Hindawi
Applied Computational Intelligence and Soft Computing
Volume 2022, Article ID 2131699, 17 pages
https://doi.org/10.1155/2022/2131699

Research Article

Mayfly Taylor Optimisation-Based Scheduling Algorithm with Deep Reinforcement Learning for Dynamic Scheduling in Fog-Cloud Computing

G. Shruthi,[1,2] **Monica R. Mundada**,[3] **B. J. Sowmya**,[3] **and S. Supreeth**[2]

[1]*M S Ramaiah Institute of Technology, VTU, Bengaluru 560054, India*
[2]*School of CSE, REVA University, Bengaluru 560064, India*
[3]*Department of Computer Science and Engineering, M S Ramaiah Institute of Technology, Bengaluru 560054, India*

Correspondence should be addressed to S. Supreeth; supreeth1588@gmail.com

Received 9 June 2022; Revised 18 July 2022; Accepted 22 July 2022; Published 28 August 2022

Academic Editor: Said El Kafhali

Fog computing domain plays a prominent role in supporting time-delicate applications, which are associated with smart Internet of Things (IoT) services, like smart healthcare and smart city. However, cloud computing is a capable standard for IoT in data processing owing to the high latency restriction of the cloud, and it is incapable of satisfying needs for time-sensitive applications. The resource provisioning and allocation process in fog-cloud structure considers dynamic alternations in user necessities, and also restricted access resources in fog devices are more challenging. The global adoption of IoT-driven applications has led to the rise of fog computing structure, which permits perfect connection for mobile edge and cloud resources. The effectual scheduling of application tasks in fog environments is a challenging task because of resource heterogeneity, stochastic behaviours, network hierarchy, controlled resource abilities, and mobility elements in IoT. The deadline is the most significant challenge in the fog computing structure due to the dynamic variations in user requirement parameters. In this paper, Mayfly Taylor Optimisation Algorithm (MTOA) is developed for dynamic scheduling in the fog-cloud computing model. The developed MTOA-based Deep Q-Network (DQN) showed better performance with energy consumption, service level agreement (SLA), and computation cost of 0.0162, 0.0114, and 0.0855, respectively.

1. Introduction

Usually, fog computing incorporates the devices and data centres at the network edge, and it offers an effective solution for resolving various limitations, like high delay time. The numerous heterogeneous devices present in the network edge are interlinked to afford flexible communication, storage, and computing services in a distributed computing system. The correlated functions in fog computing are carried out by means of a local resource pool residing in the closeness of end-users; thereby, the time taken for data transfer is reduced. Consequently, fog computing is not an alternative to cloud computing, although the fog model is an effectual extension, which permits the process at the edge and interacts with the cloud structure. Moreover, fog computing enables the demanded applications and also services to operate on devices, including Road Side Unit (RSU), routers, set-top boxes, gateways, vehicular to vehicular gateways, and access points. The utilisation of fog computing effectively decreases the consumption of energy. Besides, effectual processing and low-cost devices are employed in short distances along with better reliability for decreasing computing delays and optimising the process [1]. The cloud computing model affords storage and computing services over the Internet for providing services for various industries. However, delay-sensitive applications, such as smart city and smart health applications, need computation over a huge number of data transmitted to the central cloud data centre, and it directs to a drop in performance. Furthermore, new patterns of edge computing and fog offer new solutions by conveying resources nearer to the user and afford high energy efficiency and less latency compared to cloud services [2].

In general, fog is a significantly virtualised structure of resource pools along with a decentralised setup, in which computing resources are shared among cloud data centres and clients to enhance the data analytical and computational processing abilities. The task-associated data analytics are assumed to fog through performing data gathering tasks at the network edge, and it addresses various restrictions, like inadequate bandwidth and latency. Moreover, infrastructure and computing resources, namely, computational processing, power, available nodes, cost, and bandwidth, are optimally assigned for operating tasks in cloud or fog structures based on the needs and configurations. Task scheduling with load balancing [3] and resilient processing is a vital process in the visualisation model for decreasing the overhead and costs for both providers and users. Furthermore, it supports the significance of taking effective scheduling and also allocates the tasks among cloud and fog nodes [4]. The cloud systems generally interoperate based on load balancer and task scheduler services, which are not completely automated. Moreover, the task scheduler distributes the computing resources, which is performed at cloud or fog nodes, when the load balancer arranges workload distribution by numerous computing resources. Additionally, task scheduling techniques are utilised for scheduling tasks on computing processors to minimise total makespan without violating the specified restrictions [5, 6]. Besides, various techniques are developed for exploring the problems included in computation offloading. Furthermore, optimisation of offloading decisions and resource allocation, such as distribution of bandwidth, transmit power, and computation resources, are employed for attaining system performance gain, energy efficiency improvement, and reduction of energy consumption and delay [7–9].

In addition, the consideration of various parameters with the heterogeneity of resources and tasks executes a more difficult decision-making process. The feasible solutions are generated, and there are no deterministic polynomial techniques through the increasing amount of resources and tasks, which is termed an NP-hard problem [10]. For solving NP-hard issues, evolutionary techniques, namely, Genetic Algorithms (GA), Particle Swarm Optimisation (PSO), Ant Colony Optimisation (ACO), and Bees Life Algorithm (BLA), are utilised. Moreover, they are extensively utilised for solving task scheduling issues owing to the search capability and global optimisation of GA [11]. The employed GA executes better performance in task scheduling issues [12]. In recent days, cooperation and interplay between cloud and fog have received more attention [2]. The reinforcement learning solutions are more precise because these methods are constructed from definite measurements, and they can find the composite relationships among various interdependent parameters. The modern approaches discovered various value-based reinforcement learning approaches for optimising some factors of Resource Management Systems (RMS) in distributed surroundings. These techniques accumulate Q value function in a table or utilise the neural network for every state of edge cloud structure, and it is a probable cumulative reward in the reinforcement learning system. The tabular value-driven reinforcement learning approaches experienced the issue of restricted scalability; thus, different deep learning techniques, namely, Deep Q-Network (DQN), were designed, in which neural network approximates Q value. However, the previous works specified that value-driven reinforcement approaches are not appropriate for extremely stochastic structures [13].

The main intent of this paper is to design an effectual dynamic scheduling approach using a hybrid optimisation model. Here, the DQN model [14] is designed for predicting the energy, and also SLA verification is also carried out. Moreover, MTOA is introduced to perform dynamic scheduling processes. Accordingly, the devised MTOA is newly developed by integrating Mayfly Algorithm (MA) [15] and Taylor series [16]. Besides, the fitness measures, such as energy consumption, SLA violation, and computation cost, are considered for achieving an optimum solution.

The major contribution of this paper is explicated as follows:

> Developed MTOA-based DQN for dynamic scheduling in fog-cloud computing: an effectual dynamic scheduling approach termed MTOA-based Deep Q-Network is devised. Here, the dynamic scheduling process is performed with better convergence performance using MTOA, which is introduced by the combination of the MA and Taylor series concepts. In addition, the cost and time are less in Deep Q-Network; thus, DQN model is applied to predict the energy consumption. Besides, the fitness parameters, like energy consumption, SLA violation, and computation cost, are considered.

The rest of the paper is structured as follows: Section 2 deliberates a literature survey of existing dynamic scheduling techniques. The system model of fog-cloud computing is depicted in Section 3, and Section 4 presents a developed dynamic scheduling technique using an optimisation-enabled scheduling algorithm with DQN. Moreover, Section 5 exposes the result of the developed approach, and Section 6 designates the conclusion.

2. Motivation

The task scheduling and resource allocation are the most significant key processes in recent dynamic cloud-enabled applications. The task scheduling comprises the allocation of the task to accessible processors with minimal execution time. Moreover, the resource allocation includes a decision of allocation policy for distributing the resources to several tasks; thus, it has maximal resource utilisation. Even though the fog devices have restricted resources, which produce the resource allocation, and the provisioning process is more challenging, this challenge is considered stimulation to formulate a new dynamic scheduling technique.

2.1. Literature Survey. This section discusses the literature survey of the prevailing resource allocation and the dynamic scheduling methods in fog cloud with its advantages and confines. Tuli et al. [13] devised an Asynchronous Advantage Actor-Critic- (A3C-) based real-time scheduler model in a

stochastic edge cloud structure. Here, Residual Recurrent Neural Network (R2N2) model was utilised for capturing a huge amount of task parameters and host with temporal patterns in order to afford effectual scheduling decisions. This method permits the scheduler to rapidly adapt to the varying surroundings by means of asynchronous updates. However, this technique failed to consider data privacy and security features. To include security and data privacy aspects, Lakhan et al. [17] introduced Mobility Aware Block chain-Enabled Offloading Scheme (MABOS) in fog-cloud computing. In this approach, Vehicular Fog Cloud Network (VFCN) was presented, and it includes various components and heterogeneous computing nodes. This approach effectively reduces cost and enhances security, although this method does not include budget and fault-tolerance ability. For considering fault tolerance and budget capability, Chen et al. [18] presented Communication Resource Aware Cooperated with Computation Resources (CRACCR) model in fog-cloud computing. This technique mainly includes two factors, namely, spectral multiplexing computation consideration and Fog Node Scale Adjustment (FNSA). This approach efficiently increases energy efficiency but still does not enhance the spectrum effectiveness of fog computing structures. To increase the spectrum efficiency performance, Abbasi et al. [19] modelled GA for workload allocation in IoT-fog-cloud structure. The fog-driven IoT network was considered in this technique in which fog nodes were arbitrarily dispersed in the fog cluster. This approach highly decreased the power consumption and delay, despite being unable to resolve numerous difficult multiobjective optimisation issues.

In order to solve various multiobjective problems, Naha et al. [20] presented deadline-driven dynamic resource allocation and provisioning techniques in a fog-cloud model. Here, the resources were ranked based on the number of resource restraints in fog devices. This technique highly reduces the delay, processing time, and cost, even though this scheme was not able to manage failure handlings, like resource and communication failures. For managing various failures, Sun et al. [21] developed Energy and Time-efficient Computation Offloading and Resource Allocation (ETCORA) method in the IoT fog cloud. This technique mainly includes two processes, namely, computation offloading selection and transmission power allocation. This technique consumed minimal energy and less completion time for processing the request. However, this method was not capable of applying to realistic IoT-fog-cloud models. To adapt the system to a realistic IoT fog-cloud structure, Mishra et al. [22] modelled Analytic Hierarchy Process-(AHP-) driven resource allocation approach. The resource allocation techniques utilise predetermined weights to find the network and identify the weights from overall data. The load is efficiently balanced in this approach, and thus, the effectiveness of resources was increased while decreasing the number of tasks. Although, this approach failed to utilise deep learning techniques, namely, Convolutional Neural Network (CNN). To include deep learning methods, Yakubu et al. [23] devised an agent-driven dynamic resource allocation method in fog-cloud computing. In this method, layers and host agents were controlled by information resource tables in order to match the computing resources and tasks. This technique obtained less processing time and high Quality of Service (QoS) but generated a burden as well as oversaturation of fog resources. Narayana et al. [24] proposed an extended Particle Swarm Optimisation approach to optimise the task scheduling issue in cloud-fog situations in order to reduce the time required to finish the work and enhance the efficiency of allocating the resources. It was modelled in iFogSim, with the entire cost and makespan taken into account for assessment.

In order to optimise the energy consumption during the resource scheduling in a fog environment, Huang et al. [25] proposed a heuristic-based Particle Swarm Optimisation (PSO) algorithm based on a Lyapunov framework. Tasks scheduling with minimum energy consumption is achieved by balancing the energy consumption of IoT nodes, transmission energy consumption, and energy consumption during computation at the fog node. To schedule the resources in fog computing, Husain et al. [26] proposed a smart framework that enhances the usage of present resources. In this framework, a Master Fog is an extra layer between the cloud and fogs termed Citizen Fog (CF). This extra layer will decide on cloud deployment and CF. It uses a Comparative Attributes Algorithm (CAA) to rank the jobs, and Linear Attribute Summarized Algorithm (LASA) [27] is used to select the reachable CF with the highest computing capabilities. This framework reduces energy consumption and increases the availability of bandwidth with efficient utilisation of the other sources. Mohamed et al. [28] proposed a technique for scheduling the task in a fog-cloud environment based on the IoT request. The technique proposed to find the optimal solution is the modified artificial ecosystem (AEO) using the operators of the Salp Swarm Algorithm (SSA) called AEOSSA. This approach is evaluated using synthetic and real-world datasets with different sizes considering throughput and makespan time.

The fog computing paradigm plays an important role in allocating the resources for the IoT applications to execute the tasks. The main challenge of scheduling tasks is service cost and service execution. Najafizadeh et al. [29] proposed an architecture to minimise the service cost and service time. It uses a goal programming approach to select the best solution. Caminero et al. [30] proposed a network-aware scheduling algorithm for the execution of the application in the selected fog node, which is suitable to be executed within a deadline. This approach is the extension to the default scheduler Kubernetes. The proposed approach gives the optimal solution in that it executes all the submitted tasks within the deadline.

2.2. Challenges. The major challenges faced by present resource allocation and dynamic scheduling in fog cloud are listed as follows:

(i) In most resource allocation approaches available for fog, the cloud structure utilises static mathematical schemes for distributing resources for the incoming end-user tasks. The exploitation of present load

circumstances of fog nodes and network relations is considered in the joint optimisation process. However, ineffective resource allocation strategies decrease the system's effectiveness and increase the overall delay.

(ii) GA [19] was introduced to reduce overall delay for workload allocation in IoT-fog-cloud structure. However, this approach failed to include deep learning techniques, such as Deep Neural Networks (DNNs), for a secure mobile edge computing process.

(iii) Deadline-driven dynamic resource allocation and provisioning model was designed to make a more secure mobile edge computing structure in [20] for fog-cloud patterns. Although, this approach failed to include a more difficult simulation structure with big data IoT in fog structure for handling dynamic variations.

(iv) In order to manage various dynamic alternations, an A3C-based real-time scheduler was introduced in [13] for stochastic edge cloud structures, although this approach was not appropriate for scheduling huge quantities of hosts and edges.

(v) To make the system more suitable for the scheduling process in large edge and host networks, MABOS was developed in [17] for vehicular fog-cloud computing. However, it is a more challenging process to optimise every constraint independently.

3. System Model

This section explicates about system model of fog-cloud computing. Let us assume a set of PMs, which is illustrated as $N = \{N_1, N_2, \ldots, N_z, \ldots, N_g\}$ where g is the total number of PMs and N_z specifies z^{th} PM. The VM present in z^{th} PM is denoted as $B = \{B_1^z, B_2^z, \ldots, B_i^z, \ldots, B_y^z\}$, which is afforded by the corresponding application tasks. Every application task has a deadline E_x and run time Z_x. Moreover, a set of fog nodes is expressed as $K = \{K_1, K_2, \ldots, K_r\}$ well as a task is given $I = \{i_1, i_2, \ldots, i_x, \ldots, i_t\}$. Besides, every VM has Central Processing Unit (CPU) utilisation, bandwidth cost, memory utilisation, energy consumption, disk write throughput, disk read throughput, network received throughput, and also network transmitted throughput. The parameters of VM are assumed as dynamic because of the mobility factor in the edge paradigm. Figure 1 depicts the architecture of a fog-cloud environment.

4. Developed Dynamic Scheduling Process Using a Hybrid Optimisation-Based Deep Learning Model

The hybrid optimisation approach is developed for dynamic scheduling, and the DQN is employed for energy prediction. Originally, fog-cloud computing was done, and the parameters of VM resources are assumed as dynamic, owing to the mobility factor in the edge pattern. The SLA verification is performed, and after that, energy prediction is carried out using DQN [14]. Once the energy prediction is completed, then dynamic scheduling is done based on devised MTOA. Accordingly, the developed MTOA is designed by incorporating MA with the Taylor series. Additionally, the factors, namely, SLA verification and computation cost, are considered. Figure 2 shows the overall architecture of the dynamic scheduler model using MTOA-based DQN.

4.1. Service Level Agreement Verification. The user requests the service from the cloud, and the service is liable for detecting SLA violations based on certain rules. Depending on the defined SLA parameters, rules are generated, and it is matched using angular distance with the requested task or service for SLA verification. Usually, an SLA violation is the most important source of penalty, a violation occurs based on the Quality of Service (QoS) degradation, and it is caused by overload or network delay [31]. Furthermore, penalties are employed in provider services for assuring users. The resource scalability, execution time, resource availability, response time, and resource reliability are considered for service level agreements [32].

4.1.1. Execution Time. The execution time specifies the time consumed for estimating the users' demand, which is the most significant parameter in SLA. Besides, the implementation time mainly depends on resources and request type. The execution time is high when the resources are not appropriate [33]. The total workload running time is calculated using (1)–(3):

$$\nu_q \propto A, \tag{1}$$

$$\nu_q \propto X, \tag{2}$$

$$\nu_q = \nu_i - \nu_j, \tag{3}$$

where A indicates the resource type, ν_q represents the workload running time, X implies the request type, ν_i refers to the task initialisation time, and ν_j signifies the task finalisation time.

4.1.2. Response Time. It is referred to as the waiting time of user requests in the queue, which mainly depends on resource utilisation and network bandwidth. When the underlying resources are highly exploited, it consumes maximal time for performing a new task [34]. The total workload response time is calculated using (4)–(6):

$$\nu_m \propto M, \tag{4}$$

$$\nu_m \propto \varpi, \tag{5}$$

$$\nu_m = \nu_i - \nu_u, \tag{6}$$

where ϖ depicts the bandwidth of the network, ν_m depicts the workload response time, M represents the service scalability, ν_i represents the task initialisation time, and ν_u represents the task finalisation time.

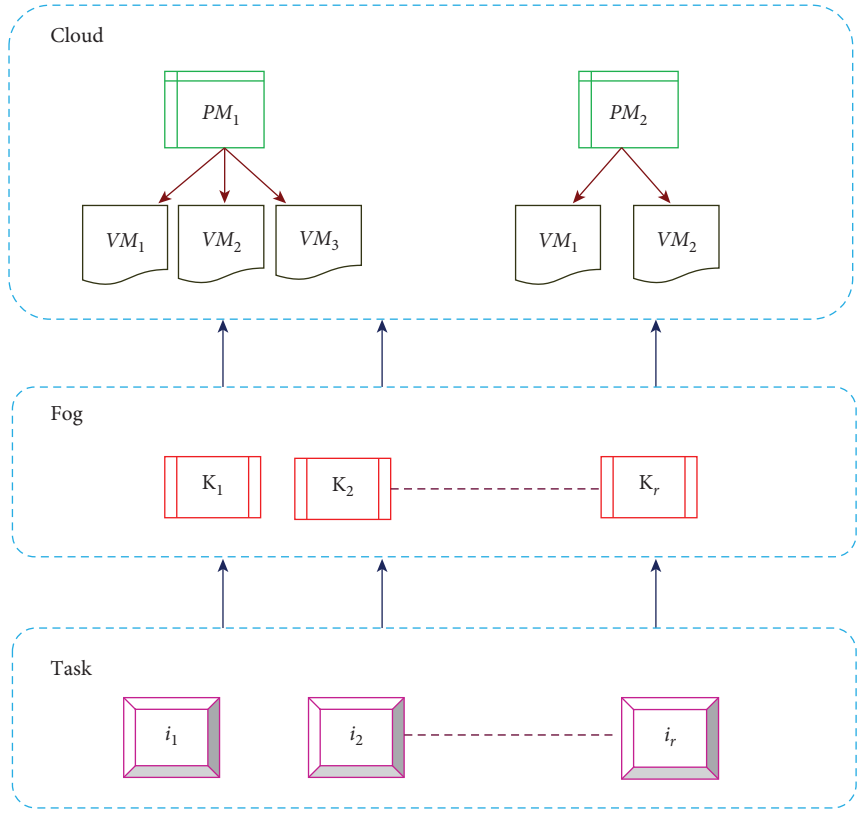

Figure 1: Architecture of a fog-cloud environment.

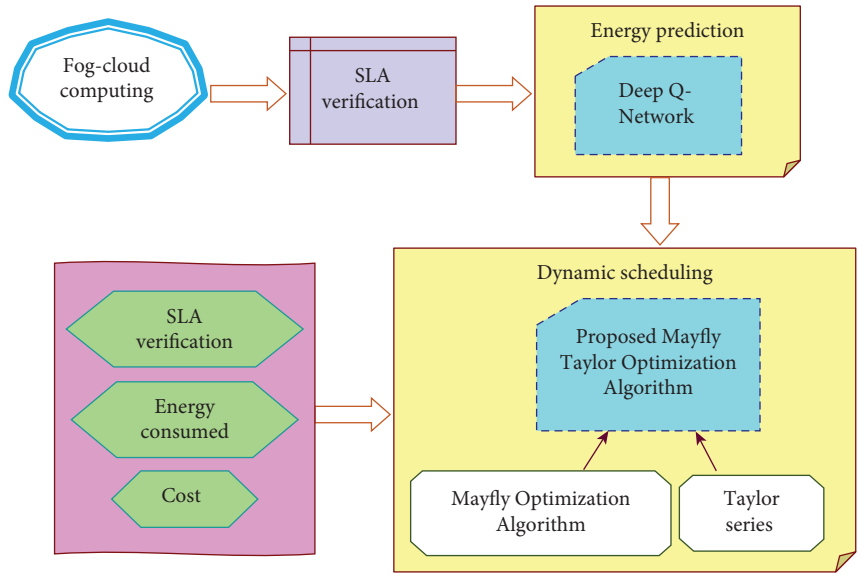

Figure 2: Overall architecture of dynamic scheduler model using MTOA-based DQN.

4.1.3. Resource Availability. The existence of agreed resources, when they are required, is specified as resource availability [34]. The resource availability is given in

$$P \propto \frac{\tau_n - \tau_l}{\tau_z}, \tag{7}$$

where τ_n is the total availability of computation time, τ_l signifies the downtime during running, and τ_z refers to the whole computation time fixed in SLA.

4.1.4. Resource Scalability. This factor is most important for QoS; if resources are nonscalable, it leads to revenue degradation and penalties [35]. More performance parameters are generally based on resource scalability. The response, as well as implementation time, may increase, while the resources are not scalable. In order to solve these time-consuming issues, external resources are rented from a third party to maximise the scalability. Besides, the resource scalability is mainly depending on capacity and SLA

violation C_v. The negative effects of nonscalable resources are given in (8)–(12):

$$\phi \propto C_v, \tag{8}$$

$$C_v \propto \frac{1}{\xi}, \tag{9}$$

$$C_v \propto \frac{1}{\mathrm{M}} \times \frac{1}{\mathrm{O}} \times \frac{1}{\mathrm{X}}, \tag{10}$$

$$\xi \propto \frac{1}{\nu_q}, \tag{11}$$

$$\mathrm{P} \propto \mathrm{M} \times \mathrm{X} \times \mathrm{O}, \tag{12}$$

where C_v shows the number of SLA violations, ξ represents the performance, P implies the customer satisfaction, M denotes the service scalability, O indicates QoS, X indicates the efficiency of services, and ν_q depicts the workload running time.

4.1.5. Resource Reliability. It is demarcated as these resources accomplish a predefined functionality for a decided time under arranged terms and circumstances [34]. The resources are dependable, while fault-tolerant is automatically recoverable. Moreover, lesser reliability decreases customer retention, and it directs to lower revenue. The resource reliability is specified in (13) as

$$\xi \propto \beta_r, \tag{13}$$

where β_r refers to the reliability.

4.2. Energy Prediction. The energy prediction process is carried out using DQN [14], in which the consumed energy of c^{th} the VM is considered as input for DQN, and the outcome of DQN is the energy consumed by the next VM $c + 1$. The energy consumption of c^{th} VM F_c is given int

$$F_c = \left[\frac{M_w}{M_v} + \frac{T_w}{T_v} + G_t + Y_t\right] * \frac{1}{4}, \tag{14}$$

where M_w and T_w represent a number of assigned CPU and memory, M_v and T_v refer to the utilised CPU and memory, G_t symbolises the disk throughput, and Y_t denotes the network throughput.

4.2.1. Architecture of Deep Q-Network. The total cost is minimal in DQN; thereby, DQN is applied for the prediction of energy. The DQN [14] model is more familiar in the reinforcement learning process, and it absorbs action-value function O^* consistent with optimum policy through decreasing the loss,

$$\begin{aligned} Q(\phi) &= \mathfrak{R}_{n,b,e,n_l}\left[\left(\mathrm{O}^*(n,b|\phi) - f\right)^2\right], \\ f &= e + \gamma \max_{n_l} \mathrm{O}^*\left(n', b'\right), \end{aligned} \tag{15}$$

where f refers to the target function, whose parameters are updated intermittently with most topical ϕ, and it assists in stabilising learning. Moreover, another essential element of alleviating DQN is the utilisation of the experience replay buffer E, which comprises tuples $n, b, e, n\prime$. Here, the agent identifies its activities by means of a neural network and joints the outcome of the neural network with random behaviours for sampling the training set. Overview of the Deep Q-Network is depicted in Figure 3.

Generally, the agents train the neural network such that it identifies collective and weighted rewards. The ideal strategy of DQN driven agent interacts directly with the workflow setting and strategies of other agents. The iterative approach for estimating the global equilibrium strategies depends on local updates Q-values and policy in every state. In general, Q-values are specified at period h for all $u \in Yn \in R$ and $b \in K(n)$ termed as$\mathrm{O}_u^h(n,b)$. Finally, every DQN agent determines the associated equilibrium approach π^h, where $\pi_n^{h+1} \in f(\mathrm{O}^{h+1}(n))$ for obtaining correlated equilibrium. Therefore, the output of DQN is the energy consumed by the next VM, which is represented as F_{c+1}.

4.3. Dynamic Scheduling Using Developed Mayfly Taylor Optimisation Algorithm. This section explicates the dynamic scheduling process based on the devised hybrid optimisation algorithm. Here, dynamic scheduling is performed by allocating resources to tasks using MTOA, which is designed by incorporating MA [15] and Taylor series [16]. Furthermore, various factors, including energy consumption, SLA violations, and computation cost, are considered.

4.3.1. Task Flow. The task flow in the fog-cloud computing structure is specified in this section. Here, task allocation of user application execution is based on the deadline and runtime of the VM. Let us consider the task to be $\mathrm{I} = \{i_1, i_2, i_3, i_4, i_5\}$, where the task is scheduled with a deadline E_x and runtime as Z_x. Table 1 specifies the deliberation of task flow [36].

Table 2 illustrates the depiction of resource allocation for all tasks. Here, tasks i_2 and i_4 are scheduled for VM_1, VM_2, and VM_3 in the time slot I_1.

4.3.2. Dynamic Scheduling Algorithmic Steps. The steps included in the developed dynamic scheduling approach are listed as follows:

(i) Configure the fog network.
(ii) Begin.
(iii) Initialise the task and VM parameters.
(iv) Perform SLA verification.
(v) Predict.
(vi) Assign the task in VM.
(vii) The task to be scheduled with the deadline E_x and run time Z_x.
(viii) Dynamic scheduling by the optimisation algorithm.

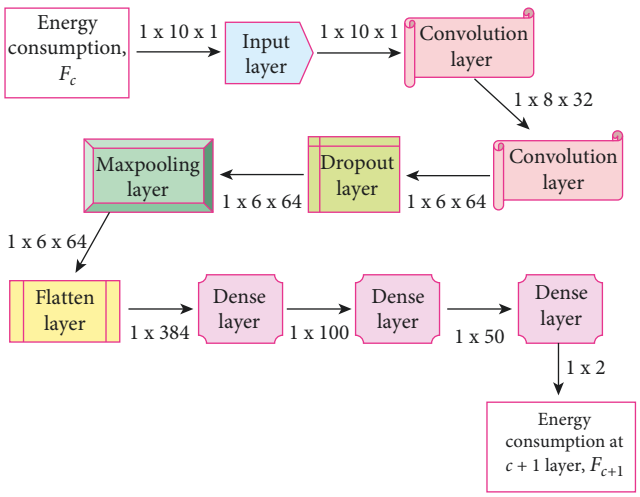

Figure 3: Overview of Deep Q-Network.

Table 1: Specification of task flow.

Task, I	i_1	i_2	i_3	i_4	i_5
Deadline	5	2	7	5	3
Run time	2	3	2	2	3

Table 2: Resource allocation for tasks.

Time slots	VM_1	VM_2	VM_3
I_1	i_2	i_4	i_4
I_2	i_2	i_3	i_4
I_3	i_1	i_3	i_5
I_4	i_3	i_1	
I_5	i_5	i_1	
I_6		i_2	

(ix) Repeat the steps.

(x) End.

4.3.3. Solution Encoding. The solution encoding for the devised dynamic scheduling process using a hybrid optimisation algorithm is deliberated in this section. Here, the task with minimal value is allocated to VM using a developed optimisation algorithm. The size of the solution encoding depends on VM size, where the dimension is 1×5. Figure 4 exposes the solution encoding of the dynamic scheduler model.

4.3.4. Fitness Function. The fitness measure is calculated in order to identify the best solution for the dynamic scheduling process with various parameters, like run time, computation cost, energy consumption, and SLA verification. The minimal fitness measure is considered the optimum solution for dynamic scheduling. The fitness value is estimated by the following expression:

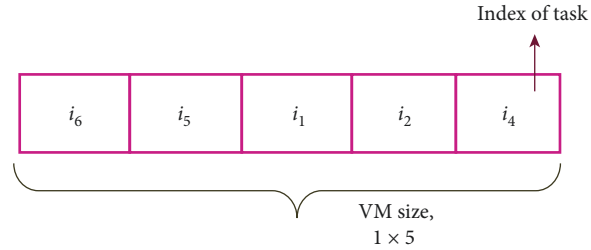

Figure 4: Dynamic scheduler model solution encoding.

$$\psi_{\text{fit}} = \sum_{n=1}^{t} Z_x + \sum_{m=1}^{y} (F_c + V_s + R_c), \tag{16}$$

where Z_x specifies the runtime of the x^{th} task, F_c symbolises the energy consumption, V_s depicts the SLA verification, R_c designates the cost, t implies a number of tasks, and y represents the number of VM. The SLA verification, energy consumption, and computation cost are explained as follows:

(i) SLA Verification. The SLA verification is estimated through the Overall Performance Degradation caused by VM Migration (PDM) [37] and SLA violation Time per Active Host (SLATAH). The SLA is calculated by (17):

$$V_s = Q_e \times S_e, \tag{17}$$

where S_e signifies the SLA violation Time per Active Host represented in (19) and Q_e is the PDM, which represents overall performance degradation produced by VM migration; these terms are expressed as follows:

$$Q_e = \frac{1}{y} \sum_{i=1}^{y} \frac{I_{ni}}{I_{si}}, \tag{18}$$

where y is the total number of VM, I_{ni} denotes the estimation of performance degradation produced by i^{th} VM, and I_{si} specifies the total CPU capacity requested by i^{th} VM.

$$S_e = \frac{1}{g} \sum_{z=1}^{z} \frac{W_{kz}}{W_{lz}}, \tag{19}$$

where g represents the total number of PM, W_{kz} indicates the z^{th} total time of the CPU utilisation, and W_{lz} is the total time of z^{th} PM being in actual state.

(ii) Energy Consumption. The energy obsession during the process of dynamic scheduling is referred to as energy consumption, and it should be low. The energy consumption is estimated by (20):

$$F_c = \frac{1}{g \times y} \left[\sum_{z=1}^{g} \sum_{i=1}^{y} (A_{zi} N_{\max} + (1 - A_{zi}) \varphi_{zi} N_{\max}) \right], \tag{20}$$

where $A_{zi} = 0.1 N_{\max} = 1$, and φ_{zi} is expressed as (21):

$$\varphi_{zi} = \frac{1}{3} \left(\frac{M_v}{M_b} + \frac{T_v}{T_b} + \frac{Z_v}{Z_b} \right), \tag{21}$$

where A_{zi} represents the constant, M_v is the CPU utilised by VM, T_v refers to the memory utilised by VM, Z_v indicates the bandwidth used by VM, M_b specifies the total CPU available in PM, T_b implies the total memory exists in VM, $N_{\max}$ denotes the maximum energy, φ_{zi} is the resource utilisation rate, and Z_b symbolises the total bandwidth available in VM.

(iii) Computation Cost. The computation cost of the system is computed based on memory utilisation and bandwidth of VM, which is estimated by (22):

$$R_c = \frac{Z_b^f + \left(1 - T_b^f\right)}{2}, \tag{22}$$

where Z_b^f is the bandwidth of f^{th} VM and T_b^f denotes memory utilisation of f^{th} VM.

4.3.5. Developed Hybrid Optimisation Algorithm for Dynamic Scheduling. This section deliberates on the devised MTOA for the effectual dynamic scheduling process. Accordingly, the developed MTOA is newly designed by integrating MA [15] and Taylor series [16]. The MA is developed by the stimulation of flight behaviour and mating procedure of mayflies. The location of every mayfly in search space indicates the potential solution to problems. This method improves the balance between exploitation and exploration behaviours; thus, it helps to escape from local optima. However, MA has some complexities from initial parameter tuning; thus, the Taylor series is included to improve the performance. The Taylor series encompasses a difficult variable function, which is the expansion of an infinite term function. Thus, the incorporation of the MA and Taylor series obtains enhanced convergence behaviour with the maximal probability of finding the optimal solution. The algorithmic process of developed MTOA is specified as follows:

(i) Initialisation. The two groups of mayflies, such as female and male populations, are formulated randomly. Here, every mayfly is arbitrarily produced in a problem space as a candidate solution, which is specified by h the dimensional vector, $D = (D_1, D_2, \ldots, D_h)$, and its performance is estimated on a predefined objective function $f(D)$. Moreover, the velocity $H = (H_1, H_2, \ldots, H_h)$ of the mayfly is demarcated as a variation of its location, and the flying route of every mayfly is the dynamic interface of individual and social flying skills. Every mayfly alters its trajectory towards its personal best location K best and the optimal location obtained by any mayfly of the swarm B best.

(ii) Fitness Function. The optimal solution is estimated depending on the fitness function for the effectual dynamic scheduling process. The fitness function with a minimal fitness rate is taken as the optimum solution, and it is calculated by expression (16).

(iii) Movement of Male Mayflies. The males collected in the swarm specify that the location of every male mayfly is altered based on its own experience and its neighbours. The movement of the male mayfly is given by (23):

$$D_p^{w+1} = D_p^w + J_p^{w+1}, \tag{23}$$

where D_p^w specifies the current location of mayfly p in time search space at a time step w, J_p^w is the location altered by including velocity J_p^{w+1}, and $D_p^0 \sim Y(D_{\min}, D_{\max})$.

Usually, the male mayflies are constantly present a few meters overhead from the water. The velocity of the male mayfly p is estimated as in

$$\begin{aligned} J_p^{w+1} = {} & J_p^w + k_1 e^{-\omega s_f^2}\left(K\,\text{best}_p - D_p^w\right) \\ & + k_2 e^{-\omega s_n^2}\left(B\,\text{best}_p - D_p^w\right). \end{aligned} \tag{24}$$

Substitute equation (24) in (23):

$$\begin{aligned} D_p^{w+1} = {} & D_p^w + J_p^w + k_1 e^{-\omega s_f^2}\left(K\,\text{best}_p - D_p^w\right) \\ & + k_2 e^{-\omega s_n^2}\left(B\,\text{best}_p - D_p^w\right), \end{aligned} \tag{25}$$

$$\begin{aligned} D_p^{w+1} = {} & D_p^w + J_p^w + k_1 e^{-\omega s_f^2} K\,\text{best}_p - k_1 e^{-\omega s_f^2} D_p^w \\ & + k_2 e^{-\omega s_n^2} B\,\text{best}_p - k_2 e^{-\omega s_n^2} D_p^w, \end{aligned} \tag{26}$$

$$\begin{aligned} D_p^{w+1} = {} & D_p^w\left(1 - k_1 e^{-\omega s_f^2} - k_2 e^{-\omega s_n^2}\right) + J_p^w \\ & + k_1 e^{-\omega s_f^2} K\,\text{best}_p + k_2 e^{-\omega s_n^2} B\,\text{best}_p. \end{aligned} \tag{27}$$

Here, the Taylor series is included in the above expression for improving the performance:

$$D(w+1) = D(w) + \frac{D\prime(w)}{1!} + \frac{D\prime\prime(t)}{2!}, \tag{28}$$

$$D\prime(t) = \frac{D(w) - D(w-e)}{e}, \tag{29}$$

$$D\prime\prime(t) = \frac{D(w) - 2D(w-e) + D(w-2e)}{e^2}. \tag{30}$$

Consider $e = 1$ in equations (29) and (30):

$$D_p^{w+1} = D_p^w + \frac{D^w - D_p^{w-1}}{1!} + \frac{D_p^w - 2D_p^{w-1} + D_p^{w-2}}{2!}, \tag{31}$$

$$D_p^{w+1} = D_p^w\left(1 + 1 + \frac{1}{2}\right) - D_p^{w-1} - \frac{2D_p^{w-1}}{2} + \frac{D_p^{w-2}}{2}, \tag{32}$$

$$D_p^{w+1} = D_p^w\left(\frac{5}{2}\right) - 2D_p^{w-1} + \frac{D_p^{w-2}}{2}, \tag{33}$$

$$D_p^w\left(\frac{5}{2}\right) = D_p^{w+1} + 2D_p^{w-1} - \frac{D_p^{w-2}}{2}, \tag{34}$$

$$D_p^w = \left[D_p^{w+1} + 2D_p^{w-1} - \frac{D_p^{w-2}}{2}\right] \times \frac{2}{5}. \tag{35}$$

Substitute equation (35) into (27):

$$D_p^{w+1} = \left[\left[D_p^{w+1} + 2D_p^{w-1} - \frac{D_p^{w-2}}{2}\right] \times \frac{2}{5}\right] \left(1 - k_1 e^{-\omega s_f^2} - k_2 e^{-\omega s_n^2}\right) - J_p^w + k_1 e^{-\omega s_f^2} K\,\text{best}_p + k_2 e^{-\omega s_n^2} B\,\text{best}_p, \tag{36}$$

$$D_p^{w+1} - \frac{2D_p^{w+1}}{5}\left(1 - k_1 e^{-\omega s_f^2} - k_2 e^{-\omega s_n^2}\right) = \left[\left(2D_p^{w-1} - \frac{D_p^{w-2}}{2}\right) \times \frac{2}{5}\right]\left(1 - k_1 e^{-\omega s_f^2} - k_2 e^{-\omega s_n^2}\right) + J_p^w + k_1 e^{-\omega s_f^2} K\,\text{best}_p + k_2 e^{-\omega s_n^2} B\,\text{best}_p, \tag{37}$$

$$\frac{5D_p^{w+1} - 2D_p^{w+1}}{5}\left(1 - k_1 e^{-\omega s_f^2} - k_2 e^{-\omega s_n^2}\right) = \left[\left(2D_p^{w-1} - \frac{D_p^{w-2}}{2}\right) \times \frac{2}{5}\right]\left(1 - k_1 e^{-\omega s_f^2} - k_2 e^{-\omega s_n^2}\right) + J_p^w + k_1 e^{-\omega s_f^2} K\,\text{best}_p + k_2 e^{-\omega s_n^2} B\,\text{best}_p, \tag{38}$$

$$D_p^{w+1} \left(\frac{5-2}{5}\right)\left(1 - k_1 e^{-\omega s_f^2} - k_2 e^{-\omega s_n^2}\right) = \left[\left(2D_p^{w-1} - \frac{D_p^{w-2}}{2}\right) \times \frac{2}{5}\right]\left(1 - k_1 e^{-\omega s_f^2} - k_2 e^{-\omega s_n^2}\right) + J_p^w + k_1 e^{-\omega s_f^2} K\,\text{best}_p + k_2 e^{-\omega s_n^2} B\,\text{best}_p \tag{39}$$

$$D_p^{w+1} \left(\frac{3}{5}\right)\left(1 - k_1 e^{-\omega s_f^2} - k_2 e^{-\omega s_n^2}\right) = \left[\left(2D_p^{w-1} - \frac{D_p^{w-2}}{2}\right) \times \frac{2}{5}\right]\left(1 - k_1 e^{-\omega s_f^2} - k_2 e^{-\omega s_n^2}\right) + J_p^w + k_1 e^{-\omega s_f^2} K\,\text{best}_p + k_2 e^{-\omega s_n^2} B\,\text{best}_p, \tag{40}$$

$$D_p^{w+1} = \left[\left(\left(2D_p^{w-1} - \frac{D_p^{w-2}}{2}\right) \times \frac{2}{5}\right)\left(1 - k_1 e^{-\omega s_f^2} - k_2 e^{-\omega s_n^2}\right) + J_p^w + k_1 e^{-\omega s_f^2} K best_p + k_2 e^{-\omega s_n^2} B best_p\right] * \frac{3}{5\left(1 - k_1 e^{-\omega s_f^2} - k_2 e^{-\omega s_n^2}\right)}, \tag{41}$$

where D_p^{w+1} is the position of p^{th} solution at $w+1$ iteration, D_p^{w-1} refers to the location of p^{th} solution at $w-1$ iteration, D_p^{w-2} signifies the location of p^{th} solution at $w-2$ iteration, and k_1 and k_2 imply the positive attraction constants utilised for scaling the contribution of the social and cognitive element. The personal best location $K\,\text{best}_p$ at the next iteration $w+1$ with the consideration of minimisation problems is estimated by the following expression:

$$K\,\text{best}_p = \begin{cases} D_p^{w+1}; & \text{if } \left(D_p^{w+1}\right) < f\left(K\,\text{best}_p\right), \\ \text{is kept the same;} & \text{Otherwise,} \end{cases} \tag{42}$$

where f indicates the objective function and estimates the solution quality. Moreover, the global best location B best at the time step w is expressed as

$$B\,\text{best} \in \min\{f\left(K\,\text{best}_1\right), f\left(K\,\text{best}_2\right), \ldots, f\left(K\,\text{best}_L\right),\}, \tag{43}$$

where L symbolises the total amount of male mayflies in the swarm. The term ω is a fixed perceptibility coefficient applied in equation (39) utilised to restrict the mayfly's visibility to others, when A_o is the Cartesian distance among D_p and $K\,\text{best}_p$, and A_j is the Cartesian distance among D_p and B best. These distances are estimated by

$$\left\|D_p - d_p\right\| = \sqrt{\sum_{z=1}^{p}\left(D_p - d_p\right)^2}, \tag{44}$$

where d_p corresponds to $K\,\text{best}_p$ or B best.

It is more significant for operating this approach that optimal mayflies in a swarm endure in order to accomplish their characteristic up and down nuptial dance. Therefore, optimal mayflies keep varying their velocities, which is computed by

$$J_p^{w+1} = J_p^w + t^* x, \tag{45}$$

where t represents the nuptial dance constantly and x symbolises the random integer with a range of $[-1, 1]$.

(iv) Mating of Mayflies. The crossover function indicates the mating procedure among two mayflies, which is explicated in the following. Here, one parent is chosen from the male population, whereas another is from the female population. The parent's selection process is similar to the technique that females are involved by males. Mostly, the selection process is done randomly or based on their fitness measure. The outcome of two off-spring is formulated as follows:

$$O_1 = X^* U + (1 - X)^* P, \tag{46}$$

$$O_1 = X^* P + (1 - X)^* U, \tag{47}$$

where U indicates the male parent, P signifies the female parent, and X symbolises the random integer in a particular range.

(v) Evaluating the Feasibility of the Solution. Here, solution feasibility is computed by means of the fitness function, which is expressed in equation (16). If a new solution is improved compared to the old solution, then the old value is replaced with a new one.

(vi) Termination. The above processes are continually repetitive until an optimal solution is obtained for dynamic scheduling. Table 3 deliberates the pseudocode of the introduced MTOA.

Thus, the combination of the MA and Taylor series model effectively increases the performance of dynamic scheduling.

5. Results and Discussion

The results and discussion of the designed dynamic scheduling process using MTOA-based DQN are specified in this section. Moreover, experimental setup, dataset description, performance metrics, comparative techniques, and discussion are deliberated in this section.

5.1. Experimental Setup. The introduced dynamic scheduling approach is implemented in JAVA with the iFogSim simulator tool with Windows 10OS. Generally, iFogSim enables the simulation of fog computing in order to estimate the resource management and scheduling procedures across edge and cloud resources with dissimilar setups.

5.2. Dataset Description. The execution of developed MTOA-based DQN for dynamic scheduling is performed using Grid Workloads Archive dataset [38, 39]. This dataset comprises performance metrics of 1,750 VMs from dispersed data centres from Bitbrains, and it is a service provider that concentrates on controlled hosting as well as the business calculation for originalities. Every file encloses the performance metrics of VM, and these files are ordered based on traces, like fast storage and Rnd. The fast storage trace includes 1,250 VMs, which are linked to fast Storage Area Network (SAN) storage devices. In addition, the Rnd trace includes 500 VMs, and it is associated with either fast SAN tools or slower Network Attached Storage (NAS) devices.

5.3. Performance Metrics. The metrics, namely, energy consumption, SLA and cost, are considered for evaluating the performance of devised MTOA-based DQN, and these metrics are already explained in Section 4.3.4.

5.4. Comparative Techniques. The existing dynamic scheduling techniques, including CAG [2], DNGSA [5], scheduling based on containers [4], and policy learning [13], are considered comparative methods for computing the performance of developed MTOA-based DQN.

5.5. Comparative Analysis. This section specifies an analysis of introduced MTOA-based DQN based on fast storage with 625 and 1250 VM and also Rnd datasets with 250 and 500 VM.

5.5.1. Analysis Using Fast Storage Data. The comparative analysis of devised MTOA-based DQN using fast storage database with 625 and 1250 VMs with regard to various metrics is illustrated in this section.

(i) For 625 VM. The analysis of the designed MTOA-based DQN for different metrics using fast storage data with 625 VM by altering the iteration is depicted in Figure 5. Figure 5(a) exposes the comparative analysis of energy consumption by shifting iteration. The energy consumption of existing dynamic scheduling methods and introduced MTOA-based DQN is 0.0585, 0.0541, 0.0514, 0.0481, and 0.0451 for the 15th iteration. The analysis of the introduced MTOA-based DQN for SLA is plotted in Figure 5(b). When the iteration is 15, the SLA of CAG is 0.0541, DNGSA is 0.0471, scheduling based on containers is 0.0354, policy learning is 0.0241, and devised MTOA-based DQN is 0.0126. The comparative analysis of the developed approach for computation cost through modifying iteration is specified in Figure 5(c). The computation cost attained by developed MTOA-based DQN is 0.233, whereas existing techniques are 0.2987, 0.2875, 0.2745, and 0.2615 in the 15th iteration. Also, the introduced MTOA-based DQN reduces energy consumption by 23%, 17%, 12%, and 6% compared to CAG, DNGSA, scheduling based on containers, and policy learning methods, respectively. SLA violation can be decreased by 77%, 73%, 64%, and 48% compared to CAG, DNGSA, scheduling based on containers, and policy learning methods. The MTOA-based DQN optimises the total computation cost by 22%, 19%, 15%, and 11% compared to CAG, DNGSA, scheduling based on containers, and policy learning methods.

(ii) For 1250 VM. Figure 6 signifies the analysis of developed MTOA-based DQN for various metrics based on fast storage data with 1250 VM by shifting iteration. The analysis of the designed MTOA-based DQN for energy consumption is illustrated in Figure 6(a). The energy consumption of CAG, DNGSA, scheduling based on containers, policy learning, and developed MTOA-based DQN is 0.0933, 0.0914, 0.0896, 0.0875, and 0.0854 when the iteration is 15. Figure 6(b) depicts a comparative analysis of the designed MTOA-based DQN for SLA by changing iterations. The SLA of the existing dynamic scheduling and introduced MTOA-based DQN is 0.0898, 0.0699, 0.0321, 0.0174, and 0.0114 for the 15th iteration. The comparative analysis of developed MTOA-based DQN for computation cost with various iterations is shown in Figure 6(c). The computation cost of the developed MTOA-based DQN is 0.1875, while the existing method attained 0.2785, 0.2357, 0.2259, and 0.2026 in the 15th iteration. Also, the introduced MTOA-based DQN reduces the energy consumption by 8%, 6%, 5%, and 2% compared to CAG, DNGSA, scheduling based on containers, and policy learning methods, respectively. SLA violation can be decreased by 87%, 84%, 64%, and 34% compared to CAG, DNGSA, scheduling based on containers, and policy learning methods. The MTOA-based DQN optimises the total computation cost by 33%, 20%, 17%, and 7% compared to CAG, DNGSA, scheduling based on containers, and policy learning methods.

Table 3: Pseudocode of devised MTOA.

Sl. No	Pseudocode of developed MTOA
1	**Input:** Total population and velocities
2	**Output:** Best solution, D_p^{w+1}
3	**Begin**
4	Initialise the population of male and female mayflies and their velocities
5	Compute the fitness function using equation (16)
6	Determine the global best B_{best}
7	While stopping condition is not satisfied
8	Update the velocities and solutions of females and males
9	Compute the solutions
10	Ranking of mayflies
11	Mate the mayflies
12	Estimate the off-springs based on equations (46) and (47)
13	Divide the off-spring into male and female arbitrarily
14	Replace worst solutions with the new solution
15	Update *Kbest* and *Bbest*
16	End while
17	Return the best solution

5.5.2. Analysis Using Rnd Data. This section illustrates the comparative analysis of developed MTOA-driven DQN based on the Rnd database with 250 and 500 VMs with respect to different metrics.

(i) For 250 VM. The analysis of the designed MTOA-based DQN for various metrics using Rnd data with 250 VM by altering iteration is depicted in Figure 7. Figure 7(a) exposes a comparative analysis of energy consumption with various iterations. The energy consumption of the existing dynamic scheduling methods and introduced MTOA-based DQN is 0.0514, 0.0486, 0.0459, 0.0421, and 0.0417 for the 15th iteration. The analysis of the introduced MTOA-based DQN for SLA is plotted in Figure 7(b). When the iteration is 15, the SLA of the CAG model is 0.0514, DNGSA is 0.0451, scheduling based on containers is 0.0441, policy learning is 0.0397, and devised MTOA-based DQN is 0.0328. The comparative analysis of the developed approach for computation cost through modifying iteration is specified in Figure 7(c). The computation cost attained by the developed MTOA-based DQN is 0.2146, whereas the existing techniques are 0.3014, 0.2625, 0.2414, and 0.2374 in the 15th iteration. Also, the introduced MTOA-based DQN reduces the energy consumption by 19%, 14%, 9%, and 1% compared to CAG, DNGSA, scheduling based on containers, and policy learning methods, respectively. SLA violation can be decreased by 36%, 27%, 26%, and 17% compared to CAG, DNGSA, scheduling based on containers, and policy learning methods. The MTOA-based DQN optimises the total computation cost by 29%, 18%, 11%, and 10% compared to CAG, DNGSA, scheduling based on containers, and policy learning methods.

(ii) For 500 VM. Figure 8signifies the analysis of developed MTOA-based DQN for the various metrics based on fast storage data with 500 VM by shifting iteration. The analysis of the designed MTOA-based DQN for energy consumption is illustrated in Figure 8(a). The energy consumption of CAG, DNGSA, scheduling based on containers, policy learning, and developed MTOA-based DQN is 0.0714, 0.0501, 0.0433, 0.0403, and 0.0198 when the iteration is 15. Figure 8(b) depicts the comparative analysis of developed MTOA-based DQN for SLA by changing iterations. The SLA of existing dynamic scheduling and introduced MTOA-based DQN is 0.0488, 0.0413, 0.0399, 0.0254, and 0.0125 for the 15th iteration. The comparative analysis of developed MTOA-based DQN for computation cost with various iterations is shown in Figure 8(c). The computation cost of the developed MTOA-based DQN is 0.0804, while the existing method attained 0.2854, 0.2413, 0.2214, and 0.1940 in the 15th iteration. Also, the introduced MTOA-based DQN reduces the energy consumption by 72%, 60%, 54%, and 51% compared to CAG, DNGSA, scheduling based on containers, and policy learning methods, respectively. SLA violation can be decreased by 74%, 70%, 69%, and 51% compared to CAG, DNGSA, scheduling based on containers, and policy learning methods. The MTOA-based DQN optimises the total computation cost by 72%, 67%, 64%, and 58% compared to CAG, DNGSA, scheduling based on containers, and policy learning methods.

5.6. Comparative Discussion. Table 4 explicates a comparative discussion of introduced MTOA-based DQN with regard to different metrics using fast storage and Rnd datasets with various numbers of VM. The energy consumption of the existing dynamic scheduling methods and introduced MTOA-based DQN is 0.0765, 0.0614, 0.0599, 0.0585, and 0.0551 for the 20th iteration using fast storage data with 625 VM. The developed dynamic scheduling model obtained less energy consumption because of the utilisation of DQN for energy prediction. When the iteration is 20, SLA of CAG is 0.0754, DNGSA is 0.0699, scheduling based on containers is 0.0565, policy learning is 0.0399, and devised MTOA-based DQN is 0.0178 based on fast storage data with 625 M. The SLA of introduced dynamic scheduling is highly reduced due to the hybridisation of optimisation algorithms. The computation cost

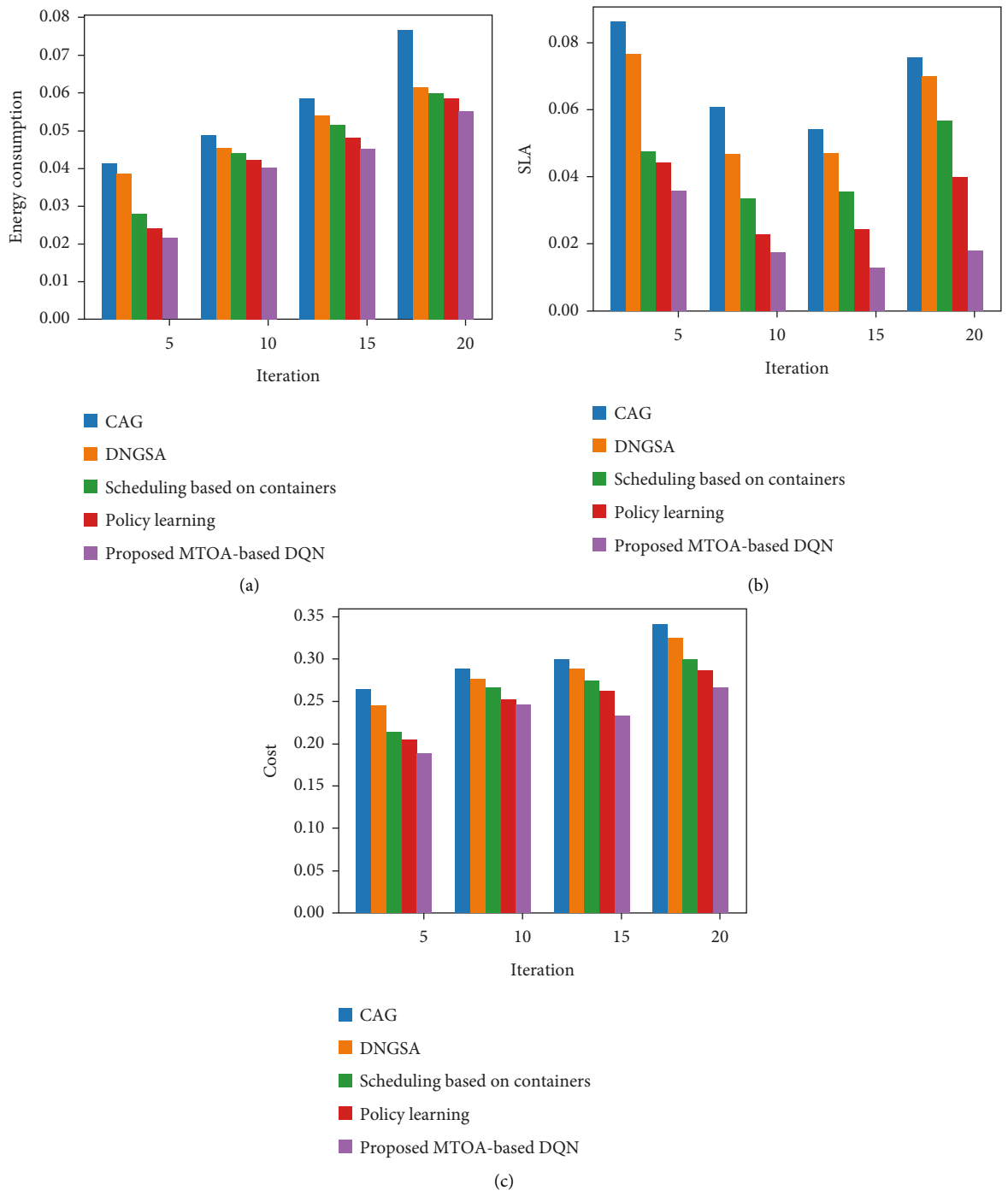

Figure 5: MTOA-based DQN with 625 VM using fast storage data: (a) energy consumption, (b) SLA, and (c) cost.

attained by the developed MTOA-based DQN is 0.2663, whereas the existing techniques are 0.3412, 0.3245, 0.2987, and 0.2854 in the 20th iteration using fast storage data with 625 VM. Also, the introduced MTOA-based DQN reduces the energy consumption by 28%, 10%, 8%, and 6% compared to CAG, DNGSA, scheduling based on containers, and policy learning methods, respectively. SLA violation can be decreased by 76%, 74%, 68%, and 55% compared to CAG, DNGSA, scheduling based on containers, and policy learning methods. The MTOA-based DQN optimises the total computation cost by 22%, 18%, 11%, and 7% compared to CAG, DNGSA, scheduling based on containers, and policy learning methods. The employed optimisation techniques are simple and easy to implement; thus, the computation cost of the developed dynamic scheduling model is reduced effectively.

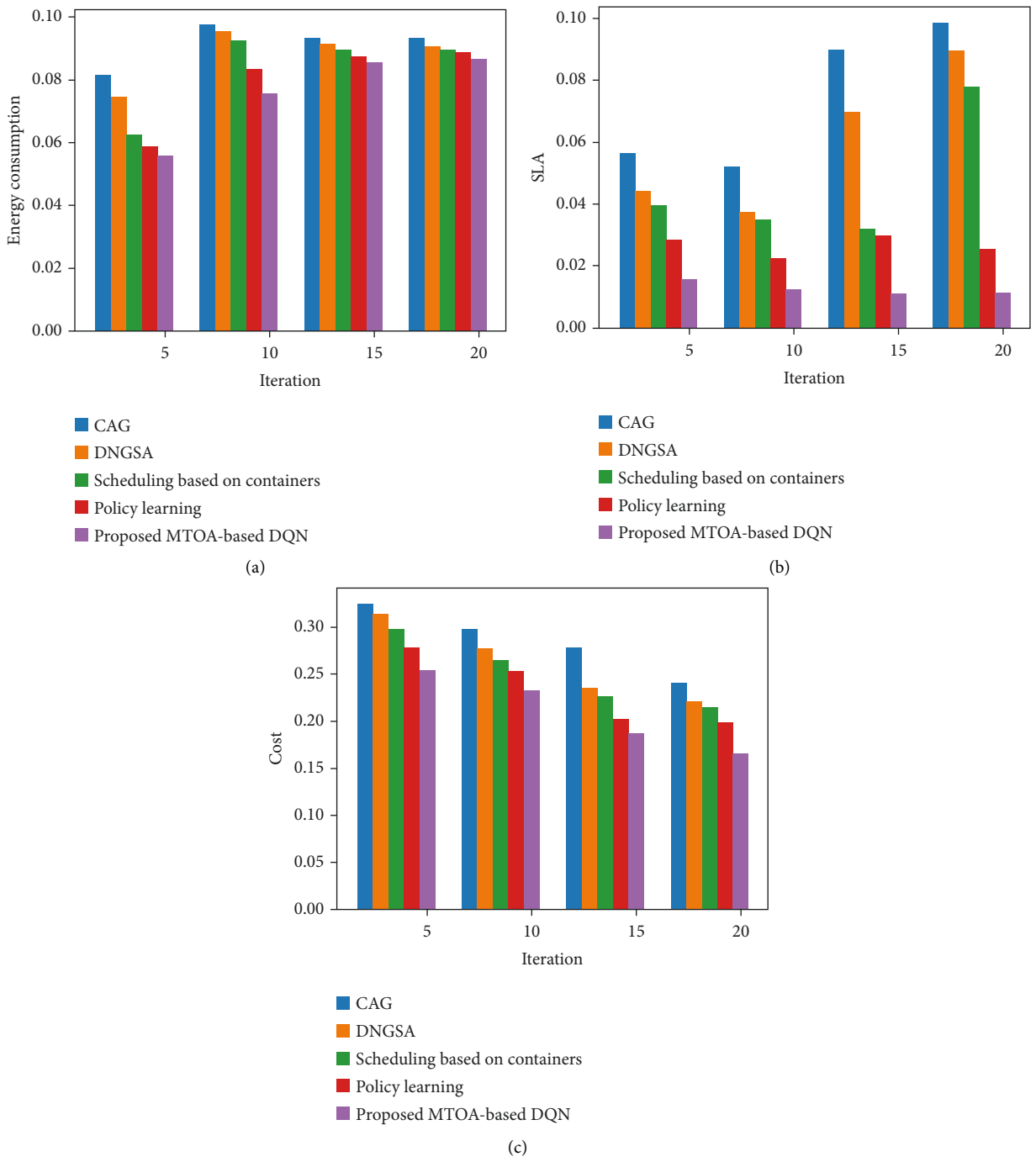

Figure 6: MTOA-based DQN with 1250 VM using fast storage data. (a) Energy consumption, (b) SLA, and (c) cost.

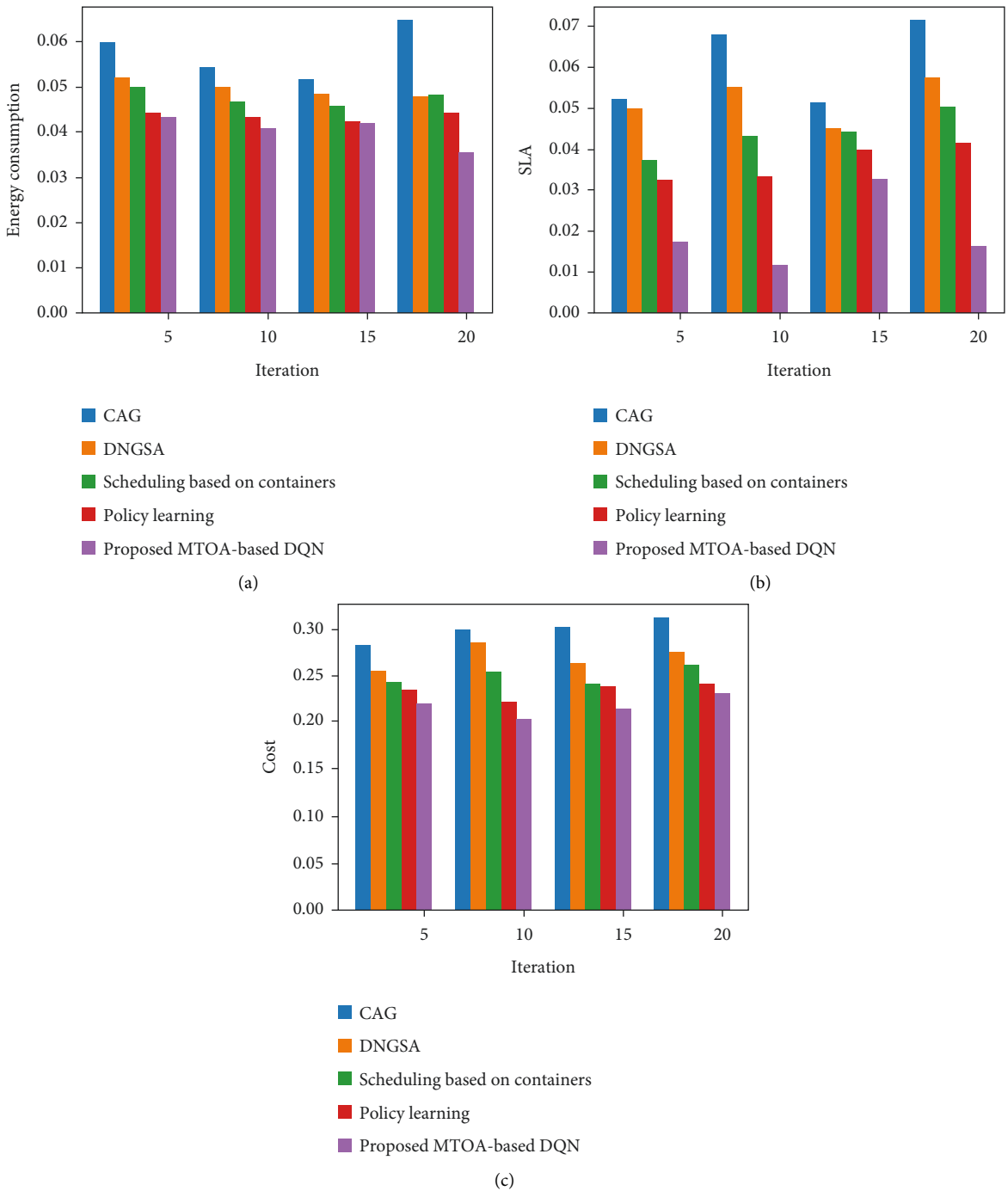

Figure 7: MTOA-based DQN with 250 VM using Rnd data. (a) Energy consumption, (b) SLA, and (c) cost.

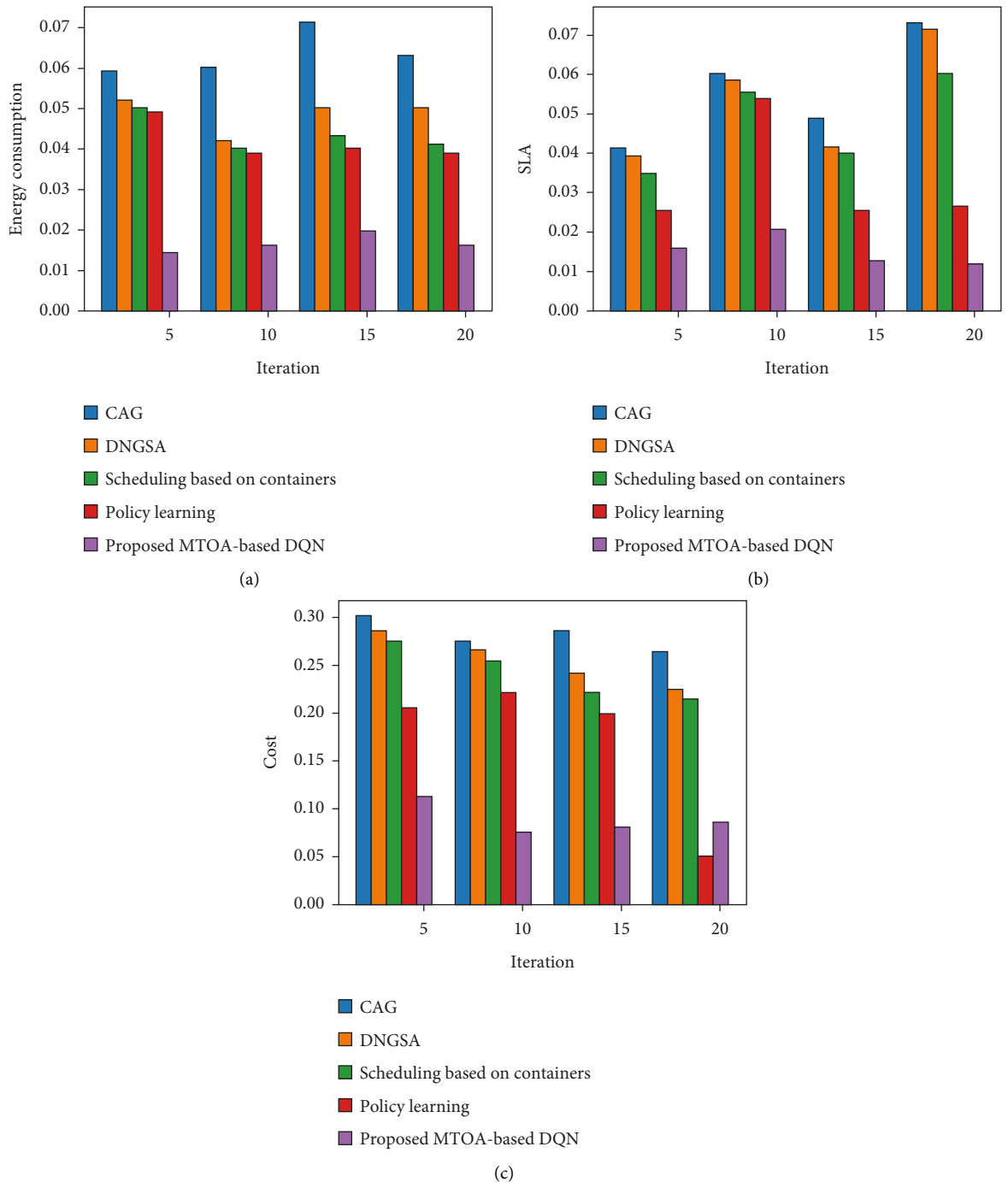

Figure 8: MTOA-based DQN with 500 VM using Rnd data. (a) Energy consumption, (b) SLA, and (c) cost.

Table 4: Comparative discussion.

Dataset/VM		Methods/metrics	CAG	DNGSA	Scheduling based on containers	Policy learning	Proposed MTOA-based DQN
Fast storage	**625**	Energy consumption	0.07654125	0.06142533	0.05985413	0.05854125	0.05514253
		SLA	0.07541258	0.06985413	0.05654125	0.03985785	0.01787909
		Cost	0.34125863	0.32451258	0.29874513	0.28541253	0.26637318
	1250	Energy consumption	0.09325413	0.09054125	0.08954725	0.08874513	0.08654125
		SLA	0.09874513	0.08954125	0.07784585	0.02547513	**0.0114253**
		Cost	0.24125833	0.22142533	0.21425325	0.19857458	0.1654125
Rnd	**250**	Energy consumption	0.06471258	0.04801425	0.04814253	0.04412583	0.03591231
		SLA	0.07142533	0.05741253	0.05014253	0.04142533	0.01597958
		Cost	0.31242533	0.27451253	0.26214253	0.24142533	0.23088294
	500	Energy consumption	0.06324126	0.05014753	0.04125325	0.03914253	**0.0162121**
		SLA	0.07325413	0.07142533	0.05987453	0.02654125	0.0116571
		Cost	0.26412583	0.22475125	0.21425325	0.15231121	**0.0855121**

6. Conclusions

The present paper deliberates on devised MTOA-based DQN model for an efficient dynamic scheduling process in a fog-cloud computing environment. Here, the Grid Workloads Archive dataset is considered in which fast storage and Rnd data are taken for processing the developed dynamic scheduling approach. Here, the MA is incorporated with the Taylor series in order to obtain a better convergence performance. In this approach, fog-cloud simulation is carried out, which enables dynamic and flexible configuration. In this approach, the parameter of VM is considered dynamic because of the mobility aspect in the edge paradigm. Additionally, SLA verification is done along with execution time, response time, resource availability, resource scalability, and resource reliability. The processing cost and time are less in DQN; thus, DQN is employed during the energy prediction process. In this dynamic scheduling process, energy consumption, SLA violation, and computation cost are considered as main fitness measures. The performance of the introduced dynamic scheduling approach is evaluated using three metrics, energy consumption, SLA and computation cost. The developed MTOA-based DQN obtained better performance with energy consumption, SLA, and computation cost of 0.0162, 0.0114, and 0.0855. Furthermore, the developed dynamic scheduling approach can be applied to a real-time data centre.

Data Availability

The labelled datasets used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

All authors declare that they have no conflicts of interest.

References

[1] M. R. Alizadeh, V. Khajehvand, A. M. Rahmani, and E. Akbari, "Task scheduling approaches in fog computing: a systematic review," *International Journal of Communication Systems*, vol. 33, no. 16, p. 4583, 2020.

[2] T. S. Nikoui, A. Balador, A. M. Rahmani, and Z. Bakhshi, "Cost-aware task scheduling in fog-cloud environment," in *Proceedings of the CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*, pp. 1–8, Tehran, Iran, June 2020.

[3] S. Supreeth and S. Biradar, "Scheduling virtual machines for load balancing in cloud computing platform," *International Journal of Science and Research*, vol. 2, pp. 437–441, 2013.

[4] L. Yin, J. Luo, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4712–4721, 2018.

[5] I. M. Ali, K. M. Sallam, N. Moustafa, R. Chakraborty, M. J. Ryan, and K. K. R. Choo, "An automated task scheduling model using non-dominated sorting genetic algorithm ii for fog-cloud systems," *IEEE Transactions on Cloud Computing*, 2020.

[6] S. Supreeth and K. K. Patil, "Virtual machine scheduling strategies in cloud computing- A review," *International Journal on Emerging Technologies*, vol. 10, no. 3, pp. 181–188, 2019.

[7] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2017.

[8] S. Supreeth and K. Patil, "Hybrid genetic algorithm and modified-particle swarm optimization algorithm (GA-MPSO) for predicting scheduling virtual machines in educational cloud platforms," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 17, no. 7, 2022.

[9] S. Supreeth and K. Patil, "VM scheduling for efficient dynamically migrated virtual machines (VMS-EDMVM) in cloud computing environment," *KSII Transactions on Internet and Information Systems*, vol. 16, no. 6, pp. 1892–1912, 2022.

[10] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "An online algorithm for task offloading in heterogeneous mobile

clouds," *ACM Transactions on Internet Technology*, vol. 18, p. 23, 2018.

[11] A. Bose, T. Biswas, and P. Kuila, "A novel genetic algorithm based scheduling for multi-core systems," in *Smart Innovations in Communication and Computational Sciences*, pp. 45–54, Springer, Berlin, Germany, 2019.

[12] T. D. Braun, H. J. Siegel, N. Beck et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810–837, 2001.

[13] S. Tuli, S. Ilager, K. Ramamohanarao, and R. Buyya, "Dynamic scheduling for stochastic edge-cloud computing environments using A3c learning and residual recurrent neural networks," *IEEE Transactions on Mobile Computing*, 2020.

[14] Y. Wang, H. Liu, W. Zheng et al., "Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 39974–39982, 2019.

[15] K. Zervoudakis and S. Tsafarakis, "A mayfly optimization algorithm," *Computers & Industrial Engineering*, vol. 145, Article ID 106559, 2020.

[16] S. AlameluMangai, B. Ravi Sankar, and K. Alagarsamy, "Taylor series prediction of time series data with error propagated by artificial neural network," *International Journal of Computer Application*, vol. 89, no. 1, 2014.

[17] A. Lakhan, M. Ahmad, M. Bilal, A. Jolfaei, and R. M. Mehmood, "Mobility aware blockchain enabled offloading and scheduling in vehicular fog cloud computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, 2021.

[18] X. Chen, Y. Zhou, L. Yang, and L. Lv, "Hybrid fog/cloud computing resource allocation: joint consideration of limited communication resources and user credibility," *Computer Communications*, vol. 169, pp. 48–58, 2021.

[19] M. Abbasi, E. Mohammadi-Pasand, and M. R. Khosravi, "Intelligent workload allocation in IoT-Fog-cloud architecture towards mobile edge computing," *Computer Communications*, vol. 169, pp. 71–80, 2021.

[20] R. K. Naha, S. Garg, A. Chan, and S. K. Battula, "Deadline-based dynamic resource allocation and provisioning algorithms in fog-cloud environment," *Future Generation Computer Systems*, vol. 104, pp. 131–141, 2020.

[21] H. Sun, H. Yu, G. Fan, and L. Chen, "Energy and time efficient task offloading and resource allocation on the generic IoT-fog-cloud architecture," *Peer-to-Peer Networking and Applications*, vol. 13, no. 2, pp. 548–563, 2020.

[22] S. Mishra, M. N. Sahoo, S. Bakshi, and J. J. Rodrigues, "Dynamic resource allocation in fog-cloud hybrid systems using multicriteria AHP techniques," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8993–9000, 2020.

[23] I. Z. Yakubu, L. Muhammed, Z. A. Musa, Z. I. Matinja, and I. M. Adamu, "A multi agent based dynamic resource allocation in fog-cloud computing environment," *Trends in Sciences*, vol. 18, no. 22, p. 413, 2021.

[24] P. Narayana, C. Jatoth, and P. Parvataneni, "Optimising resource scheduling based on extended particle swarm optimization in fog computing environments," *Concurrency and Computation Practice and Experience*, vol. 33, 2021.

[25] C. Huang, H. Wang, L. Zeng, and T. Li, "Resource scheduling and energy consumption optimization based on Lyapunov optimization in fog computing," *Sensors*, vol. 22, no. 9, p. 3527, 2022.

[26] B. H. Husain and S. Askar, "Smart resource scheduling model in fog computing," in *Proceedings of the 2022 8th International Engineering Conference on Sustainable Technology and Development (IEC)*, pp. 96–101, Erbil, Iraq, February 2022.

[27] Md R. Hossain, Md Whaiduzzaman, A. Barros et al., "A scheduling-based dynamic fog computing framework for augmenting resource utilisation," *Simulation Modelling Practice and Theory*, vol. 111, Article ID 102336, 2021.

[28] M. Abd Elaziz, L. Abualigah, and I. Attiya, "Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments," *Future Generation Computer Systems*, vol. 124, 2021.

[29] A. Najafizadeh, A. Salajegheh, A. M. Rahmani, and A. Sahafi, "Privacy-preserving for the internet of things in multi-objective task scheduling in cloud-fog computing using goal programming approach," *Peer-to-Peer Networking and Applications*, vol. 14, pp. 3865–3890, 2021.

[30] A. C. Caminero and R. Muñoz-Mansilla, "Quality of service provision in fog computing: network-aware scheduling of containers," *Sensors*, vol. 21, no. 12, p. 3978, 2021.

[31] M. Macías, J. O. Fitó, and J. Guitart, "Rule-based SLA management for revenue maximisation in cloud computing markets," in *Proceedings of the 2010 International Conference on Network and Service Management*, pp. 354–357, Niagara Falls, Canada, October 2010.

[32] A. Badshah, A. Ghani, G. A. ShahaboddinShamshirband, and A. Pescapè, "Performance-based service-level agreement in cloud computing to optimise penalties and revenue," *IET Communications*, vol. 14, no. 7, pp. 1102–1112, 2020.

[33] S. Shin, Y. Kim, and S. Lee, "Deadline-guaranteed scheduling algorithm with improved resource utilisation for cloud computing," in *Proceedings of the 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 814–819, Las Vegas, NV, USA, January 2015.

[34] A. Maarouf, A. Marzouk, and A. Haqiq, "A novel penalty model for managing and applying penalties in cloud computing," in *Proceedings of the 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, pp. 1–6, Marrakech, Morocco, November 2015.

[35] K. Tsakalozos, V. Verroios, M. Roussopoulos, and A. Delis, "Live VM migration under time-constraints in share-nothing IaaS-clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 8, pp. 2285–2298, 2017.

[36] J. Devagnanam and N. M. Elango, "Design and development of exponential lion algorithm for optimal allocation of cluster resources in cloud," *Cluster Computing*, vol. 22, no. 1, pp. 1385–1400, 2019.

[37] Z. Zhou, Z. Hu, and K. Li, "Virtual machine placement algorithm for both energy-awareness and SLA violation reduction in cloud data centers," *Scientific Programming*, vol. 2016, Article ID 5612039, 11 pages, 2016.

[38] *GWA-T-12 Bitbrains Dataset*, Delft University of Technology, Delft, Netherlands, 2021, http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains.

[39] N.-M. Dang-Quang and M. Yoo, "An efficient multivariate autoscaling framework using Bi-lstm for cloud computing," *Applied Sciences*, vol. 12, no. 7, 2022.