Journal of Autonomous Intelligence (2024) Volume 7 Issue 5
doi: 10.32629/jai.v7i5.1554

ORIGINAL RESEARCH ARTICLE

Adaptive Threshold Fuzzy C-Means (ATFCM) VMmigration and resource optimization based dynamic scheduling for edge-cloud computing environments

S. Supriya[1,*], K. Dhanalakshmi[2]

[1] *Departmentof Computer Science, Kongunadu Arts and Science College, Coimbatore 641029, India*

[2] *Department of Information Technology, Kongunadu Arts and Science College, Coimbatore 641029, India*

*** Corresponding author:** S. Supriya, supriyasundaram@gmail.com

ABSTRACT

The method of delivery of information technology services has changed according to cloud computing. The latest generation of IoT applications benefits from low latency response offered by edge-cloud computing architecture. The migration procedure suffers when there is insufficient network capacity available. This further increases the difficulty of scheduling and resource monitoring. In this study, (1) Average Migration Time (AMT), Average Energy Consumption (AEC), Average Response Time (ART), and Average Service Level Agreement Violations (SLAV) evaluation parameters are minimised using Mutation Donkey and Smuggler Optimisation (MDSO). The amount of load that they can manage, data centre servers are categorised into four groups using Adaptive Threshold Fuzzy C-Means (ATFCM) clustering: extremely low load, mild load, medium load, maximum load. ATFCM moves thevirtual machine (VM) on maximum loaded or extremely low loaded hosts to very lowly loaded hosts. Utilising host information from Resource Monitoring Service (RMS), Residual Recurrent Neural Network (R2N2). Asynchronous advantage actor critical (A3C) learning is acknowledged for its ability to swiftly adapt to dynamic settings with fewer data, whereas R2N2 for rapid updating of model parameters. When contrasted to modern methods, trials done on practical applications data sets in areas of energy usage, SLA, response time, and running costs.

Keywords: deep reinforcement learning; edge computing; Residual Recurrent Neural Network (R2N2); Asynchronous Advantage Actor-Critic; Mutation Donkey and Smuggler Optimization (MDSO) algorithm

ARTICLE INFO

Received: 18 January 2024
Accepted: 26 February 2024
Available online: 15 May 2024

1. Introduction

Internet of Things (IoT) applications are growing rapidly in multiple domains. These edge devices have put a lot of strain on the Cloud Computing (CC) infrastructure and centralized control for real-time operation and control of data storage and processing. Due to significant network latency, deployed conventional cloud-centric IoT apps do not adapt to time-sensitive applications.

Several versions of virtual resources are created by CC and are housed on a real server. These physical servers are widely distributed in huge Data Centers (DC) in numerous places. DC assign various computing resources, including processors, storage, and networks, to end users in accordance with their needs[1]. DCs use a lot of energy in order to run on a regular basis. In addition, users' reliance on CC has multiplied with the growth of digital ecosystems including smart cities, healthcare, and IoT. Additionally, demands for real-time data storages and processes with CC have grown phenomenally along with

IoT.

Edge Computing (EC) interface can efficiently reduce the workload and enable real-time processing while maintaining data security. Due to expense and viability factors, the resources at network's edge are limited. To assist additional apps and improve Quality of Services (QoS), correct use of Edge resources is essential. Edge computations are a complex paradigm. The capacity, response time, speed, and energy consumption of computing servers among remote clouds and local edge nodes dramatically vary as a result of heterogeneity. Edge paradigm's mobility element, the bandwidth among processing nodes and data sources is always changing, demanding continuing dynamic tuning to meet application requirements.

Heuristics or rule-based rules have prevailed task scheduling techniques in Edge-Cloud systems[2]. Heuristics frequently function effectively in normal situations; however, they do not take into consideration the dynamic settings caused by workloads and hybrid computational paradigms[3]. Moreover, they struggle to adapt to ongoing system changes, which are typical in Edge-Cloud settings[4]. Given this, a reinforcement learning (RL) based scheduling method offers a practical way to optimise the system dynamically.

Most value-based RL approaches function miserably in Edge-Cloud deployments because they are unsuitable for highly stochastic situations, as demonstrated by prior research[5]. There aren't many works that can apply policy gradient methods[6], enhance for just one QoS parameter, and steer clear of using asynchronous updates in highly stochastic environments if you want faster flexibility. Furthermore, temporal patterns in workload, network, or node behaviour have not been utilised in any of the previous studies to improve scheduling decisions. Furthermore, the centralised scheduling approach used in these studies is unsuitable for contexts that are hierarchical or decentralised.

The effective utilization of data center resources is an important task more users migrate to the cloud. Depending on the workload, certain servers might be overcrowded while others might be underutilized. If a physical server breaks without VM migration, every VM that is running on it will also fail. Furthermore, the service quality is impacted by the time required to re-instantiate such VM on different servers. All of these problems have been overcome by the introduction of live VM migration. In cloud data centers, it makes server consolidation, energy conservation, fault tolerance, maintenance, and traffic management simpler. Consequently, serving as a critical technology for data center administration. As a result of their close communication, moving one VM frequently necessitates moving all other associated VMs.

This research provides an innovative way to maximize the migration of virtual machines utilizing ATFCM clustering. MDSO algorithm has been introduced to represent the complicated workload patterns and resource variability. Additionally, it uses asynchronous policy gradient techniques to resolve the scheduling issue in dynamic edge-cloud environments. These techniques use residual recurrent neural networks (R2N2) to constantly modify system dynamics for producing improved outcomes.

2. Literature review

Aujla and Kumar[7] introduced an effective plan for Software-Defined Networking (SDN)-based MEnSuS of cloud data centers. Support Vector Machine (SVM) driven workload categorization technique for classifications. Furthermore, a two-phase game is developed for workload scheduling in order to ensure edge-cloud sustainability of DC. In the cloud environment, several consolidation solutions are also provided to enhance energy efficiency the most use of computer and network resources. The evaluation findings using Google workload traces show the the proposed system's effectiveness.For all these cases, using realistic weather traces, the mapping of energy generated by Renewable Energy Sources (RES) and energy consumption is done to assess the impact of proposed scheme on sustainability.

Nayyer et al.[8] developed a Load Balancing for Resource Optimization (LBRO), a platform for collaborative cloudlets that takes user preferences into account when addressing load balancing issues in

edge computing. The proposed model not only addresses resource scarcity of cloudlets but also resolves the under-provisioning of resources at peer cloudlets thus maximizing resource utilization at cloudlet level. The experimental results show decreased load and stable resource utilization at cloudlets without jeopardizing performance of applications running on them. The suggested method greatly outperforms the traditional edge-based strategy with regard to of Central Processing Unit (CPU), storage, and disk utilization, according to a comparison of the two approaches.

Li et al.[9] proposed a mechanism for data transport and a strategy for allocating resources that is adaptable. The edge cloud cluster's adaptive resource allocations are made possible by predictions. Resource allocations with lowest service costs for the edge cloud clusters are selected using adaptive resource allocation approaches. Cluster load balances and data dependability are guaranteed by data movements. In the experiment, the load prediction accuracy and resource reconfiguration time under different loads are evaluated.Then compare the related work in the edge cloud and verify the effectiveness of proposed algorithm. Finally, the feasibility of the data migration algorithm is verified by experiments.Several studies show that the suggested approach may greatly increase system efficiency by reduced cost controls, increased, load balances and data integrity.

Zeng et al.[10] suggested a model-free method that, with no prior expertise, can suit the network dynamics well. To achieve this, a model-free DRL technique was presented to manage network edge resources effectively. They created agents for mobility-aware data processing service migrations which adhered to their DRL design principles. The results of the studies demonstrate that the agent is capable of automatically recognizing user mobility patterns and controlling the service migration between edge servers to reduce runtime operational costs. There is also a clear discussion of some prospective future research problems.

Tuli et al.[11] suggested an A3C based real-time scheduler for stochastic Edge-Cloud environments allowing decentralized learning, concurrently across multiple agents.R2N2 architecture to capture a large number of host and task parameters together with temporal patterns to provide efficient scheduling decisions. Stochastic Edge-Cloud instances provide decentralised learning among several agents through the construction of an A3C-based real-time scheduler. Implementing R2N2 architecture enables scheduling choices to be made effectively. It is capable of gathering a broad variety of temporal patterns, host, and task data. Studies on real-world datasets show a considerable increase in energy usages, response times, SLA, and operating expenditures by 14.40%, 31.90%, 7.74%, and 4.64%, respectively, as compared to current approaches.

Nabavi et al.[12] suggesteda multiple-goal VM placements in edge clouds which optimized network traffic and power of data centres using Seagull optimization. By centering VM communications on a single PM, the amount of data transported via the network is decreased, and the quantity of energy used by PM is decreased by VM consolidation to fewer, more energy-efficient PM. Two distinct network topologies, VL2 and three-tier, have been tested using CloudSim to demonstrate that the suggested strategy may successfully minimize traffic and power usage in ECDC.The experimental results show that proposed method can decrease energy consumption by 5.5% while simultaneously reducing network traffic by 70% and the power consumption of the network components by 80%.

3. System model and problem formulation

This study takes into account that the underlying architecture consists of both cloud and edge nodes. The system structure is shown in broad strokes in **Figure 1**. The distributed heterogeneous resources that make up the edge-cloud environment are linked to the network edge through a multi-hop remote cloud. Various application functionalities are hosted by the computer resources. Although edge devices provide

significantly faster reaction times since they are located closer to the clients, resource limitations restrict their computing capacity. More distant cloud resources, however, should provide much faster responses for users.

The technology involves Resource Monitoring Applications, VM Migration, and Scheduling are all included in the Resource Management System (RMS) for management. Tasks with their QoS and SLA criteria are sent to RMS by IoT users and devices. On the basis of the optimization goals, it sets new tasks that periodically determine if current tasks should be transferred to new hosts. The RAM, bandwidth, CPU, and disk limitations and the estimated completion times have an impact on the RMS option. The effect is replicated using Workload Generation Module (WGM), stochastic task generators that adhere to dynamic workload paradigm for job executions.

A Deep Reinforcement Learning Module (DRLM), which the Migration and Scheduler services interface with, offers placement recommendations for each work to the previous services. Run numerous schedulers with distinct partitioning of jobs and nodes in place of a single scheduler. Asynchronous updates are made possible by policy learning of DRLM, which alloted schedulers unique instances of global neural networks. The DRLM's proposal is assessed by the Constraint Satisfaction Module (CMS), sometimes referred to as the RMS, taking into account constraints such when a task has started migrating or the target host is loaded.

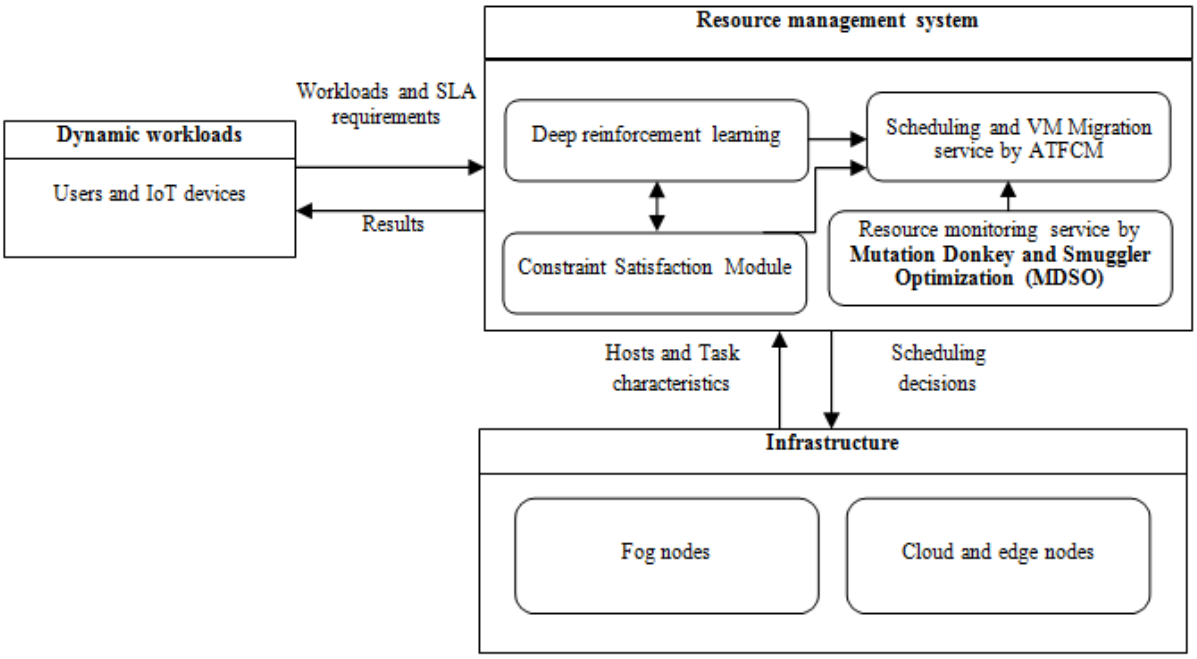

Figure 1. System model.

Workload Model:Each task involves dynamic workload, and task generation is stochastic. Separate execution time to scheduling intervals of similar length, as was done in earlier research[13]. As seen in **Figure 2**, the ordering of the scheduling intervals is represented by numbers.

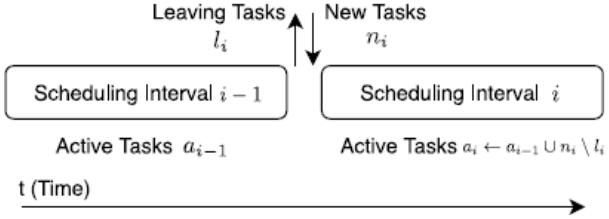

Figure 2. Dynamic task workload model.

i^{th}schedule intervals SI_i, starting at times t_i, stay till next intervals denoted as t_{i+1}. Active tasks in $SI_i$are those that were running on the hosts and are identified by the letter a_i.Additionally, at the start of SI_i, the completed tasks are designated as l_i, and the WGM sends new tasks, which are designated as n_i.New tasks n_i are created in the system as the tasks $l_i$are removed. Therefore, tasks that were active for SI_i were $a_i$is$a_{i-1} \cup n_i \backslash l_i$.

Problem Formulation: Represent loss of the interval SI_i as $Loss_i$.H_i represents i^{th} hosts in enumerated Host samples i.e. $[H_0, H_1, \dots, H_n]$ which imply their allocations for tasks Tas $\{T\}$. Executing Hosts' parameters, tasks from previous intervals ($a_{i-1} \backslash l_i$), and new tasks n_i are included in SI_i, designated as $State_i$. For each task, the scheduler must choose $a_i (= a_{i-1} \cup n_i \backslash l_i)$the host that is going to be assigned to or migrated to, which is $Action_i$ for SI_i. Let $m_i \subseteq a_{i-1} \cup l_i$ be the migratabletasks. So, $Action_i = \{h \in$ Hosts for task $T | T \in m_i \cup n_i\}$ which is a decision about task migration in $m_i$and allocationdecision for tasks in n_i. Scheduler, which indicates that the Model is a function: $State_i \rightarrow Action_i$. $Loss_i$ of an intervalrelying on the way hosts are assigned their roles i.e., $Action_i$by the Model, n= Edge-Cloud Data enter Host counts. Consequently, the issue can be expressed as given by Equation (1) for the best Model,

$$\sum_i Loss_i \; subjectto \forall i, Action_i = Model(State_i) \forall i \forall T \in m_i \cup n_i, \{T\} \leftarrow Action_i(T) \quad (1)$$

Having established the loss function and input-output requirements, specify the process for updating the Model following each scheduling interval.

4. Reinforcement learning model

This study takes into account that the underlying architecture consists of both cloud and edge nodes. The system structure is shown in broad strokes in **Figure 1**.

Model for policy gradient learning that uses reinforcement learning to solve the issue.

Input Specification: The scheduling model's input is the $State_i$, a collection of host-specific characteristics that include CPU, bandwidth, RAM, and disc usage and capacity[14] along with Hosts' parameters including reaction time, cost per units of time, million instructions per second (MIPS), and number of tasks given to it. It is possible to guarantee low energy usage by allocating several tasks to a limited group of hosts. Every host in the feature vector FV_i^{Hosts} specifies these parameters. In a_i, the jobs are divided into two distinct sets:n_i and $a_{i-1} \backslash l_i$.

Output Specification: The model must, depending on the input $State_i$, allocate a host to each job in a_i at the start of the interval SI_i. The result ($Action_i$) are host assignments of new job tasks $\in n_i$ as well as migration decisions of executing tasks from the previous interval $\in a_{i-1} \backslash l_i$.Every work that is migrated needs to be able to migrate to the new host; hence, the job needs to fit within the feasible boundaries. Furthermore, once a host h is allocated to a job T, it shouldn't get overworked; that is, h is suitable for T. $Action_i$ is therefore taken through Equation (2) in order for the interval SI_i, $\forall T \in n_i \cup m_i, \{T\} \leftarrow Action_i(T)$,

$$Action_i = \begin{cases} h \in Hosts \forall t \in n_i \\ h_{new} \in Hosts \forall t \in m_i \; if \; t \; is \; to \; be \; migrated \end{cases} \quad (2)$$

$Action_i$fits $t \forall t \in n_i \cup m_i$and is susceptible. Vectors of preferred allocations of hosts for tasks might be the result for neural networks which provides ranked lists of hosts instead of one host for each activity.

5. Proposed methodology

MDSO algorithm assists in producing better outcomes of metrics including ART, AEC, AMT, and Average SLAV. ATFCM clustering divides data centre hosts into four groups based on the load they can

support: extremely low load, mild load, medium load, and maximum load. The VM on maximum loaded or extremely low loaded hosts is moved to extremely low loaded hosts using the ATFCM algorithm, while the VM on lowly loaded and medium loaded sites is left in place. Afterward, based on ATFCM clustering, threshold has been generated based on the fuzzy function. The R2N2 model uses demands and host features from RMS to forecast the upcoming scheduling choices. With less data, A3C learning recognizes and adapts dynamically and quickly than R2N2.

AEC: An interval's AEC is described as the infrastructure's energy use regulated by environmental peak powers and $\alpha_h \in [0,1]$ are added to energy used by hosts $h \in Hosts$, considering user demands and deployment plans which can be customized for cloud and edge nodes. Powers are normalized using Equation (3),

$$AEC_i^{Hosts} = \frac{\sum_{h \in Hosts} \alpha_h \int_{t=t_1}^{t_{t+1}} P_h(t)dt}{\sum_{h \in Hosts} \alpha_h P_h^{\max}(t_{i+1} - t_i)} \tag{3}$$

here $P_h(t)$is host h,power function with time,and$P_h^{max}$is h maximum possible power.

ART:ART represents average response times for all departing tasks (l_{i+1}) during periods SI_i , standardized by longest intervals up to those point in time.ART is defined asfollows Equation (4),

$$ART_i = \frac{\sum_{t \in l_{i+1}} Response\ Time\ (t)}{|l_{i+1}| \max_i \max_{t \in l_i} Response\ Time\ (t)} \tag{4}$$

AMT: AMT for all active jobs (a_i) during an interval SI_i is defined as the longest feasible migration time up to the previous interval, standardized by that longest time. AMTis defined as follows Equation (5),

$$AMT_i = \frac{\sum_{t \in a_i} Migration\ Time(t)}{|a_i| \max_i \max_{t \in l_i} Response\ Time\ (t)} \tag{5}$$

Average SLA Violations (SLAV): Mean values of SLA breaches for leaving a position (l_{i+1}).Average SLA Violations (SLAV), which is the total (i) SLA violation times for active host matrices and (ii) matrices of performance deteriorations due to migrations, as identified by SLA(t) of tasks T during intervals SI_i and depicted as Equation (6):

$$SLAV_i = \frac{\sum_{t \in l_{i+1}} SLA(t)}{|l_{i+1}|} \tag{6}$$

This model modifies the variables to fulfill restrictions in addition to minimizing $Loss_i$.

5.1. MDSO algorithm

The MDSO algorithm is clearly shown in **Figure 3**.

DSO algorithm is designed to minimize energy use measures for cloud and edge nodes by modeling the actions after that of donkeys. The algorithm simulates the movement of donkeys in an edge-cloud environment by choosing and searching for routes. For deploying the search activity and RMS in an edge-cloud context, two modes—donkeys and smugglers—have been developed. In the Smuggler mode, all potential routes are identified before the resource optimization is identified. Several donkey activities, including run, face & support and face & suicide, are utilized in the donkey's approach[15].

Part I: Smuggler (Non-Adaptive): Here, the smuggler will examine every resource that could be used from the tasks to the VM before making a decision upon particular metrics. The fitness function in the smuggler section is designed to identify an ideal resource based on many criteria, including AEC, ART, AMT, and SLAV. When this is finished, the donkey will be sent to the finest monitored VM schedule. Each task's parameters will be entered by the operator, and the solutions and fitness will be assessed in the

smuggler section. The responses will be clustered altogether according to how fit they are. The donkey will be sent according to the finest monitoring service that has been selected and briefly explained in Algorithm 1.

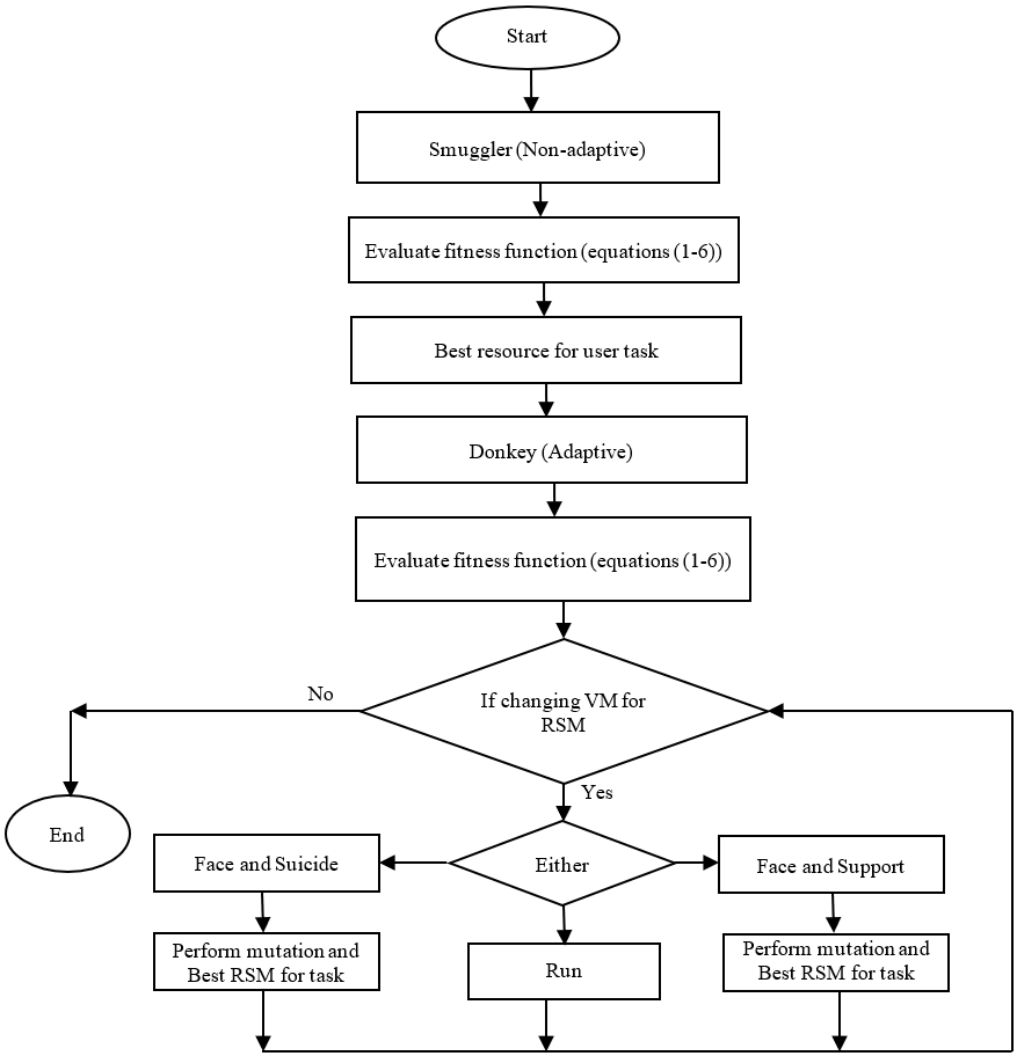

Figure 3. MDSO algorithm.

Part II: Donkey (Adaptive): The decisions made in adaptive routing depends on the resources in present VM state. According to Donkey's conduct, this response will be provided. The optimal resources were identified for user demands once the user task specifications were entered into each resource monitoring parameter. One of the following things will happen when the choke packet findings show that the ideal solution's fitness has decreased or is no longer good (another option currently has a greater fitness);

1) Run: A different optimal path (best solution) should be taken. When the ideal resource monitoring solution established in non-adaptive portions of non-optimal methods are eliminated and updated optimal resource monitors are set in accordance with the recent modifications.
2) Face and Suicide: selecting the ideal resource monitoring solution search path. There is no need to recalculate the fitness of the potential population; instead, discard the present route (VM) and employ the other resource that is ideal for the user job while correcting the blocked one. If any changes that lower the approach's fitness render the optimal answer found in the first portion of the procedure no longer the best. Once the first choice is eliminated and the situation returns to ideal, choose the next best option in the set of options.
3) Face and Support: Provide 2nd best VM solutions in case ideal resource solutions are discovered by smugglers begin to show overcrowding or congestion to prevent dropping solutions.

Algorithm 1 Mdso algorithm

1: Read the number of request and Task of user,$n_1, n_2,$

2: Smuggler Part

3: arbitrarily create the initial population of VM solutions.

4: **For**row=1 to n_1

5: **For**col=1 to n_2

6: parameters (rows, cols) =$rand_i$([decision ranges of variables],n_1, n_2)

7: end

8: end

9: For (e=1 to n_1) do

10: Examine fitness of results (Equations1-6)

11: Update potential RSM solutions' in population

12: End

13: EstablishthefinestRSM for each task

14: Demonstrate your best option

15: Provide the donkey part with the finest RSM solution.

16: Donkey Part

16.1. Determine whether there has been an upgrade in fitness.

16.2. If there is less of a change in the ideal solution's fitness

17: Runupdate the best solution

$$Best_{suicidsolution} = f(x_i) - f(mbest_{solution}) \quad (7)$$

$f(mbest_{solution})$denotes the mutation results of Equation (7) VM based on the user task.

18: **Face &Suicide convert** second-best fitness solutions from available options.

19: **Face &Support** Utilize the population's second-best option, which will serve the same function as the ideal solution, to support the best solution. (Population fitness is not updated.) (Equations (8) and (9))

$$secondbest_{solution} = f(mbest_{solution}) - f(x_i) \quad (8)$$

$$best_{supportsolution} = mbest_{solution} + secondbest_{solution} \quad (9)$$

20: End If

5.2. VM migration by ATFCM clustering

The CPU, memory disk, and bandwidth of servers in data centers all have an impact on the amount of energy they use. Energy usage and SLA violations in data centers can be decreased with proper VM mobility across servers. Yet, frequent VM migration may have a negative effect on the efficiency of any applications that use the VM. ATFCM Clustering Algorithm 2 is discussed below. Cost issues with moving VMs, as well as performance issues brought on by moving VMs Equations (10) and (11),

$$C = k. \int_{t_0}^{t_0 + T_{m_j}} u_j(t) dt \quad (10)$$

$$T_{m_j} = \frac{M_j}{B_j} \quad (11)$$

When variable C denotes the overall efficiency degradation brought on by VM_j, variable k denotes the average performance degradation brought on by VM, and the value of k is roughly equivalent to 0.1 (10%) of CPU use,$function\ u_j(t)$ is equivalent to the CPU use of VM_j, variable $t_0$signifies start time of migrations, T_{m_j} represents completion times, $M_j$corresponds to complete memory utilizations of VM_j, and B_j represents available bandwidths.

SLA Violation Metrics: SLA violations have to be considered in all VM migration algorithms. SLA violations are now classified in two ways.

(1) Overall Performance Degradations Caused by VM Migrations (PDM): It is specified by Equation (12),

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{d_j}}{C_{r_j}} \tag{12}$$

here variable Msignifies VM counts in data centers while C_{d_j} implies estimated performance degradations based on VM_j migrations, and C_{r_j}refers to the entire CPU capacity that VM_j required during the course of its existence.

(2) SLATAH (SLA Violation Time per Active Host): This represents percentages of total SLA violations where CPU usages of active host reached 100%, as per Equation (13).

$$SLATAH = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{s_i}}{T_{a_i}} \tag{13}$$

here Nindicates host counts in data centers, T_{s_i} stands for total time consumed by host i's CPU attaining cent percent usage, leading to SLA violations, and T_{a_i} corresponds to total times host i consume in active states. When the SLATAH happened, the CPU of the active host was 100% used, which means the virtual machine on the host might not have received the desired CPU capacity. Two effective methods for determining the SLA violation individually are SLATAH and PDM. That being said, the SLA is defined as follows Equation (14),

$$SLA = PDM \times SLATAH \tag{14}$$

Energy usage and SLA violations are both a part of energy conservation. Less energy usage and SLA violations result from increased energy efficiency. Consequently, the following describes the way the energy performance metric is explained in Equation (15),

$$E = \frac{1}{P \times SLA} \tag{15}$$

here Eimplies energy efficiencies of data centers y, P stands for their energy consumptions, and SLA is a data center's SLA violation. Equation (11) demonstrates that energy efficiency increases as E increases.

The energy efficiency of data centers can be raised through VM migration. But there are some significant issues that need to be resolved: (1) when a host is anticipated to be at its maximum capacity, wherein a host's VM can be moved to another host; (2) when it is decided to leave all VM on a host unmodified when it is expected to be minimally or moderately loaded; (3) when all VM are expected to be moved to another host due to underutilization; (4) selecting one or more VM that need to be migrated from hosts that are completely or partially loaded; and (5) finding new hosts to house migrating VM.

An algorithm called ATFCM is suggested for VM migration. Three criteria are automatically determined by the ATFCM algorithm Th_{low}, Th_{me}, $and\ Th_{max}$ ($0 \leq Th_{low} < Th_{me} < Th_{max} \leq 1$), that leads to the classifications of data center hosts as hosts with very low, low, medium, and maximum loads. Host is considered to be low-loaded when its CPU use is less than or equal to Th_{low}. The host must be put into sleep mode and all virtual machines must be moved from the very low loaded host to a different host with low load to lower the amount of energy used. A host is regarded to be low loaded when its CPU usage ranges from Th_{low} and Th_{me}.Because excessive VM migration outcomes in performance degradation and high SLA violations, all VM on lowly loaded hosts must be retained in place as a way to prevent high SLA violations. A host is considered to be medium loaded when its CPU usage is among Th_{me}, $and\ Th_{max}$. It seems that

the host is medium loaded; due to the performance decrease and severe SLA breaches caused by excessive VM migration, all virtual machines on medium-loaded hosts must remain unchanged; Whenever a host's CPU usage exceeds Th, the host is regarded as being maximally loaded; Some of the virtual machines on hosts that are fully loaded have to be migrated to hosts that are underutilized as a way to reduce SLA violations.

Data centre hosts are divided into four groups using ATFCM clustering based on the load they handle: extremely low load, low load, medium load, and maximum load hosts. Let us assume that the collection of n computers to be separated based on CPU utilization is represented by the expression $VM = \{vm_1, \ldots, vm_n\}$. Here, $vm_i \in R^d$ for $i = 1, \ldots, n; c$ represents cluster counts, where $2 \leq c < n$. Equation (16)[16] depicts fuzzy clustering.

$$P: minimize J_m(U, V) = \sum_{i=1}^{n} \sum_{j=1}^{c} (u_{ij})^m \|vm_i, v_j\|^2 \tag{16}$$

here U = $u_{ij}$is each $VM_i's$ membership matrix to clusters j; $V = \{v_1, \ldots, v_c\}$ are collections of centroids, where v_j represents cluster j's centroid;(The host's CPU use at time i is represented by vm_i; the size of n is determined empirically). The extent that the clusters overlap is indicated by the weighting exponent, or m. $m > 1$; $d = \|vm_i, v_j\|^2$specifies the Euclidean distance among the $vm_i$and the centroid $v_j$for $i = 1, \ldots, n$ and $j = 1, \ldots, c$. MinimizingJ_m, which represents estimated $V\ and\ U$ models in Equations (17) and (18),

$$v_j = \frac{\sum_{i=1}^{n} (u_{ij})^m vm_i}{\sum_{i=1}^{n} (u_{ij})^m}\ 1 \leq j \leq c \tag{17}$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|vm_i, v_j\|^2}{\|vm_i, v_k\|^2}\right)^{\frac{1}{m-1}}} 1 \leq i \leq n, 1 \leq j \leq c \tag{18}$$

where $vm_i$and $v_j$are the space-specific vectors R^d and are definite as follows Equations (19) and (20),

$$vm_i = (vm_1, \ldots, vm_d), 1 \leq i \leq n \tag{19}$$

$$v_j = (v_1, \ldots, v_d), 1 \leq j \leq c \tag{20}$$

Limitations of fuzzy clusters are formalized in Equations (21)–(23)[17],

$$u_{ij} \in [0,1], 1 \leq i \leq n, 1 \leq j \leq c \tag{21}$$

$$\sum_{j=1}^{c} u_{ij} = 1, 1 \leq i \leq n \tag{22}$$

$$0 < \sum_{j=1}^{c} u_{ij} < n, 1 \leq j \leq c \tag{23}$$

According to Equation (21), a vm_i's degree of membership in clusters j need to be between 0 and 1. Equation (22) states total of a vm_imembership degrees in several clusters must necessarily equal 1. Equation (23), which states that there cannot be any empty clusters and that there cannot be a single cluster, requires which the total of membership degrees in clusters are > 0 and < n. Thresholds (Th_{low} , Th_{me}, $and\ Th_{max}$) in ATFCM algorithms can be depicted as Equation (24)–(26),

$$Th_{low} = 0.5(1 - r * d) \tag{24}$$

$$Th_{me} = 0.75(1 - r * d) \tag{25}$$

$$Th_{max} = 1 - r * d \tag{26}$$

here $r \in R^+$ is an algorithmic variable that controls how quickly the system consolidates virtual machines. Higher r results in more energy usage but fewer SLA breaches brought on by VM consolidation.

Algorithm 2 Atfcm clustering

Input: Number of tasks $T = (t_1, \ldots, t_n), c, m, \varepsilon$

Output: U,V

1: Initialization

2: t:=0

3: $U^{(t)} \coloneqq \{u_{11}, \ldots., u_{ij}\}$ is randomly generated depending on the CPU utilization

4: Calculate centroids by equation (17)

5: Modify membership matrices with equation (18)

6: Convergence

7: If max {abs($u_{ij}^{(t)} - u_{ij}^{(t+1)}$)} $< \varepsilon$

8: stop the algorithm

9: else

10: $U^{(t)} \coloneqq U^{(t+1)}$ and t:=t+1

11: Go to step 4

12: end of the algorithm

5.3. Stochastic dynamic scheduling using policy gradient learning

Each scheduling session starts with the following: Task requests and job specifications, such as bandwidth, computation, and SLA needs, are received by the RMS and used by DRL algorithm with host features from RMS to forecast potential schedules in future. (3) Outputs of DRL are used by constraint satisfaction modules to make migrations and schedules possible. (4) When new jobs are detected, RMS alerts users or IoT devices for submitting requests directly to relevant edges or cloud devices. (5) Loss functions are computed and variables DRL variables are adjusted accordingly. Q-tables or neural network function approximates simulate these jobs in stochastic circumstances which result in deterministic non-flexible rules. The technique utilizes policy gradient optimization to estimate the policy and optimize it, utilizing $Loss_i^{PG}$ as a sign to update the network. Utilizing an R2N2 network, estimate the function from $Action_i^{PG}$ for each interval SI_i. The ability of an R2N2 system to capture complex temporal correlations among outputs and inputs is advantageous. After the present interval, a single network forecasts policy (actor head) and cumulative loss (critic head). One may modify the network variables and determine the optimal scheduling choice for each interval by repeatedly preprocessing the interval state and submitting it, together with the penalties and losses to R2N2. This enables the framework to quickly adjust to users, environments, and specific application demands. GRUs, or gated recurrent units, imitate the temporal features of task and host qualities such task RAM, CPU, and bandwidth needs as well as the hosts' RAM, CPU, and bandwidth capability, are used to generate recurrent layers.Features of elements e are represented as f_e, max and min values as $\underset{f_e}{max}\ and\ \underset{f_e}{min}$individually. These maximum and minimum values are computed utilizing two heuristic-based scheduling rules: Maximum-Migration-Time (MMT) for task selection, and Local-Regression (LR) for task allocation both of which depend on a sample dataset[18]. Following that, Equation (27) is used to standardize features,

$$e = \begin{cases} 0\ if\ \max f_e = \min f_e \\ \min\left(1, max\left(0, \dfrac{e - \min f_e}{\max f_e - \min f_e}\right)\right), else \end{cases} \tag{27}$$

R2N2 models receive these pre-processed inputs and flatten them for passing them through thick layers. By initially generating grouped lists of hosts and decreasing probabilities in O_i for each i, outputs created O are converted into $Action_i^{PG}$.Gradients have negative signs to reduce total losses and are proportionate to these quantities. Mean Square Error (MSE) of expected cumulative losses with the cumulative losses following one-step look-aheads are second gradient terms. Every scheduling interval, CSM converts the output $Action_i^{PG}$ to $Action_i$ and sends it to RMS.

6. Results and discussion

This section summarizes experimental configurations, evaluation metrics, dataset, and presents a comprehensive examination of the results, contrasting the model with other industry-standard methods. Properties like edge node response times, costs, and powers may be used with CloudSim. Software for Constraint Satisfactions with preprocessing and output conversions were built. Tasks and host monitoring services provided by CloudSim computed losses. Hosts in simulation environments were assigned tasks, or cloudlets and dispersed amongst VM.

Dataset: Tasks (cloudlets) VM were assigned to the simulation environment, and hosts are assigned to them after that. By assigning the ith generated. Bijections from cloudlets to VM can be assumed by connecting the Cloudlet to the ith produced VM and removing the VM when the related Cloudlet is finished for the current task setting in the edge-cloud environment. The real-world, publicly-available Bitbrain dataset is used to produce the dynamic workload for cloudlets. Real-time resource utilisation statistics for business-critical applications utilising Bitbrain architecture is available in the Bitbrain dataset[19,20]. This dataset was chosen because it illustrates real-world patterns of infrastructure utilisation and may be utilised to create accurate input feature vectors for learning algorithms. It contains almost a thousand virtual machine workload records from two different types of computers. The dataset contains workload statistics for each time stamp, divided into five minutes. These figures include the required number of CPU cores, the CPU utilisation measured in MIPS, the amount of RAM that was requested, and details on the disc (write/read) and network (receive/transmit) bandwidth. Divide the dataset into two halves, each having a 25.00% and a 75.00% virtual machine workload. The larger component is used to train the R2N2 network, whereas the smaller portions are used in tests, sensitivity analyses, and cross-references with similar studies[21]. Cloud layers' cost models are based on Microsoft Azure IaaS cloud service.

Metrics:The following metrics has been used for results comparison.

Average Response Time (ART) shown as follows,

$$ART = \frac{\sum_{t \in l_{i+1}} Response\ Time(t)}{|l_{i+1})} \tag{28}$$

SLAVdefine as follows,

$$SLAV = \frac{\sum_i SLAV_i . |l_{i+1}|}{\sum_i l_i} \tag{29}$$

Average Task Completion Time: It is determined by adding the average scheduling time of a task, the task's execution time, and the server's response time during the most recent scheduling interval. The overall number of tasks completed, the percentage of tasks finished within the anticipated execution time (based on the desired MIPS), the number of task migrations each time interval, and the total migration time transfer over the course of the period are all taken into account.

Results comparison Methods: A variety of heuristics have been employed in the dynamic scheduling methodology. For a number of sub-problems, including task/VM selection and host overload detection, the top three heuristics from this collection of sub-heuristics have been determined. These variations locate

target hosts using Best Fit Decreasing (BFD) method. Additionally, contrast the results with two well-liked RL techniques that are frequently applied in the literature.

LR-MMT: This dynamic workload distribution method selects tasks based on task selection and overload detection methods (LR and MMT).

The Maximum Correlation Policy (MC) and Median Absolute Deviation (MAD) heuristics are used in MAD-MC to dynamically schedule workloads based on task selection and overload detection.

DDQN: The Deep Q-Learning based RL approach has been used in a number of researches published in the literature.

DRL (REINFORCE): Using a linked neural network and policy gradients, the REINFORCE technique.

Figure 4 demonstrates that, when compared to the other scheduling schemes, the one suggested offers the shortest average response time. The suggested approach performs 10.26% lower than the baseline algorithms' top algorithm, A3C-CCTSO-R2N2. Since the suggested framework does not have multiple migrations or AMT inherent in the loss function, it specifically checks if nodes are clouds and edge nodes and distributes work using RMS (MDSO). This shows that the proposed system has lesser ART of 6.21 ms, whereas other methods such as DDQN, REINFORCE, A3C-R2N2, and A3C-CCTSO-R2N2 has increased ART of 8.74ms, 8.20ms, 7.46ms, and 6.92 ms respectively (refer **Table 1**).

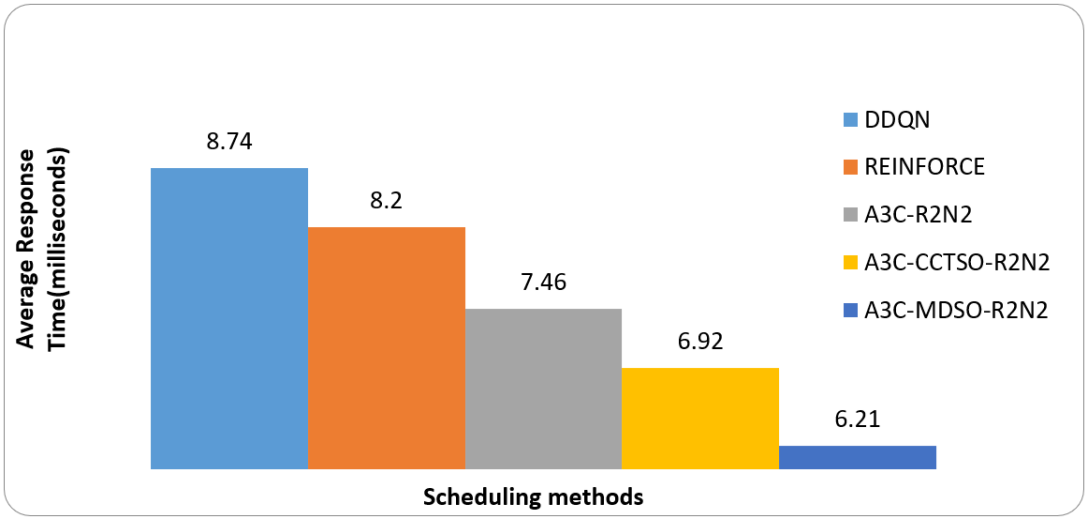

Figure 4. Average response time vs. scheduling methods.

Table 1. Average response time of methods' schedules.

Scheduling Methods	Response Time (ms)
DDQN	8.74
REINFORCE	8.20
A3C-R2N2	7.46
A3C-CCTSO-R2N2	6.92
A3C-MDSO-R2N2	6.21

According to **Figure 5**, the suggested approach has 34.61% less SLA breaches than the A3C-R2N2 policy, and it also has the lowest rate of SLA violations overall. Also reduced migrations and astute work schedules prevent significant losses resulting from SLA breaches. The proposed system has lower SLA violations (0.034), whereas other approaches (refer to **Table 2**) have higher ART of 0.072, 0.064, 0.052, and 0.040, respectively, such as LR-MMT, MAD-MC, DDQN, REINFORCE, and A3C-R2N2.

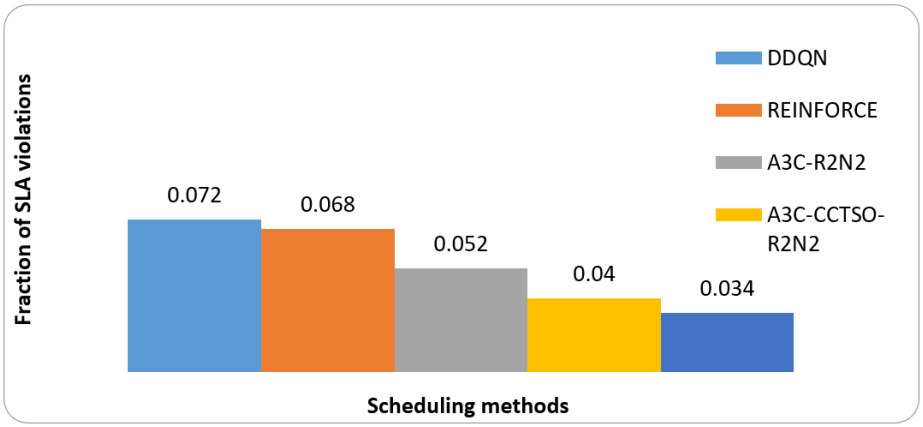

Figure 5. Fraction of SLAviolations vs. scheduling methods.

Table 2. SLA violation fractions of methods' schedules.

Scheduling Methods	Fraction of SLA Violations
DDQN	0.072
REINFORCE	0.064
A3C-R2N2	0.052
A3C-CCTSO-R2N2	0.040
A3C-MDSO-R2N2	0.034

As seen in **Figure 6**, the suggested model has high task completions which guarantee distributions of work amongsdt few cloud VMs and save cost. Proposed system has higher number of completed tasks as 1264, whereas other methods such as DDQN, REINFORCE, A3C-R2N2, and A3C-CCTSO-R2N2 has lesser number of completed tasks of 895, 978, 1127, and 1150 respectively (refer **Table 3**).

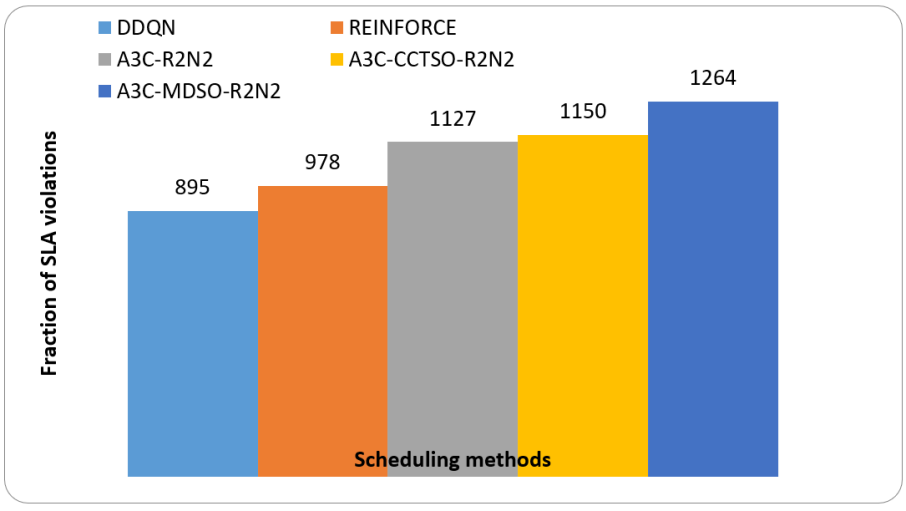

Figure 6. Number of completed tasks vs. scheduling methods.

Table 3. Complted task counts of methods' schedules.

Scheduling Methods	Fraction of SLA Violations
DDQN	895
REINFORCE	978
A3C-R2N2	1127
A3C-CCTSO-R2N2	1150
A3C-MDSO-R2N2	1264

Figure 7 shows the results of number of task migrations with respect to simulation time. The results are evaluated by several methods likeA3C-MDSO-R2N2, DDQN, REINFORCE, A3C-R2N2, and A3C-CCTSO-R2N2. From the results it shows that the proposed system has lesser number of task migrations of 16, whereas other methods such as DDQN, REINFORCE, A3C-R2N2, and A3C-MDSO-R2N2 has 30, 25, 20, and 18 for simulation time of 20 Hours (refer **Table 4**).

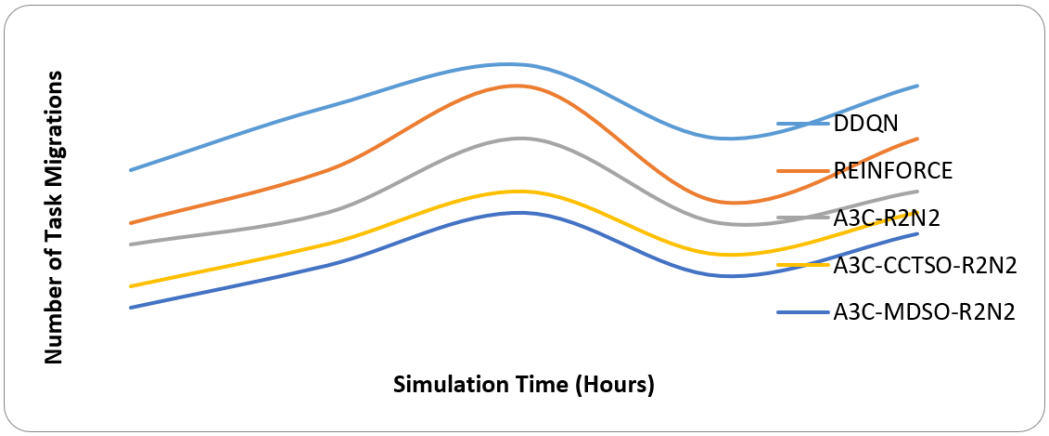

Figure 7. Task migration counts of methods during intervals.

Table 4. Number of task migration vs. scheduling methods.

Scheduling Methods	Simulation time (Hours)				
	0	**5**	**10**	**15**	**20**
DDQN	22	28	32	25	30
REINFORCE	17	22	30	19	25
A3C-R2N2	15	18	25	17	20
A3C-CCTSO-R2N2	11	15	20	14	18
A3C-MDSO-R2N2	9	13	18	12	16

Figure 8 shows the results of Interval Migration Time in task with respect to simulation time. The results are evaluated by several methods like A3C-MDSO-R2N2, DDQN, REINFORCE, A3C-R2N2 and A3C-CCTSO-R2N2. It shows that the proposed system has takes lesser total migration time of 1.19 Seconds, whereas other methods such as DDQN, REINFORCE, A3C-R2N2, and A3C-CCTSO-R2N2 has 9.50 Seconds, 7.80 seconds, 7.50 seconds, and 5.0 seconds for simulation time of 20 Hours(refer **Table 5**).

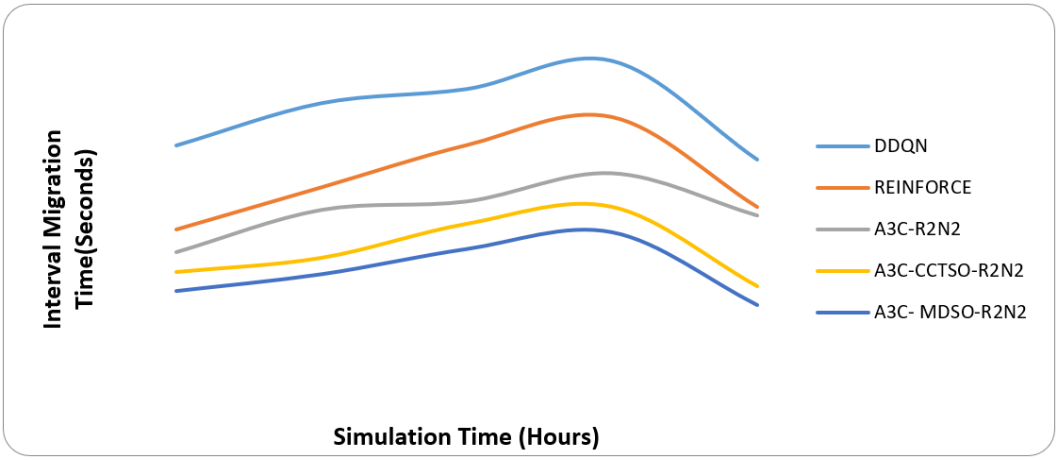

Figure 8. Total migration times during interval of methods' schedules.

Table 5. Total migration time in each interval vs.scheduling methods.

Scheduling Methods	Simulation time (Hours)				
	0	**5**	**10**	**15**	**20**
DDQN	10	11.5	12	13	9.5
REINFORCE	7	8.5	10	11	7.8
A3C-R2N2	6.2	7.7	8	9	7.5
A3C-CCTSO-R2N2	5.5	6	7.2	7.8	5
A3C-MDSO-R2N2	4.8	5.4	6.3	6.9	4.3

7. Conclusion and future work

A complete real-time task scheduler and migration for combined cloud and edge computing systems are the goals of this work. MDSO algorithm has been introduced for minimizing the metrics like AEC, ART, AMT, and SLAV. The technique simulates the movement of donkeys in an edge-cloud environment by choosing and searching for routes. For deploying the search behavior and RMS in an edge-cloud context, two modes, donkeys and smugglers have been developed. Virtual machine migration can improve the energy efficiency of data centres. For VM migration, ATFCM clustering has been implemented. Four classifications of hosts exist within a data centre: extremely low load, moderate load, medium load, and maximum load hosts. Data centre servers' RAM, disc, CPU, and bandwidth all affect how much energy they need. Appropriate virtual machine mobility among servers helps reduce data centre energy consumption and SLA breaches. For stochastic dynamic scheduling, an A3C-based policy gradient reinforcement learning algorithm is used. The A3C-R2N2 scheduler was implemented to increase efficiency; it can account for all important task and host characteristics. Based on the real Bitbrain dataset, CloudSim 5.0 shows the superiority of the model over existing methods. The existing design has limitations when it comes to scheduling jobs and edge nodes. Future research may focus on scalable reinforcement learning techniques like as Impala. Prepare to investigate issues related to data security and privacy as well.

Author contributions

Conceptualization, SS and KD; methodology, SS; validation, SS and KD; draft manuscript preparation, SS; visualization, SS and KD. All authors have read and agreed to the published version of the manuscript.

Conflict of interest

The authors declare no conflict of interest.

References

1. Katal A, Dahiya S, Choudhury T. Energy efficiency in cloud computing data centers: a survey on software technologies. Cluster Computing. 2023; 26(3): 1845-1875.
2. Mohiuddin I, Almogren A. Workload aware VM consolidation method in edge/cloud computing for IoT applications.Journal of Parallel and Distributed Computing. 2019;123: 204-214.
3. Skarlat O, Nardelli M, Schulte S, et al. Optimized IoT service placement in the fog. Service Oriented Computing and Applications. 2017; 11(4): 427-443.
4. Pham XQ, Man ND, Tri NDT, et al. A cost-and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. International Journal of Distributed Sensor Networks. 2017; 13(11): 1-16.
5. Mnih V, Badia AP, Mirza M, et al. Asynchronous methods for deep reinforcement learning. Proceedings of the International conference on machine learning. 2016; 1928-1937.
6. Mao H, Alizadeh M, Menache I, Kandula S. Resource management with deep reinforcement learning. Proceedings of the 15th ACM Workshop on Hot Topics in Networks. 2016; 50-56.
7. Aujla GS, Kumar N. MEnSuS: An efficient scheme for energy management with sustainability of cloud data centers in edge–cloud environment.Future Generation Computer Systems. 2018; 86: 1279-1300.

8. Nayyer MZ, Raza I, Hussain SA, et al. LBRO: Load Balancing for Resource Optimization in Edge Computing. IEEE Access. 2022; 10: 97439-97449.
9. Li C, Sun H, Tang H, Luo Y. Adaptive resource allocation based on the billing granularity in edge-cloud architecture.Computer Communications. 2019; 145: 29-42.
10. Zeng D, Gu L, Pan S, et al. Resource management at the network edge: A deep reinforcement learning approach.IEEE Network. 2019; 33(3): 26-33.
11. Tuli S, Ilager S, Ramamohanarao K, Buyya R. Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks.IEEE transactions on mobile computing. 2020; 21(3): 940-954.
12. Nabavi S, Wen L, Gill SS, Xu, M. Seagull optimization algorithm based multi-objective VM placement in edge-cloud data centers.Internet of Things and Cyber-Physical Systems. 2023; 3: 28-36.
13. Akbari M, Hassan R, Alizadeh SH. An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems. Engineering Applications of Artificial Intelligence. 2017; 61: 35-46.
14. Kim D, Son J, Seo D, et al. A novel transparent and auditable fog-assisted cloud storage with compensation mechanism. Tsinghua Science and Technology. 2019; 25(1): 28-43.
15. Shamsaldin AS, Rashid TA, Al-Rashid Agha RA, et al. Donkey and smuggler optimization algorithm: A collaborative working approach to path finding.Journal of Computational Design and Engineering. 2019; 6(4): 562-583.
16. Gosain A, Dahiya S. Performance analysis of various fuzzy clustering algorithms: a review.Procedia Computer Science. 2016; 79: 100-111.
17. Pérez-Ortega J, Rey-Figueroa CD, Roblero-Aguilar SS, et al. POFCM: A Parallel Fuzzy Clustering Algorithm for Large Datasets.Mathematics. 2023; 11(8): 1-16.
18. Beloglazov A, Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurrency and Computation: Practice and Experience. 2012; 24(13): 1397-1420.
19. Shen S, van Beek V, Iosup A. Statistical characterization of business-critical workloads hosted in cloud datacenters. 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. 2015; 465-474.
20. Available online: http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains (accessed on 25 December 2023).
21. Supriya S, Dhanalakshmi K. Residual Recurrent Neural Network (R2N2) and Intelligent Resource Optimization based Dynamic Scheduling for Edge-Cloud Computing Environments.International Journal of Intelligent Systems and Applications in Engineering. 2023; 12(8s): 160–172.