

The A3C-CCTSO-R2N2 algorithmic framework for precise edge-cloud parameter estimation

Gangadharan Manonmani¹, Krishnasamy Ponmozhi², Krishnasamy Balasubramanian¹

¹Department of Computer Applications, Kalasalingam Academy of Research and Education, Krishnankoil, India

²Department of Master of Computer Applications, SRM University, Chennai, India

Article Info

Article history:

Received Jul 9, 2024

Revised Aug 6, 2025

Accepted Sep 1, 2025

Keywords:

Acoustic echo cancellation

Adaptive resonance theory

Cloud computing task

scheduling optimization

Edge computing

Internet of things

Recurrent residual network

ABSTRACT

Efficient resource allocation is crucial in fog computing environments due to dynamic conditions and different user requirements; this work addresses the scheduling issues of internet of things (IoT) applications in such situations. Our proposed method, chaotic crossover tuna swarm optimizer (CCTSO), is based on metaheuristics and aims to reduce energy usage, reaction time, and SLA breaches; it should help with these problems. Improved system responsiveness and dependability are outcomes of the suggested approach's use of machine learning models for scheduling decision prediction and dynamic workload adaptation. The framework achieves a good balance between performance and energy efficiency by adjusting critical parameters and application settings. By reducing energy usage, reaction time, and operational cost while retaining reduced service level agreement (SLA) violation rates, our solution greatly outperforms previous techniques, according to experimental assessments. In real-world implementations, our results demonstrate that CCTSO is a strong solution for fog-based IoT scheduling, providing greater scalability and adaptability. Taken together, the results of this study provide a strong algorithmic foundation for better resource management in cloud, fog, and edge computing environments.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Gangadharan Manonmani

Department of Computer Applications, Kalasalingam Academy of Research and Education

Krishnankoil, Tamil Nadu, India

Email: gmanonmani24@gmail.com

1. INTRODUCTION

The advancement of the internet of things (IoT) has resulted in the generation of vast amounts of data at a rapid pace. Applications need a strong enough computing infrastructure to support user demands to exploit this data for analysis and goal-directed action [1]. Many time-sensitive applications, including traffic monitoring, emergency response, and healthcare, find it challenging to integrate cloud-centric IoT software due to increasing network latency. The IoT has limited consumer resources, such as storage capacity and signal resolution. To help it successfully unload its tasks, traditional cloud computing was developed [2], [3]. But if a lot of task data must be transmitted, between local PCs and cloud centers outsourcing localized jobs to a remote cloud center may cause serious delays. This kind of delay is unacceptable for nearly all scenarios and applications that rely on latency. Furthermore, transferring a large amount of data over a network that is linked to a central server could cause the network to become extremely congested. New technologies are required to address the increasing demands on resources and quality of service (QoS) requirements [4]. The edge cloud computing (ECC) paradigm is a promising approach that provides IoT applications with a shorter latency response [5]-[7]. ECC has been recommended because network edges have limited processing power and respond slowly in times-sensitive applications [8].

To improve service quality, a new component called ECC is added between the cloud computing environment and the on-premises level [9]. In contrast to cloud computing, which utilizes a core network transport for data, edge-to-local (ECC) placement positions server hardware closest to the edge. When providing high downstream bandwidth, ECC can reduce system latency by preventing the undesired delay brought on the networks [10]. Consequently, there has been an increasing focus on ECC in both academic and industry research [11]. In an ECC setting, outsourcing computation has been the subject of a great deal of research in the last few years [12]. Planning the Edge program's computational paradigm is difficult, nevertheless, for a variety of reasons. Regarding energy usage, response time, speed, and capacity, computer servers adapt to the heterogeneity between adjacent edge nodes and distant clouds. Furthermore, there might be a variety of computer kinds between the periphery and cloud levels between the periphery and cloud levels. Moreover, the edge paradigm's mobility component results in constant bandwidth fluctuations between data sources and processing nodes, necessitating constant dynamic adjusting to satisfy application needs. The ECC's uncertain task arrival rate, job duration, and resource requirements exacerbate the scheduling issue. Dynamic task scheduling is required to lower energy consumption and improve the QoS of applications in unpredictable situations by effectively utilizing multi-layer resources. Cloud centers use several forms of task scheduling algorithms, including hybrid algorithms, heuristic computations, and meta-heuristic strategies that use swarm intelligence and biological motivation [13].

Several performance indicators have been incorporated into the scheduling process, including system usage, latency, load balancing, execution time, and cost of a network connection [14]. The heuristic work scheduling technique can be used to determine the best solution. But it's prone to incomplete selection and can't always be counted on to yield the best results. A heuristic method that mixes random algorithms and local search is improved upon by the meta-heuristic approach [15], [16]. It permits the exploration and enlargement of the search space and manages a great deal of search space knowledge. It can also learn and master new information by applying learning processes, which allows it to find approximations of ideal solutions. The dynamic conditions arising from requirements and the growth of the EdgeCloud computing paradigm are not taken into account by heuristic models. Furthermore, they struggle to adapt to frequent system modifications that occur in EdgeCloud settings. To achieve this, a scheduling strategy based on reinforcement learning (RL) may be beneficial for the system's dynamic optimization [17].

The models have greater accuracy and capable of identifying intricate relationships between multiple interdependent components since they are built using real data. Recently, many value-based RL techniques have been used to optimize resource management systems (RMS) in remote systems [18]. These systems either use neural networks or use tables to store Q-value functions. The states of ECC settings indicate expected cumulative rewards in an RL setup. Moreover, no previous research has improved scheduling decisions by utilizing temporal trends in workload, network, or node activity. Additionally, a centrally managed scheduling policy is used in these studies, which is undesirable for decentralized or hierarchical systems. The scheduling problem in stochastic edge-cloud systems is mapped and resolved using asynchronous policy gradient approaches. These methods, which use recurrent neural networks to find patterns in behavior, may continuously adjust to the system's dynamics to yield better outcomes. The scheduling problem in stochastic edge-cloud systems is mapped and resolved using asynchronous policy gradient approaches. These methods, which use recurrent neural networks to find patterns in behavior, may continuously adjust to the system's dynamics to yield better outcomes.

This research presents the chaotic crossover tuna swarm optimizer (CCTSO), which aims to minimize indices such as acoustic echo cancellation (AEC), adaptive resonance theory (ART), adaptive multimodal transformer (AMT), cost (C), and service level agreement violation (SLAV). Unlike conventional deep QNetwork (DQN) systems, the proposed method may quickly adjust the allocation strategy based on host behavior, changing workloads, and QoS requirements. Additionally, this study demonstrates how to use an R2N2-based strategy that accounts for temporal trends in planning in a hybrid edge-cloud situation. Organization of the work section 2 will see the literature review, section 3 followed by the system model of this work, section 4 indicates a model for RL, section 5 followed by the proposed work, results of this work in section 6, and finally the conclusion of this work in section 7.

2. REVIEW OF THE LITERATURE

In order to improve the reduction of service latency and lower the overall system cost (SC) with available resources, Wang *et al.* [19] offer a QoS-guaranteed edge user data deployment technique. The proposed method has to be theoretically tested and its performance analyzed to compare it with three other benchmark approaches that use real-world datasets. Trials demonstrate the suggested strategy's effectiveness and efficiency in comparison to the current practices. The hybrid edge cloud (HEC) is a unique architectural technique for cloud decentralized governance that Zhou *et al.* [20] designed to alleviate the burden on computer resources that are centrally managed, such as server facilities. HEC makes use of smart device

resources to lower communication latency and free up network capacity. Utilizing smart devices' computational power and creating a decentralized infrastructure that is robust and scalable in the future of hyperconnected, The benefits of contemporary network technologies, such as 5G and Wi-Fi6, are combined in HEC. To maximize resource utilization and reduce transmitting objections, Zheng *et al.* [21] improved the transfer of work under delay limits in the edge-cloud computing system. While considering optimal decisions on task offloading and resource allocation, deep reinforcement learning (DRL) is applied. To formulate this optimization issue and identify the best method for burden offloading, a stochastic decision process is employed, followed by DQN. The DQN-edge-cloud (DQNEC) computational technique was developed to adjust the policy and offload the work as effectively as feasible in a flexible edge-cloud system while taking resource utilization into consideration. When it comes to optimizing task offloads with low task rejection rates and resource utilization at cheap prices, DQNEC outperforms heuristic approaches, according to simulations. The use of an R2N2 with A3C learning was suggested by Kim *et al.* [22] as a quick way to adjust to shifting conditions with little knowledge. A real-time scheduler based on A3C is offered for stochastic ECC systems that offer concurrent decentralized learning across several agents. In order to make scheduling decisions more efficient, the R2N2 architecture is introduced. In addition to temporal patterns, it might gather additional host and task data. The provided framework can be altered to fit the requirements of the application and change a number of hyper-parameters. Deep learning was introduced to the frontier computing environment by Kim *et al.* [23] as a tool for the IoT. A novel offloading technique is suggested to increase the effectiveness of edge computing-based deep learning IoT applications because of the present edge nodes' insufficient processing power. The performance evaluation uses an offloading technique in an edge computing environment to finish a number of deep-learning tasks. Evaluation findings show that the proposed method outperforms other IoT optimization strategies. Qin *et al.* [24] introduced a deep learning technique dubbed LASER for the random execution and replication of time-sensitive tasks. Machine learning has provided a practical solution to a number of classification and prediction problems. The deep neural network (DNN) may offer more accurate regression (prediction) than traditional machine learning methods since it consists of multiple layers of hidden units positioned between the input and output layers.

The quantitative analysis, based speculative CV technique is called LASER with SRQuant. By reducing the overall (virtual) task duration on the device, they seek to lower the cost of speculative execution while raising probability of completion before deadlines (PoCD), or the likelihood that MapReduce jobs will be completed on schedule. It is best to evaluate and compare the two strategies using traditional experimentation. Daneshfar *et al.* [25] presented a novel DRLbased resource allocation and task scheduling system to lower energy costs for largescale cloud service providers (CSPs). These CSPs handle a tonne of user requests every day and have a huge number of servers. When training in changing environments, such as consumer demand trends and actual

The two-stage RP-TS deep Q learning processor aims to automatically make long-term decisions at the optimal power cost limit. First, Vaquero and Rodero-Merino [26] use the Gaussian process regression approach to forecast resource utilization in the future. The optimal number of real hosts for each monitoring window is then ascertained by applying the convex optimization technique. This amount of interest is calculated to ensure that only a few systems remain capable of providing a satisfactory service. After that, a similar migration directive is sent out to shut down the idle physical servers and pile up virtual machines (VMs) to achieve the energy savings goal.

3. MODELING THE SYSTEM AND FORMULATING THE PROBLEM

It is assumed in this work that the underlying architecture consists of all the peripheral and cloud nodes. The broad strokes of Figure 1 represent the system model. The network hierarchy includes a variety of resources, ranging from edges to several hops. The environment of edge clouds is formed by distant clouds. Numerous application operations are hosted on computers. The difference in response times and processing power between these servers is substantial. Although edge devices provide significantly faster reaction times since they are located closer to the customers, resource limitations limit their computing capacity. On the other hand, when cloud resources VMs are located several hops away from users, their reaction time increases significantly. On the other hand, cloud nodes can perform multiple jobs simultaneously since they have greater processing capacity and more resources.

Infrastructure is scheduled, migrated, and monitored by the RMS. The RMS receives jobs from IoT devices and users together with their SLA and QoS specifications. It regularly decides whether to schedule new operations under the optimization goals or transfer ongoing work to new hosts. The development's anticipated completion dates or deadlines, CPU, RAM, bandwidth, and storage all affect the RMS choice. The stochastic task generator workload generation module (WGM) is used to create tasks to reproduce this effect.

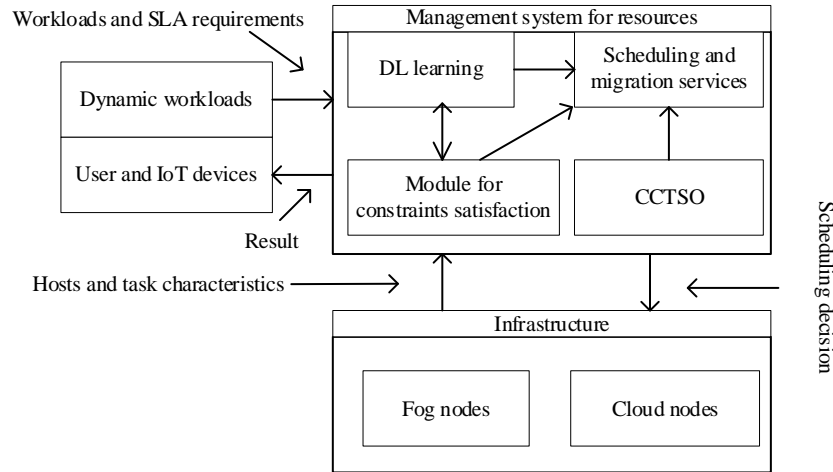


Figure 1. System model

The deep reinforcement learning module (DRLM) provides places for every activity on the server by interacting with the scheduling and migration services. Use a variety of different schedulers in DRLM, each with different duties and node partitions. One node or several edge cloud nodes can be subject to these schedulers. Previous research has shown that the computational load can be distributed over multiple servers, allowing multiple agents to learn changes in parameters simultaneously and facilitating faster learning within the constraints of different devices.

It is believed that all cloud and edge nodes will synchronize their schedulers with regional gradients so that each host model can be updated independently. Each planner has a unique instance of the global neural network thanks to a policy learning model in the DRLM, allowing for asynchronous updates. The constraint satisfaction module (CSM), another essential part of RMS, evaluates constraints such as whether a job is being relocated or if the destination server has enough capacity. even to judge whether the DRLM concept is appropriate.

The workload for each task varies, and they are all generated at random. A division of the performance time into equal-length periodic periods is done in other studies [8]. The intended intervals are numbered by the historical order of events, as shown in Figure 2.

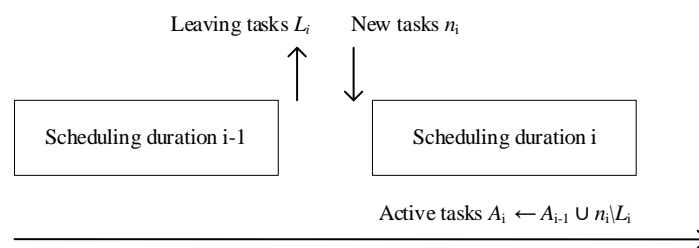


Figure 2. Work load model for dynamic tasks

Let the duration of the i^{th} scheduling interval be denoted as s_{i_1} , which starts at time T_i and ends at time T_{i+1}^{th} . Each interval s_{i_1} is associated with a set of active tasks, denoted as A_i .

At the beginning of interval s_{i_1} :

L_i : the set of tasks that were completed (and thus removed from the system).

n_i : the set of newly assigned tasks.

A_{i-1} : the set of active tasks carried over from the previous interval.

Hence, the set of active tasks at the beginning of interval s_{i_1} is:

$$A_i = (A_{i-1} \cap n_i) / L_i$$

where tasks completed in the previous interval (L_i) are removed, and newly assigned tasks (n_i) are added.

The performance of the scheduler during each interval is evaluated using a loss function, denoted as $loss_i$. A lower value of $loss_i$ indicates better scheduling performance. Additionally, h_i represents the i -th host in the set of available hosts.

Formulation of the problem: the measurement known as the scheduler's efficacy is gauged by the loss given for each scheduling period. Better scheduling has been provided by a lower loss value. State the time interval. s_i as $loss_i$ in a variety of hosts, h is recognized as the i^{th} host. The edge cloud environment refers to the collection of hosts and its enumeration is $[h_0, h_1, \dots, h_n]$. Assume that throughout execution, there are n hosts in total. To the host assigned to a task, append the symbol T . A scheduling tool can be defined as a mapping from a state of the system to an action, such as selecting a host for new assignments or determining whether to move ongoing work. The system's condition at the start of s_i indicated as S_i . This includes the hosts' parameter settings, the remaining open jobs from the prior interval, and $(A_{i-1} \setminus L_i)$ and new assignments (n_i). The hosts that schedulers assign or move must be selected, which indicates A_i for s_i regarding every task in $A_i (=A_{i-1} \cup n_i \setminus L_i)$. Allow the migratory duties to be $M_i \subseteq A_{i-1} \cup L_i$. Consequently, $Action_i = \{h \in \text{host tasks } t \mid t \in M_i \cup n_i\}$, which is a choice about migration for actions in M_i and distribution for tasks in n_i . Consequently, the function scheduler assigned A model that implies $S-i$ entails $A-i$. Using the entire number of servers in the edge-cloud data center as input, the model calculates the time-dependent work allocation among hosts. Hence, the problem for a perfect model can be described as (1).

$$\sum_i loss_i \text{ belongs to } \forall i, A_i = \text{model}(S_i) \forall i \forall t \in M_i \cup n_i, \{t\} \leftarrow A_i(t) \quad (1)$$

4. MODEL FOR REINFORCEMENT LEARNING

Strengthening since learning models are appropriate for policy gradient learning; they are introduced to address the problems outlined in section 3. Specification for input: S_i , it is the input for the scheduling model and is composed of host characteristics such as CPU, RAM, bandwidth, disk capacity, and usage [16]. Features such as the host's million instructions per second (MIPS), response time, cost per unit time, power characteristics, cost per hour, cost per minute, and total number of assigned jobs are all encompassed. Different hosts would have different input/output (I/O), memory/RAM, and computational (CPU) capabilities. Due to the memory, processing, and input/output limitations of tasks in an edge-cloud setting, these characteristics are crucial for scheduling choices. Another way to ensure low energy use is to distribute several jobs among a small group of active hosts while masking the inactive hosts. Completing I/O-intensive operations and preventing SLA violations may be possible with faster disk read/write rates on the host. Within the feature vector known as fv_i^{hosts} , every host has a specific set of these settings. The tasks in a_i are separated into n_i and $a_{i-1} \setminus L_i$, two different teams. Included in the first set of characteristics are the task's CPU, RAM, bandwidth, and storage requirements.

4.1. Specifications for output

Assigning hosts for tasks in the proposed model depends on the input State _{i} , a_i at the start of the duration S_{L_i} and outcomes referenced action A_i comprise host assignments for newly created activities. n_i Migration choices for ongoing tasks from earlier periods belong to $a_{i-1} \setminus L_i$. Every task that is moved needs to be transferable to the new server. ASM_i which belongs to A_i based on the viability standards. Furthermore, anytime a host H tasked with a certain assignment shouldn't accumulate too much after being assigned that is H is suitable for T . Consequently,

$$A_i(T) = \{H \in \text{HOSTS} \mid T \in n_i H_{\text{new}} \in \text{HOSTS} \mid T \in M_i \text{ if } T \text{ is to be changed}\} \quad (2)$$

It is possible to train neural networks to prioritize tasks for hosts. This means that rather than providing a list of hosts for each task, the model provides an ordered list of hosts. In addition, actions that are not under control will have repercussions. Specifically, this encompasses two aspects of penalization: the host allocation penalty is the sum of all hosts assigned a higher priority but ended up failing to finish a job and were added to each task, while the migration penalty is the percentage of tasks that the simulation attempted to migrate but failed to do so.

5. PROPOSED WORK

AEC, ART, AMT, and SLAV are among the metrics that the CCTSO is intended to lower. Many hyper-parameter values have been tuned using the CCTSO algorithm because various users have distinct

wants and application configurations. R2N2 forecasts future scheduling decisions based on these requirements and the resource monitoring service server's features. While A3C adaptation is known to react swiftly to changing conditions with fewer data points, R2N2 updates model parameters quickly. An A3C-based real-time scheduler is proposed to enable several agents to learn concurrently and decentralized in the stochastic edge-cloud scenario.

AEC: all edge and cloud servers' total power consumption, normalized to the environment's maximum carrying capacity for a specific duration, is referred to as "AEC." To be sure, the major power source for the cloud and the energy-saving devices for the edge may affect the energy sources used by the cloud and periphery nodes. Include a factor $\alpha_H \in [0,1]$. Depending on user needs and deployment strategy, it is possible to modify the amount of energy needed by a host for peripheral and cloud nodes. Here's how the power gets normalized:

$$AEC = \frac{\sum_{H \in \text{HOSTS}} \alpha_H \int_{T=T_1}^{T=T_1+1} P_H(t) dt}{\sum_{H \in \text{HOSTS}} \alpha_H P_H^{\text{MAX}} (T_{I+1} - T_I)} \quad (3)$$

Wherein the power function of host H about time is represented by $P_H(t)$ whereas the greatest power of H is denoted by P_H^{MAX} .

ART: the maximum response time to date, adjusted for the average reaction time for the entire task. L_{I+1} during S_{I_i} is the normal reaction time. The task execution time multiplied by the task response time is the task response time, the anticipated time of host reaction. ART is explained,

$$ART_I = \frac{\sum_{T \in L_{I+1}} RT(t)}{|L_{I+1}| RT(t)} \quad (4)$$

Where RT is the response time

AMT: during a period, the AMT of all active tasks $a_i S_{L_i}$ each job's average migration time is normalized by the longest possible migration time during the preceding period. The meaning of AMT is explained.

$$AMT = \frac{\sum_{T \in a_i} MT(t)}{|a_i| RT(t)} \quad (5)$$

Where MT is the migration time.

SLAV: the average number of SLA violations for leaving a work ($li+1$) over an interval SI_i is known as the average SLA violations (SLAV). Task T's SLA (t) is the sum of two measures, such as; i) performance degradation due to migrations and ii) time overruns per active host. Consequently,

$$MT = \frac{\sum_{T \in L_{I+1}} SLA(t)}{|L_{I+1}|} \quad (6)$$

The scientific term for the tuna, or oceanic predatory fish, is Tunnini. Based on RMS, it has been applied to decrease host characteristics. The first strategy is spiral foraging. To reduce host characteristics and migrate to shallower waters where they might be more easily targeted by RMS, tuna swim in a spiral pattern while feeding. The following strategy is known as parabolic foraging. Each tuna envelops the final host by forming a parabolic curve as they follow it.

Starting over: to start the optimization process, CCTSO uniformly generates initial host populations at random in the search space.

$$X_I^{\text{INT}} = \text{Rand.} (u_B - L_B) + L_B, I = 1, 2, \dots, NP \quad (7)$$

Where NP is a tally of tuna communities according to host attributes, X_I^{INT} is the i^{th} individual baseline, u_B and L_B is the upper and lower search space for boundaries. Furthermore, the vector "Rand" has values between 0 and 1, exhibiting a uniform distribution.

Using a spiral: to facilitate the exchange of host-specific information among nearby tuna, tuna have developed a behavior in which they actively pursue earlier individuals. Below is the mathematical formula for the spiral foraging technique.

$$X_I^{T+1} = \{\alpha_1 \cdot (X_{\text{BEST}}^T + \beta \cdot |X_{\text{BEST}}^T - X_I^T|) + \alpha_2 \cdot X_I^T, I = 1 \alpha_1 \cdot (X_{\text{BEST}}^T + \beta \cdot |X_{\text{BEST}}^T - X_I^T|) + \alpha_2 \cdot X_{I-1}^T, I = 1, 2, 3, \dots, NP \quad (8)$$

$$\alpha_1 = A + (1 - A) \cdot \frac{T}{T_{MAX}} \quad (9)$$

$$\alpha_2 = (1 - A) - (1 - A) \cdot \frac{T}{T_{MAX}} \quad (10)$$

$$\beta = e^{bL} \cos \cos(2\pi b) \quad (11)$$

$$L = e^{3 \cos \cos \left(\left(\left(T_{MAX} + \frac{1}{T} \right) - 1 \right) \pi \right)} \quad (12)$$

Where X_l^{T+1} is the l th iteration of the host characteristics, X_{BEST}^T is the best host characteristic for an individual, α_1 and α_2 is a constant that is used to determine the tuna's adherence to the most beneficial asset and the previous resource during the first phase; and T is the iteration number at this stage. Mass coefficients are what regulate the host features' propensity to migrate concerning the best resources and the preceding resource.

6. RESULTS

In this section, we compare the model with many conventional methods in the business to give you a full picture of the outcomes. Further on the experimental setting, evaluation measures, and data collection methods are provided. By utilizing cloud sim, one may activate features like power, cost, and reaction time for edge nodes. The constraint fulfillment module has its own dedicated software and input/output preprocessing. Building the loss function requires the usage of the cloud sim performance monitoring and storage service. In the simulation setting, hosts are used to distribute tasks to VMs, which are called cloudlets. Jobs are sent from the cloud to the virtualized servers via the VM. When you create a new cloudlet, assign it to a VM. When the cloudlet is complete, delete the VM. Take into account the present setup of tasks in the edge-cloud context as a shift from cloudlets to VMs.

The open-source bitbrain real-world data gathering is used to build the cloudlet dynamic workload. Datasets maintained by bitbrain reveal metrics regarding the consumption of infrastructure resources by critical business activities. In order to generate reliable input feature vectors for machine learning models, over 1,000 VM workload records from two different types of machines were selected. These logs accurately reflect real-world infrastructure utilization patterns.

The workload data includes information on cores per CPU, multithreading instruction per second, random access memory (RAM), and network and storage bandwidth, with each time stamp in the dataset spaced five minutes apart. Dissect the data gathering into two parts, assigning a VM burden of 27.00% and 77.00%, respectively. The larger portion is utilized for training the R2N2 network, while the smaller portion is for testing, sensitivity analysis, and cross-referencing with other relevant programs. Costs associated with the cloud layer are based on Microsoft Windows Infrastructure as a service.

An average job completion time is calculated by adding the most recent scheduling period's average scheduling time, task execution time, and server response time. Task completion rate, percentage of tasks completed within the targeted MIPS's expected execution time, number of task migrations performed per period, and total time transmitted during migration are all metrics that are measured.

Methods for evaluating results: a number of heuristics have been applied to the problem of dynamic planning. In order to address the subproblems of task/VM selection and server overload detection, three of the most successful solutions were selected. By utilizing the best fit decreasing (BFD) algorithm, each of these variants identifies the target host. Also, compare and contrast the results with two well-known RL methods that are commonly used in the literature:

- LR-MMT: the local regression (LR) and minimum migration time (MMT) techniques are used in this dynamic workload scheduling technique to identify workloads and indicate overloading.
- MAD-MC: loads jobs dynamically using the maximum correlation policy (MC) and median absolute deviation (MAD) heuristics for task selection and overload detection, respectively.
- DDQN: the deep Q-learning method for RL has been used in several academic publications.
- Reinforce: fully integrated neural network that makes use of the reinforce technology based on policy gradients.

The suggested A3C-CCTSO-R2N2 scheduling strategy, in comparison to the other scheduling policies, has the shortest average response time, as shown in Figure 3. The suggested model divides work based on RMS (CCTSO) and asks directly if a given node is a cloud or perimeter node, eliminating the need for multiple migrations and integrating AMT into the loss function. This shows that the suggested system has

a lower ART of 6.92 ms compared to earlier approaches like LR-MMT, MAD-MC, DDQN, REINFORCE, and A3C-R2N2, which have increasing ART of 9.40 ms, 8.95 ms, 8.76 ms, 8.24 ms, and 7.48 ms accordingly (refer to Table 1).

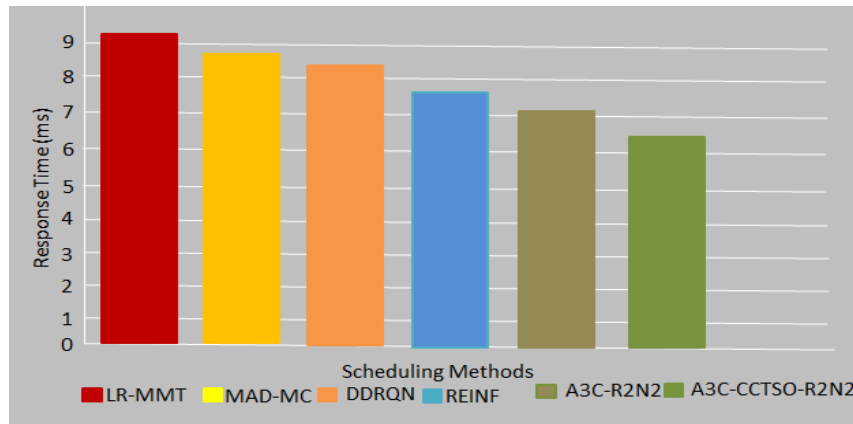


Figure 3. Average response time with different scheduling methods

Table 1. Average response time with diverse scheduling methods

Methods for scheduling	Response time (ms)
LR-MMT	9.40
MAD-MC	8.95
DDQN	8.76
REINFORCE	8.24
A3C-R2N2	7.48
A3C-CCTSO-R2N2	6.94

Figure 4 shows that compared to the A3C-R2N2 policy, the A3C-CCTSO-R2N2 model has fewer service level agreement (SLA) breaches. Once again, this is because of less migrations and smarter work scheduling to prevent the huge value loss that happens when SLAs are violated. See Table 2 for a comparison of the proposed system's ART with that of alternative methods; LR-MMT, MAD-MC, DDQN, REINFORCE, and A3C-R2N2 all produce higher ART values of 0.095, 0.086, 0.074, 0.066, and 0.054 respectively.

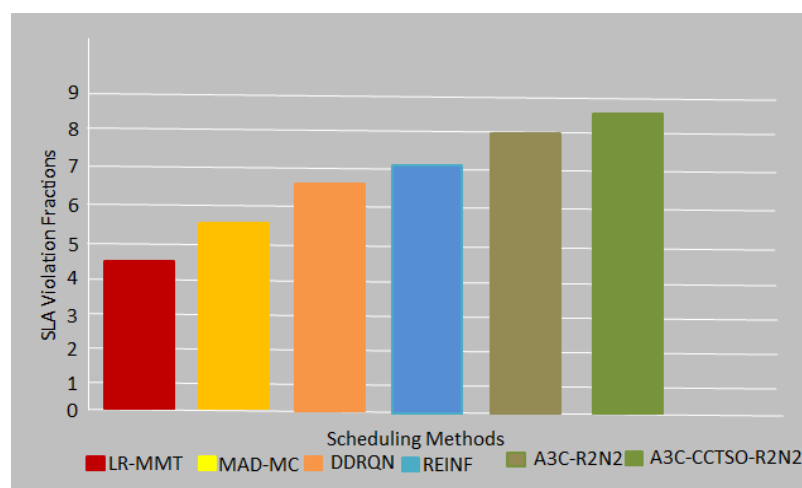


Figure 4. SLV violation fraction with diverse scheduling methods

Table 2. SLA violation fractions with different scheduling methods

Methods for scheduling	SLA violation fractions
LR-MMT	0.095
MAD-MC	0.086
DDQN	0.074
REINFORCE	0.066
A3C-R2N2	0.054
A3C-CCTSO-R2N2	0.042

The higher proportion of tasks completed by the A3C-CCTSO-R2N2 model is based on its capacity to ensure that work can be distributed to as few cloud VMs as possible in order to minimize expense (Figure 5). The proposed system finished 1152 tasks, compared to 792, 824, 897, 980, and 1129 tasks for the LR-MMT, MAD-MC, DDQN, REINFORCE, and A3C-R2N2 methods, respectively. Furthermore, the quantity of activities finished by each methodology is displayed in Table 3.

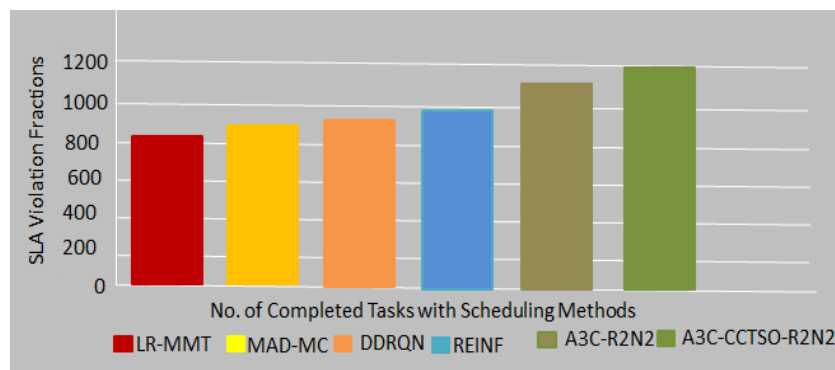


Figure 5. SLA violations with scheduling methods tasks completed numbers

Table 3. SLA violation fractions with different scheduling methods

Methods for scheduling	SLA violation fractions
LR-MMT	792
MAD-MC	824
DDQN	897
REINFORCE	980
A3C-R2N2	1129
A3C-CCTSO-R2N2	1152

7. CONCLUSION

Improving response times and service quality with edge and cloud resources in unpredictable settings with changing workloads is tough. In this paper, we offer real-time, end-to-end task scheduling for cloud and hybrid devices. There will be a decrease in metrics such as AEC, ART, AMT, SLAV, and CCTSO. The CCTSO algorithm has been used to fine-tune a large number of hyper-parameter values according to application variables and user needs. Tuna learn their hosts' characteristics via a combination of spiral and parabolic foraging. A3C is an innovative policy-gradient-based RL method for SDS. A scheduling method based on A3C-R2N2 that takes critical host and job characteristics into account enhances performance. An improved version of the iFogSim Toolkit called CloudSim was used in the experiment. Compared to previous approaches such as LR-MMT, MAD-MC, DDQN, REINFORCE, and A3C-R2N2, the proposed system has fewer SLA breaches (0.042) and greater ART (0.095, 0.086, 0.074, 0.066, and 0.054 respectively). In contrast to previous methods such as LR-MMT, MAD-MC, DDQN, REINFORCE, and A3C-R2N2, which had ART increases of 9.40 ms, 8.95 ms, 8.76 ms, 8.24 ms, and 7.48 ms, respectively, this research demonstrates a lower ART of 6.92 ms. It is possible that future RL methods will integrate A3C with R2N2 3D reconstruction and CCTSO optimization. Autonomous robots and virtual systems might benefit from agents learning in 3D-interactive virtual environments.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Gangadharan	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	
Manonmani														
Krishnasamy		✓				✓		✓	✓	✓	✓	✓		
Ponmozhi														
Krishnasamy	✓		✓	✓			✓			✓	✓		✓	✓
Balasubramanian														

C : **C**onceptualization

M : **M**ethodology

So : **S**oftware

Va : **V**alidation

Fo : **F**ormal analysis

I : **I**nterpretation

R : **R**esources

D : **D**ata Curation

O : **O**riginal Draft

E : **E**valuation

Vi : **V**isualization

Su : **S**upervision

P : **P**roject administration

Fu : **F**unding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.




REFERENCES

- [1] M. Aazam, M. St-Hilaire, C. -H. Lung, and I. Lambadaris, "MeFoRE: QoE based resource estimation at Fog to enhance QoS in IoT," *2016 23rd International Conference on Telecommunications (ICT)*, Thessaloniki, Greece, 2016, pp. 1-5, doi: 10.1109/ICT.2016.7500362.
- [2] T. Achterberg, "SCIP: Solving constraint integer programs," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 1–41, 2009, doi:10.1007/s12532-008-0001-1.
- [3] A. Anand and G. de Veciana, "Measurement-based scheduler for multi-class QoE optimization in wireless networks," *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, Atlanta, USA, 2017, pp. 1-9, doi: 10.1109/INFOCOM.2017.8057014.
- [4] A. Ansari and A. A. Bakar, "A Comparative Study of Three Artificial Intelligence Techniques: Genetic Algorithm, Neural Network, and Fuzzy Logic, on Scheduling Problem," *2014 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology*, Kota Kinabalu, Malaysia, 2014, pp. 31-36, doi: 10.1109/ICAET.2014.15.
- [5] P. Bellavista and A. Zanni, "Feasibility of Fog computing deployment based on docker containerization over Raspberry Pi", in *Proceedings of the 18th International Conference on Distributed Computing and Networking, ACM*, no. 16, pp. 1-10, 2017, doi:10.1145/3007748.3007777.
- [6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *MCC '12: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, 2012, pp. 13–16, doi: 10.1145/2342509.2342513.
- [7] A. Brogi and S. Forti, "QoS-Aware Deployment of IoT Applications Through the Fog," in *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185-1192, Oct. 2017, doi: 10.1109/JIOT.2017.2701408.
- [8] T. Wang *et al.*, "Data collection from WSNS to the cloud based on mobile fog elements," *Future Generation Computer Systems*, vol. 105, pp. 864-872, Apr. 2020, doi: 10.1016/j.future.2017.07.031.
- [9] T. Wang, G. Zhang, A. Liu, M. Z. A. Bhuiyan, and Q. Jin, "A Secure IoT Service Architecture with an Efficient Balance Dynamics Based on Cloud and Edge Computing," in *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4831-4843, Jun. 2019, doi: 10.1109/JIOT.2018.2870288.
- [10] T. Wang, Z. Peng, S. Wen, G. Wang, B. Wang, and A. Liu, "A survey of fog computing in wireless sensor networks: Concepts, applications, and issues," *Ad Hoc & Sensor Wireless Networks*, vol. 44, no. 1-4, pp. 109–130, 2019.
- [11] E. Lopez-Falcon *et al.*, "Adaptive encrypted cloud storage model," *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, Moscow and St. Petersburg, Russia, 2018, pp. 329-334, doi: 10.1109/EIConRus.2018.8317099.
- [12] Z. Xia, X. Wang, L. Zhang, Z. Qin, X. Sun, and K. Ren, "A Privacy-Preserving and Copy-Deterrence Content-Based Image Retrieval Scheme in Cloud Computing," in *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2594-2608, Nov. 2016, doi: 10.1109/TIFS.2016.2590944.
- [13] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the Edge: A Scalable IoT Architecture Based on Transparent Computing," in *IEEE Network*, vol. 31, no. 5, pp. 96-105, 2017, doi: 10.1109/MNET.2017.1700030.
- [14] Z. Cai, X. Zheng, and J. Yu, "A Differential-Private Framework for Urban Traffic Flows Estimation via Taxi Companies," in *IEEE Transactions on Industrial Informatics*, vol. 15, no. 12, pp. 6492-6499, Dec. 2019, doi: 10.1109/TII.2019.2911697.
- [15] Z. Cai and Z. He, "Trading Private Range Counting over Big IoT Data," *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, Dallas, TX, USA, 2019, pp. 144-153, doi: 10.1109/ICDCS.2019.00023.




- [16] J. W. Shang, C. K. Wang, C. P. Wang, G. Y. Guo, and J. Qian, "An attribute-based community search method with graph refining," *The Journal of Supercomputing*, vol. 76, pp. 7777–7804, 2020, doi: 10.1007/s11227-017-1976-z.
- [17] X. Zheng and Z. Cai, "Privacy-Preserved Data Sharing Towards Multiple Parties in Industrial IoTs," in *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 968–979, May 2020, doi: 10.1109/JSAC.2020.2980802.
- [18] F. Firouzi *et al.*, "Internet-of-things and big data for smarter healthcare: From device to architecture, applications, and analytics," *Future Generation Computer Systems*, vol. 78, pp. 583–586, 2018, doi: 10.1016/j.future.2017.09.016.
- [19] C. Wang, C. Wang, Z. Wang, X. Ye, J. X. Yu, and B. Wang, "Deep Direct: Learning Directions of Social Ties with Edge-Based Network Embedding," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2277–2291, Dec. 2019, doi: 10.1109/TKDE.2018.2877748.
- [20] B. Zhou *et al.*, "Online Internet traffic monitoring system using spark streaming," in *Big Data Mining and Analytics*, vol. 1, no. 1, pp. 47–56, March 2018, doi: 10.26599/BDMA.2018.9020005.
- [21] X. Zheng, Z. Cai, J. Yu, C. Wang, and Y. Li, "Follow But No Track: Privacy Preserved Profile Publishing in Cyber-Physical Social Systems," in *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1868–1878, Dec. 2017, doi: 10.1109/JIOT.2017.2679483.
- [22] D. Kim, E. Ko, J. Son, Y. Kim, and J. Seo, "A Lightweight and Transparent Compensation Mechanism for Fog-Cloud Storage Framework," *2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService)*, Bamberg, Germany, 2018, pp. 254–259, doi: 10.1109/BigDataService.2018.00045.
- [23] Y. Kim, D. Kim, J. Son, W. Wang, and Y. Noh, "A New Fog-Cloud Storage Framework with Transparency and Auditability," *2018 IEEE International Conference on Communications (ICC)*, Kansas City, USA, 2018, pp. 1–7, doi: 10.1109/ICC.2018.8422479.
- [24] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A Software Defined Networking architecture for the Internet-of-Things," *2014 IEEE Network Operations and Management Symposium (NOMS)*, Krakow, Poland, 2014, pp. 1–9, doi: 10.1109/NOMS.2014.6838365.
- [25] N. Daneshfar, N. Pappas, and V. Angelakis, "Poster abstract: Resource allocation with service availability & QoS constraints in mobile fog networks," *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Atlanta, GA, USA, 2017, pp. 1018–1019, doi: 10.1109/INFCOMW.2017.8116539.
- [26] L. M. Vaquero and L. Roderio-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014, doi: 10.1145/2677046.2677052.

BIOGRAPHIES OF AUTHORS






Gangadharan Manonmani    received her B.Sc. degree in Physics from Shrimati Indira Gandhi College, Trichy, MCA from The Standard Fireworks Rajaratnam College for Women, Sivakasi, and M.Phil. from Periyar University, Salem. She is working as an Assistant Professor in the Department of Information Technology, Hajee Karutha Rowther Howdia College, Uthamapalayam-625533, Tamil Nadu, India. Her areas of interest include neural networks, fog computing, and the application of blockchain. She can be contacted at email: gmanonmani24@gmail.com.



Krishnasamy Ponmozhi    received her B.Sc. degree in Computer Science from Fatima College, Madurai, and MCA from Seethalakshmi Ramaswami College, Trichy. She was awarded a Doctorate from Mother Teresa Women's University, Kodaikanal. She Published 1 Book, 5 Book Chapters, 10 International Conference and 30 Journals. Her area of interest includes image processing, networking, and natural language processing. She can be contacted at email: chezhiyan71@gmail.com.



Krishnasamy Balasubramanian    received his under graduate degree from Madurai Kamaraj University and post graduate degree from Bharathidasan University, Trichy. He was completed Ph.D. at Gandhigram Rural Institute–Deemed to be University, India. He is currently working as Professor and Head at Kalasalingam Academy of Research and Education – Deemed to be University, Krishnankoil. He has more than 25 years of experience to his credit in teaching and research. His areas of interest include image processing, artificial intelligence and data science. He can be contacted at email: ksbsala75@gmail.com.