} is Classification of disbursed $n1$ tasks, t_{n1}^{mips} is $n1$ tasks runtime capacity. It is calculated and identified with the help of runtime capacity of task, duration of task, length of task. If the runtime capacity of task t_{n1}^{mips} is high and even if the length of task t_{n1}^{len} is less it will be treated as HPC task by task classifier.

$$if(t_{n1}^{iden} > H) \text{ classify task as HPC}$$

If length of task t_{n1}^{len} is high but runtime capacity t_{n1}^{mips} of task is less then task execution time will be less and it will be treated as HTC task.

$$if(t_{n1}^{iden} < L) \text{ classify task as HTC}$$

We need dynamic threshold values to classify the HPC and HTC tasks over a period of time. For that, we use the statistical metrics to update the thresholds dynamically:

$$H = mean(t_{n1}^{iden}) + k * std_dev(t_{n1}^{iden}) \quad (2)$$

$$L = mean(t_{n1}^{iden}) - k * std_dev(t_{n1}^{iden}) \quad (3)$$

Here, k is a scaling factor that is tuned based on the specific requirements of the system.

This is an iterative process to identify the misclassified tasks and adjusting k value accordingly, and update the threshold value.

B. CALCULATION OF VM PRIORITIES

After classifying the task as either HPC or HTC, tasks are submitted to Task scheduler in which VM priorities calculated based on electricity cost at corresponding datacenters. VM priorities (vm_{k1}^{prio}) in each datacenter is calculated using (4).

$$vm_{k1}^{prio} = \frac{ele_{k1}^{high\ cost} dc_{j1} * ld_{CPU_{n1}}}{ele_{k1}^{cost} dc_{j1}} \quad (4)$$

$ele_{k1}^{high\ cost} dc_{j1}$ is highest electricity cost at $j1$ datacenters, $ele_{k1}^{cost} dc_{j1}$ is electricity cost of $j1$ datacenters. After calculating VM priorities, classified tasks which are of HPC and HTC are scheduled onto VMs in such a way that HPC tasks to be scheduled onto high prioritized VMs i.e. in other words, VMs with low electricity cost is considered as high prioritized VM and HPC tasks are scheduled onto those VMs. On the other side, HTC tasks as mentioned above which runs with less runtime capacity when it is compared with HPC tasks which have more run time capacities. Therefore, these tasks are to be scheduled onto VMs with low priority which are of with high electricity cost.

HPC tasks, characterized by high runtime capacity even if their length is short, are prioritized for VMs with lower

electricity costs. Thus, HTC tasks, which have a higher length but lower runtime capacity, are scheduled onto VMs with higher electricity costs. The VM priorities in each data center are calculated using the formula in the above mentioned (4). This approach ensures that HPC tasks are assigned to VMs with lower electricity costs, maximizing cost efficiency and minimizing energy consumption.

The algorithm's sensitivity to changes in electricity costs directly influences scheduling performance. As electricity costs change, the VM priorities are recalculated to reflect these changes, ensuring that the most cost-effective resources are always utilized. This dynamic adjustment allows the scheduler to adapt to real-time variations in electricity pricing, maintaining optimal resource allocation. However, significant volatility in electricity prices may require frequent recalculations, which could introduce overhead and impact the overall performance. Despite this, the ability to prioritize tasks based on current electricity costs ensures that the scheduling algorithm remains both cost-effective and energy-efficient, enhancing the overall performance and sustainability of the data center operations

C. CALCULATION OF WORKLOAD ON VMs AND PHYSICAL NODES

Scheduling these tasks onto appropriate prioritized VMs for a scheduler is a challenge as it need to know whether that VM is fully loaded and need to also now about its capacity. Therefore, before scheduling a task onto VM, current workload running on VMs calculated by (5).

$$work_{vm_{k1}}^{load} = \sum work_{vm_{k1}}^{load} \quad (5)$$

$work_{vm_{k1}}^{load}$ is current workload on considered $k1$ VMs. All these VMs considered in our research are resided in the physical machine nodes, therefore we calculated workload on physical machine nodes are calculated using (6).

$$work_{hn_{i1}}^{load} = \frac{work_{vm_{k1}}^{load}}{hn_{i1}} \quad (6)$$

$work_{hn_{i1}}^{load}$ is Current workload on considered hn_{i1} Physical host nodes.

D. CALCULATION OF PROCESSING CAPACITY OF VMs

After evaluating workloads on VMs, Physical Machine nodes our interested is to calculate processing capacities of all VMs as it is important to know processing capacity of a VM based on type of task to be scheduled and it is calculated using (7).

$$proc_{k1}^{vm} = proc^{no} * proc^{mips} \quad (7)$$

$proc_{k1}^{vm}$ is processing capacity of considered $k1$ VMs. VM priorities and type of tasks are fed to HDDPG scheduler in which based on type of task, run time processing capacity it will be assigned to VM with appropriate priority based on electricity cost and suitable processing capacity of a VM while minimizing energy consumption, makespan.

E. MAKESPAN MODEL

In this proposed scheduling algorithm, makespan, Energy Consumption considered as evaluation parameters. The main reason to choose makespan as parameter is any task scheduling algorithm in cloud computing is it directly effects performance of task scheduler as if execution time of a task is delayed or increased then it directly increases the value of makespan and quality of service of cloud provider can also be degraded. Therefore, initially execution time of tasks calculated using (8).

$$exe_{t_{n1}} = \frac{exe_t}{proc_{k1}^{vm}} \quad (8)$$

$exe_{t_{n1}}$ is execution time of considered $n1$ tasks. All tasks are pertained to have a finish time and as well as deadline. Note that, finish time of considered tasks should be less than deadline of tasks. All the considered $n1$ tasks should be less than or equal to deadlines of tasks. It is evaluated using (9) and (10).

$$ft_{t_{n1}}^{time} = \sum vm_{k1} + exe_{t_{n1}} \quad (9)$$

$$ft_{t_{n1}}^{time} \leq ddl_{t_{n1}} \quad (10)$$

$ft_{t_{n1}}^{time}$ is Finish time of considered $n1$ tasks, $ddl_{t_{n1}}$ Deadline of considered $n1$ tasks. After evaluation of execution time, finish time in eqns. 6 to 8, makespan is evaluated using (11), (12).

$$mkp_{n1} = \min(ft_{t_{n1}vm_{k1}}^{time}) \quad (11)$$

$$\min(ft_{t_{n1}vm_{k1}}^{time}) = \sum_{i=1, j=1}^{n1, k1} \partial_{i,j}(ft_{t_{n1}vm_{k1}}^{time}) \quad (12)$$

mkp_{n1} is Makespan of considered $n1$ tasks. The aim of makespan evaluation is to minimize execution time to improve makespan of considered $n1$ tasks. In this research, a deadline constraint is induced to make sure that all considered tasks to finish their execution before its posed deadline constraint. From eqn.11, $\partial_{i,j}$ parameter is induced when task t_{n1} is assigned to VM vm_{n1} it is to be represented as 1 or otherwise it is set to 0. After careful evaluation of makespan, another important parameter identified is energy consumption.

F. ENERGY CONSUMPTION MODEL

The reason to choose energy consumption as an evaluation parameter is most of the cloud service providers incurs high energy consumption with ineffective scheduling process adopted in cloud environment. In other words, task scheduling is effective only when tasks are classified based on their processing capacity assigned to the matched VMs which can be suitable for corresponding tasks. It is calculated using (13).

$$vm_{k1} = \begin{cases} active & \alpha_{k1} \\ idle & \sigma_{k1} \end{cases} \quad (13)$$

$$ene_{vm_{k1}}^{cons} = ft_{t_{n1}}^{time} * \alpha_{k1} + (mkp_{n1} - ft_{t_{n1}}^{time}) * \sigma_{k1} \quad (14)$$

$$ene_{cons}^{act} = (ene_{max} - ene_{min}) * re^{util} + ene_{min} \quad (15)$$

$$Totene_{cons} = \sum ene_{vm_{k1}}^{cons} + ene_{cons}^{act} \quad (16)$$

$Totene_{cons_{k1}}$ is Energy consumption of considered $k1$ VMs.

G. RESOURCE UTILIZATION MODEL

It discusses mathematical formulation of resource utilization as an effective scheduling mechanism directly impacts utilization of resources. Therefore, utilization of resources is considered as another evaluating parameter. Generally, in cloud paradigm, a resource can be a CPU, I/O, bandwidth. In this research, our considered resource is CPU. Therefore, running load on CPUs of all considered $n1$ physical host nodes are calculated by (17).

$$ld_{CPU_{n1}} = \sum_{i1=1}^{s1} \frac{usg(i1)}{CPU_{n1}} \quad (17)$$

where $s1$ indicates active tasks on physical host node, $usg(i1)$ indicates usage of $i1$ CPU among $n1$ physical host nodes.

H. SCHEDULING OVERHEAD MODEL

After evaluation of utilization of resources, scheduling overhead is discussed as for effective utilization of resources, there is a certain importance of addressing overhead while scheduling process in this paradigm. It is mathematically formulated by (18).

$$sch_{od} = \sum_{i=1}^{n1} t_{n1} * mem[t_{n1}vm_{k1}] \quad (18)$$

I. SCALABILITY EFFICIENCY MODEL

The importance of scalability in cloud paradigm is how task scheduler efficiently handles sudden increase of workloads. It is formulated using (19), (20).

$$ef_{scale} = \frac{time_{ideal}(n1)}{time_{actual}(n1, k1)} * 100 \quad (19)$$

where $time_{ideal}(n1)$ is ideal time to execute task on a single VM to the number of VMs while assuming tasks are divided fairly.

$$time_{ideal}(n1) = \frac{time_{single}(n1)}{k1} \quad (20)$$

IV. METHODOLOGY OF PROPOSED HDDPGTS

This section discusses methodology used for proposed Hybrid workload Deep Deterministic policy gradient based task scheduler(HDDPGTS). DDPG [50] is a policy gradient based algorithm which adaptively learns policies by using actor network and evaluates it through critic networks i.e. on target evaluation networks. This gradient policy based algorithm is essentially combined with Deep Q- Network which increases adaptability with dynamic workload in the cloud environment. It is basically based on actor critic framework which performs actions based on states through which it gets input using gradient policy. On the other side Critic network evaluates reward generated by action space for every iteration and it continuously tracks the reward generated and reward expected in every iteration. It consists of four networks in this approach 1. Policy network which guides action space to take considerable action based on input state spaces 2. Evaluation network which observes generated action and gives a reward. 3. Target policy network which sets target

reward to be generated by action space and 4. Target evaluation network is which need to evaluate the generated reward whether it is inclined to the reward set by the target policy network.

A. STATE SPACE

This subsection discusses about state space which is an input sequence of tasks given at a specific time interval T and they are represented as $st_T = \{st_1, st_2, \dots, st_T\}$ and assumed that $st_T = \{feainf o_T^{hn_{i1}}, feainf o_T^{n1}\}$. Where $feainf o_T^{hn_{i1}}$ indicates feature information about physical host nodes considered in our research, $feainf o_T^{n1}$ indicates feature information about tasks considered in research.

B. ACTION SPACE

This subsection discusses about action space which is an action to be performed with the input sequence generated by state space. In this research, it is to be considered as mapping action of tasks to VMs with in a time interval of T . It is represented as $act = \{act_0, act_1, act_2, \dots, act_T\}$ where $act_T = \{dec_{n1k1}\}$. It means that mapping procedure will be depends on decision variable for a time interval T mentioned as above.

C. POLICY

This subsection discusses policy to be used in DDPG which guides action space to map set of tasks in state space to appropriate VMs. This is to be indicated as $\lambda(act_T|st_T)$.

D. REWARD FUNCTION

After mapping tasks to VMs by action space with the help of policy, an outcome will be generated called as reward which is either a positive or negative reward. In this DDPG approach, it is to be evaluated by target evaluation network, target policy network but not to be submitted to the global network as in the conventional actor critic algorithms. The policy gradient applied at the time of mapping procedure to check whether the mapping is inclined to target policy and target evaluation networks. Reward function in this research indicated by using (21).

$$rew_T = \min(Tot_{enecons}, mkp_{n1}) \quad (21)$$

Unlike in conventional actor critic algorithms, in DDPG evaluation reward to be validated using target evaluation, policy networks.

E. TARGET POLICY NETWORKS

In policy network, action performed using policy with gradient parameters θ^λ . For one iteration, it is indicated as $act_T = \lambda(st_T|\theta^\lambda)$. For the next iteration is updated as act_T^\sim which is generated by the next state st_T^\sim . Target policy network looks for next act_T^\sim and it updates θ^λ to θ^{λ^\sim} . It is expressed in (22).

$$M(\theta^\lambda) = E\theta^\lambda[rew_1 + \tau rew_2 + \tau^2 rew_3 + \dots] \quad (22)$$

F. TARGET EVALUATION NETWORKS

In the evaluation network, expected target of mapping process is validated by updating q-function in iterative approach. It updates q function parameter θ^q . q-function is indicated for next iterations are expressed using (23).

$$q^\lambda(st_T, act_T) = E[rew(st_T, act_T) + \tau q^\lambda(st_{T+1}, \lambda(st_{T+1}))] \tag{23}$$

where τ is a discount factor in above eqn.17. The quality of policy and its evaluation is expressed using (24).

$$M_\omega(\lambda) = \int p^\omega q^\lambda(st_T, \lambda(st_T)) ds \tag{24}$$

$$= E_{s \sim p^\omega}[q^\lambda(st_T, \lambda(st_T))] \tag{24}$$

$$\lambda = argmax_\lambda M(\lambda) \tag{25}$$

where $q^\lambda(st_T, \lambda(st_T))$ is expected reward generated when policy λ is considered with states indicated as st_T for scheduling process. $M_\omega(\lambda)$ is expectation of $q^\lambda(st_T, \lambda(st_T))$ when state spaces st_T is disbursed on p^ω .

G. UPDATION OF POLICY AND EVALUATION NETWORKS

After evaluating the dispersed rewards generated from policy guided actions and expected actions from target networks with different states, updation of policy, evaluation networks are expressed using below (26), (27),(28).

$$policy\ network = \begin{cases} \lambda(st_T | \theta^\lambda), & \text{gradient update} \\ \lambda^\sim(st_T^\sim | \theta^{\lambda^\sim}), & \text{Soft update} \end{cases} \tag{26}$$

$$q\ network = \begin{cases} q(st_T, act_T | \theta^q), & \text{gradient update} \\ q^\sim(st_T^\sim, act_T | \theta^{\lambda^\sim}), & \text{Soft update} \end{cases} \tag{27}$$

Finally in the above equations soft update of both policy and evaluation networks are calculated by eqn.22.

$$soft\ update = \begin{cases} \theta^{\lambda^\sim} \leftarrow \vartheta \theta^\lambda + (1 - \vartheta) \theta^{\lambda^\sim} \\ \theta^{q^\sim} \leftarrow \vartheta \theta^q + (1 - \vartheta) \theta^{q^\sim} \end{cases} \tag{28}$$

H. PROPOSED HYBRID WORKLOAD DEEP DETERMINISTIC POLICY GRADIENT BASED TASK SCHEDULER(HDDPGTS)

Figure 2 discusses flow of Hybrid Workload Deep Deterministic policy gradient based algorithm. It is initialized with actor, critic networks with gradient parameters. Later in the next step target policy, evaluation networks are to be initialized with expected policies. After initialization of these networks, randomly tasks are initialized and later task classification, VM priorities are calculated using eqns.1, 2. Apply mapping procedure of tasks using action space with sample selection of state spaces and later calculate the reward with eqn.19. After evaluating reward check obtained values for makespan, energy consumption. Check the parameters are

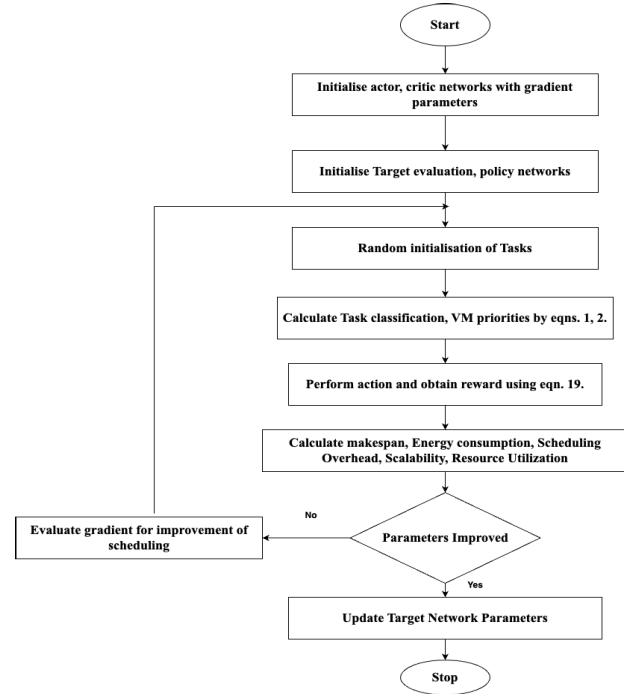


FIGURE 2. Flow of HDDPGTS algorithm.

Input: $t_{n1}, vm_{k1}, hn_{i1}, dtc_{j1}, \theta^\lambda, \theta^{\lambda^\sim}, \theta^q, \theta^{q^\sim}, \vartheta, \omega, \tau$
 Output: Optimal Mapping of set of tasks to VMs i.e. act_T
 start

Initialize both actor, critic networks $\lambda(st_T | \theta^\lambda), q(st_T, act_T | \theta^q)$
 Initialize both target networks q^\sim, λ^\sim by using weights $\theta^{\lambda^\sim} \leftarrow \theta^\lambda$ and $\theta^{q^\sim} \leftarrow \theta^q$.
 Initialize tasks randomly
 Calculate task classification using eqn.1.
 Calculate priorities of VMs using eqn.4.
 Select initial state st_1 and check the reward for st_1
 Implement mapping i.e. action $act_T = \lambda(st_T | \theta^\lambda)$ with random exploration of tasks.
 Check reward rew_T , and identify next state st_{T+1}
 Store values of $(st_T, act_T, rew_T, st_{T+1})$ in q-table.
 Identify random values of $(st_i, act_i, rew_i, st_{i+1})$ of a mini-batch of tasks from N transitions.
 Evaluate q-value, as $z_i = rew_i + \tau q^\sim(st_{i+1}, \lambda^\sim(st_{i+1} | \theta^{\lambda^\sim}) | \theta^{q^\sim})$
 Update policy network using eqn. 26.
 Update critic network using eqn.27.
 Update target networks using eqn.28.
 Check the reward rew_T using eqn.21.
 Check if parameters are optimized
 Else
 $act_T \leftarrow \lambda(st_T | \theta^\lambda)$
 Continue this process till all iterations are completed.
 stop

improved or not and if they improved update as best parameters, end the scheduling procedure. If selected parameters are

not best identified values apply the gradient policy to improve these parameters till all iterations are completed.

I. ALGORITHM TIME COMPLEXITY

The time complexity of our proposed algorithm as follows: In DDPG, the initialization steps for the actor, critic, target actor and target critic networks, initializing tasks randomly, task classification and VM priorities are defined respectively by $O(1)$, $O(n)$, and $O(m)$ complexities where n is the number of tasks and m is the number of VMs. In the iterative loop of each iteration, the process includes selecting initial states and checking rewards takes, storing values in the q-table, sampling minibatches, evaluating q-values and updating networks, all these take $O(b)$ per iteration, where b is the minibatch size. The loop parameter T emphasizes that the computational complexity inside the loop is $T \cdot b$. Hence, when taking into account all the preprocessing steps and the iterative loop, the overall time complexity of the algorithm will be $C \triangleq CK(CP, CE) = O(T \cdot b + n + m)$, where T , b are much bigger than n, m , thus the time complexity of proposed algorithm is $O(T \cdot b)$

V. SIMULATION AND RESULTS

Simulation and results of Hybrid workload Deep Deterministic Policy gradient task scheduler (HDDPGTS) in section V. It consists of various subsections in which Configuration settings of simulation discussed in subsection A, makespan evaluation by HDDPGTS discussed in Subsection, evaluation of Energy Consumption using HDDPGTS discussed in Subsection C, Results analysis discussed in Subsection D. Extensive simulations conducted using fabricated datasets with statistical distributions indicated as uniform distribution indicated as U1, normal distribution indicated as N2, Left skewed distribution indicated as L3, right skewed distribution indicated as R4. After completion of simulation using different distributions, then we used real time super-computing logs HPC2N indicated as H5, NASA indicated as N6. Entire simulations are conducted on Cloudsim. Our proposed HDDPGTS compared over existing DQN [52], A2C [53] algorithms to evaluate parameters energy consumption, makespan. In Simulation setup, we have mentioned that we used both fabricated, realtime worklogs. Fabricated worklogs represents different statistical distributions consists of U1 which consists of all equal sized small, medium, large tasks. N2 distribution consists high number of medium, less number of large, small tasks. L3 distribution consists high number of small tasks, less large tasks. R4 distribution consists less small number of tasks, high number of large tasks. Moreover, real time worklogs are captured as H5, N6 identified from HPC2N [54], NASA [55] parallel worklogs.

A. CONFIGURATION SETTINGS FOR HDDPGTS

This subsection discussed settings used in proposed HDDPGTS algorithm and they are mentioned in Table 3.

TABLE 3. Configuration settings used in HDDPGTS.

Entity	Quantity
Tasks	1000
VMs	100
Length of Tasks	700,000
Physical host node Memory	128GB
Physical host Bandwidth	1500 MBPS
VM Memory	8 GB
Storage of Physical host node	8 TB
Storage of Virtual Machine	32 GB
Virtual Machine Bandwidth	20 MBPS
Physical Host node Operating System	Windows
Considered Datacenters	80

B. EVALUATION OF MAKESPAN BY HDDPGTS

This subsection discussed about calculation of makespan using proposed HDDPGTS. In our research makespan is chosen as a primary evaluating parameter. Reason for choosing it as parameter is scheduling of tasks mainly relies on task execution time which is an important component in makespan which directly affects quality of service of CSP. Increase in makespan lead to delay in execution of tasks as variety of tasks with different capacities and therefore in this research, a mechanism is induced which classify tasks based on length and capacity and then assign them to an appropriate VMs based on processing capacity and VM priorities based on electricity cost and schedules tasks onto appropriate VMs using HDDPGTS. Figure 3 indicates makespan generated by U1 distribution.

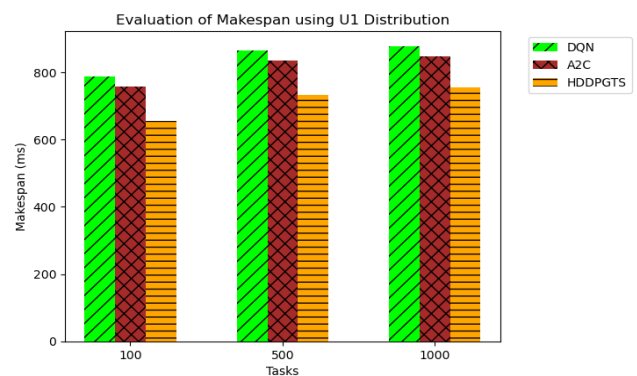


FIGURE 3. Makespan evaluation for HDDPGTS using U1 distribution.

Initially as mentioned our proposed HDDPGTS evaluated against DQN, A2C algorithms. Simulation conducted with 100 iterations. Initial makespan generated with 100,

500, 1000 tasks with U1distribution for DQN is 787.23, 865.82, 878.17 respectively. Makespan generated for 100, 500, 1000 tasks with U1 distribution for A2C is 756.91, 848.28, 835.23 respectively. Finally, proposed HDDPGTS generated makespan for 100, 500, 1000 tasks with U1 distribution is 656.18, 732.19, 754.67. From the results mentioned in Figure 3 it clearly specifies makespan is significantly minimized over DQN, A2C algorithms for U1 workload.

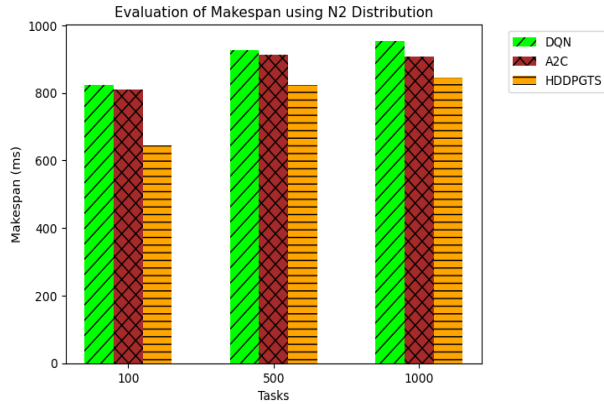


FIGURE 4. Makespan evaluation for HDDPGTS using N2 distribution.

Makespan generated with 100, 500, 1000 tasks with N2distribution for DQN is 823.17, 926.37, 954.17 respectively. Makespan generated for 100, 500, 1000 tasks with N2 distribution for A2C is 809.28, 913.42, 908.35 respectively. Finally, proposed HDDPGTS generated makespan for 100, 500, 1000 tasks with N2 distribution is 645.56, 823.42, 843.66. From results mentioned in Figure 4 it is clearly specifies makespan is significantly minimized over DQN, A2C algorithms for N2 workload.

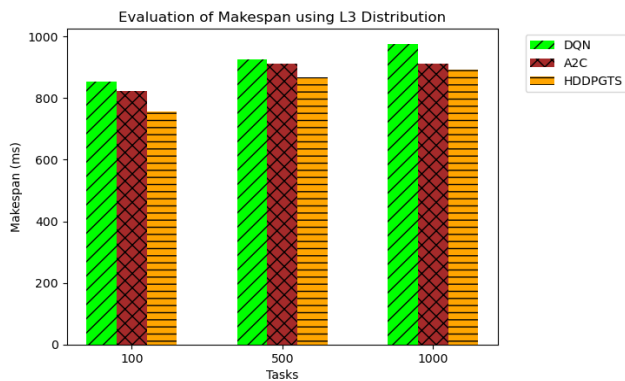


FIGURE 5. Makespan evaluation for HDDPGTS using L3 distribution.

Makespan generated with 100, 500, 1000 tasks with L3 distribution for DQN is 853.45, 925.73, 976.38 respectively. Makespan generated for 100, 500, 1000 tasks with L3 distribution for A2C is 822.45, 912.82, 911.87 respectively. Finally, proposed HDDPGTS generated makespan for 100, 500, 1000 tasks with L3 distribution is 756.38, 867.38,

892.13. From the results mentioned in Figure 5 it is clearly specifies makespan is significantly minimized over DQN, A2C algorithms for L3 workload.

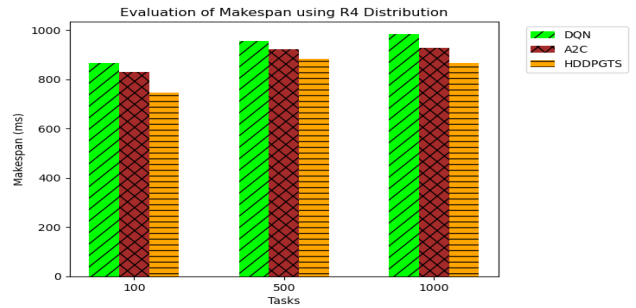


FIGURE 6. Makespan evaluation for HDDPGTS using R4 distribution.

Generated makespan with 100, 500, 1000 tasks with R4 distribution for DQN is 867.27, 956.17, 984.27. Generated makespan for 100, 500, 1000 tasks with R4 distribution for A2C is 830.12, 923.34, 928.35 respectively. Finally, proposed HDDPGTS generated makespan for 100, 500, 1000 tasks with R4 distribution is 745.12, 884.16, 867.28. From results mentioned in Figure 6 it is clearly observed that makespan is significantly minimized over DQN, A2C algorithms for R4 distribution.

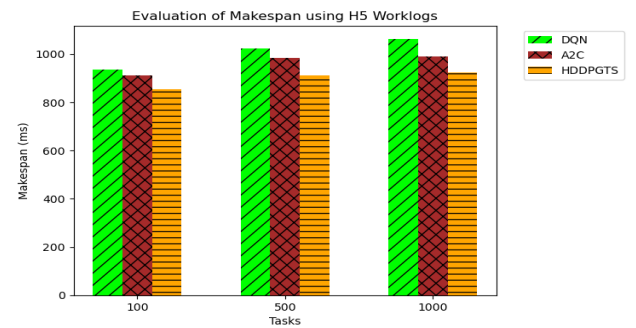


FIGURE 7. Makespan evaluation for HDDPGTS using H5 worklogs.

For 100, 500, 1000 tasks makespan generated with H5 worklogs for DQN is 935.18, 1024.12, 1063.48 respectively. For 100, 500, 1000 tasks makespan generated with H5 Worklogs for A2C is 912.74, 983.47, 988.63 respectively. Finally, for proposed HDDPGTS for 100, 500, 1000 tasks makespan generated is 853.28, 912.24, 924.33. From results mentioned in Figure 7 it is clearly specifies makespan is significantly minimized over DQN, A2C algorithms with H5 worklogs.

For 100, 500, 1000 tasks makespan generated with N6 worklogs for DQN is 942.28, 1018.36, 1057.11 respectively. For 100, 500, 1000 tasks makespan generated with with N6 Worklogs for A2C is 921.37, 947.38, 956.29 respectively. Finally, proposed HDDPGTS makespan for 100, 500, 1000 tasks with N6 worklogs is 862.31, 909.17, 915.69 respectively. From results mentioned in Figure 8 it is clearly observed makespan is significantly minimized over DQN, A2C algorithms with H5 worklogs.

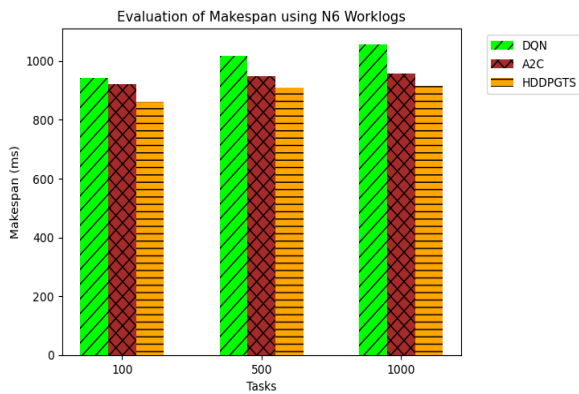


FIGURE 8. Makespan evaluation for HDDPGTS using H5 worklogs.

C. ENERGY CONSUMPTION EVALUATION BY HDDPGTS

This subsection discusses about energy consumption which is a second evaluation parameter for HDDPGTS. Energy consumption is important evaluation parameter in task scheduling in cloud computing as ineffective mapping of tasks to VMs without considering VMs processing capacity and task processing capacity and its runtime then more energy will be wasted in executing those tasks on VMs in corresponding datacenters and it also impacts electricity costs for cloud service providers. Therefore, Cloud Service Provider will incur a huge loss due to ineffective task scheduling. This motivates us to evaluate this parameter to improve scheduling process using HDDPGTS.

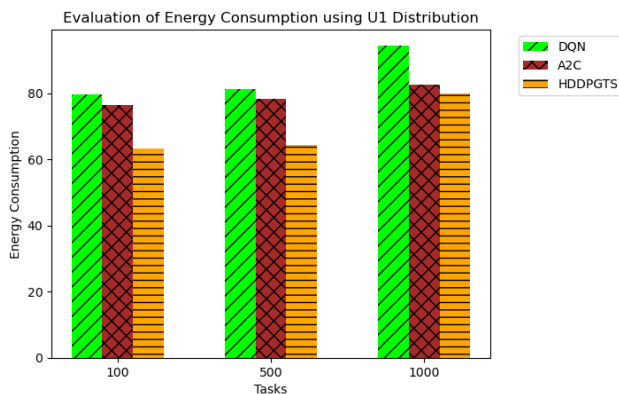


FIGURE 9. Energy consumption evaluation for HDDPGTS using U1 distribution.

Initially as mentioned that our proposed HDDPGTS evaluated against DQN, A2C algorithms. For 100, 500, 1000 tasks energy consumption generated with U1 distribution with DQN is 79.57, 81.36, 94.47 respectively. For 100, 500, 1000 tasks energy consumption generated with with U1 distribution for A2C is 76.54, 78.29, 82.54 respectively. Finally, for proposed HDDPGTS with 100, 500, 1000 tasks energy consumption generated with U1 distribution is 63.17, 64.22, 79.93. From results mentioned in Figure 9 it is clearly

specifies energy consumption is significantly minimized over DQN, A2C algorithms for U1 distribution.

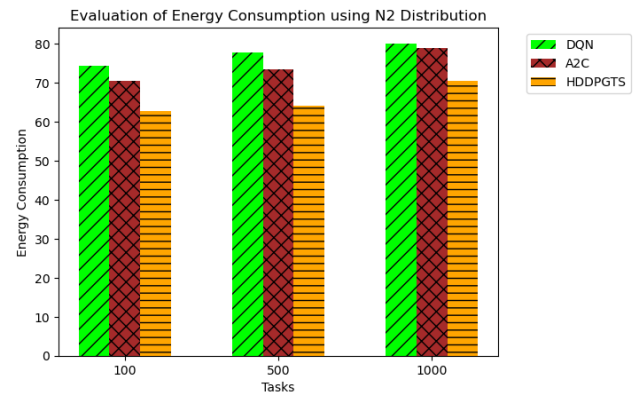


FIGURE 10. Energy consumption evaluation for HDDPGTS using N2 distribution.

For 100, 500, 1000 tasks energy consumption generated with N2 distribution for DQN is 74.37, 77.87, 80.17 respectively. For 100, 500, 1000 tasks energy consumption generated with N2 distribution for A2C is 70.52, 73.47, 78.91 respectively. Proposed HDDPGTS with 100, 500, 1000 tasks energy consumption generated with N2 distribution is 62.77, 64.12, 70.58 respectively. From results mentioned in Figure 10 it is clearly signifies energy consumption is significantly minimized over DQN, A2C algorithms for N2 distribution.

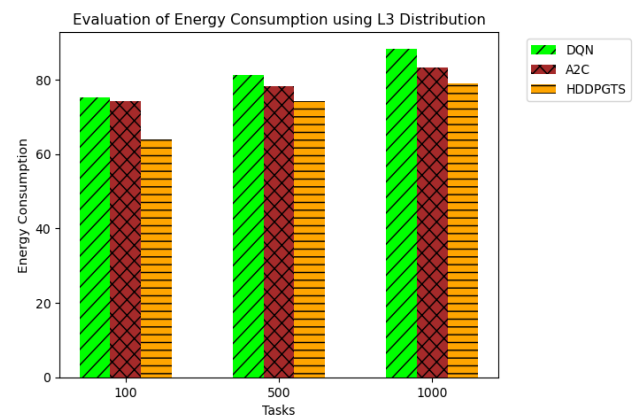


FIGURE 11. Energy consumption evaluation for HDDPGTS using L3 distribution.

For 100, 500, 1000 tasks energy consumption generated with L3 distribution for DQN is 75.26, 81.23, 88.37 respectively. For 100, 500, 1000 tasks energy consumption generated with L3 distribution for A2C is 74.13, 78.17, 83.24 respectively. Proposed HDDPGTS with 100, 500, 1000 tasks energy consumption generated with L3 distribution is 64.04, 74.28, 79.11. From results mentioned in Figure 11 it is clearly shows energy consumption is significantly minimized over DQN, A2C for L3 distribution.

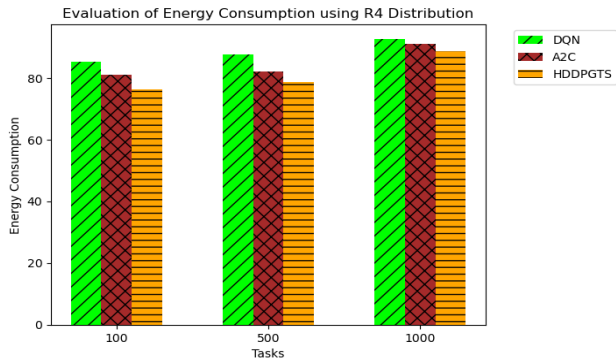


FIGURE 12. Energy consumption evaluation for HDDPGTS using R4 distribution.

For 100, 500, 1000 tasks energy consumption generated with R4 distribution for DQN is 85.23, 87.51, 92.67 respectively. For 100, 500, 1000 tasks energy consumption generated with R4 distribution for A2C is 81.07, 82.13, 90.91 respectively. Proposed HDDPGTS with 100, 500, 1000 tasks energy consumption generated with R4 distribution is 76.21, 78.58, 88.73. From results mentioned in Figure 12 it is clearly observed that energy consumption is significantly minimized over DQN, A2C for R4 distribution.

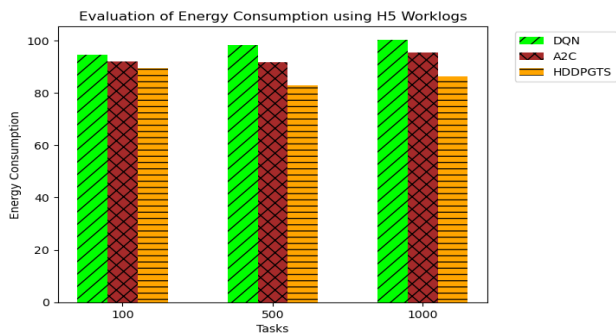


FIGURE 13. Energy consumption evaluation for HDDPGTS using H5 worklogs.

For 100, 500, 1000 tasks energy consumption generated with H5 worklogs for DQN is 94.57, 98.29, 100.36 respectively. For 100, 500, 1000 tasks energy consumption generated with H5 Worklogs for A2C is 92.13, 91.66, 95.39 respectively. Proposed HDDPGTS with 100, 500, 1000 tasks energy consumption generated for H5 worklogs is 89.44, 82.78, 86.38. From results mentioned in Figure 13 it is clearly observed Energy Consumption is significantly minimized over DQN, A2C algorithms with H5 worklogs.

For 100, 500, 1000 tasks energy consumption generated with N6 worklogs for DQN is 98.62, 105.47, 112.58 respectively. For 100, 500, 1000 tasks energy consumption generated with N6 Worklogs for A2C is 91.36, 94.91, 97.85 respectively. Proposed HDDPGTS with 100, 500, 1000 tasks energy consumption generated for N6 worklogs is 85.37, 87.19, 89.87. From results mentioned in Figure 14

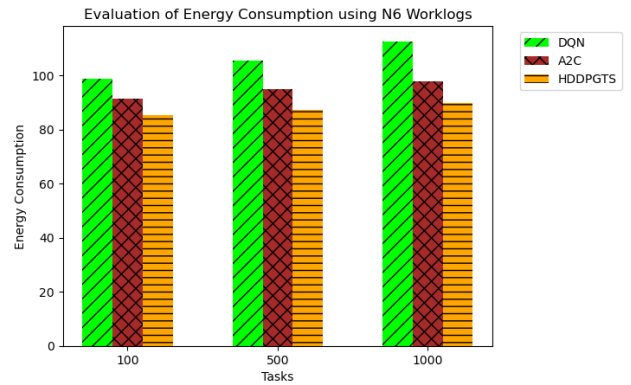


FIGURE 14. Energy consumption evaluation for HDDPGTS using N6 worklogs.

it is clearly observed, Energy Consumption is significantly minimized over DQN, A2C algorithms with N6 worklogs.

D. RESOURCE UTILIZATION EVALUATION BY HDDPGTS

It discusses about utilization of resources as it directly affects overall performance of scheduler and cost optimization of resources in perspective of cloud users. Therefore, utilization of resources with proposed HDDPGTS using statistical distributions, realtime workloads.

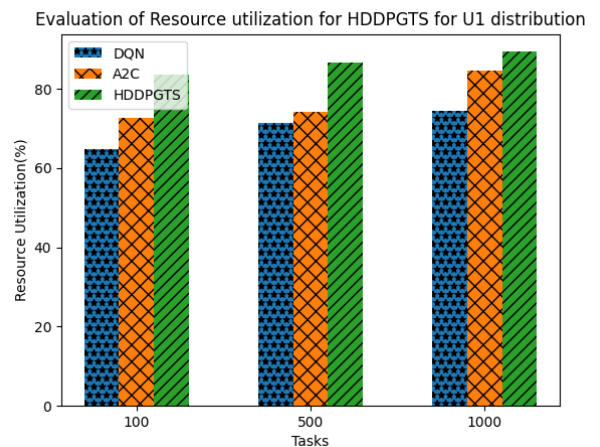


FIGURE 15. Resource utilization evaluation for HDDPGTS using U1 distribution.

For 100-1000 tasks resource utilization evaluated with U1 distribution for DQN is 64.68, 71.37, 74.57 respectively. Resource utilization evaluated with U1 distribution for A2C is 72.56, 74.19, 84.58 respectively. Proposed HDDPGTS with 100-1000 tasks resource utilization generated with U1distribution is 83.49, 86.58, 89.38 respectively. From results mentioned Figure 15 it is clearly observed that Resource utilization is significantly improved over DQN, A2C for U1 distribution.

For 100-1000 tasks resource utilization evaluated with N2 distribution for DQN is 66.78, 69.39, 71.69 respectively. Resource utilization evaluated with N2 distribution for A2C

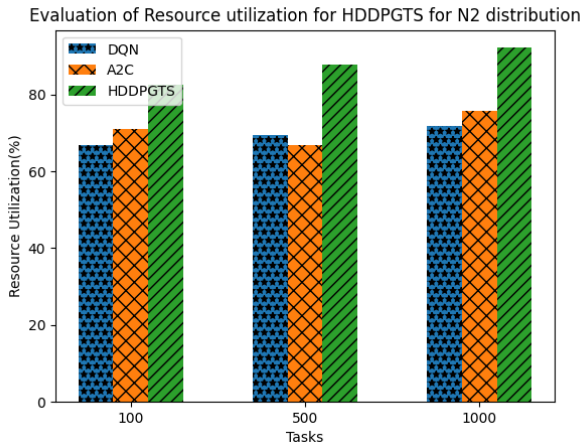


FIGURE 16. Resource utilization evaluation for HDDPGTS using N2 distribution.

is 71.07, 66.86, 75.86 respectively. Proposed HDDPGTS with 100-1000 tasks resource utilization generated with N2 distribution is 82.57, 87.68, 92.18 respectively. From results mentioned Figure 16 it is clearly observed that Resource utilization is significantly improved over DQN, A2C for N2 distribution.

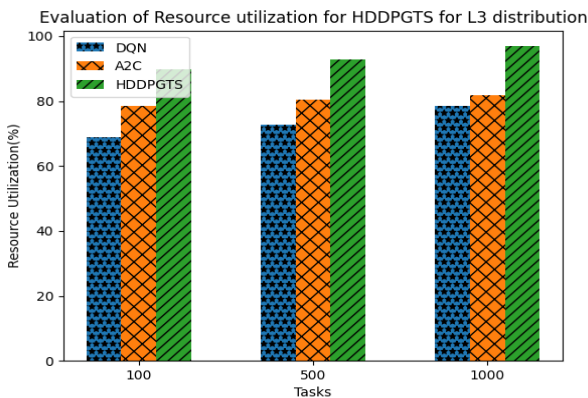


FIGURE 17. Resource utilization evaluation for HDDPGTS using L3 distribution.

For 100-1000 tasks resource utilization evaluated with L3 distribution for DQN is 68.76, 72.59, 78.57 respectively. Resource utilization evaluated with L3 distribution for A2C is 78.45, 80.37, 81.87 respectively. Proposed HDDPGTS with 100-1000 tasks resource utilization generated with L3 distribution is 89.92, 92.78, 96.89 respectively. From results mentioned Figure 17 it is clearly observed that Resource utilization is significantly improved over DQN, A2C for L3 distribution.

Resource utilization evaluated with R4 distribution for DQN with 100-1000 tasks is 57.92, 64.35, 72.76 respectively. Resource utilization evaluated with R4 distribution for A2C is 83.67, 78.04, 86.25 respectively. Proposed HDDPGTS with 100-1000 tasks resource utilization generated with R4 distribution is 86.18, 92.32, 95.78 respectively. From results

mentioned Figure 18 it is clearly observed that Resource utilization is significantly improved over DQN, A2C for R4 distribution.

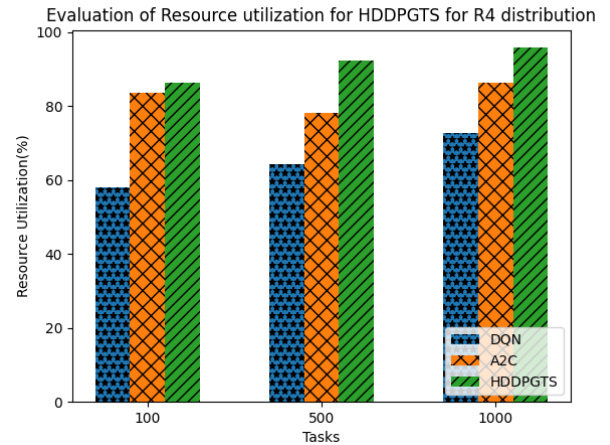


FIGURE 18. Resource utilization evaluation for HDDPGTS using R4 distribution.

Resource utilization evaluated with H5 worklogs for DQN with 100-1000 tasks is 54.37, 65.78, 72.16 respectively. Resource utilization evaluated with H5 worklogs for A2C is 65.67, 71.88, 74.29 respectively. Proposed HDDPGTS with 100-1000 tasks resource utilization generated with H5 worklogs is 84.33, 88.12, 91.54 respectively. From results mentioned in Figure 19 it is clearly observed that Resource utilization is significantly improved over DQN, A2C for H5 workload.

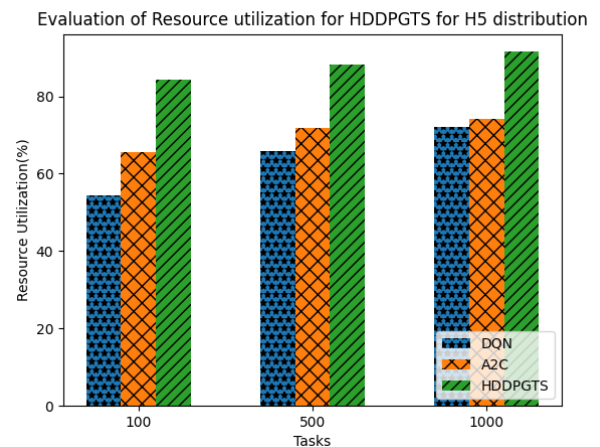


FIGURE 19. Resource utilization evaluation for HDDPGTS using H5 worklogs.

Resource utilization evaluated with N6 worklogs for DQN with 100-1000 tasks is 50.37, 63.24, 68.53 respectively. Resource utilization evaluated with N6 worklogs for A2C is 72.17, 78.43, 81.11 respectively. Proposed HDDPGTS with 100-1000 tasks resource utilization generated with H5 worklogs is 89.46, 90.36, 96.28 respectively. From results mentioned in Figure 20 it is clearly observed that Resource

utilization is significantly improved over DQN, A2C for N6 workload.

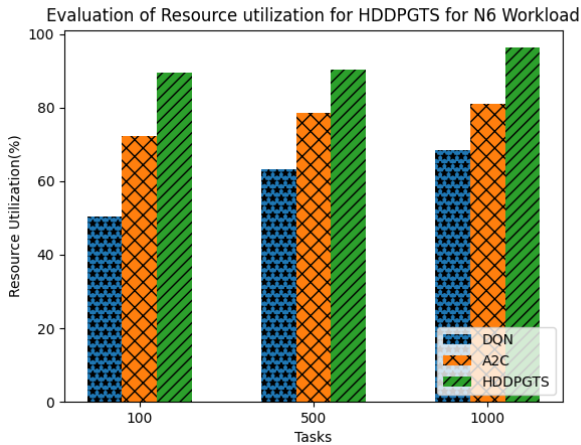


FIGURE 20. Resource utilization evaluation for HDDPGTS using N6 worklogs.

E. SCALABILITY EFFICIENCY EVALUATION BY HDDPGTS

It discusses scalability efficiency of task scheduler with different workload distributions of proposed HDDPGTS. Reason to choose scalability as evaluation parameter is to balance workload and act upon diversified tasks comes to cloud platform. Therefore, to evaluate robustness of proposed HDDPGTS with diversified workloads, scalability is calculated here in this research.

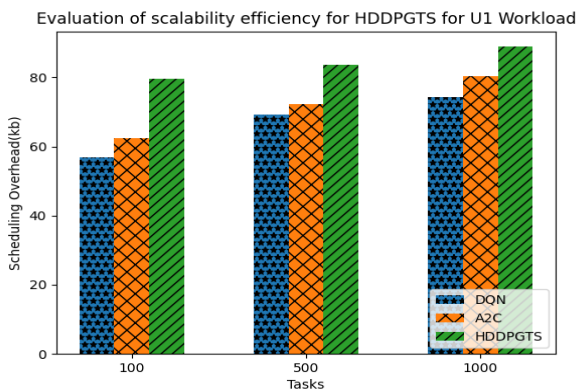


FIGURE 21. Scalability evaluation for HDDPGTS using U1 distribution.

Scalability evaluated with U1 distribution for DQN with 100-1000 tasks is 56.88, 69.27, 74.38 respectively. Scalability evaluated with U1 distribution for A2C is 62.45, 72.18, 80.33 respectively. Proposed HDDPGTS with 100-1000 tasks scalability generated with U1 distribution is 79.55, 83.51, 88.87 respectively. From results mentioned Figure 21 it is clearly observed that Scalability is significantly improved over DQN, A2C for U1 distribution.

Scalability evaluated with N2 distribution for DQN with 100-1000 tasks is 64.34, 71.56, 79.92 respectively. Scalability

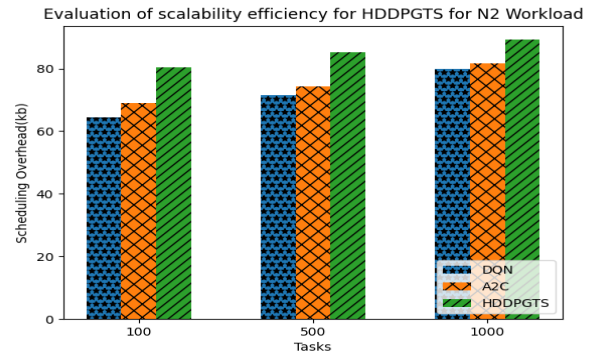


FIGURE 22. Scalability evaluation for HDDPGTS using N2 distribution.

evaluated with N2 distribution for A2C is 69.04, 74.38, 81.78 respectively. Proposed HDDPGTS with 100-1000 tasks scalability generated with N2 distribution is 80.37, 85.18, 89.22 respectively. From results mentioned Figure 22 it is clearly observed that Scalability is significantly improved over DQN, A2C for N2 distribution.

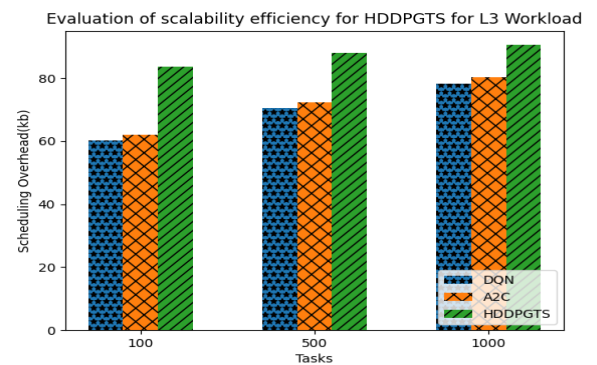


FIGURE 23. Scalability evaluation for HDDPGTS using L3 distribution.

Scalability evaluated with L3 distribution for DQN with 100-1000 tasks is 60.23, 70.58, 78.37 respectively. Scalability evaluated with L3 distribution for A2C is 62.18, 72.43, 80.33 respectively. Proposed HDDPGTS with 100-1000 tasks scalability generated with L3 distribution is 83.58, 87.99, 90.56 respectively. From results mentioned Figure 23 it is clearly observed that Scalability is significantly improved over DQN, A2C for L3 distribution.

Scalability evaluated with R4 distribution for DQN with 100-1000 tasks is 63.55, 67.21, 71.42 respectively. Scalability evaluated with R4 distribution for A2C is 67.57, 74.86, 82.15 respectively. Proposed HDDPGTS with 100-1000 tasks scalability generated with R4 distribution is 85.62, 89.09, 92.36 respectively. From results mentioned Figure 24 it is clearly observed that Scalability is significantly improved over DQN, A2C for R4 distribution.

Scalability evaluated with H5 Worklogs for DQN with 100-1000 tasks is 78.17, 79.88, 86.78 respectively. Scalability evaluated with H5 worklogs for A2C is 80.45, 82.37, 88.99 respectively. Proposed HDDPGTS with 100-1000 tasks

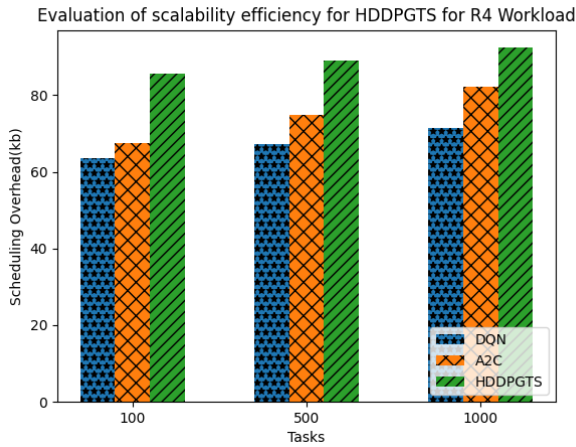


FIGURE 24. Scalability evaluation for HDDPGTS using R4 distribution.

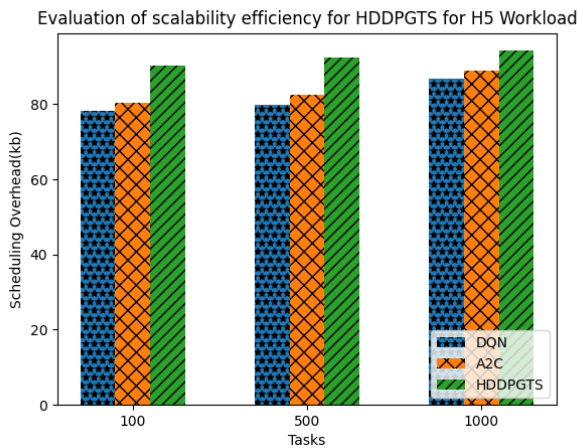


FIGURE 25. Scalability evaluation for HDDPGTS using H5 worklogs.

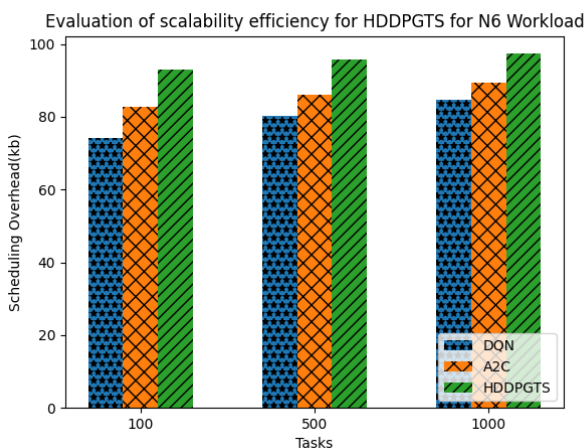


FIGURE 26. Scalability evaluation for HDDPGTS using N6 worklogs.

scalability generated with H5 worklogs is 90.18, 92.37, 94.21 respectively. From results mentioned Figure 25 it is clearly observed that Scalability is significantly improved over DQN, A2C for H5 worklogs.

Scalability evaluated with N6 Worklogs for DQN with 100-1000 tasks is 74.23, 80.33, 84.78 respectively. Scalability evaluated with N6 worklogs for A2C is 82.73, 86.11, 89.25 respectively. Proposed HDDPGTS with 100-1000 tasks scalability generated with N6 worklogs is 92.87, 95.76, 97.32 respectively. From results mentioned Figure 26 it is clearly observed that Scalability is significantly improved over DQN, A2C for N6 worklogs.

F. SCHEDULING OVERHEAD EVALUATION BY HDDPGTS

It discusses evaluation of scheduling overhead which is a crucial parameter in scheduling process. It affects memory usage pattern as diversified tasks may arrive at cloud platform and the way proposed HDDPGTS schedules tasks impacts scheduling overhead in this research. Therefore, it is evaluated using different workloads.

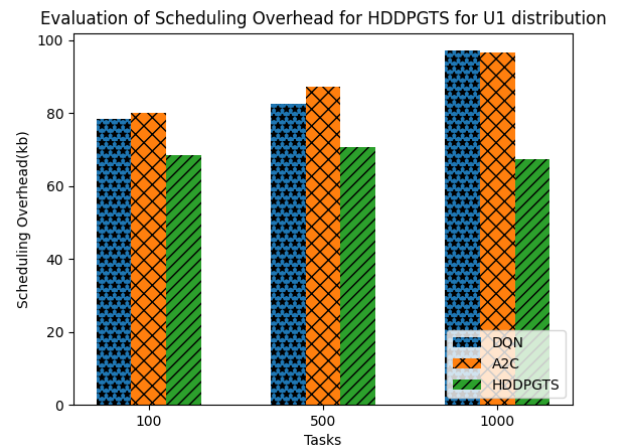


FIGURE 27. Scheduling overhead evaluation for HDDPGTS using U1 distribution.

Scheduling overhead evaluated with U1 distribution for DQN with 100-1000 tasks is 78.56, 82.57, 97.15 respectively. Scheduling overhead evaluated with U1 distribution for A2C is 79.97, 87.36, 96.73 respectively. Proposed HDDPGTS with 100-1000 tasks, scheduling overhead generated with U1 distribution is 68.53, 70.74, 67.36 respectively. From results mentioned Figure 27 it is clearly observed that Scheduling overhead is minimized over DQN, A2C for U1 distribution.

Scheduling overhead evaluated with N2 distribution for DQN with 100-1000 tasks is 85.78, 96.17, 100.36 respectively. Scheduling overhead evaluated with N2 distribution for A2C is 89.37, 78.17, 98.35 respectively. Proposed HDDPGTS with 100-1000 tasks, scheduling overhead generated with N2 distribution is 70.12, 72.18, 69.18 respectively. From results mentioned Figure 28 it is clearly observed that Scheduling overhead is minimized over DQN, A2C for N2 distribution.

Scheduling overhead evaluated with L3 distribution for DQN with 100-1000 tasks is 78.92, 86.72, 98.36 respectively. Scheduling overhead evaluated with L3 distribution for A2C is 56.35, 68.77, 89.18 respectively. Proposed HDDPGTS with 100-1000 tasks, scheduling overhead generated with L3

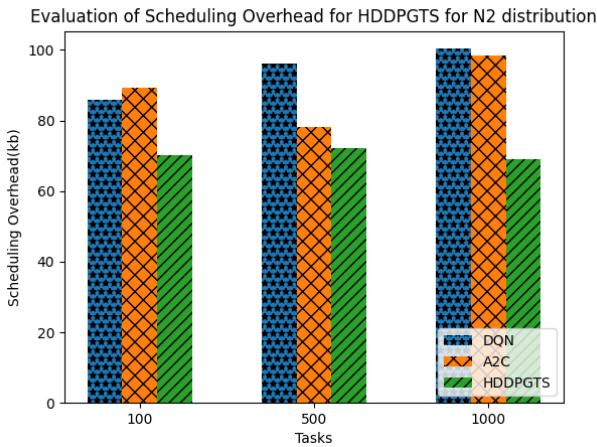


FIGURE 28. Scheduling overhead evaluation for HDDPGTS using N2 distribution.

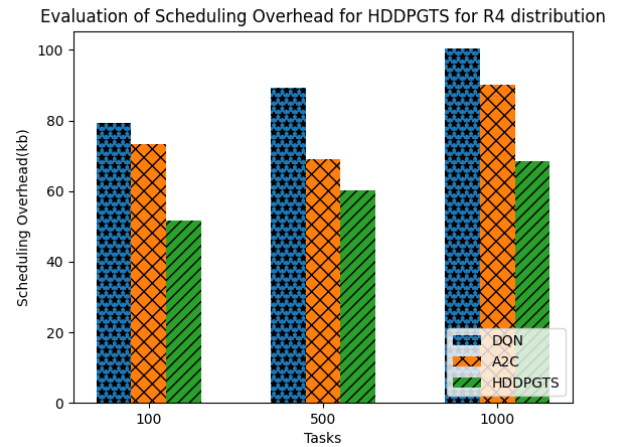


FIGURE 30. Scheduling overhead evaluation for HDDPGTS using R4 distribution.

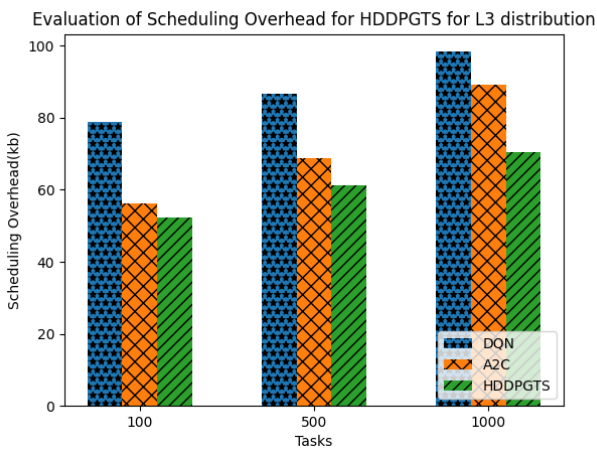


FIGURE 29. Scheduling overhead evaluation for HDDPGTS using L3 distribution.

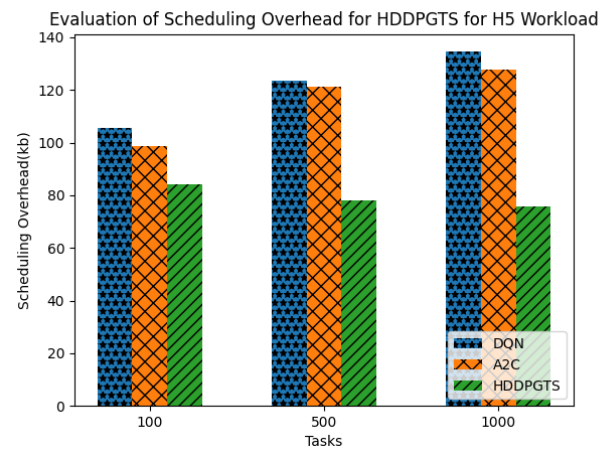


FIGURE 31. Scheduling overhead evaluation for HDDPGTS using H5 workload.

distribution is 52.21, 61.33, 70.57 respectively. From results mentioned Figure 29 it is clearly observed that Scheduling overhead is minimized over DQN, A2C for L3 distribution.

Scheduling overhead evaluated with R4 distribution for DQN with 100-1000 tasks is 79.17, 89.15, 100.35 respectively. Scheduling overhead evaluated with R4 distribution for A2C is 73.21, 69.18, 90.25 respectively. Proposed HDDPGTS with 100-1000 tasks, scheduling overhead generated with R4 distribution is 51.66, 60.24, 68.37 respectively. From results mentioned Figure 30 it is clearly observed that Scheduling overhead is minimized over DQN, A2C for R4 distribution.

Scheduling overhead evaluated with H5 workload for DQN with 100-1000 tasks is 105.64, 123.56, 134.56 respectively. Scheduling overhead evaluated with H5 workload for A2C is 98.57, 121.33, 127.87 respectively. Proposed HDDPGTS with 100-1000 tasks, scheduling overhead generated with H5 distribution is 84.32, 78.18, 75.67 respectively. From results mentioned Figure 31 it is clearly observed that Scheduling overhead is minimized over DQN, A2C for H5 workload.

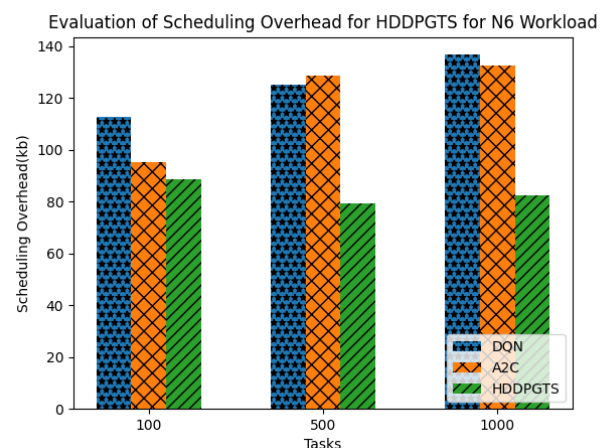


FIGURE 32. Scheduling overhead evaluation for HDDPGTS using N6 workload.

Scheduling overhead evaluated with N6 workload for DQN with 100-1000 tasks is 112.64, 125.28, 136.73 respectively. Scheduling overhead evaluated with N6 workload for A2C

is 95.23, 128.76, 132.54 respectively. Proposed HDDPGTS with 100-1000 tasks, scheduling overhead generated with N6 distribution is 88.53, 79.45, 82.35 respectively. From results mentioned Figure 31 it is clearly observed that Scheduling overhead is minimized over DQN, A2C for N6 workload.

G. DISCUSSION AND RESULT ANALYSIS

Simulation Analysis of proposed HDDPGTS discussed in this subsection. The below tables 4, 5, 6, 7, 8, 9 shown improvement of makespan for proposed HDDPGTS over DQN [52], A2C [53] algorithms.

TABLE 4. Improvement of makespan for HDDPGTS for u1 distribution.

Tasks	DQN	A2C
100	16.65%	13.31%
500	15.43%	12.34%
1000	14.06%	11.03%

TABLE 5. Improvement of makespan for HDDPGTS for N2 distribution.

Tasks	DQN	A2C
100	21.58%	20.23%
500	11.11%	9.85%
1000	11.58%	7.12%

TABLE 6. Improvement of makespan for HDDPGTS for L3 distribution.

Tasks	DQN	A2C
100	11.37%	8.03%
500	6.30%	4.98%
1000	8.63%	2.16%

TABLE 7. Improvement of makespan for HDDPGTS for R4 distribution.

Tasks	DQN	A2C
100	14.08%	10.24%
500	7.53%	4.24%
1000	11.89%	6.58%

TABLE 8. Improvement of makespan for HDDPGTS for H5 worklogs.

Tasks	DQN	A2C
100	8.76%	6.51%
500	10.92%	7.24%
1000	13.08%	6.50%

TABLE 9. Improvement of makespan for HDDPGTS for N6 worklogs.

Tasks	DQN	A2C
100	8.49%	6.41%
500	10.72%	4.03%
1000	13.38%	4.25%

From the above tables, It clearly observed that proposed HDDPGTS is adaptive to different workloads and minimizes makespan over DQN, A2C algorithms. In view of improvement for HDDPGTS N2 distribution gave best makespan over the other distributions and the reason for that is N2 distribution consists of all types of medium, short, long tasks which proves that proposed approach is adaptive to all types of workload distributions. When it is compared with other approaches DQN, A2C using realtime worklogs H5 worklogs significantly improved makespan over other algorithms and even if it consists of high performance tasks in H5 worklogs HDDPGTS generated schedules in order to improve makespan.

The below tables 10, 11, 12, 13, 14, 15 shown improvement of Energy consumption for proposed HDDPGTS over DQN, A2C algorithms.

TABLE 10. Improvement of energy consumption for HDDPGTS for U1 distribution.

Tasks	DQN	A2C
100	20.61%	17.47%
500	21.07%	17.97%
1000	15.39%	3.16%

TABLE 11. Improvement of energy consumption for HDDPGTS for N2 distribution.

Tasks	DQN	A2C
100	21.71%	19.21%
500	17.66%	15.98%
1000	11.96%	10.56%

TABLE 12. Improvement of energy consumption for HDDPGTS for L3 distribution.

Tasks	DQN	A2C
100	14.91%	13.61%
500	8.56%	4.98%
1000	10.48%	4.96%

TABLE 13. Improvement of energy consumption for HDDPGTS for R4 distribution.

Tasks	DQN	A2C
100	10.58%	5.99%
500	10.20%	4.32%
1000	4.25%	2.40%

From the above tables, it clearly observed that proposed HDDPGTS is adaptive to different workloads and minimizes energy consumption over DQN, A2C algorithms. In view of improvement for HDDPGTS N2 distribution gave best energy consumption over the other distributions and the reason for that is N2 distribution consists of all types of medium, short, long tasks which proves that proposed approach is

TABLE 14. Improvement of energy consumption for HDDPGTS for H5 worklogs.

Tasks	DQN	A2C
100	15.42%	7.76%
500	15.78%	9.69%
1000	13.93%	9.45%

TABLE 15. Improvement of energy consumption for HDDPGTS for N6 worklogs.

Tasks	DQN	A2C
100	13.44%	6.56%
500	17.33%	8.13%
1000	20.17%	8.16%

adaptive to all types of workload distributions. When it is compared with other approaches DQN, A2C using realtime worklogs H5 worklogs significantly improved energy consumption over other algorithms and even if it consists of high performance tasks in H5 worklogs HDDPGTS generated schedules in order to improve energy consumption.

The below tables 16, 17, 18, 19, 20, 21 shown improvement of utilization of resources for proposed HDDPGTS over DQN, A2C algorithms.

TABLE 16. Improvement of resource utilization (%) for HDDPGTS for U1 distribution.

Tasks	DQN	A2C
100	77.47	86.90
500	82.43	85.68
1000	83.43	94.62

TABLE 17. Improvement of resource utilization (%) for HDDPGTS for N2 distribution.

Tasks	DQN	A2C
100	80.87	86.07
500	79.14	76.25
1000	77.77	82.29

TABLE 18. Improvement of resource utilization (%) for HDDPGTS for L3 distribution.

Tasks	DQN	A2C
100	76.46	87.24
500	78.23	86.62
1000	81.09	84.49

From the above tables, it clearly observed that proposed HDDPGTS is adaptive to different workloads and improves resource utilization over DQN, A2C algorithms. In view of improvement for utilization of resources for HDDPGTS U1 distribution gave significant improvement in resource utilization over DQN and the reason for that is U1 distribution consists of all types of medium, short, long tasks

TABLE 19. Improvement of resource utilization (%) for HDDPGTS for R4 distribution.

Tasks	DQN	A2C
100	67.20	97.08
500	69.70	84.53
1000	75.96	90.05

TABLE 20. Improvement of resource utilization (%) for HDDPGTS for H5 workload.

Tasks	DQN	A2C
100	64.47	77.87
500	74.64	81.57
1000	78.82	81.15

TABLE 21. Improvement of resource utilization (%) for HDDPGTS for N6 workload.

Tasks	DQN	A2C
100	56.30	80.67
500	69.98	86.79
1000	71.17	84.24

which proves that proposed approach is adaptive to all types of workload distributions. Proposed HDDPGTS improves resource utilization with R4 distribution over A2C approach as it consists of high large number of tasks. When it is compared with other approaches DQN, A2C using realtime worklogs H5, N6 worklogs there is a significant improvement in resource utilization over DQN with H5 worklogs and over A2C it gives best utilization of resources even if it consists of high performance tasks in H5 worklogs HDDPGTS generated schedules in order to improve resource utilization.

The below tables 22, 23, 24, 25, 26, 27 shown improvement of Scalability for proposed HDDPGTS over DQN, A2C algorithms.

TABLE 22. Improvement of scalability (%) for HDDPGTS for U1 distribution.

Tasks	DQN	A2C
100	71.5	78.5
500	82.94	86.43
1000	83.69	90.39

TABLE 23. Improvement of scalability (%) for HDDPGTS for N2 distribution.

Tasks	DQN	A2C
100	80.05	85.90
500	84.01	87.32
1000	89.57	91.66

From the above tables, it clearly observed that proposed HDDPGTS is adaptive to different workloads and improves scalability over DQN, A2C algorithms with diversified workloads. In view of scalability for HDDPGTS N2 distribution

TABLE 24. Improvement of scalability (%) for HDDPGTS for L3 distribution.

Tasks	DQN	A2C
100	72.06	74.39
500	80.21	82.31
1000	86.53	88.70

TABLE 25. Improvement of scalability (%) for HDDPGTS for R4 distribution.

Tasks	DQN	A2C
100	74.22	78.91
500	75.44	84.02
1000	77.32	88.94

TABLE 26. Improvement of scalability (%) for HDDPGTS for H5 worklogs.

Tasks	DQN	A2C
100	86.68	89.21
500	86.47	89.17
1000	92.11	94.45

TABLE 27. Improvement of scalability (%) for HDDPGTS for N6 worklogs.

Tasks	DQN	A2C
100	79.92	89.08
500	83.88	89.92
1000	87.11	91.70

gave significant improvement in scalability over DQN, A2C and the reason for that is N2 distribution consists of all types of medium, short, long tasks which proves that proposed approach is adaptive to all types of workload distributions. When it is compared with other approaches DQN, A2C using realtime worklogs H5, N6 worklogs there is a significant improvement in scalability over DQN with H5 worklogs and over A2C it gives best scalability with N6 worklogs even if it consists of high performance tasks in H5, N6 worklogs HDDPGTS generated schedules in order to improve scalability.

The below tables 28, 29, 30, 31, 32, 33 shown improvement of Scheduling overhead for proposed HDDPGTS over DQN, A2C algorithms.

TABLE 28. Improvement of scheduling overhead (%) for HDDPGTS for U1 distribution.

Tasks	DQN	A2C
100	87.23	85.69
500	85.67	80.97
1000	69.33	69.63

From the above tables, it clearly observed that proposed HDDPGTS is adaptive to different workloads and minimizes overhead of scheduling in terms of memory capacity over DQN, A2C algorithms with diversified workloads. In view of scalability for HDDPGTS U1 distribution gave

TABLE 29. Improvement of scheduling overhead (%) for HDDPGTS for N2 distribution.

Tasks	DQN	A2C
100	81.74	78.46
500	75.05	92.33
1000	68.93	70.34

TABLE 30. Improvement of scheduling overhead (%) for HDDPGTS for L3 distribution.

Tasks	DQN	A2C
100	66.15	92.65
500	70.72	89.18
1000	71.74	79.13

TABLE 31. Improvement of scheduling overhead (%) for HDDPGTS for R4 distribution.

Tasks	DQN	A2C
100	65.25	70.56
500	67.57	87.07
1000	68.13	75.75

TABLE 32. Improvement of scheduling overhead (%) for HDDPGTS for H5 workload.

Tasks	DQN	A2C
100	79.81	85.54
500	63.27	64.43
1000	56.23	59.17

TABLE 33. Improvement of scheduling overhead (%) for HDDPGTS for N6 workload.

Tasks	DQN	A2C
100	78.59	92.96
500	63.41	61.70
1000	60.22	62.13

best scalability improvement over DQN and the reason for that is N2 distribution consists of all types of medium, short, long tasks. For A2C best scalability improvement given by L3 workload which consists of more small tasks and less large tasks which proves that proposed approach is adaptive to all types of workload distributions. When it is compared with other approaches DQN, A2C using realtime worklogs H5, N6 worklogs there is a significant improvement in scalability over DQN with H6 worklogs and over A2C it gives best scalability with N6 worklogs even if it consists of high performance tasks in H5, N6 worklogs HDDPGTS generated schedules in order to minimize scheduling overhead. Therefore, from analysis of results, our proposed HDDPGTS scheduler is adaptive to the diversified worklogs and improves all mentioned above parameters which proves that proposed approach is robust over diversified workloads. Moreover that, HDDPGTS improved scalability over both

types of statistical distributions, real time worklogs while minimizing scheduling overhead.

VI. CONCLUSION AND FUTURE WORK

Task Scheduling in cloud paradigm is prodigious challenge as various (Hybrid) types of tasks (mostly HPC and HTC) generated from various resources. Scheduling these types of tasks to precise VMs is a critical issue. Many existing authors used metaheuristic, machine learning based algorithms to design task schedulers but still there is a chance to optimize scheduling is highly dynamic in cloud environment. Aim of our task scheduling process is to identify and classify the type of traffic under two categories as either HPC or HTC based on their length, run time capacity. After classification these tasks are to be fed to DDPG task scheduler to make decision to schedule tasks onto corresponding VMs based on electricity cost while tracking the underlying resources in VMs. Extensive simulations are conducted using Cloudsim with different fabricated workload distributions, realtime worklogs to facilitate HPC and HTC traffic in simulation. Finally, our proposed HDDPGTS evaluated against DQN, A2C algorithms to check effectiveness of HDDPGTS in terms of makespan, energy consumption, scheduling overhead, utilization of resources, scalability. From results, it shown that there is a significant improvement over DQN, A2C algorithms while improving above mentioned parameters. In future, our aim is to deploy HDDPGTS in AWS cloud environment to check effectiveness of it for other operational parameters in cloud environment.

ACKNOWLEDGMENT

The authors would like to thank the support of Research Supporting Project Number (RSP2024R421), King Saud University, Riyadh, Saudi Arabia.

REFERENCES

- [1] R. O. Aburukba, M. AliKarrar, T. Landolsi, and K. El-Fakih, "Scheduling Internet of Things requests to minimize latency in hybrid fog-cloud-computing," *Future Gener. Comput. Syst.*, vol. 111, pp. 539–551, Oct. 2020.
- [2] T. Zheng, J. Wan, J. Zhang, and C. Jiang, "Deep reinforcement learning-based workload scheduling for edge computing," *J. Cloud Comput.*, vol. 11, no. 1, p. 3, Dec. 2022.
- [3] S. Subbaraj and R. Thiyagarajan, "Performance oriented task-resource mapping and scheduling in fog computing environment," *Cognit. Syst. Res.*, vol. 70, pp. 40–50, Dec. 2021.
- [4] S. Ghanavati, J. Abawajy, and D. Izadi, "Automata-based dynamic fault tolerant task scheduling approach in fog computing," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 1, pp. 488–499, Jan. 2022.
- [5] M. Abdel-Basset, D. El-Shahat, M. Elhoseny, and H. Song, "Energy-aware metaheuristic algorithm for industrial-Internet-of-Things task scheduling problems in fog computing applications," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12638–12649, Aug. 2021.
- [6] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog computing for energy-aware load balancing and scheduling in smart factory," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4548–4556, Oct. 2018.
- [7] A. Ghasemi and A. T. Haghghat, "A multi-objective load balancing algorithm for virtual machine placement in cloud data centers based on machine learning," *Computing*, vol. 102, no. 9, pp. 2049–2072, Sep. 2020.
- [8] S. Ijaz, E. U. Munir, S. G. Ahmad, M. M. Rafique, and O. F. Rana, "Energy-makespan optimization of workflow scheduling in fog-cloud computing," *Computing*, vol. 103, no. 9, pp. 2033–2059, Sep. 2021.
- [9] Y. Sun, F. Lin, and H. Xu, "Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II," *Wireless Pers. Commun.*, vol. 102, no. 2, pp. 1369–1385, Sep. 2018.
- [10] R. O. Aburukba, T. Landolsi, and D. Omer, "A heuristic scheduling approach for fog-cloud computing environment with stationary IoT devices," *J. Netw. Comput. Appl.*, vol. 180, Apr. 2021, Art. no. 102994.
- [11] S. P. Praveen, H. Ghasempoor, N. Shahabi, and F. Izanloo, "A hybrid gravitational emulation local search-based algorithm for task scheduling in cloud computing," *Math. Problems Eng.*, vol. 2023, pp. 1–9, Feb. 2023.
- [12] S. Chaudhary, V. K. Sharma, R. N. Thakur, A. Rathi, P. Kumar, and S. Sharma, "Modified particle swarm optimization based on aging leaders and challengers model for task scheduling in cloud computing," *Math. Problems Eng.*, vol. 2023, pp. 1–11, Jun. 2023.
- [13] H. K. Patnaik, M. R. Patra, and R. Kumar, "A workflow based approach for task scheduling in cloud environment," *Mater. Today, Proc.*, vol. 80, no. 10, pp. 3305–3311, Dec. 2023.
- [14] M.-L. Chiang, H.-C. Hsieh, Y.-H. Cheng, W.-L. Lin, and B.-H. Zeng, "Improvement of tasks scheduling algorithm based on load balancing candidate method under cloud computing environment," *Expert Syst. Appl.*, vol. 212, Feb. 2023, Art. no. 118714.
- [15] G. Agarwal, S. Gupta, R. Ahuja, and A. K. Rai, "Multiprocessor task scheduling using multi-objective hybrid genetic algorithm in fog-cloud computing," *Knowledge-Based Syst.*, vol. 272, Jul. 2023, Art. no. 110563.
- [16] A. Y. Hamed, M. Kh. Elnahary, F. S. Alsubaei, and H. H. El-Sayed, "Optimization task scheduling using cooperation search algorithm for heterogeneous cloud computing systems," *Comput., Mater. Continua*, vol. 74, no. 1, pp. 2133–2148, 2023.
- [17] C. Chandrashekar, P. Krishnados, V. Kedalu Poornachary, B. Ananthkrishnan, and K. Rangasamy, "HWACOA scheduler: Hybrid weighted ant colony optimization algorithm for task scheduling in cloud computing," *Appl. Sci.*, vol. 13, no. 6, p. 3433, Mar. 2023.
- [18] S. Badri, D. M. Alghazzawi, S. H. Hasan, F. Alfayez, S. H. Hasan, M. Rahman, and S. Bhatia, "An efficient and secure model using adaptive optimal deep learning for task scheduling in cloud computing," *Electronics*, vol. 12, no. 6, p. 1441, Mar. 2023.
- [19] D. Chen and Y. Zhang, "Diversity-aware marine predators algorithm for task scheduling in cloud computing," *Entropy*, vol. 25, no. 2, p. 285, Feb. 2023.
- [20] X. Fu, Y. Sun, H. Wang, and H. Li, "Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm," *Cluster Comput.*, vol. 26, no. 5, pp. 2479–2488, Oct. 2023.
- [21] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, "Q-learning based dynamic task scheduling for energy-efficient cloud computing," *Future Gener. Comput. Syst.*, vol. 108, pp. 361–371, Jul. 2020.
- [22] Z. Tong, H. Chen, X. Deng, K. Li, and K. Li, "A scheduling scheme in the cloud computing environment using deep Q-learning," *Inf. Sci.*, vol. 512, pp. 1170–1191, Feb. 2020.
- [23] F. S. Alsubaei, A. Y. Hamed, M. R. Hassan, M. Mohery, and M. K. Elnahary, "Machine learning approach to optimal task scheduling in cloud communication," *Alexandria Eng. J.*, vol. 89, pp. 1–30, Feb. 2024.
- [24] S. Swarup, E. M. Shakshuki, and A. Yasar, "Task scheduling in cloud using deep reinforcement learning," *Proc. Comput. Sci.*, vol. 184, no. 184, pp. 42–51, Nov. 2021.
- [25] M. Sharma and R. Garg, "An artificial neural network based approach for energy efficient task scheduling in cloud data centers," *Sustain. Comput., Informat. Syst.*, vol. 26, Jun. 2020, Art. no. 100373.
- [26] C. Shetty, H. Sarojadevi, and S. Prabhu, "Machine learning approach to select optimal task scheduling algorithm in cloud," *Turkish J. Comput. Math. Educ. (TURCOMAT)*, vol. 12, no. 6, pp. 2565–2580, Apr. 2021.
- [27] H. A. M. Balla, C. G. Sheng, and W. Jing, "Reliability-aware: Task scheduling in cloud computing using multi-agent reinforcement learning algorithm and neural fitted Q," *Int. Arab J. Inf. Technol.*, vol. 18, no. 1, pp. 36–47, 2021.
- [28] Y. Wang, S. Dong, and W. Fan, "Task scheduling mechanism based on reinforcement learning in cloud computing," *Mathematics*, vol. 11, no. 15, p. 3364, Aug. 2023.
- [29] B. Kruekaew and W. Kimpan, "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning," *IEEE Access*, vol. 10, pp. 17803–17818, 2022.

- [30] M. T. Islam, S. Karunasekera, and R. Buyya, "Performance and cost-efficient spark job scheduling based on deep reinforcement learning in cloud computing environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 7, pp. 1695–1710, Jul. 2022.
- [31] M. S. A. Khan and R. Santhosh, "Task scheduling in cloud computing using hybrid optimization algorithm," *Soft Comput.*, vol. 26, no. 23, pp. 13069–13079, Dec. 2022.
- [32] Z. Tong, X. Deng, H. Chen, J. Mei, and H. Liu, "QL-HEFT: A novel machine learning scheduling scheme base on cloud computing environment," *Neural Comput. Appl.*, vol. 32, no. 10, pp. 5553–5570, May 2020.
- [33] Z. Peng, D. Cui, J. Zuo, Q. Li, B. Xu, and W. Lin, "Random task scheduling scheme based on reinforcement learning in cloud computing," *Cluster Comput.*, vol. 18, no. 4, pp. 1595–1607, Dec. 2015.
- [34] K. Siddesha, G. V. Jayaramaiah, and C. Singh, "A novel deep reinforcement learning scheme for task scheduling in cloud computing," *Cluster Comput.*, vol. 25, no. 6, pp. 4171–4188, Dec. 2022.
- [35] R. Jain and N. Sharma, "A quantum inspired hybrid SSA–GWO algorithm for SLA based task scheduling to improve QoS parameter in cloud computing," *Cluster Comput.*, vol. 26, no. 6, pp. 3587–3610, Dec. 2023.
- [36] M. Abdullahi, M. A. Ngadi, S. I. Dishing, and S. M. Abdulhamid, "An adaptive symbiotic organisms search for constrained task scheduling in cloud computing," *J. Ambient Intell. Humanized Comput.*, vol. 14, no. 7, pp. 8839–8850, Jul. 2023.
- [37] P. Pirozmand, H. Jalalinejad, A. A. R. Hosseinabadi, S. Mirkamali, and Y. Li, "An improved particle swarm optimization algorithm for task scheduling in cloud computing," *J. Ambient Intell. Humanized Comput.*, vol. 14, no. 4, pp. 4313–4327, Apr. 2023.
- [38] K. Dubey and S. C. Sharma, "A hybrid multi-faceted task scheduling algorithm for cloud computing environment," *Int. J. Syst. Assurance Eng. Manage.*, vol. 14, no. S3, pp. 774–788, Jul. 2023.
- [39] M. Yadav and A. Mishra, "An enhanced ordinal optimization with lower scheduling overhead based novel approach for task scheduling in cloud computing environment," *J. Cloud Comput.*, vol. 12, no. 1, p. 8, Jan. 2023.
- [40] X. Zhang, "A fine-grained task scheduling mechanism for digital economy services based on intelligent edge and cloud computing," *J. Cloud Comput.*, vol. 12, no. 1, p. 30, Mar. 2023.
- [41] P. K. Bal, S. K. Mohapatra, T. K. Das, K. Srinivasan, and Y.-C. Hu, "A joint resource allocation, security with efficient task scheduling in cloud computing using hybrid machine learning techniques," *Sensors*, vol. 22, no. 3, p. 1242, Feb. 2022.
- [42] F. A. Saif, R. Latip, Z. M. Hanapi, and K. Shafinah, "Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing," *IEEE Access*, vol. 11, pp. 20635–20646, 2023.
- [43] M. Mokni, S. Yassa, J. E. Hajlaoui, M. N. Omri, and R. Chelouah, "Multi-objective fuzzy approach to scheduling and offloading workflow tasks in fog–cloud computing," *Simul. Model. Pract. Theory*, vol. 123, Feb. 2023, Art. no. 102687.
- [44] J. A. J. Sujana, T. Revathi, T. S. S. Priya, and K. Muneeswaran, "Smart PSO-based secured scheduling approaches for scientific workflows in cloud computing," *Soft Comput.*, vol. 23, no. 5, pp. 1745–1765, Mar. 2019.
- [45] M.-T. Zhou, T.-F. Ren, Z.-M. Dai, and X.-Y. Feng, "Task scheduling and resource balancing of fog computing in smart factory," *Mobile Netw. Appl.*, vol. 28, no. 1, pp. 19–30, Feb. 2023.
- [46] S. Iftikhar, M. M. M. Ahmad, S. Tuli, D. Chowdhury, M. Xu, S. S. Gill, and S. Uhlig, "HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments," *Internet Things*, vol. 21, Apr. 2023, Art. no. 100667.
- [47] M. Abd Elaziz, L. Abualigah, and I. Attiya, "Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments," *Future Gener. Comput. Syst.*, vol. 124, pp. 142–154, Nov. 2021.
- [48] M. M. Razaq, S. Rahim, B. Tak, and L. Peng, "Fragmented task scheduling for load-balanced fog computing based on Q-learning," *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–9, Mar. 2022.
- [49] M. Abdel-Basset, N. Moustafa, R. Mohamed, O. M. Elkomy, and M. Abouhawwash, "Multi-objective task scheduling approach for fog computing," *IEEE Access*, vol. 9, pp. 126988–127009, 2021.
- [50] T. Long, P. Chen, Y. Xia, Y. Ma, X. Sun, J. Zhao, and Y. Lyu, "A deep deterministic policy gradient-based method for enforcing service fault-tolerance in MEC," *Chin. J. Electron.*, vol. 33, no. 4, pp. 899–909, Jul. 2024.
- [51] Z. Liu, L. Wan, J. Guo, F. Huang, X. Feng, L. Wang, and J. Ma, "PPRU: A privacy-preserving reputation updating scheme for cloud-assisted vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 1, no. 1, pp. 1–16, Oct. 2023.
- [52] Z. Tong, F. Ye, B. Liu, J. Cai, and J. Mei, "DDQN-TS: A novel bi-objective intelligent scheduling algorithm in the cloud environment," *Neurocomputing*, vol. 455, pp. 419–430, Sep. 2021.
- [53] J. Lu, J. Yang, S. Li, Y. Li, W. Jiang, J. Dai, and J. Hu, "A2C-DRL: Dynamic scheduling for stochastic edge–cloud environments using A2C and deep reinforcement learning," *IEEE Internet Things J.*, vol. 11, no. 9, pp. 16915–16927, May 2024.
- [54] (2016). *HPC2N: The HPC2N Seth Log*. [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload/l_hpc2n/
- [55] (2016). *NASA: The NASA Ames IPCS/860 Log*. [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload/l_nasa_ipsc/
- [56] S. A. Murad, Z. R. M. Azmi, A. J. M. Muzahid, M. K. B. Bhuiyan, M. Saib, N. Rahimi, N. J. Prottasha, and A. K. Bairagi, "SG-PBFS: Shortest gap-priority based fair scheduling technique for job scheduling in cloud environment," *Future Gener. Comput. Syst.*, vol. 150, pp. 232–242, Jan. 2024.
- [57] P. Banerjee, S. Roy, A. Sinha, M. M. Hassan, S. Burje, A. Agrawal, A. K. Bairagi, S. Alshathri, and W. El-Shafai, "MTD-DHJS: Makespan-optimized task scheduling algorithm for cloud computing with dynamic computational time prediction," *IEEE Access*, vol. 11, pp. 105578–105618, 2023.
- [58] S. A. Murad, Z. R. M. Azmi, A. J. M. Muzahid, M. M. H. Sarker, M. S. U. Miah, M. K. B. Bhuiyan, N. Rahimi, and A. K. Bairagi, "Priority based job scheduling technique that utilizes gaps to increase the efficiency of job distribution in cloud computing," *Sustain. Comput., Informat. Syst.*, vol. 41, Jan. 2024, Art. no. 100942.
- [59] R. Jeyaraj, A. Balasubramaniam, N. Guizani, and A. Paul, "Resource management in cloud and cloud-influenced technologies for Internet of Things applications," *ACM Comput. Surveys*, vol. 55, no. 12, pp. 1–37, Dec. 2023.
- [60] Deepak, M. K. Upadhyay, and M. Alam, "Load balancing techniques in fog and edge computing: Issues and challenges," in *Proc. IEEE Int. Conf. Comput., Power Commun. Technol. (IC2PCT)*, Feb. 2024, pp. 25–34.
- [61] M. Alam, M. Shahid, S. Mustajab, and F. Ahmad, "Security driven dynamic level scheduling under precedence constrained tasks in IaaS cloud," *Int. J. Inf. Technol.*, vol. 16, no. 2, pp. 721–729, Feb. 2024.
- [62] M. Alam, M. Shahid, and S. Mustajab, "Security challenges for workflow allocation model in cloud computing environment: A comprehensive survey, framework, taxonomy, open issues, and future directions," *J. Supercomput.*, vol. 80, no. 8, pp. 11491–11555, May 2024.
- [63] F. Ahmad, M. Shahid, M. Alam, Z. Ashraf, M. Sajid, K. Kotecha, and G. Dhiman, "Levelized multiple workflow allocation strategy under precedence constraints with task merging in IaaS cloud environment," *IEEE Access*, vol. 10, pp. 92809–92827, 2022.



S. SUDHEER MANGALAMPALLI (Member, IEEE) is currently an Associate Professor with the Department of Computer Science and Engineering, Manipal Institute of Technology, Bengaluru, Manipal Academy of Higher Education known as MAHE (Institution of Eminence, Deemed to be University). He is a passionate Researcher. He is a member of the Institution of Engineers. Towards his profile, he has 34 publications, which are indexed in Scopus and SCI databases. He authored three text books. His research interests include scheduling in cloud computing, edge computing, and fog computing. He is an active reviewer for various SCI indexed journals.



GANESH REDDY KARRI (Member, IEEE) is currently an Associate Professor Senior Grade-2 with the School of Computer Science and Engineering, VIT-AP University. He is a passionate Researcher. He is a Certified Trainer for Security Analyst profile by NAASCOM. He is guiding six full time Ph.D. scholars with VIT-AP University. Towards his Profile, he has 15 publications indexed in Scopus and SCI indexing. His research interests include cloud computing, network security, fog computing, and edge computing.



SACHI NANDAN MOHANTY (Senior Member, IEEE) received the Ph.D. degree from IIT Kharagpur, India, in 2015, with MHRD scholarship from the Government of India, and the Ph.D. degree from IIT Kanpur, in 2019. He has authored/edited 32 books, published by IEEE-Wiley, Springer, Wiley, CRC Press, NOVA, and DeGruyter. He has guided nine Ph.D. Scholar. He has published 120 international journals of international repute. His research interests include data mining, big data

analysis, cognitive science, fuzzy decision-making, brain-computer interface, cognition, and computational intelligence. He was elected as a fellow of the Institute of Engineers, European Alliance Innovation (EAI) Springer, and a Senior Member of the Computer Society Hyderabad Chapter. He has received four best paper awards during the Ph.D. studies at IIT Kharagpur, International Conference, Beijing, China, and the others at International Conference on Soft Computing Applications organized by IIT Roorkee, in 2013. He was awarded the Best Thesis Award first prize by the Computer Society of India, in 2015. He is also a Reviewer of *Robotics and Autonomous Systems* (Elsevier), *Computational and structural Biotechnology Journal* (Elsevier), *Artificial Intelligence Review* (Springer), and *Spatial Information Research* (Springer).



SHAHID ALI received the B.S. degree in communication engineering from the University of Engineering and Technology Peshawar, Pakistan, in 2015, the M.S. degree in aerospace engineering from Beijing Institute of Technology, Beijing, China, and the Ph.D. degree in signal and information processing from the School of Electronics, Peking University. His research interests include channel estimation, MIMO, OFDM, and channel capacity in wireless communications.



ATIF M. ALAMRI received the Ph.D. degree in computer science from the School of Information Technology and Engineering, University of Ottawa, Canada. In addition, he has been the Director General of Planning, since March 2017. He is currently an Associate Professor of software engineering with the College of Computer and Information Sciences (CCIS), King Saud University. As the Director General of Planning, the most significant responsibilities are to strategically plan

and oversee the development of a five-year education strategy and primary plan to identify the gaps in existing initiatives based on the Kingdom's education primary plan to determine the priorities of the ministry. Timely overseeing the proposals on revisions of portfolio and primary plan initiatives to address the gaps. Supporting the development strategies and initiatives of the K-12 planning, higher education planning, shared services planning, and financial planning. Setting the overall education system objectives is a crucial duty towards overseeing the development of targets and initiatives to achieve targets for the short, medium, and long term for the ministry directorates, offices, and schools respectively. Aligning with the strategic plan by overseeing the definition of detailed targets and KPIs. Lastly, the most important liability of the Director General of Planning is to plan the operations and allocating budget effectively. This involves overseeing the development of the operational plan and budget allocation. Overseeing the compilation of data from other departments to project total funding requirements in order to deliver the desired outcome. Supervising the analyses to understand strategic budgeting tradeoff across initiatives and allocate funds to strategic initiatives across the ministry.



SALMAN A. ALQAHTANI has authored two scientific books and authored/coauthored around 200 journals and conference papers on the topic of his research interests, since 2004. His current research interests include 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 grids, the intelligent IoT solutions for smart cities, dynamic spectrum access, coexistence issues on heterogeneous networks in 5G, industry 4.0 issues, the Internet of Everything, and mobile cloud computing. In addition, his interests also include performance evaluation and analysis of high-speed packet-switched networks, system models, 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.

...