DOI: 10.21917/ijct.2024.0478

MULTI-AGENT INDEPENDENT NON-COOPERATIVE REINFORCEMENT LEARNING FOR LOAD BALANCING IN CLOUD HETEROGENEOUS NETWORKS

Tanu Kaistha[1] and Kiran Ahuja[2]

[1]*Department of Electronics and Communication Engineering, IK Gujral-Punjab Technical University, India*
[2]*Department of Electronics and Communication Engineering, DAV Institute of Engineering and Technology, India*

Abstract

Cloud services are now seeing significant advancements and have witnessed a growing demand. Hence, implementing Load Balancing is necessary to enhance resource usage by effectively distributing workload among numerous Virtual Machines (VMs). The present research aims to address task scheduling challenges and achieve efficient load balancing for all VMs by implementing a novel non-cooperative load balancing algorithm called Multi-agent Independent Deep Q Networks (MAIDQN-LB) in cloud computing heterogeneous networks. The list of tasks is passed to MAIDQN-LB, which will search for a list of VM to be allocated, maintaining the load of all VMs. This procedure facilitates the identification of optimized VMs, the allocation of workloads based on the optimal solution derived from the analysis and optimize the performance parameters. The performance analysis considers essential parameters, including makespan time, average turnaround time, average response time, degree of imbalance (DI), task rejection rate (TRR), and convergence loss. The findings indicate that MAIDQN-LB demonstrates superior performance compared to the current system, exhibiting enhancements of 1.82% and 0.05% regarding DI and TRR.

Keywords:

Non-cooperative, Multi-Agent, Independent, Deep Q Networks, Cloud Heterogeneous Networks

1. INTRODUCTION

The use of the cloud for computing heterogeneous networks frequently and concurrently experiences a significant amount of requests from various geographical origins. Users have access to various computing services through cloud computing, which relies on virtual machines as its primary resource. Online access is provided for all these services, including data storage, program applications, computer servers, and network infrastructure [1]. Cloud resources consist of multiple elements, such as storage capacity, memory allocation, and network capabilities, which can be offered as a service to diverse consumers. The assignment of these requests to different cloud providers is done randomly, leading to an unequal distribution of workload among nodes, where specific nodes bear excessive load while others are underutilized [2]. This imbalance has detrimental implications for the system's functioning, highlighting the necessity for implementing effective Load Balancing (LB) solutions in heterogeneous networks within the context of cloud environment. The principal aim of LB is to mitigate the risk of any component becoming excessively burdened, guaranteeing the optimal utilization of resources and the effective and dependable operation of the system [3].

LB is of two types, static and dynamic. Dynamic is further classified into cooperative and non-cooperative LB [4]. Cooperative LB involves the joint decision-making of participating entities, such as servers, nodes, or resources, to achieve LB. Individuals exchange data regarding their present condition, encompassing their ongoing tasks, computational capabilities, and accessibility. However, the necessity of information sharing, and collaboration might lead to increased communication overhead among the institutions involved. Utilizing a central load balancer introduces a potential vulnerability as it becomes a singular point of failure, posing a risk of disrupting the LB mechanism [5]. On the other side, non-cooperative refers to a scenario where individual entities, such as servers, nodes, or resources, make autonomous LB decisions without engaging in information sharing. Each individual entity makes decisions on the management of its incoming workload exclusively based on its local information, which includes factors such as its present utilization level and the resources currently accessible to it [6]. Non-cooperative LB removes the drawback of cooperative sharing by making its own decisions. LB mainly aims to optimize the performance parameters by minimizing the makespan and response time according to different users' needs [7]. To improve the performance of LB, several non-cooperative artificial intelligence techniques [8] are there. One of the techniques is independent reinforcement learning (IRL) [9]. IRL is a non-cooperative LB technique that focuses on decentralized or distributed learning systems in which several agents or learners operate in the same environment simultaneously, and each agent learns its own policy autonomously by interacting with the environment.

To improve the processing speed of tasks and minimize the response time, a non-cooperative multi-agent independent deep Q network for LB (MAIDQN-LB) in cloud heterogeneous networks is proposed. Initially, a buffer of tasks has been passed to MAIDQN to determine the most appropriate VM for the given work. The MAIDQN model effectively chooses VM actions by considering the present LB state of all the VMs. By doing so, the appropriate VM is selected for allocation. The performance of the proposed MAIDQN-LB is computed using six parameters comprising makespan time, average turnaround time, average response time, loss, degree of imbalance (DI), and task rejection rate (TRR). The main contributions of the research are:

- To develop a non-cooperative multi-agent independent deep Q network for efficient LB in cloud heterogeneous networks.
- To minimize the response and makespan time while balancing the load using the proposed MAIDQN-LB.
- To assess the system's efficiency by optimizing its performance concerning different parameters, including average turnaround time, average response time, makespan time, DI, TRR, and convergence loss.

The subsequent section of the paper will be delineated as follows. Section 2 encompasses the literature review, which is subsequently followed by the proposed research outlined in

Section 3. Section 4 provides a detailed description of the experimental setup and presents the results obtained. Section 5 offers a conclusion of the findings and discusses potential avenues for further research.

2. LITERATURE REVIEW

Various researchers have worked on LB techniques to balance the workload in heterogeneous networks; for example, Sourav et al. [10] presented a non-cooperative model for balancing the workloads among the cloudlets in a cloud computing heterogeneous environment. A computational algorithm has been developed that utilizes the Nash equilibrium. From findings, it has been found that the cloudlets can optimize their utilities effectively by implementing the Nash equilibrium to offload the workload from overloaded machines. Ali et al. [11] introduce the concept of fuzzy logic for LB in a cloud environment. The fuzzy logic work performed well by reducing the response time and exhibited superior performance compared to alternative methods. Stavros et al. [12] introduced a task LB technique by utilizing finite state Markov prose to manage the workload over multiple VMs. A centralized server called a load balancer (LBer) is employed for fair task allocation among VMs. The experimental findings prove that the suggested algorithm performed better than various existing state-of-the-art methods regarding response time and DI. Similarly, Amine et al. [13] introduced Modified particle swarm optimization (PSO) along with game theory for LB in heterogeneous networks. A non-cooperative game theory was used, reducing the multiple users' response time. The simulation results indicate that the presented approach was superior in terms of response time. A load scheduling algorithm called HDCBS was anticipated by Wenwei et al. [14] for LB in heterogeneous cloud data centers. A queuing theory has been employed for the representation of computing nodes and to compute the average response time. The convex optimization theory assigns the work to different data centers. The simulation findings indicate that the HDCBS improved the performance by 95% in the context of task scheduling.

Moving ahead, Octavio et al. [15] presented an agent-based cooperative LB technique to balance the load in data centers. Agents have been utilized that employ heuristics to determine the VM to which a task can be migrated. The findings indicate that agents, by engaging in independent and adaptive collaboration, have effectively distributed and managed workloads in a manner that surpasses centralized methods. Avadh et al. [16] presented a CO-evolutionary framework based on Differential Evolution (CODE) to solve the LB problem aiming to abate both the response time and the imbalance in server utilization. The experimental findings demonstrate that CODE effectively lessens both the response time of jobs and the imbalance in server utilization. Ali et al. [17] introduced a hybrid method [18] that combines state-action-reward-state-action (SARSA) learning with a genetic algorithm for LB in cloud heterogeneous networks. During the initial stage, the intelligent agents engage in task scheduling as part of the learning process when they explore the workflow. Next, every resource is assigned to an agent, with the objective of optimizing its consumption through the learning process of the respective agent. The process involves carefully selecting a suitable collection of tasks that optimizes the utilization of available resources. A genetic algorithm has been employed to facilitate the convergence of the agents inside the suggested method, with the ultimate objective of achieving global optimization. The genetic algorithm utilized in this study aimed to optimize resource consumption and LB by considering job deadlines in its fitness function. The experimental findings demonstrate that the suggested method effectively decreases makespan, boosts resource usage, and improves LB compared to the existing literature. Devaraj et al. [19] introduced a hybrid of firefly (FF) and an improved multi-objective PSO (IMPSO) called FFIMPSO to balance the load on various VMs. Firefly computed the search space, and IMPSO was used to discover the enhanced response required to schedule the task on multiple VMs. The findings were compared with individual FF, improved PSO (IPSO), and combined FF-IPSO. It was also compared with various existing algorithms comprising round robin (RR) [20], shortest job first (SJF) [20], first come first serve (FCFS) [20], weighted RR (WRR) [20], diffusive LB (DLB) [21] and LB Bayes and clustering (LB-BC) [21]. From the findings, it was found the FFIMPSO outperformed all the existing works efficiently by balancing the load on the VMs.

Table.1. LB techniques in Cloud Heterogeneous Networks

Work	Technique	Performance Parameters					
		Makespan Time	Turn Around Time	Response Time	DI	TRR	Loss
Sourav et al. [10]	Pure-strategy Nash Equilibrium	✓	×	✓	×	×	×
Ali et al. [11]	Fuzzy Logic	×	×	✓	×	×	×
Stavros et al. [12]	Markov Decision Process	✓	×	✓	✓	×	×
Amine et al. [13]	Modified PSO and Nash Equilibrium	✓	×	✓	×	×	×
Wenwei et al. [14]	Queuing and convex optimization theory.	✓	×	✓	×	✓	×
Octavia et al. [15]	Agent-based LB	×	×	✓	✓	✓	×
Avadh et al. [16]	CODE	✓	×	✓	×	×	×
Ali et al. [17]	SARSA and genetic algorithms	✓	×	✓	×	×	×
Devraj et al. [19]	FFIMPSO	✓	✓	✓	×	×	×
Sami et al. [22]	WFBLBA, WFDBLBA	✓	×	✓	×	×	×
MAIDQN-LB	Multi-agent Independent Deep Q Networks	✓	✓	✓	✓	✓	✓

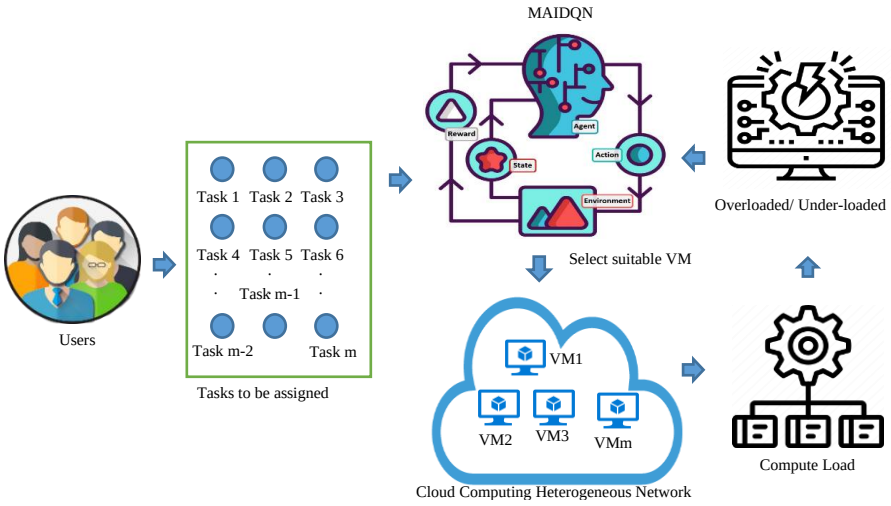


Fig.1. Workflow of MAIDQN-LB

Sami et al. [22] presented novel LB algorithms called Worst-Fit-Based LB Algorithm (WFBLBA) that effectively address the needs of cloud service providers by addressing the bin stretching problem. The experiment was conducted using the CloudSim [23] simulator and the findings demonstrated that the WFBLBA exhibit superior makespan and waiting time compared to the existing works. The Table.1 gives a discussion about the related work on LB techniques.

3. PROPOSED WORK - MAIDQN-LB

In this research, multi-agent independent non-cooperative deep Q network LB (MAIDQN-LB) has been proposed to balance the workload of Virtual Machines (VMs). The main aim of the proposed work is to reduce response time. Instead of single agent, multiple agents act independently to select the specific VM to assign the load. These agents interact independently in a non-cooperative environment, meaning they have their own objectives and do not work together cooperatively. Initially, a buffer of tasks has been submitted to MAIDQN to determine the most appropriate VM for the given work. The MAIDQN model effectively chooses VM actions by considering the present LB state of all the VMs and provides the selected VM to the cloud computing heterogeneous platform model. The environment offers a reward that signifies whether the job has been declined or completed successfully to train the MAIDQN. During the training phase, the MAIDQN model transitions from initial inaccuracies to improved decision-making for selecting the most appropriate VM for a given task. The performance of the presented MAIDQN-LB is computed using six parameters comprising makespan time, average turnaround time, average response time, loss, DI, and TRR. The complete working of MAIDQN-LB is given in Fig.1:

3.1 CLOUD COMPUTING HETEROGENEOUS NETWORK

The cloud computing heterogeneous network consists of m number of virtual machines (VMs) comprising VM_1, VM_2, …., VM_m, which are required to perform some suitable tasks. The processing speed is considered one of the most crucial aspects of VMs. This study examines the disparity in processing speed among VMs and observes that, when subjected to identical conditions, a VM with a higher rate accomplishes a given task in a shorter duration than a VM with a lower speed. The determination of the earliest start time (EST) for a task is contingent upon both the task's arrival time and the VMs idle time. If the task arrival time is greater than the idle time, the arrival time is considered as EST. Conversely, if the task arrival time is less than the idle time, then the idle time is regarded as the EST. The idle time can be task startup time if it is the first VM or the last time the task is completed if it is other than the first VM. For each task, MAIDQN selects the VM which can execute the task. The load on VMs is getting more significant as the tasks are being executed. If it goes to the maximum capacity of the VM, the task will be rejected for execution. Once the tasks are executed, a load is calculated for VMs, and if the load is more than the MAIDQN procedure is repeated until the balancing is done.

3.2 MULTI-AGENT INDEPENDENT DEEP Q NETWORK (MAIDQN)

In the present research, MAIDQN-LB is proposed to balance the workload in cloud heterogeneous networks. MAIDQN is an extension of Deep Q-Networks (DQN). In DQN [24], the agent interacts with the environment iteratively to make decisions. Each time the agent interacts, the corresponding Q-values of the agents representing action-state values are updated accordingly, which enables the agent to approximate the best policy for a solitary agent inside the provided environment. On the other side, in multi-agent independent DQN, the interactions have been done by multiple agents in which the action taken by one agent will automatically influence the experiences of the different agents involved in the interaction. The complete working of MAIDQN is expressed as follows:

- *Environment Modeling and Initialization:* The environment is modeled for multi-agent interactions using a 5-step Markov decision process (MDP) including states (*st)*, actions *(ac)*, discount factor (γ), reward function (r), and transition probability (p). The state observations, action spaces, and rewards have been provided to the agents by the

environment. Once the environment is modeled, each agent is assigned an individual DQN, which includes target networks, hyperparameters, and experience replay buffers.

- *Action Selection:* The action is selected by an agent at each iteration by making a decision considering the present state of the agent and its unique DQN. This selection is based on the epsilon greedy approach, i.e., the agent selected the random action with a specific probability (p). On the other side, the highest Q-value action can also be selected. Based on the selected action, the combined actions of all the agents have been used to interact with the environment, which in turn returns the next state and reward function. This is done for each episode or epoch.
- *Experience collection and Replay:* Once the action has been selected, the next state, action, and reward have been collected by the agents, which are stored in the experience replay buffer. This replay buffer is used during several points in training where some mini-batches of experience have been utilized to update the DQN of each individual agent. The DQN algorithm operates a neural network to estimate the Q-function, denoted as *Q(st, ac)*. The neural network receives the environmental states as input and generates Q-values for every feasible action (ac). The neural network's weights are iteratively adjusted throughout the training process using the stochastic gradient descent algorithm to minimize the loss function. The loss function is computed using a Bellman equation, which quantifies the discrepancy between the predicted and actual Q-values. The Bellman equation is given by the following Eq.(1):

$$Q(s_t,a_c) \leftarrow Q(s_t,a_c)+\alpha*[r+\gamma*\max(Q(s_t',a_c'))-Q(s_t,a_c)] \quad (1)$$

where the temporal difference (TD) is given by $r+\gamma*\max(Q_{target}(s',a'))-Q(s,a)$, the learning rate is denoted by α, and the reward and discount factor is given by r and γ. The maximum Q value among all states and actions is given by max $(Q(s_t', a_c'))$.

- *Multi-agent Independence*: In the MAIDQN framework, individual agents uphold their Deep DQN, and the learning process is conducted autonomously without any interdependence on other agents. Agents gather their experiences and execute Q-learning updates according to their own observations and behaviors. The autonomy provided enables agents to acquire their respective policies without direct communication or collaboration with other agents.
- *Update DQN and target network:* The weights of each individual DQN are updated using TD to minimize the Q-value loss. This is followed by the updation of the target network, which is done by copying the weights of DQN. By doing this, the learning process is stabilized, and it helps in preventing divergence.
- *Convergence:* The training procedure is iterated until the agents of DQN reach a state of convergence, approximating the ideal action-value function. Subsequently, the agents' policies are employed to make decisions within the given environment.

In the current research, the state space consists of a load of each individual VM ($load_1, load_2, \ldots load_m$), and the action state is a selection of VM from multiple VMs, i.e., VM_1, VM_2, …. VM_m, for efficient LB. When the DQN algorithm chooses a VM for a given job, the action value 1 is assigned to the particular VM, 0 is assigned to the remaining VMs. For instance, in the scenario when the m^{th} virtual machine is chosen to carry out the assigned task, the vector is represented as (0,0,...,1). Algorithm 1 gives a pseudocode of MAIDQN-LB.

Algorithm 1: Pseudocode of MAIDQN-LB

Input: Tasks, VMs
Output: Balanced VMs, Performance parameters
Begin:
1. User Request for task assignment
2. Task (T) to be allocated present in a queue
3. For each task in T // Call MAIDQN
4. Select for suitable VM from a cloud computing heterogeneous network
5. Create a state space for VMs load, i.e., $load_1, load_2, \ldots load_m$
6. Create an action state of VM_1, VM_2, …. VM_m
7. Select an action considering present space and independent DQN
8. Compute Experience and Q-values for individual independent DQN
9. Update DQN and Target network
10. Compute Convergence loss
11. End For
12. Compute Load on each VM
13. If (under-loaded or overloaded)
14. Repeat steps 3-11
15. Else
16. Stop
17. Compute Performance parameters

End

4. EXPERIMENTAL SETUP AND RESULTS

In this current research, MAIDQN-LB is presented for LB in cloud computing heterogeneous networks. To implement the proposed work, Python 3.9 has been utilized. The minimum hardware requirement includes Windows 8, 8 Gb Ram, and i5 processor. CloudSim is used to perform VM simulations. The libraries used are tensorflow, matplot lib, numpy, random, math, and time.

4.1 PERFORMANCE PARAMETERS

To compute the performance of proposed MAIDQN, six parameters comprising makespan time, average turnaround time, average response time, DI, TRR, and convergence loss at different hyper-parameters have been computed and are described below:

- *Makespan time:* The total time to compute all the tasks is known as makespan time.
- *Turnaround time:* It is the time interval between the arrival of a task and completion of a particular task.
- *Response time:* Response time pertains to the duration a system or component requires to react to a particular event or solicitation.
- *Convergence loss:* Convergence loss pertains to the numerical representation of the loss function throughout the iterative training procedure of MAIDQN as it approaches a state of stability.

- *Degree of Imbalance:* It is the imbalance of loads between the VMs, and the smaller the value of the DI, the more the load is balanced between VMs. The DI is computed using the following Eq.(2):

$$DI=(Maximum\ load-Minimum\ load)/(Average\ load) \quad (2)$$

- *Task Rejection Rate:* When assigning the tasks, a task is rejected if the corresponding task violates the maximum load condition. The TRR is computed by the given Eq.(3):

$$TRR=(No.\ of\ rejected\ tasks)/(Total\ tasks) \quad (3)$$

4.2 RESULTS

To compute the results of MAIDQN-LB, different types of scenarios have been considered. In the first case scenario, the different number of tasks comprising 2000, 4000, 6000, 8000, and 10000 tasks have been assigned to 8 VMs maintaining the load balance. The results showed that the MAIDQN performed effectively with a total turnaround time of 20.57 ms, 28,63 ms, 34.4 ms, 40.12 ms, and 49.28 ms, respectively. Similarly, the average response time for all the tasks has been computed, and it has been found that the MAIDQN performs effectively with a minimum response time of 18.36 ms, 25.32 ms, 29.43 ms, 35.45 ms, and 45.11 ms for 2000, 4000, 6000, 8000, and 10000 tasks respectively. The results of turnaround time and response time are given in Table 2.

Table.2. Average turnaround time and response time for different no. of tasks assigned to 8 VMs

No. of Tasks	Turn Around time (ms)	Response Time (ms)
2000	20.57	18.36
4000	28.63	25.32
6000	34.4	29.43
8000	40.12	35.45
10000	49.28	45.11

Moreover, the line plots of turnaround time and response time have been plotted for 2000, 4000, 6000, 8000, and 10,000 tasks assigned to 8 VMs have been plotted and shown in Fig.2.

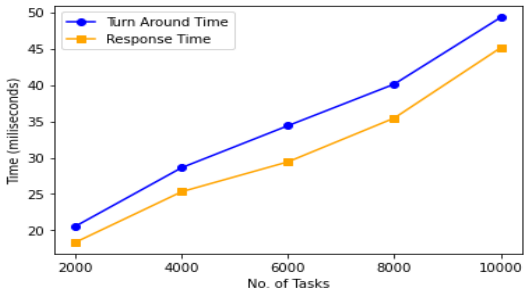


Fig.2. Line plots for turnaround time and response time for different no. of tasks

The trend in line plots shows the effectiveness of the proposed MAIDQN-LB. In the second case scenario, the number of VMs has been increased to 4, 8, 16, 24, and 32 VMs, and 8000 tasks have been allocated. Then the corresponding turnaround and response times are computed and shown in Table 3. The results show that the MAIDQN-LB performed effectively by reducing the response time and turnaround time, respectively. Moreover, the makespan time is also computed and it is found that MAIDQN-LB performed well with a makespan time of 80 s, 92 s, 113 s, 125 s, and 136 s for 4, 8, 16, 24, and 32 VMs respectively.

Table.3. Turnaround time and response time for different no. of VMs

No. of VMs	Turn Around time (ms)	Response Time (ms)
4	50.5	45.16
8	40.12	35.45
16	28.20	21.36
24	21.10	15.5
32	15.28	9.31

The line plots have also plotted for turnaround time and response time for different no. of VMS, and it has been found that MAIDQN-LB performed effectively with 50.5 ms, 40.12 ms, 28.20 ms, 21.10 ms, and 15.28 ms of turnaround time and 45.16 ms, 35.45 ms, 21.36 ms, 15.5 ms, and 9.31 ms of response time respectively. Fig.3 shows the line plot for turnaround and response times for different no. of VMs.

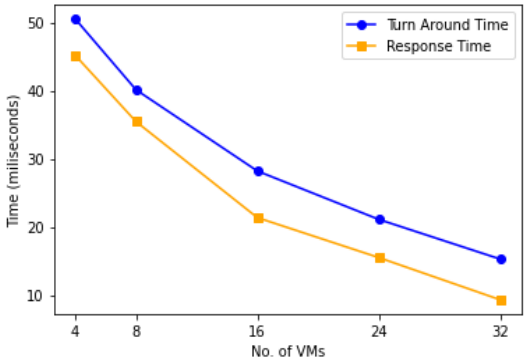


Fig.3. Line plots for turnaround time and response time for different no. of VMs

In the third case scenario, the DI and TRR has been computed for different activation functions [25] comprising relu, selu, elu, tanh, and sigmoid functions, and shown in Table 4 below. The total VMs used is 8, and 8000 tasks are assigned to them. The MAIDQN-LB runs for 100 episodes with a learning rate of e^-1 (~0.36788). The experimental findings show that the MAIDQN-LB performed effectively with a DI and TRR values of 0.150, 0.176, 0.170, 0.207, 0.344, and 0.05994. 0.05996, 0.005991, 0.05933, and 0.3000 for the above-mentioned activation functions.

Table.4. DI and TRR of different activation functions.

Activation Function	DI	TRR
relu	0.150	0.0599493
selu	0.176	0.0599697
elu	0.170	0.0599162
tanh	0.207	0.0593345
sigmoid	0.344	0.3000553

Moving ahead, the bar plot and line plot combination have been plotted for the DI and TRR, as shown in Fig.4. As lesser the

value of the DI and TRR, the more balanced the load on VMs is. Therefore, the results show that the MAIDQN-LB performed best on the tanh activation function, followed by the relu, selu, and elu functions, and performed worst for the sigmoid activation function.

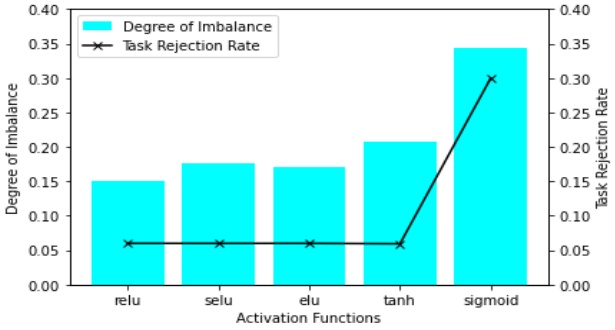


Fig.4. Plot of a DI and TRR for different activation functions

In the following case, the DI and TRR of MAIDQN-LB for different learning rates comprising e^-1 (~0.3678), e^-2 (~0.1353), e^-3 (~0.0497), and e^-4 (0.0183) have been computed with relu activation function and for 100 episodes. Table 5 shows the results of MAIDQN-LB, and it has been found that the MAIDQN-LB gives 0.069, 0.150, 0.104, and 0.295 degrees of imbalance values and 0.050, 0.230, 0.278, and 0.350 TRR value for e^{-1}, e^{-2}, e^{-3}, and e^{-4} respectively.

Table.5. DI and TRR for different learning rates

Learning Rates	DI	TRR
e^{-1}	0.069	0.05036
e^{-2}	0.150	0.23076
e^{-3}	0.104	0.27864
e^{-4}	0.295	0.35097

Furthermore, the plots for the DI and TRR for MAIDQN-LB's different learning rates and shown in Fig.5. The MAIDQN-LB performed best on e^{-1} learning rate and worst on e^{-4} learning rate values. In the next case, the convergence loss value has been computed for 8000 tasks assigned to 8 VMs. The MAIDQN-LB runs for 100 episodes with different learning rates and different activation functions. The convergence plot is shown in Fig.6(a) and Fig.6(b). the Fig.6(a) represents the convergence plot on different learning rates. The convergence loss is higher for e^{-3}, followed by e^{-2}, e^{-4}, and e^{-1} for the first 35 episodes. This is followed by lower loss for all the learning rates for the remaining episodes. The Fig.6(b) shows the convergence loss of MAIDQN-LB on different activation functions, and it shows the highest convergence for the Selu activation function. After 40 episodes, fewer fluctuations and a lower loss value are achieved. From the loss value, it is found that MAIDQN-LB performed best on the e^{-1} learning rate, and the Relu activation function shows a lesser loss than other learning rates and activation functions.

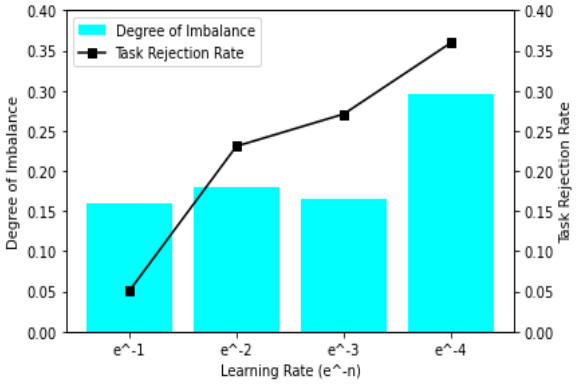
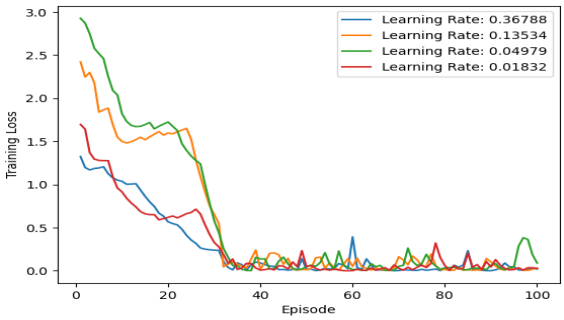
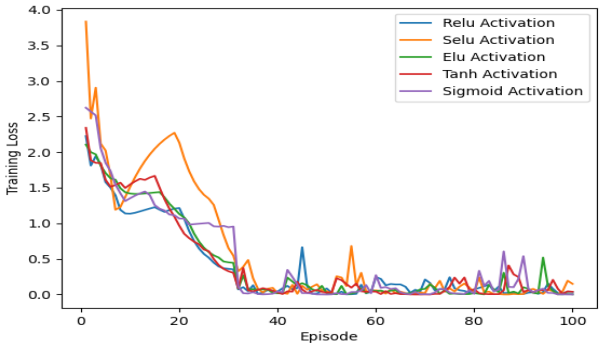


Fig.5. Plots on the DI and TRR for different learning rates.



(a) Different Learning rates



(b) Different activation functions

Fig.6. Convergence loss for MAIDQN-LB on different learning rates and activation functions

Moving ahead, the MAIDQN-LB is compared with several existing works comprising opportunistic LB (OLB) [26], round robin, and random LB techniques, and their corresponding DI and TRRs have been computed. The Table.6 shows the comparison results, and it has been found that MAIDQN-LB performed better than the existing OLB, round robin, and random in case of a DI. It gives a DI value of 0.23, 0.13, 0.15, and 0.27 for 2000, 5000, 8000, and 10,000, respectively. Only for TRR, OLB performed better; for rest in all cases, MAIDQN-LB performed best with 0.07, 0.06, 0.05, and 0.06 TRR values for 2000, 5000, 8000, and 10000 tasks, respectively.

Table.6. Comparison results of MAIDQN-LB with existing works in increasing no. of tasks.

Task / Work	OLB		Round Robin		Random		MAIDQN	
	DI	TRR	DI	TRR	DI	TRR	DI	TRR
2000	0.85	0.01	1.3	0.08	0.9	0.17	0.23	0.07
5000	0.82	0.01	1.5	0.09	0.95	0.175	0.13	0.06
8000	0.81	0.01	1.5	0.10	0.96	0.16	0.15	0.05
10000	0.80	0.01	1.5	0.11	0.99	0.18	0.27	0.06

Additionally, line plots have been made for comparison of MAIDQN-LB with existing works, as shown in Fig.7. Findings show that MAIDQN-LB shows an improvement of 0.62%, 0.69%, 0.66%, and 0.53% when compared to the second-best performing OLB technique in case of DI. Similarly, MAIDQN-LB improves the performance by 0.01%, 0.03%, 0.05%, and 0.05% when compared with round robin regarding TRR.

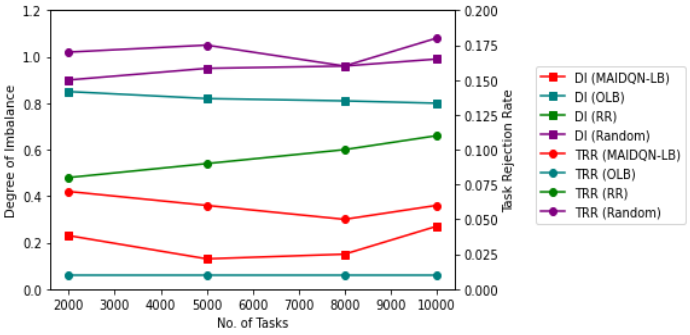


Fig.7. Line plot for comparison of MAIDQN-LB with increasing no. of tasks

In the next case scenario, MAIDQN-LB is compared with existing work in terms of increasing no. of VMS (VM4, VM8, VM16, VM32) and with 8000 tasks. Table 7 shows the comparison results, and it has been found that MAIDQN-LB performed best with 0.05, 0.15, 0.20, and 0.28 degrees of imbalance and 0.08, 0.5, 0.03, and 0.02 TRR value for VM4, VM8, VM16, and VM32 respectively.

Table 7. Comparison of MAIDQN-LB with existing works for increasing no. of VMs

VMs / Work	OLB		Round Robin		Random		MAIDQN-LB	
	DI	TRR	DI	TRR	DI	TRR	DI	TRR
4	0.70	0.01	1.1	0.17	0.9	0.20	0.05	0.08
8	0.81	0.01	1.5	0.10	0.96	0.16	0.15	0.05
16	1.3	0.01	1.7	0.05	1.3	0.10	0.20	0.03
32	2.1	0.01	2.5	0.06	2.2	0.08	0.28	0.02

Moreover, that line plots for comparison are also plotted and shown in Fig.8. The 8000 tasks have been assigned to 4,8,16, and 32 VMs and it has been found that MAIDQN-LB improves the performance by 0.65%, 0.66%, 1.1%, and 1.82% in terms of DI when compared with OLB. Similarly, it shows an improvement of 0.09%, 0.05%, 0.02%, and 0.04% compared to the round-robin regarding TRR.

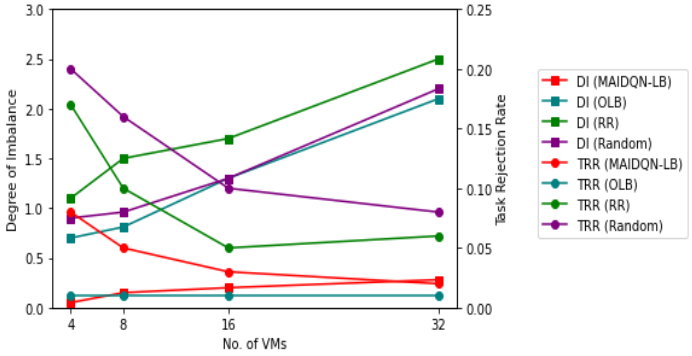


Fig.8. Comparison plots of MAIDQN-LB with increasing no. of VMs

Additionally, MAIDQN-LB is also compared with some static load balancing comprising RR [19], SJF [19], and FCFS [19], and some dynamic load balancing techniques comprising Firefly [19], IPSO [19], FF-IPSO [19], FFIMPSO [19], and LW-PSO [27] in terms of Average TAT, and RT for 32 VMs. Table 8 shows the comparison results for TAT, RT, and load and it is found that RSLbestPSO performed best with average TAT of 15.28ms and average RT of 9.31ms, respectively. It takes less TAT and RT as compared to existing works.

Table.8. Comparison results of MAIDQN-LB in terms of Average TAT, and RT

Work	Average TAT (ms)	Average RT (ms)
RR [19]	41.98	30.5
SJF [19]	41.56	30.24
FCFS [19]	41.87	30.84
Firefly [19]	55.54	48.87
IPSO [19]	57.74	49.23
FF-IPSO [19]	22.13	15.21
FFIMPSO [19]	21.09	13.58
LW-PSO [27]	20.56	12.96
MAIDQN-LB	15.28	9.31

5. CONCLUSION AND FUTURE WORK

Cloud services have experienced notable breakthroughs and a noticeable increase in demand. Therefore, incorporating LB is vital to optimizing resource utilization by efficiently distributing workload across several Virtual Machines (VMs). The main goal of this research is to tackle the difficulties associated with job scheduling and attain optimal LB for all VMs within cloud heterogeneous networks. To achieve this, a non-cooperative LB method known as Multi-agent Independent Deep Q Networks (MAIDQN-LB) is proposed to balance the workload by lessening the response makespan times. In the user request model, the job is first submitted to the MAIDQN model, which then selects the best virtual machine (VM) for the supplied work. The cloud heterogeneous platform receives the selected VMs from the MAIDQN model after considering the present LB status of the VMs. The MAIDQN model is trained using the environment, which offers a reward that indicates whether the task has been

approved or successfully finished. The MAIDQN model changes from initial inaccuracies to increasing accuracy during training in its decision-making process for choosing the best VM for a particular task. The performance of proposed MAIDQN-LB is computed using six metrics comprising makespan time, average turnaround time, average reaction time, loss, DI, and TRR. The results indicate that the proposed MAIDQN-LB performed effectively when 8000 tasks were assigned to 8VMs with 40.12ms, 35.45ms, 125s, 0.15, and 0.05 value of average turnaround time, average response time, makespan time, DI, and TRR respectively. In the future, it is anticipated that other dynamic LB techniques will be employed to distribute the workload across heterogeneous cloud networks effectively. Moreover, MAIDQN-LB can be improved by dynamically reevaluating its task allocation decisions based on real-time feedback from VMs. If certain VMs become overloaded or underutilized, the algorithm can dynamically redistribute tasks to balance the load more effectively. Additionally, it can consider factors like VM performance metrics (e.g., CPU utilization, memory usage) and prioritize task allocation accordingly.

REFERENCES

[1] A. Rashid and A. Chaturvedi, "Cloud Computing Characteristics and Services a Brief Review", *International Journal of Computer Sciences and Engineering*, Vol. 7, No. 2, pp. 421-426, 2019.

[2] S. S. Tripathy, "State-of-the-Art Load Balancing Algorithms for Mist-Fog-Cloud Assisted Paradigm: A Review and Future Directions", Springer, 2023.

[3] N. Kumar and N. Mishra, "Load Balancing Techniques: Need, Objectives and Major Challenges in Cloud Computing- A Systematic Review", *International Journal on Computer Applications*, Vol. 131, No. 18, pp. 11-19, 2015.

[4] D.A. Shafiq, N.Z. Jhanjhi and A. Abdullah, "Load Balancing Techniques in Cloud Computing Environment: A Review", *Journal of King Saud University - Computer and Information Sciences*, Vol. 34, No. 7, pp. 3910-3933, 2022.

[5] M.H. Al Bowarab, N.A. Zakaria and Z. Zainal Abidin, "Load Balancing Algorithms in Software Defined Network", *International Journal of Recent Technology and Engineering*, Vol. 7, No. 6, pp. 686-693, 2019.

[6] K.S. Gill, S. Saxena and A. Sharma, "GTM-CSec: Game Theoretic Model for Cloud Security based on IDS and Honeypot", *Computer Security*, Vol. 92, pp. 101-109, 2020.

[7] L. Zhang, K. Li, Y. Xu, J. Mei, F. Zhang and K. Li, "Maximizing Reliability with Energy Conservation for Parallel Task Scheduling in a Heterogeneous Cluster", *Information Science*, Vol. 319, pp. 113-131, 2015.

[8] A. Dhillon, A. Singh and V.K. Bhalla, "Biomarker Identification and Cancer Survival Prediction using Random Spatial Local Best Cat Swarm and Bayesian Optimized DNN", *Applied Soft Computing*, Vol. 146, pp. 110649-110658, 2023.

[9] J. Foerster, "Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning", *Proceedings of International Conference on Machine Learning*, Vol. 3, pp. 1879-1888, 2017.

[10] S. Mondal, G. Das and E. Wong, "A Game-Theoretic Approach for Non-Cooperative Load Balancing among Competing Cloudlets", *IEEE Open Journal of the Communications Society*, Vol. 1, No. 1, pp. 226-241, 2020.

[11] A. Pourghaffari, M. Barari and S. Sedighian Kashi, "An Efficient Method for Allocating Resources in a Cloud Computing Environment with a Load Balancing Approach", *Concurrency and Computation: Practice and Experience*, Vol. 31, No. 17, pp. 1-15, 2019.

[12] S. Souravlas, S.D. Anastasiadou, N. Tantalaki and S. Katsavounis, "A Fair, Dynamic Load Balanced Task Distribution Strategy for Heterogeneous Cloud Platforms Based on Markov Process Modeling", *IEEE Access*, Vol. 10, pp. 26149-26162, 2022.

[13] A. Mrhari and Y. Hadi, "A Load Balancing Algorithm in Cloud Computing Based on Modified Particle Swarm Optimization and Game Theory", *Proceedings of IEEE World Conference on Complex Systems*, pp. 1-6, 2019.

[14] W. Cai, J. Zhu, W. Bai, W. Lin, N. Zhou and K. Li, "A Cost Saving and Load Balancing Task Scheduling Model for Computational Biology in Heterogeneous Cloud Datacenters", *Journal of Supercomputing*, Vol. 76, No. 8, pp. 6113-6139, 2020.

[15] J.O. Gutierrez-Garcia and A. Ramirez-Nafarrate, "Agent-Based Load Balancing in Cloud Data Centers", *Cluster Computing*, Vol. 18, No. 3, pp. 1041-1062, 2015.

[16] A. Kishor and R. Niyogi, "Multi-Objective Load Balancing in Distributed Computing Environment: An Evolutionary Computing Approach", *Proceedings of the ACM Symposium on Applied Computing*, pp. 170-175, 2020.

[17] A. Asghari, M.K. Sohrabi and F. Yaghmaee, "Task Scheduling, Resource Provisioning, and Load Balancing on Scientific Workflows using Parallel SARSA Reinforcement Learning Agents and Genetic Algorithm", *Journal of Supercomputing*, Vol. 77, No. 3, pp. 2800-2828, 2021.

[18] K. Ahuja, B. Singh and R. Khanna, "Network Selection in Wireless Heterogeneous Environment by C-P-F Hybrid Algorithm", *Wireless Personal Communications*, Vol. 98, No. 3, pp. 2733-2751, 2018.

[19] A.F.S. Devaraj, M. Elhoseny, S. Dhanasekaran, E.L. Lydia and K. Shankar, "Hybridization of firefly and Improved Multi-Objective Particle Swarm Optimization Algorithm for Energy Efficient Load Balancing in Cloud Computing Environments", *Journal of Parallel and Distributed Computing*, Vol. 142, pp. 36-45, 2020.

[20] D. Chaudhary and B. Kumar, "An Analysis of the Load Scheduling Algorithms in the Cloud Computing Environment: A Survey", *Proceedings of International Conference on Industrial and Information Systems*, pp. 1-6, 2015.

[21] J. Zhao, K. Yang, X. Wei, Y. Ding, L. Hu and G. Xu, "A Heuristic Clustering-Based Task Deployment Approach for Load Balancing using Bayes Theorem in Cloud Environment", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 27, No. 2, pp. 305-316, 2016.

[22] S. Dhahbi, M. Berrima and F.A.M. Al Yarimi, "Load Balancing in Cloud Computing using Worst-Fit Bin-Stretching", *Cluster Computing*, Vol. 24, No. 4, pp. 2867-2881, 2021.

[23] A. Sundas and S.N. Panda, "An Introduction of CloudSim Simulation tool for Modelling and Scheduling", *Proceedings of International Conference on Emerging Smart Computing and Informatics*, pp. 263-268, 2020.

[24] W. Du and S. Ding, "A Survey on Multi-Agent Deep Reinforcement Learning: from the Perspective of Challenges and Applications", *Artificial Intelligence Review*, Vol. 54, No. 5, pp. 3215-3238, 2021.

[25] S. Sharma and A. Anidhya, "Understanding Activation Functions in Neural Networks", *International Journal of Engineering Applied Sciences and Technology*, Vol. 4, No. 12, pp. 310-316, 2020.

[26] G. Patel, R. Mehta and U. Bhoi, "Enhanced Load Balanced Min-min Algorithm for Static Meta Task Scheduling in Cloud Computing", *Procedia Computer Science*, Vol. 57, pp. 545-553, 2015.

[27] M. Malik and S. Suman, "Lateral Wolf Based Particle Swarm Optimization (LW-PSO) for Load Balancing on Cloud Computing", *Wireless Personal Communications*, Vol. 125, No. 2, pp. 1125-1144, 2022.