The International Journal of Interdisciplinary Organizational Studies
ISSN: 2324-7649 (Print), ISSN: 2324-7657 (Online)
Volume 19, Issue 1, January-June, 2024

CLUSTERING BASED VM MIGRATION AND BIDIRECTIONAL RECURRENT NEURAL NETWORK (BIRNN) FOR EDGE-CLOUD COMPUTING ENVIRONMENTS

[1*]S. Supriya, [2]Dr. K. Dhanalakshmi
[1*]Research Scholar, Department of Computer Science, Kongunadu Arts and Science College Coimbatore – 641029, Email: supriyasundaram@gmail.com
[2]Head of the Department, Department of Information Technology, Kongunadu Arts and Science College, Coimbatore – 641029, Email: kdhanalakshmimca@gmail.com

ABSTRACT: Utilization of Internet of Things (IoT) gadgets and the information generated by these gadgets has grown significantly. It utilizes cloud and mobile-edge resources with ease. Because of the limited resource capacities, IoT flexibility aspects, resource diversity, network ranking, and problematic nature, it is difficult to schedule application jobs effectively in these kinds of contexts. The issue cannot be effectively solved by current heuristics and RL (Reinforcement Learning)-methods since they are not generalizable or quickly adaptable. This research introduces an innovative way to effectively utilize multi-layer resources in stochastic situations through dynamic task scheduling in the Edge-Cloud systems. Firstly, Fuzzy Election Based Optimization Algorithm (FEBOA) has been introduced for minimizing the metrics such as ART (Average Response Time), Average Energy Consumption (AEC), SLAV (Service Level Agreements Violations), and Average Migration Time (AMT). Secondly, Adaptive Threshold Fuzzy C-Means (ATFCM) clustering has been introduced for VM load detection. Features of host from RMS (Resource Monitoring Service) are utilized through Bidirectional Recurrent Neural Network (BiRNN) method to detect the next scheduling decisions. For forward and reverse input TS (Task Scheduling), BiRNN takes advantage of data to regulate connections. Each RNN cell's input (task) and output (host allocated to task) dependencies were defined for the BiRNN classifier in the forward as well as the reverse direction. Lastly, a scheduler using Asynchronous-Advantage-Actor-Critic (A3C) is shown, which is capable of taking into account all relevant task and host parameters in order to make task scheduling that would improve efficiency. Power usage, timing of responses, SLA, and operational charges are used to quantify the outcomes.

INDEX TERMS: (EC) Edge Computing, (CC) Cloud Computing, (TS) Task Scheduling, Bidirectional Recurrent Neural Network (BiRNN) and Fuzzy Election Based Optimization Algorithm (FEBOA).

1. INTRODUCTION

Over 50 billion, IoT gadgets would be linked to the internet in the upcoming years as an outcome of the recent explosive growth of the IT (Information Technology) industry [Rababah et al., 2020; Shekhar and Gokhale 2017]. Furthermore, the emergence of intricate and computationally demanding IoT applications which frequently create and process massive amounts of data, as it is facilitated by the accessibility of reliable, fast internet as well as communication tools. Because CC provides immediate access to a vast amount of computational resources for DA (Data

Analytics) and service processing, it has been recognized as a feasible and viable technology to enable such growth [Elgendy et al., 2018].

However, the massive volume of information created by IoT gadgets needs to be sent and analyzed in the moment, while CC resources are centralized and located away from IoT gadgets. For this reason, the CC model is unsuitable for apps requiring high (QoS) Quality of Service, immediate interactivity, and decreased latency because of delays in the network [Sahni etv al., 2017]. Complex computations can be delegated to more capable gadgets to relieve from these restrictions and satisfy the communication/processing latency requirements.

For this novel class of IoT uses, Edge-Cloud is a suitable computing model that offers a few delay reactions [Yousefpour et al., 2019], [Tuli et al., 2019], [Wang et al., 2019]. Here, in addition to the distant cloud, the network's edge has insufficient processing power to respond quickly to critical applications. It is vital to use Edge resources as efficiently as possible in order to support more apps and optimize their QoS at the same time. A scheduler that effectively handles applications and underlying resources must be used to accomplish this. The process of assigning resources (computations) to the specific user making a request for services is known as scheduling. To optimize resource utilization, scheduling a specific job at the edge or cloud might be a tough operation in the Edge Cloud. The scheduling challenge is further complicated by the problematic behavior of the Edge-Cloud background, that affects several aspects of the work, including the duration, arrival rate, and demand for resources.

In order to effectively employ the multi-layer resources in unpredictable situations and saving money and energy, also enhancing the QoS of apps, dynamic task scheduling was established. The main difficulties include balancing loads, accessibility, dependability, and adaptive resource allocation. Therefore, in the edge cloud background, SOTA (State-Of-The-Art) scheduling algorithm approaches are required at all times. An approach to scheduling that depends on RL offers hope for dynamic system optimization [Fox et al., 2019]. Diverse value-RL algorithms are analyzed in current studies and offered for optimizing various elements of RMS in distributed computing backgrounds [Xu et al., 2017], [Li et al., 2018]. However, prior study indicates that these value-RL algorithms perform poorly in Edge-Cloud deployments because they are unsuitable for highly chaotic situations [Mnih et al., 2016]. There aren't many works that can take advantage of policy gradient techniques [Mao et al., 2016], optimize for just one QoS parameter, and neglect asynchronous updating in order to remain more adaptable in extremely unpredictable conditions. Furthermore, no previous study has taken advantage of dynamic trends in task load, net, and node behaviors for improving the decisions of scheduling. Moreover, the diverse resources (e.g., varying computation resource capabilities) comprise the edgecloud backgrounds.

Additionally, the centralized scheduling policy is employed in this study remains inappropriate for decentralized or multilevel backgrounds. Rather than allocating additional properties, VMs in the overloaded to under loaded edge devices. To ensure effective computing and run in the edge servers, there may be less migration as possible among neighboring edge servers [Tao et al., 2019]. Because the majority of edge devices are mobile, there will be a boom in demand for shared resources in the edge servers, creating a dynamic edge computing backgrounds. By minimizing

resource contention, it is possible to utilize these shared resources without any conflicts. The scheduling issue in CC falls into the category of NP (Nondeterministic Polynomial) challenging issues, because of its enormous solution space, which makes finding the best solution as well as time-consuming. Identifying an undesirable answer immediately is the ideal case in a cloud background. It has been demonstrated that metaheuristic-based approaches can solve these kinds of issues almost perfectly in an appropriate period of time. The advantages of this algorithms were simplicity of idea, easy application, independence from particular issues, lack of requirement for the reasoning procedure, and effectiveness in solving challenging issues [Gonzalez et al., 2022]. In order to handle optimization issues and produce better quasi-optimal solutions, metaheuristic algorithms execute well when the 2 signs of exploration and exploitation are combined with the concepts of global and local search [Mejahed and Elshrkawey 2022]. The purpose of this research is to build the Election-Based Optimization Algorithm (EBOA), a new metaheuristic procedure that depends on popular movement and voting process model [Trojovský and Dehghani 2022].
The Edge-Cloud system framework, including all the necessary elements to provide resources with diversity and offloading TS. The improvement of VM migration has led to the ATFCM (Adaptive Threshold Fuzzy C-Means clustering). The presentation of the FEBOA aims to take the intricate dynamics of resource heterogeneity and workloads. Scheduling technique depends on Bidirectional Recurrent Neural Networks (BiRNNs) for taking use of time-based patterns in a hybrid Edge-Cloud configuration. The suggested model can swiftly adjust its allocation policy in response to host behavior, dynamic workloads, and QoS demands because of to the Policy Gradient. Furthermore, by utilizing the adaptive loss function, the suggested scheduling paradigm could be adjusted to maximize the required QoS metrics in accordance with the presentation requirements.

2. LITERATURE REVIEW

A unique DNCPSO (Directional and Non-local-Convergent PSO) was suggested by [Xie et al., 2020]. It uses non-linear inertia weight via selection and mutation processes with directional search procedure. This can significantly lessen the duration and expenses while obtaining an impacting outcome. The outcomes of simulation tests conducted on a variety of practical and randomized workflow conditions indicate that the DNCPSO is significantly efficient and effective to outperform existing standard and modernized algorithms.
A two-level scheduling optimization technique for an edge cloud background has been suggested by [Li et al., 2020]. The majority of tasks could be allocated to edge data centers under the AFS (Artificial Fish Swarm) TS approach, which is used for first-level scheduling. The TS to a centralized cloud data center if the edge data center is unable to finish due to insufficient resources. The task is then separated into tasks of similar size. Next, second-level scheduling is suggested, considering the node load balance. It suggests edge cloud task scheduling to shorten completion times, and also presents centralized cloud TS to lower overall costs. The experimental findings demonstrate that the suggested strategy outperforms the others by reducing the overall expenses and latencies as well as timing of completion.

A Deep Q-learning TS (DQTS) that integrates the benefits of the DNN (Deep Neural Network) and the Q-learning procedure was suggested by Tong et al. [Tong et al., 2020]. The purpose of this novel technique for resolving the risk of handling DAG (Directed Acyclic Graph) workloads in CC settings. This approach applies the commonly employed Deep Q-learning (DQL) technique to TS, where DQL serves as a major inspiration for fundamental model learning. Tests that compare the duration and load balanced variance in TS are carried out according to WorkflowSim advancements. Verification of the effectiveness of optimization and learning capabilities in DQTS is done through both simulation and real-world tests. DQTS offers benefits by the scalability, limitation, and learning capabilities.

Tuli et al. [Tuli et al., 2020] suggested Residual RNN (Recurrent Neural Network) (R2N2) for updating paradigm parameters quickly and A3C learning to swiftly adapt to dynamic circumstances with a smaller amount of information. Consequently, a real-time scheduler depends on A3C is presented for unpredictable Edge-Cloud systems that enable decentralized learning to occur simultaneously among several agents. In order to produce effective scheduling decisions, a multitude of cloudlets or tasks, factors and time-based designs are captured by the R2N2 framework. The recommended paradigm is adaptable and can alter many hyper-parameters suitable for the demands of the particular app.

A modified HHO (Harris Hawks optimization) technique was presented by [Zivkovic et al., 2021] to address the cloud-edge workflow scheduling problem. The Opposition-Based Learning (OBL) process used in the improved HHO methodology improves the original HHO method by addressing the cloud-edge workflow scheduling issue. Cost and makespan are the two basic goals that are pursued in simulations. The suggested tests made use of actual workflow models, and they assessed the suggested algorithm by contrasting it with alternative strategies that have been tried under similar test and simulation environments in the current literature. The suggested enhanced HHO method surpassed other cutting-edge techniques by lowering costs and duration metrics for performance, according to outcomes of executed tests.

To support DS (Dynamic Scheduling) in fog-CC, [Shruthi et al., 2022] created a Deep Q-Network (DQN) based on Mayfly Taylor Optimization based scheduling Algorithm (MTOA). By combining the ideas of the Taylor series and MA, MTOA, which improves convergence performance and it is used in this instance to carry out the DS procedure. Furthermore, DQN requires fewer resources as well as time, hence it is used for estimating power usage. Fitness criteria take into account computation costs, power consumption, and SLA violations. Rules are developed based on the specified SLA parameters, and they are matched, via angular distance, with the task or service that is being requested for SLA verification. Following the completion of the energy estimated, dynamic scheduling is carried out using the calculated MTOA.

A Deep RL structure for process scheduling has been suggested by [Jayanetti et al., 2022]. A new hierarchical action space is intended to facilitate the differentiation of edge nodes from cloud nodes. Proximal policy optimization methodology is added to the hybrid A3C-scheduling architecture to improve its ability to handle complicated workflow scheduling issues in edge-cloud scenarios. Utilizing the use of power, time required for execution, meets the deadline %, and % of

tasks finished as performance measures, the suggested framework's effectiveness was tested against different standard methods. Therefore, the outcomes show how effective the suggested method is in finding the best balance among the opposing objectives of reducing the implementation time and power consumption.

The migration cost resulting from scheme adjustments made throughout the modelling and optimization phase was taken into account by [Zhang et al., 2022]. The (NSGA-II) Non-Dominated Sorting Genetic Algorithm II-Seq2Seq algorithm is presented in order to develop a low-cost scheduling strategy. It uses the historical scheduling strategy to create an alternative scheduling strategy. WorkflowSim has been customized for the purpose of conducting tests. Through experimentation, it is discovered that the approach presented in this study may generate an adaptively FT (Fine-Tuning) scheduling system while also adapting to changes in task load. The population optimization process of NSGA is expedited in numerous successive scheduling tests, which drastically reduce the extent of time required to get the scheduling strategy.

A three-step scheduling methodology was presented by [Li et al., 2022] to integrate the container distribution in a cloud-edge background with the scheduling of container-based workflows. To allow for virtual CPU (vCPU) sharing between many containers, the initial step involves allocating vCPU to each container. Next, in an edge or cloud context, the containers are scheduled onto VMs via a two-step resource deployment process. Various goals are taken into account, such as reducing duration, load imbalance, and power usage, from the viewpoint of containerized workflows and cloud-edge resources. 3 evolution strategies: The Co-Evolution Strategy (CES), the Basic Non-Co-Evolution Strategy (B-NECS), and the Hybrid Non-Co-Evolution Strategy (H-NECS) are created and integrated with two multi-objective method framework to produce a group of non-dominated solutions. The suggested framework executes better than the existing 2-stage scheduling framework, and H-NECS beats other strategies, according to simulation data.

A hybrid approach was presented by [Xue et al., 2022] to address the resource scheduling problem with subtask dependency and parallelism. This research fully utilizes the capabilities of GA (Genetic Algorithm) and DQN to increase the algorithm's convergence speed. DQN generates the initial GA population. In order to assess the suggested algorithm's efficacy, this study chooses three actual scientific workflows for experimentation. According to the test outcomes, the hybrid method can improve the optimization effect immediately and converge soon.

3. SYSTEM MODEL AND PROBLEM FORMULATION

It is presumed that both edge and cloud nodes make up the fundamental infrastructure for this study. Figure 1 displays an outline of the structure model. Dispersed diverse properties in the network structure, spanning from the network edge to the multi-hop remote cloud, comprise the edge-cloud background. For various app cloudlets, the computing properties serves as hosts. Although the edge devices have limited computational capacity and are resource-constrained, they offer significantly faster reaction times because they are closer to the users. The RMS that includes scheduling, VM migration, and resource monitoring services is in

charge of the structure. The RMS makes a choice based on the projected completion timeframes or deadlines for CPU, RAM, bandwidth, and disk demands.

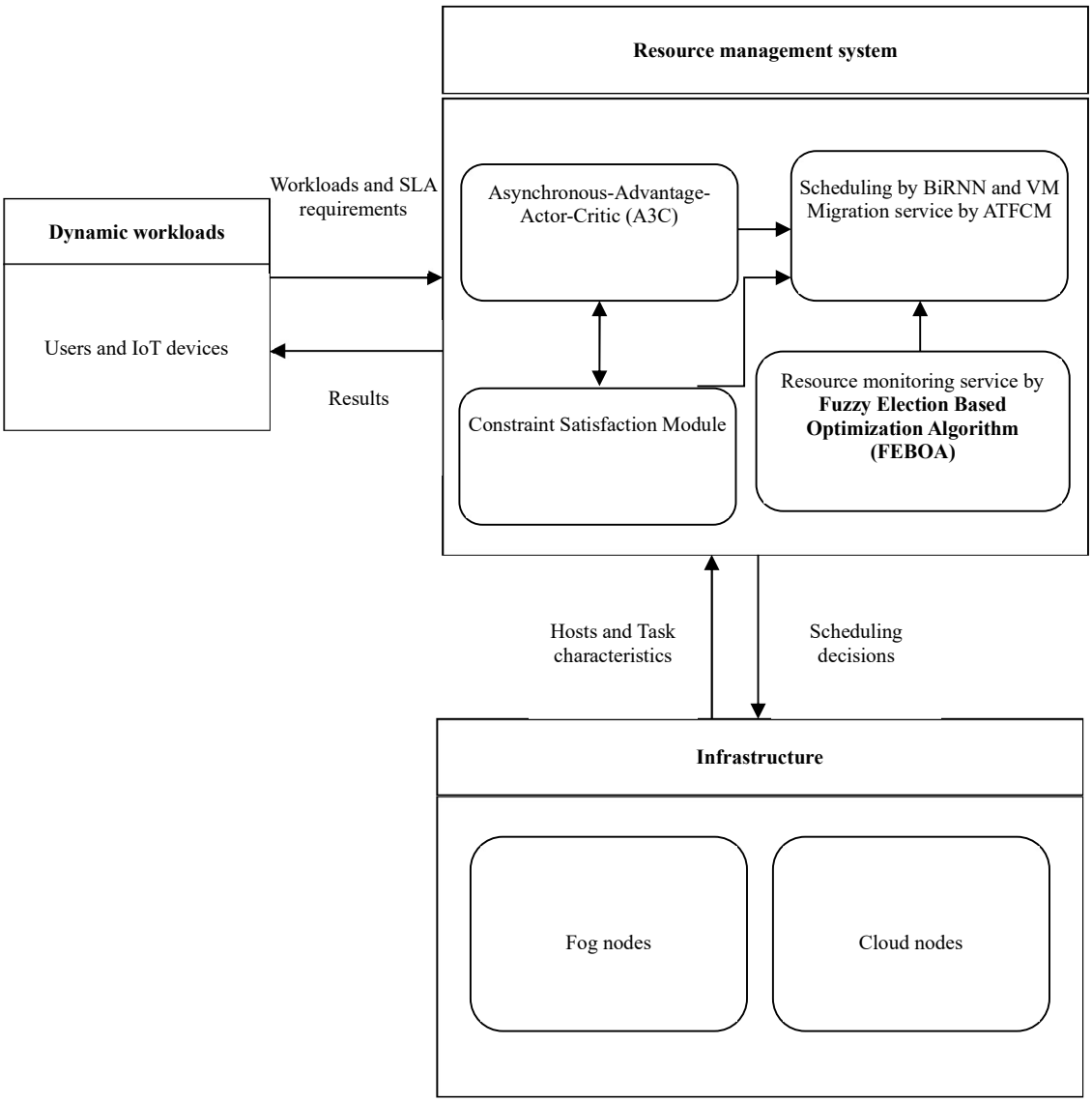

FIGURE 1. SYSTEM MODEL

Workload Model: Every cloudlet has an active cloudlet, and task generation is stochastic. Partition the time of processing into scheduling intervals of the same length, as has been done in previous research [Mnih et al., 2016]. As seen in Figure 2, the scheduling intervals are numbered according to the sequence in which they occur.

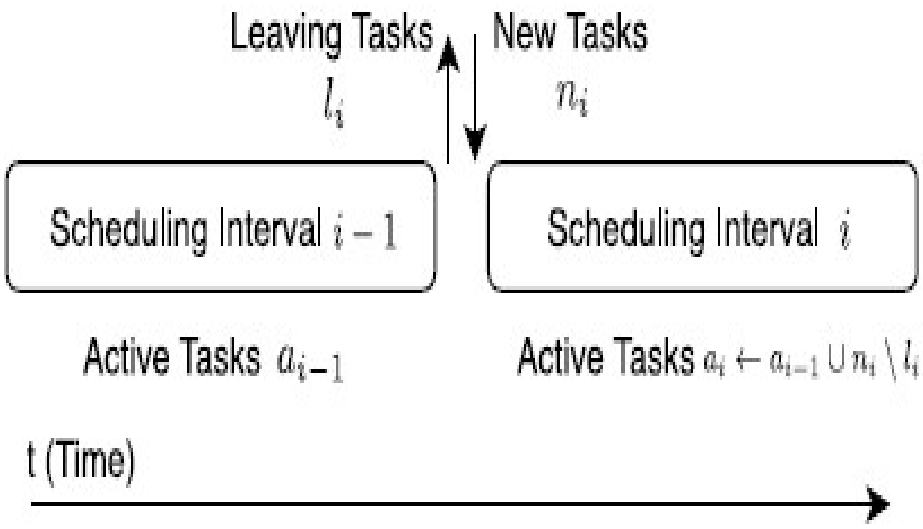

FIGURE 2. DYNAMIC TASK WORKLOAD MODEL

SI_i, the i[th] scheduling interval, is displayed. It initiates at time t_i and continues up to t_{i+1}, the jump to the succeeding interval. The activities that were being performed on the hosts and designated as a_i, are the active tasks in every SI_i. Additionally, at the initiation of SI_i, the collection of tasks that are finished is indicated by l_i, and the new tasks that the WGM sends are indicated by n_i. New tasks, n_i, have been added to the structure, while the tasks l_i are removed from it. As an outcome, the active tasks at the beginning of the interval SI_i are $a_i$is$a_{i-1} \cup n_i \setminus l_i$.

Problem Formulation: $Loss_i$ can be represented by the loss of the SI_i interval. In an enumeration of (H) hosts, H_i is labelled as the i[th] host. The set of H in an edge-cloud environment is referred to as Hosts, and they are enumerated as $[H_0, H_1, \ldots, H_n]$. Assign a hostname {T} to a task T as well. The system's initial state at the initiation of SI_i, known as State$_i$, is made up of new tasks (n_i), any active tasks from the previous interval that remains in evolution ($a_{i-1} \setminus l_i$), and the parameter values for the hosts.

Each task, $a_i (= a_{i-1} \cup n_i \setminus l_i)$, requires the scheduler to choose which host to allocate or migrate ; for SI_ithis decision is known as $Action_i$. The migratable tasks are represented by $m_i \subseteq a_{i-1} \cup l_i$. $Action_i$ that includes a decision of migration for tasks in m_i and tasks in n_i is the allocation decision, thus, $Action_i = \{h \in$ Hosts for task $T | T \in m_i \cup n_i\}$. Scheduler therefore indicates that Model is a function: $State_i \rightarrow Action_i$. The framework's assignment of tasks to hosts, or, $Action_i$ determines the $Loss_i$ of an interval, here, n is the amount of hosts in the Edge-Cloud Data Enter. Therefore, the issue can be stated as indicated by Equation (1) for a suitable framework.

$$\underset{Model}{\text{minimize}} \sum_{i} Loss_i \ \ subject\ to\ \forall i, Action_i = Model(State_i)\ \forall i\ \forall T \in m_i \bigcup n_i, \{T\} \leftarrow Action_i(T) \quad (1)$$

Describing the process for model updating following each scheduled intervals after the input-output requirements and the loss function is established.

4. REINFORCEMENT LEARNING MODEL

RL paradigm that works well with Policy Gradient Learning (PGL) and the issue description.

Input Specification: The $State_i$, which comprises of the host's characteristics such as RAM, disk space, CPU, and the bandwidth, is the input used by the scheduling model [Akbari et al., 2017]. In addition, it contains information about the host's power features, reaction time, rate for each unit time, and MIPS (Million Instructions Per Second). A tiny cluster of hosts with numerous tasks assigned to them could guarantee low consumption of power. In a FV (Feature Vector) known as FV_i^{Hosts}, each of these parameters have been assigned for each host. There are 2 discrete sets of tasks in n_i and $a_{i-1} \backslash l_i$.

Output Specification: Depending on the input $State_i$, the framework must assign a host to all task in a_i at the initiation of the interval SI_i. The result, also known as $Action_i$, is a migration choice for any active tasks from the preceding interval that remain $\in\ a_{i-1} \backslash l_i$ and a H allocation for all $task \in n_i$ new task. Every task that is moved must be able to be moved to the new host in order for the task to be considered feasible within the given limitations. Furthermore, if a host (h) is allocated to T, it must not turn into overloaded afterwards; in other words, h is appropriate for T. Consequently, $Action_i$via Equation (2) so that, for the interval SI_i, $\forall T \in n_i \cup m_i, \{T\} \leftarrow Action_i(T)$,

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

If, $t\ \forall\ t \in n_i\ \cup\ m_i$ then $Action_i$ is appropriate. The output of NN can be a vector representing each host's preferred allocation for each task. This indicates that the framework gives a ranked list of hosts as an alternative of designating one host for each task.

5. PROPOSED METHODOLOGY

To minimize metrics such as Average SLAV, ART, AMT, and AEC, the FEBOA was presented. Based on the load handled, data center hosts using ATFCM clustering are categorized into 4 categories: extremely low load, low load, medium load, and maximum load hosts. A model known as the BiRNN is presented for predicting the upcoming decisions for scheduling. BiRNN is utilized for updating model parameters quickly, and A3C learning is employed to promptly adjust to active circumstances having fewer information.

Average Energy Consumption (AEC): The infrastructure's energy consumption normalized by the environment's maximum power is known as AEC, and it applies to any interval. Enlarge the power utilized through a host, $h \in Hosts$ through a factor $\alpha_h \in [0,1]$, varies based on the operator's necessities and the cloud and edge nodes deployment strategy. In this way, the power is normalized:

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

The maximum power of h is represented by $P_h^{\max}$, and $P_h(t)$ is the host (h) of power function by time.

Average Response Time (ART): When calculating the ART for SI_i interval, one must take into account the ART to the current interval and normalize it by the ART for each of the leaving tasks (l_{i+1}). The general description of ART is as follows:

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

Average Migration Time (AMT): AMT for each of the active tasks (a_i) within an interval SI_i is determined by taking the AMT to the present intervals and normalizing it. The definition of AMT is as follows:

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

Average SLA Violations (SLAV): The average amount of SLAV for leaving the task (l_{i+1}) throughout an interval SI_i is known as average SLAV. Task T of SLA(t) is the outcome of 2 measures: (i) the number of times an active host violates the SLA and (ii) the amount of PD (Performance Degradation) brought by migrations. Consequently,

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

This approach adjusts its parameters according to restrictions along with reducing $Loss_i$.

5.1. Fuzzy Election Based Optimization Algorithm (FEBOA)

The procedure in which members of a resource monitoring group select a task from among requests is known as an election. Each and every person in that society, including those who did not vote for him, is impacted by the VM's election as leader. The ability to select and vote for the superior

candidate increases with VM awareness. FEBOA is a planned population-based metaheuristic algorithm. Every task assigned to the population in the FEBOA is a suggested fix for the resource monitoring. Equation (7) is used to describe the FEBOA population as a matrix known as the population matrix.

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} x_{1.1} \ldots x_{1,j} \ldots x_{1,m} \\ \vdots \quad \ddots \quad \vdots \quad \iddots \quad \vdots \\ x_{i.1} \cdots x_{i,j} \cdots x_{i,m} \\ \vdots \quad \iddots \quad \vdots \quad \ddots \quad \vdots \\ x_{N.1} \cdots x_{N,j} \cdots x_{N,m} \end{bmatrix}_{N \times m} \tag{7}$$

Here, N symbolizes the dimension of the EBOA population, m denotes the total amount of decision variables: AEC, ART, AMT, and Average SLAV, and the FEBOA population matrix for resource monitoring is referred as X . The i^{th} FEBOA member is denoted by X_i. The i^{th} FEBOA member's j^{th} scheduled task is represented as $x_{i,j}$. Equation (8) determines the beginning position of VM in the search space at random.

$$x_{i,j} = lb_j + r.\left(ub_j - lb_j\right), i = 1,2,\ldots,N, j = 1,2,\ldots,m \tag{8}$$

Here, $r \in [0,1]$ is a random number in the interval and the lower and upper bounds can be denoted by lb_j and ub_j, the j^{th}decision variables (AEC, ART, AMT, and SLAV) of the lb_j and ub_j. Equation (9) [Trojovský and Dehghani 2022] specifies these assessed values for the resource monitoring's objective function through a vector.

$$OF = \begin{bmatrix} OF_1 \\ \vdots \\ OF_I \\ \vdots \\ OF_N \end{bmatrix}_{N \times 1} = \begin{bmatrix} OF(X_1) \\ \vdots \\ OF(X_i) \\ \vdots \\ OF(X_N) \end{bmatrix}_{N \times 1} \tag{9}$$

Here, the ith FEBOA's obtained OF (Objective Function) value can be denoted as OF_i and the achieved OF vector values of the FEBOA population can be symbolized as OF. The OF's values act as conditions for assessing the suggested solutions' quality, with the function's best value designating the best member and the worst value designating the worst member. The two stages of exploration and exploitation that make up the FEBOA algorithm population update process are covered here.

Phase 1: Voting process and holding elections (exploration).

FEBOA members received votes for an individual during the election. As a result, equation (10) is used to simulate people's awareness within the VM. Those having greater objective function values during the simulation procedure.

$$A_i = \begin{Bmatrix} \dfrac{OF_i - OF_{worst}}{OF_{best} - OF_{worst}}, & OF_{best} \neq OF_{worst}; \\ 1, & else, \end{Bmatrix} \quad (10)$$

Here, the OF's best and worst values are represented as $OF_{best}$and OF_{worst}, respectively, and the realization of the ith EBOA member is A_i. It should be stated that, in the resource monitoring process, the objective function's minimum value is associated with OF_{best} and maximum value with OF_{worst}; whereas the issues of maximization, on the other hand, it is presumed in the FEBOA which the least amount of candidates (N_C) is two (i.e., $N_C \geq 2$)), indicating as 2 or more candidates may file to run in the election. If not, that VM will vote for alternative candidates at random. Equation (11) provides a numerical explanation of this voting procedure.

$$V_i = \begin{cases} C_1, & A_i > r; \\ C_{k,} & else, \end{cases} \quad (11)$$

Here, the best optimal VM for each resource monitoring can be referred as C_1, the vote of the i^{th} candidate in the VM can be denoted as V_i, and the k th candidate can be denoted as C_k, randomly chosen number k from the $\{2,3,\ldots,N_C\}$ set. candidate's positions within the FEBOA are updated with the elected leader's influence and direction. The leader leads the procedure of updating the FEBOA population in a manner that all members are assigned a novel position initially. If updating to the recently created position increases the OF's value, then it is acceptable. In the event that not, the appropriate member holds their prior position. Equations (12–13) are used in the FEBOA to model this update process.

$$x_{i,j}^{new,P1} = \begin{cases} x_{i,j} + r.\left(L_j - I.x_{i,j}\right), & OF_L < OF_i; \\ x_{i,j} + r.\left(x_{i,j} - L_j\right), & else, \end{cases} \quad (12)$$

$$X_i = \begin{cases} X_i^{new.P1}, OF_i^{new,P1} < OF_i; \\ X_i, \quad else, \end{cases} \quad (13)$$

Here, $OF_i^{new,P1}$ is the value of the OF, L is the elected leader, and OF_L is the value of the OF. The integer I arbitrarily chosen from the values of 1 or 2. A novel created position for the ith FEBOA member is denoted as $X_i^{new.P1}$, as it represents the jth dimension.

Phase 2: Public movement to raise awareness (exploitation)

The level of awareness among society's members greatly influences their ability to make wise decisions throughout the exploration procedure. Any suggested algorithm's nearby local search may yield a better result. A random location is taken into consideration in the locality of every member in the search space for replicating this local search procedure. The updated condition is then used to assess the resource monitoring's objective function in order to see if it is superior to the member's current VM. The local search is effective and the relevant member's VM position is updated if the novel VM location has a superior value for the OF. Eqns. 14–15 are used to model this update procedure intended for improving awareness among individuals in the EBOA.

$$x_{i,j}^{new,P2} = x_{i,j} * fw + (1 - 2r).R.\left(1 - \frac{t}{T}\right).x_{i,j} \quad (14)$$

$$X_i = \begin{cases} X_i^{new,P2}, OF_i^{new,P2} < OF_i \\ X_i, else \end{cases} \quad (15)$$

Here, $OF_i^{new,P2}$is the OF's value, R is the constant = 0.02, the iteration contour is t, and the highest amount of iterations be T. A recently generated VM location for the ith FEBOA member can be denoted as $X_i^{new,P2}$. Its jth dimension can be represented by $X_{i,j}^{new,P2}$. The algorithm's weight is generated by fw in equation (14). For every language phrase, a range of values between 0 and 1 along with some tolerance zones are assigned. Next, a triangular fuzzy integer is created to represent the j[th] vague (fuzzy) weight.

$$\widetilde{w}_j = (v_j - \delta_j^l, v_j, v_j + \delta_j^u) \quad (16)$$

Here, the zone for the weight value of the j[th] criterion can be defined as $\delta_j^l, \delta_j^u (j = 1,2,\ldots,k)$, and the weight that most closely matches the language expression of criterion relevance is $v_j (j = 1,2,\ldots,k)$.

Start EBOA.
Input problem information: variables, objective function, and constraints.
Set EBOA population size (N) and iterations (T).
Generate the initial population matrix at random.
Evaluate the objective function.
For t = 1 to T
Update best and worst population members.
Phase 1: Voting process and holding elections (exploration).
Calculate A using Eq. (4).
Determine candidates based on awareness criteria.
Simulate holding election and voting using Eq. (5).
Count the votes and determine the election winner as leader.
For i = 1 to N
Calculate X new,P1 i using Eq. (6).

Update Xi using Eq. (7).
Phase 2: Public movement to raise awareness (exploitation).
Calculate X new,P2 i using Eq. (8).
Update Xi using Eq. (9).
end
Save best proposed solution so far.
end
Output best quasi-optimal solution obtained with the EBOA.
End EBOA.

5.2. VM migration by Adaptive Threshold Fuzzy C-Means (ATFCM) clustering

The Bandwidth, RAM, CPU, and disk of computers in data centers are all related to the amount of energy they use. In data centers, proper virtual machine migration across servers can lower energy usage and SLA violations. On the other hand, frequent virtual machine migrations may have a detrimental effect on the virtual machine's application performance.

SLA Violation Metrics: A crucial component of any VM migration procedure is SLA violation. There are now two approaches to characterize the SLAV.

Total PDM (Performance Degradation Affected by VM Migration): It is indicated by equation (17),

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{d_j}}{C_{r_j}} \tag{17}$$

In the data center, the amount of VMs can be represented through the parameter M, the estimated performance deterioration brought on by the migration of VM$_j$ can be denoted as C_{d_j} and the total CPU capacity that VMs have demanded over the course of their lifetime is C_{r_j}.

(2) SLATAH (SLAV Time per Active Host): Equation (18) indicates that it refers to the portion of the entire SLA violation period that the CPU operation of the AH attained 100%.

$$SLATAH = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{s_i}}{T_{a_i}} \tag{18}$$

Here, the amount of host in the data center is N, the entire time that host i of the CPU utilization has reached 100% can be T_{s_i}, which leads to SLA violations, and the over-all time that host i has been in an active state is T_{a_i}. The VM on the host unable for the given demanded CPU ability due to the active host's 100% CPU load, which is the rationale behind the SLATAH. Two efficient techniques to independently assess the SLA violation are PDM and SLATAH. Consequently, the definition of a SLA violation is expressed as:

$$SLA = PDM \times SLATAH \tag{19}$$

SLAVs and energy consumption are included in the efficiency of energy. Reducing consumption of energy and SLAVs in data centers could be attained through enhancing energy efficiency. Consequently, the following is the definition of the energy efficiency metric:

$$E = \frac{1}{P \times SLA} \tag{20}$$

Here, the data center's power consumption can be denoted as P, SLA for a data center's SLA violation, and E for a data center's energy efficiency. Equation (20) demonstrates that energy efficiency increases with increasing E.

Data centers can increase their energy efficiency by using virtual machine migration. Nonetheless, there are a few major issues that need to be resolved: When a host is thought to be moderate or low-loaded, it is decided to maintain the rest of the VMs on that host unaffected. (3) When a host is thought to be low-loaded, all of the VMs on that host need to be transferred to a different host. (4) Choosing which VM or more could be transferred from the host's maximum loaded; (5) locating the novel host for every VM transferred from the maximum loaded or low-loaded hosts.

Data center hosts using ATFCM clustering are classified into 4 types based on the load they handle: extremely low load, low load, medium load, and maximum load hosts. Let us assume that the set of n machines to be separated based on CPU use is represented by the notation $\mathrm{VM} = \{\mathrm{vm}_1, \ldots, \mathrm{vm}_\mathrm{n}\}$, here $\mathrm{vm}_\mathrm{i} \in \mathbb{R}^\mathrm{d}$ for $\mathrm{i} = 1, \ldots, \mathrm{n}$; the cluster count is denoted by c, when $2 \leq \mathrm{c} < n$. The difficulty of fuzzy clustering is illustrated in Equation (21) [Gosain and Dahiya 2016].

$$P\text{: } minimize\ J_m(U,V) = \sum_{i=1}^{n}\sum_{j=1}^{c}\left(u_{ij}\right)^m \left\|vm_i, v_j\right\|^2 \tag{21}$$

Here, the set of centroids can be represented by $V = \{v_1, \ldots, v_c\}$, and the centroid of cluster j is v_j; (the host processing power at time i can be denoted as vm_i , through the empirical value, the size of n could be chosen). Then, the correlation matrix of all VM i to each cluster j is represented

as U = u_{ij} . The fuzzy factor, m, which represents the weighting exponent, expresses the extent to which the clusters overlap; for $i = 1,\ldots,n$ and $j = 1,\ldots,c$, .The Euclidean distance separating the centroid, v_j, and the VM, vm_i, is given by $d = \left\|vm_i, v_j\right\|^2$, when $m > 1$. It is possible to minimize J_m, an estimated model of V and U, in the following way [Pérez-Ortega et al., 2023],

$$v_j = \frac{\sum_{i=1}^{n}(u_{ij})^m \, vm_i}{\sum_{i=1}^{n}(u_{ij})^m} \; 1 \le j \le c \tag{22}$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{c}\left(\frac{\left\|vm_i, v_j\right\|^2}{\|vm_i, v_k\|^2}\right)^{\frac{1}{m-1}}} \; 1 \le i \le n, 1 \le j \le c \tag{23}$$

When the vectors that belongs to $\mathbb{R}^{\mathrm{d}}$ space are: $vm_i$and v_j are expressed as,

$$vm_i = (vm_1, \ldots, vm_d), 1 \le i \le n \tag{24}$$

$$v_j = (v_1, \ldots, v_d), 1 \le j \le c \tag{25}$$

Equations (26) to (28) formalize the fuzzy clustering limitations,

$$u_{ij} \in [0,1], 1 \le i \le n, 1 \le j \le c \tag{26}$$

$$\sum_{j=1}^{c} u_{ij} = 1, 1 \le i \le n \tag{27}$$

$$0 < \sum_{j=1}^{c} u_{ij} < n, 1 \le j \le c \tag{28}$$

According to equation (23) a vm_i degree of membership in a cluster j has value from zero to one. According to equation (26), the total of a degrees of vm_i membership in various clusters must equal 1. Lastly, the following definitions might be applied to the 3 thresholds in the ATFCM technique: (Th_{low} , Th_{me}, $and Th_{max}$).

$$Th_{low} = 0.5(1 - r * d) \tag{29}$$

$$Th_{me} = 0.75(1 - \mathrm{r} * \mathrm{d}) \tag{30}$$

$$Th_{max} = 1 - \mathrm{r} * \mathrm{d} \tag{31}$$

Here, an algorithmic parameter $r \in \mathbb{R}^+$ indicates the rapid manner of the system combines VMs. As it consolidation occurs, higher r results in reduced SLA breaches but higher energy consumption.

5.3. STOCHASTIC DYNAMIC SCHEDULING USING PGL

At the starting point of each scheduling period, the whole system functions as follows: (1) the RMS received task demands as well as task parameters such as computation, bandwidth, and SLA requirements. (2) The DRL model uses these specifications along with the host attributes in RMS to detect the subsequent scheduling decisions. (3) Using the DRL model's output, the constraint satisfaction module determines potential migration and scheduling choices. (4) The IoT gadget is notified through the RMS to submit its request for the new tasks directly to the associated edge or cloud server that is scheduled for that task. (5) The DRL framework's loss function is calculated, and its parameters are adjusted. Use a NN function approximator or Q-Table to simulate this function. The latter will give an objective strategy that is unadaptable in stochastic environments. In contrast, the methodology uses $Loss_i^{PG}$ as a signal for updating the network while attempting to estimate the policy itself and improve it via PG approaches. Utilize a BiRNN to estimate the function from $State_i$ to $Action_i^{PG}$ for each SI_i interval. The power of the BiRNN lies in its capability to represent complicated activist correlations among both inputs and outputs. Either policy (actor head) as well as cumulative loss after the (critic head) present interval are expected by a single net. By continuously pre-processing and sending the interval state to the BiRNN framework that includes the loss and penalties for updating the network parameters, the optimum scheduling option for each scheduling intervals may be determined. As a result, the framework can instantly alter to the needs of the operator, the setting, and particular applications. The Gated Recurrent Unit (GRU) [Supriya & Dhanalakshmi, 2023]., which models the temporal features of the task and host features, such as the hosts' CPU, RAM, and bandwidth capacity, and the task's demands, is used to build the recurrent layers.

A bidirectional RNN model for forward order input plus another BiRNN framework for the reverse order of input constitute the BiRNN framework. While the reverse order utilizes the input order from n to 1, the forward order utilizes the input order from 1 to n. The LSTM cell has evolved into the GRU, which has fewer parameters. Because GRU contain gating units, they can adaptively extract dependencies from large amounts of sequential VM information without losing any of the data from the previous segment of the order [Sharma and Casas 2020].

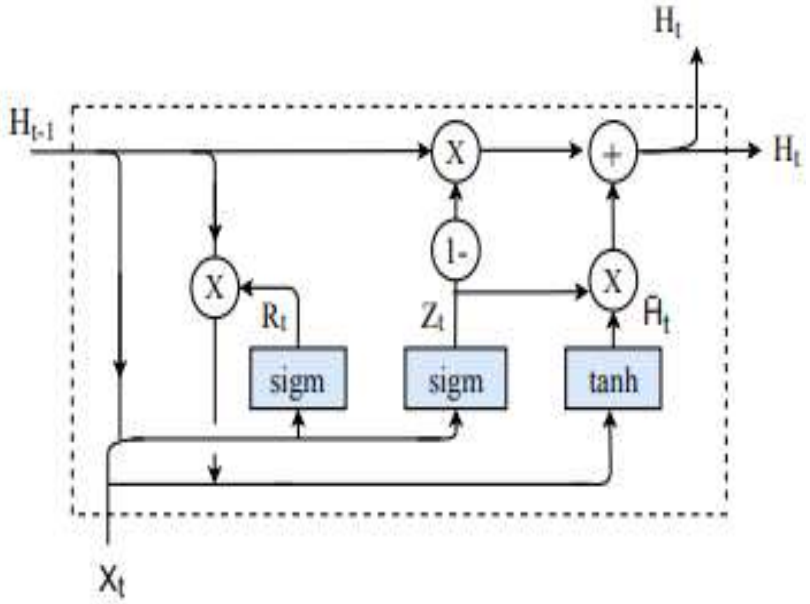

FIGURE 3. GRU CELL STRUCTURE

Compared to the LSTM cell, the GRU cell's architecture is substantially easier as seen in Figure 5. Tasks on GRU cells is defined by equations (32–35).

$$Z_t = sigm(W_z * [X_t, H_{t-1}] + B_z) \tag{32}$$

$$R_t = sigm(W_r * [X_t, H_{t-1}] + B_r) \tag{33}$$

$$\bar{H}_t = tanh(W_{\mathrm{h}} * [X_t, R_t \odot H_{t-1}] + B_{\mathrm{h}}) \tag{34}$$

$$H_t = Z_t \odot \bar{H}_t + (1 - Z_t) \odot H_{t-1} \tag{35}$$

The Reset gates, Previous hidden state, Update gate, Current hidden state are represented as R_t , $\bar{H}_t$, Z_t, H_t respectively, The weight matrices for update and reset gates are denoted as W_z and W_r.The weight matrix for hidden gate is W_h. The bias matrices for hidden state, reset gate, and update are denoted as B_h, B_r , B_z respectively are all examples of bias matrices.

The element's feature e is denoted as f_e, and the values for f feature maximum and minimum are denoted as, $\max_{f_e}$ and $\min_{f_e}$ respectively. 2 heuristic-based scheduling policies: LR for allocating tasks and MMT (Maximum-Migration-Time) for TS are used to determine these minimum and maximum values according to a sample dataset [27]. Subsequently, the standardization of features is carried out using equation (36).

$$e = \begin{cases} 0 \; if \; \max f_e = \min f_e \\ \min\left(1, max\left(0, \dfrac{e - \min f_e}{\max f_e - \min f_e}\right)\right), else \end{cases} \tag{36}$$

The BiRNN model takes this pre-processed input, decreases it, and then runs it via the dense layers. Through initially generating the ordered list of hosts $\mathrm{SortedHosts_i}$ with reducing probabilities in

O_i for all i, the output obtained O is transformed to $Action_i^{PG}$.Gradients have a negative sign to lower the overall loss and are balanced to this capacity. The MSE (Mean Square Error) of the estimated cumulative loss compared to the cumulative loss following an a single-stage is the 2nd gradient element. During scheduling interval, the output $Action_i^{PG}$ is transformed by CSM as $Action_i$ and delivered to the RMS. Consequently, the BiRNN has a forward pass for every interval. By continuously pre-processing and sending the interval state through the BiRNN model incorporating the loss and penalty for updating the network parameters, the optimum scheduling option for every scheduling intervals may be determined. As a result, the framework could promptly alter to the desires of the operator, the setting, and certain applications.

6. RESULTS AND DISCUSSION

Describe the experimental setting, performance metrics, dataset, and provide an extensive study of the outcomes by contrasting the model with several typical techniques in this section. To enable the utilization of characteristics like timing of response, expenses, and edge node power, CloudSim are utilized. Further software was created for the input pre-processing, output conversion, and Constraint Satisfaction Module. In CloudSim, the host and task monitoring services are used to determine the loss function. The tasks, or cloudlets, are allocated to VMs in the simulated setting, and the VMs are subsequently allotted to hosts.

DATASET: Coudlets, are allocated to VMs in the simulation setting, and VMs are subsequently assigned to hosts. In the edge-cloud setting where the cloudlets are currently established that takes into consideration switching from tasks to VMs by assigning the i^{th} formed tasks to the i^{th} generated VM and discarding the VM after the related task is finished. The freely available, actual Bitbrain dataset is the basis for the dynamic workload generation for tasks. Real-time resource usage figures of business-critical workloads hosted on Bitbrain architecture are available in Bitbrain's dataset [28]. The logs of more than 1000 VM workloads hosted on 2 distinct types of machines are included in this dataset because they exhibit actual architecture utilization patterns, which are helpful in building accurate input FVs for learning models.

Workload statistics, with the amount of required CPU cores, CPU utilization by MIPS, and requested RAM by network (receive/transmit) and disk (read/write) bandwidth features, are included in the dataset for all time-stamp (separated by five minutes). The BitBrain dataset is available for download at http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains. Separate the dataset into 2 parts, with respective VM workloads of 25.00% and 75.00%. The R2N2 network is trained on the larger segment, then evaluation, sensitivity assessments, and comparing with other relevant research are conducted on the previous separations. Microsoft Azure IaaS cloud service is the basis of the cloud layer's cost model [Shen et al., 2015].

Metrics: The metrics listed below have been utilized for assessing the outcomes.

ART is formulaized as follows,

$$ART = \frac{\sum_{t \in l_{i+1}} Response\ Time(t)}{|l_{i+1})} \quad (37)$$

The expression of SLAV as given below,

$$SLAV = \frac{\sum_i SLAV_i . |l_{i+1}|}{\sum_i l_i} \quad (38)$$

Average Task Completion Time: It is calculated by incrementing the typical TS, task performance, and host response times for the tasks that were executed throughout the maximum new scheduling interval.

The total tasks accomplished, the % of tasks completed in the estimated processing time (depending on specified MIPS), the percentage of migration of tasks during all interval, and the overall time spent on migrations during all intervals.

Results comparison Methods Several heuristics have been put out in the context of dynamic scheduling. These are association of several sub-heuristics for various sub-issues, like task/VM selection and host overload detection, from which the top 3 heuristics have been chosen. Best Fit Decreasing (BFD) heuristics are used by all of these differences to determine the target host. Moreover, compare the results with 2 common standard RL procedures that are implemented in the existing research.

DDQN: The DQL-RL strategy has been applied in numerous study.

DRL (REINFORCE):PG-REINFORCE technique through FC (Fully Connected) NN.

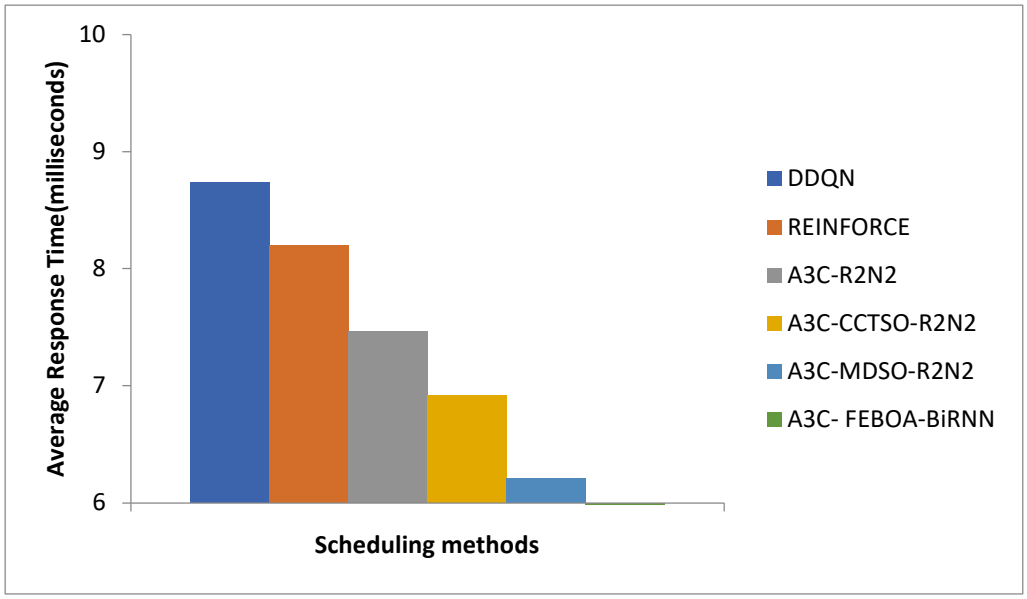

FIGURE 4. ART VS. SCHEDULING TECHNIQUES

Figure 4 illustrates that the suggested solution offers the lowest average response time compared to all alternative scheduling policies. The suggested approach outperforms the best foundation

algorithm, A3C-CCTSO-R2N2, by 5.958%. Given that the suggested model assigns jobs by RMS (MDSO) doesn't need several migrations and incorporates AMT into the loss function, it clearly accepts information regarding whether a node is an edge or cloud node. This indicates that although existing approaches like DDQN, REINFORCE, A3C-R2N2, A3C-CCTSO-R2N2, and A3C-MDSO-R2N2 have higher ART of 8.74 ms, 8.20 ms, 7.46 ms, 6.92 ms, and 6.21 ms similarly, the suggested structure has a less ART of 6.21 ms (Refer Table 1).

TABLE 1. ART VS. SCHEDULING TECHNIQUES

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
A3C- FEBOA-BiRNN	5.84

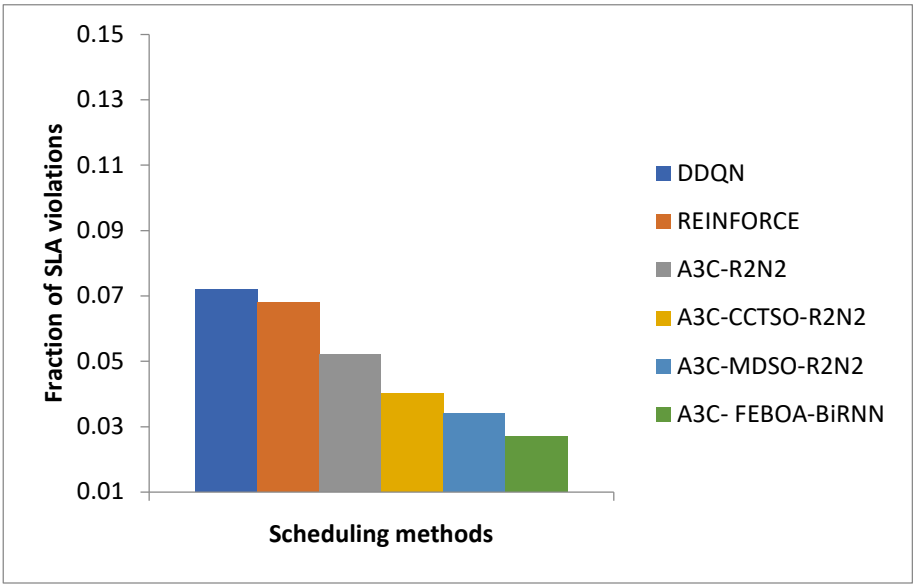

FIGURE 5. FRACTION OF SLA VIOLATIONS VS. SCHEDULING TECHNIQUES

The suggested model has 20.58% fewer SLAV than the A3C-MDSO-R2N2 policy, as illustrated in Figure 5. Once more, this is the result of fewer migrations and clever TS to avoid significant loss values from SLAV. The suggested system has lower SLAVs (0.027), while other approaches (see Table 2) have higher ART of 0.072, 0.064, 0.052, 0.040, and 0.034, respectively. These approaches include LR-MMT, MAD-MC, DDQN, REINFORCE, A3C-R2N2, A3C-CCTSO-R2N2, and A3C-MDSO-R2N2.

TABLE 2. FRACTION OF SLA VIOLATIONS VS. SCHEDULING TECHNIQUES

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
A3C- FEBOA-BiRNN	0.027

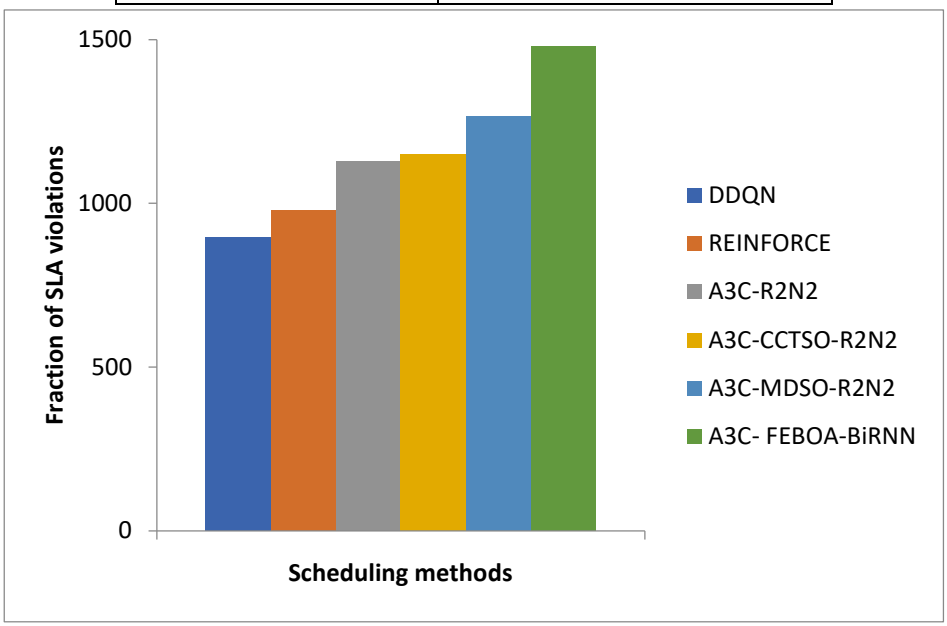

FIGURE 6.NUMBER OF COMPLETED TASKS VS. SCHEDULING TECHNIQUES

As can be seen in Figure 6, the suggested approach has a greater percentage of cloudlets completed and can guarantee cloudlets are assigned to the fewest number of cloud VMs in order to minimize costs. The total amount of tasks completed for the suggested framework is 1478, while the number of finished tasks for other approaches, including DDQN, REINFORCE, A3C-R2N2, A3C-CCTSO-R2N2, and A3C-MDSO-R2N2, is 895, 978, 1127, 1150, and 1264, respectively (refer to Table 3).

TABLE 3. NUMBER OF COMPLETED TASKS VS. SCHEDULING TECHNIQUES

Scheduling Methods	Fraction of SLA Violations
DDQN	895
REINFORCE	978
A3C-R2N2	1127
A3C-CCTSO-R2N2	1150
A3C-MDSO-R2N2	1264
A3C- FEBOA-BiRNN	1478

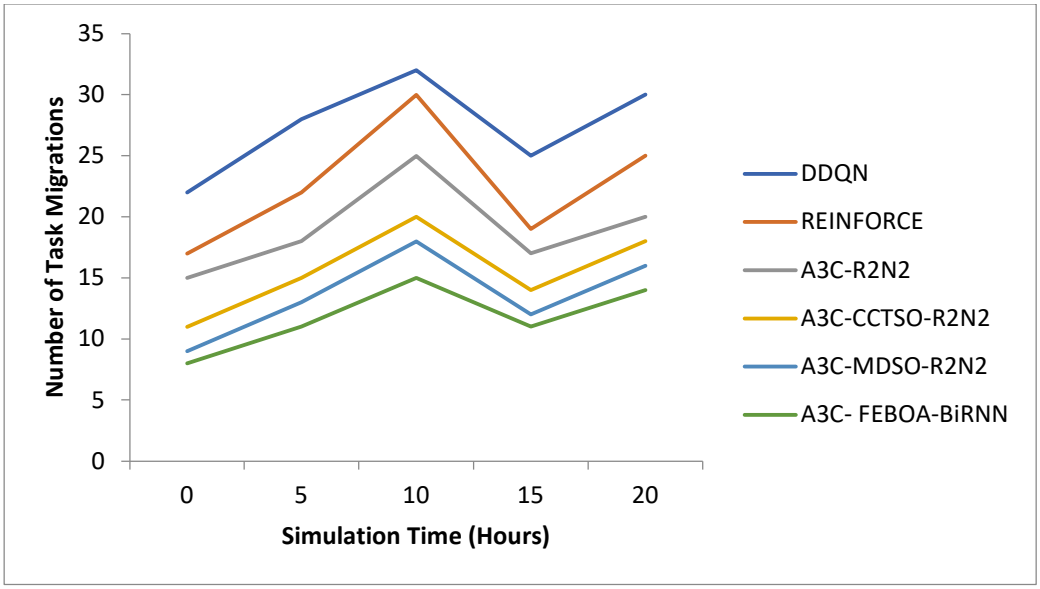

FIGURE 7.NUMBER OF TASK MIGRATION IN EACH INTERVAL VS. SCHEDULING TECHNIQUES

The outcomes of the number of task migrations in relation to simulation time are presented in Figure 7. Numerous methods, comprising DDQN, REINFORCE, A3C-R2N2, A3C-CCTSO-R2N2, A3C-MDSO-R2N2, and A3C-FEBOA-BiRNN, are used to analyze the outcomes. It demonstrates that, for a 20 (hour) simulation time, the suggested system has less task migrations14 than other approaches like DDQN, REINFORCE, A3C-R2N2, A3C-CCTSO-R2N2, and A3C-MDSO-R2N2—which have 30, 25, 20, 18, and 16 task migrations (Refer Table 4).

TABLE 4. NUMBER OF TASK MIGRATION VS. SCHEDULING TECHNIQUES

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
A3C- FEBOA-BiRNN	8	11	15	11	14

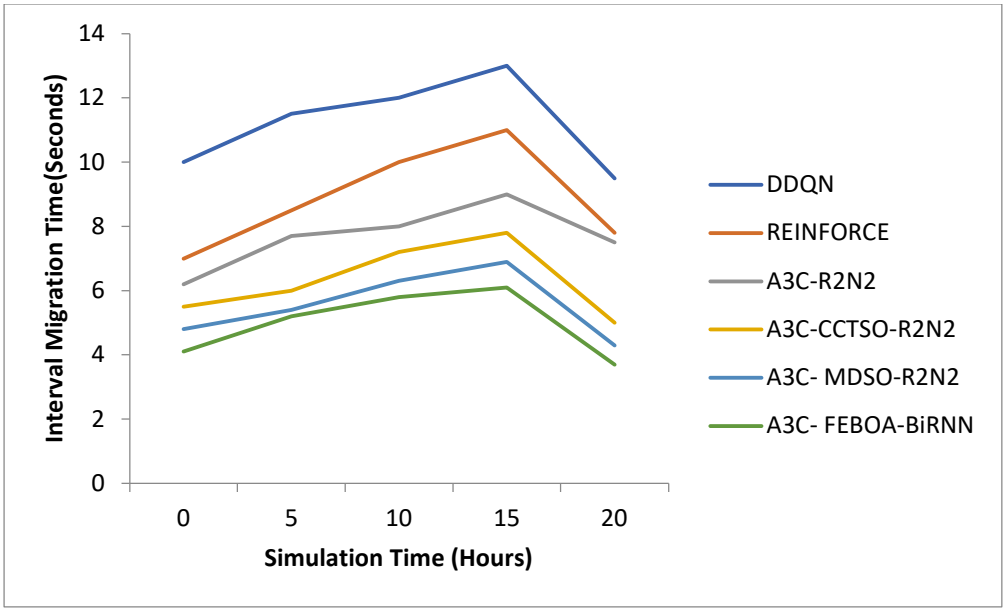

FIGURE 8.TOTAL MIGRATION TIME IN EACH INTERVAL VS. SCHEDULING TECHNIQUES

The outcomes of the task's interval migration time in relation to simulation time are presented in Figure 8. Numerous systems, comprising DDQN, REINFORCE, A3C-R2N2, A3C-CCTSO-R2N2, A3C-MDSO-R2N2, and A3C-FEBOA-BiRNN, are used to analyze the outcomes. The results indicate that the suggested system takes 3.70 seconds, while other approaches (refer to Table 4) take 9.50 seconds, 7.80 seconds, 7.50 seconds, 5.0 seconds, and 4.30 seconds for a simulation time of 20 hours. These methods include DDQN, REINFORCE, A3C-R2N2, A3C-CCTSO-R2N2, and A3C-MDSO-R2N2.

Table 5. Total Migration Time in each interval vs. scheduling techniques

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
A3C- FEBOA-BiRNN	4.1	5.2	5.8	6.1	3.7

7. CONCLUSION AND FUTURE WORK

Create a structural system model in this study for scheduling in Edge-Cloud background using RNNs. The Algorithm for FEBOA has been presented to minimize metrics like AEC, ART, AMT, and SLAV. VM has the ability to select and support the superior candidate. FEBOA is a planned population-based metaheuristic algorithm. The two stages of FEBOA are

exploration and exploitation. The weight of the FEBOA is produced via the triangular fuzzy membership function. Data centers can increase their energy efficiency by using VM migration. For VM migration with data centers, ATFCM clustering is established. For the purpose of stochastic dynamic scheduling, PG-RL (A3C) is used. Through asynchronous updates, A3C enables the scheduler to swiftly adjust to vigorously varying backgrounds, and BiRNN can rapidly learn network weights by taking use of the temporal behaviors of tasks and workloads. CloudSim uses the real-world Bitbrain dataset to demonstrate the model's superiority over current approaches. Metrics such as Average Task Completion Time, SLAV, and ART are used to quantify the results. Future research can, however, focus on scalable RL models such as Impala. Additionally, make plans to look into the security and privacy of information in the future.

REFERENCES

1. Rababah B, Alam T, Eskicioglu R (2020), "The next generation internet of things architecture towards distributed intelligence: Reviews, applications, and research challenges", J Telecommun Electron Comput Eng 12(2), pp.11-19.
2. Shekhar, S andGokhale, A (2017) Dynamic resource management across cloud-edge resources for performance-sensitive applications. In: 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). IEEE, Madrid. pp 707–710.
3. Elgendy I, Zhang W, Liu C, Hsu C-H (2018) An efficient and secured framework for mobile cloud computing. IEEE Trans Cloud Comput 9(1):79–87.
4. Sahni Y, Cao J, Zhang S, Yang L (2017) Edge mesh: A new paradigm to enable distributed intelligence in internet of things. IEEE access 5:16441–16458.
5. Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J. and Jue, J.P., 2019. All one needs to know about fog computing and related edge computing paradigms: A complete survey. Journal of Systems Architecture, 98, pp.289-330.
6. Tuli S., R. Mahmud, S. Tuli, and R. Buyya, "FogBus: A Blockchainbased Lightweight Framework for Edge and Fog Computing," Journal of Systems and Software, vol. 154, pp. 22 – 36, 2019.
7. Wang J., K. Liu, B. Li, T. Liu, R. Li, and Z. Han, "Delay-sensitive multi-period computation offloading with reliability guarantees in fog networks," IEEE Transactions on Mobile Computing, 2019.
8. Fox, G., Glazier, J.A., Kadupitiya, J.C.S., Jadhao, V., Kim, M., Qiu, J., Sluka, J.P., Somogyi, E., Marathe, M., Adiga, A. and Chen, J., 2019, Learning everywhere: Pervasive machine learning for effective high-performance computation. In 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 422-429.
9. Xu M., S. Alamro, T. Lan, and S. Subramaniam, "Laser: A deep learning approach for speculative execution and replication of deadline-critical jobs in cloud," in Proceedings of the 26th International Conference on Computer Communication and Networks (ICCCN). IEEE, 2017, pp. 1–8.

10. Li, H., Ota, K. and Dong, M., 2018. Learning IoT in edge: Deep learning for the Internet of Things with edge computing. *IEEE network*, *32*(1), pp.96-101.
11. Mnih V., A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in Proceedings of the International conference on machine learning, 2016, pp. 1928–1937.
12. Mao H., M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in Proceedings of the 15th ACM Workshop on Hot Topics in Networks, 2016, pp. 50–56.
13. Tao Z, Xia Q, Hao Z et al (2019) A survey of virtual machine management in edge computing. Proceed IEEE 107(8):1482–1499.
14. Gonzalez, M., López-Espín, J.J., Aparicio, J. and Talbi, E.G., 2022. A hyper-matheuristic approach for solving mixed integer linear optimization models in the context of data envelopment analysis. *PeerJ Computer Science*, *8*, pp.1-24.
15. Mejahed, S. and Elshrkawey, M., 2022. A multi-objective algorithm for virtual machine placement in cloud environments using a hybrid of particle swarm optimization and flower pollination optimization. PeerJ Computer Science, 8, pp.1-27.
16. Trojovský, P. and Dehghani, M., 2022. A new optimization algorithm based on mimicking the voting process for leader selection. *PeerJ Computer Science*, *8*, pp.1-40.
17. Xie, Y., Zhu, Y., Wang, Y., Cheng, Y., Xu, R., Sani, A.S., Yuan, D. and Yang, Y., 2019. A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud–edge environment. Future Generation Computer Systems, 97, pp.361-378.
18. Li, C., Wang, C. and Luo, Y., 2020. An efficient scheduling optimization strategy for improving consistency maintenance in edge cloud environment. The Journal of Supercomputing, 76, pp.6941-6968.
19. Tong, Z., Chen, H., Deng, X., Li, K. and Li, K., 2020. A scheduling scheme in the cloud computing environment using deep Q-learning. *Information Sciences*, *512*, pp.1170-1191.
20. Tuli, S., Ilager, S., Ramamohanarao, K. and Buyya, R., 2020. Dynamic scheduling for stochastic edge-cloud computing environments using A3C learning and residual recurrent neural networks. IEEE transactions on mobile computing, 21(3), pp.940-954.
21. Zivkovic, M., Bezdan, T., Strumberger, I., Bacanin, N. and Venkatachalam, K., 2021. Improved harris hawks optimization algorithm for workflow scheduling challenge in cloud–edge environment. In Computer Networks, Big Data and IoT: Proceedings of ICCBI 2020 (pp. 87-102). Singapore: Springer Singapore.
22. Shruthi, G., Mundada, M.R., Sowmya, B.J. and Supreeth, S., 2022. Mayfly taylor optimisation-based scheduling algorithm with deep reinforcement learning for dynamic scheduling in fog-cloud computing. Applied Computational Intelligence and Soft Computing, vol.2022, no. 2131699,pp.1-17.
23. Jayanetti, A., Halgamuge, S. and Buyya, R., 2022. Deep reinforcement learning for energy and time optimized scheduling of precedence-constrained tasks in edge–cloud computing environments. Future Generation Computer Systems, 137, pp.14-30.

24. Zhang, M., Yang, Z., Yan, J., Ali, S., Ding, W. and Wang, G., 2022. Task-load aware and predictive-based workflow scheduling in cloud-edge collaborative environment. Journal of Reliable Intelligent Environments, 8(1), pp.35-47.
25. Li, F., Tan, W.J. and Cai, W., 2022. A wholistic optimization of containerized workflow scheduling and deployment in the cloud–edge environment. Simulation Modelling Practice and Theory, 118, p.102521.
26. Xue, F., Hai, Q., Dong, T., Cui, Z. and Gong, Y., 2022. A deep reinforcement learning based hybrid algorithm for efficient resource scheduling in edge computing environment. Information Sciences, 608, pp.362-374.
27. Mnih V., A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," In Proceedings of the International conference on machine learning, 2016, pp. 1928–1937.
28. Akbari M., R. Hassan, and S. H. Alizadeh, "An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems," Engineering Applications of Artificial Intelligence, vol. 61, pp. 35–46, 2017.
29. Trojovský, P. and Dehghani, M., 2022. A new optimization algorithm based on mimicking the voting process for leader selection. PeerJ Computer Science, 8, pp.1-40.
30. Gosain, A. and Dahiya, S., 2016. Performance analysis of various fuzzy clustering algorithms: a review. Procedia Computer Science, 79, pp.100-111.
31. Pérez-Ortega, J., Rey-Figueroa, C.D., Roblero-Aguilar, S.S., Almanza-Ortega, N.N., Zavala-Díaz, C., García-Paredes, S. and Landero-Nájera, V., 2023. POFCM: A Parallel Fuzzy Clustering Algorithm for Large Datasets. Mathematics, 11(8), pp.1-16.
32. Sharma, R.K. and Casas, M., 2020, Wavefront parallelization of recurrent neural networks on multi-core architectures. In Proceedings of the 34th ACM International Conference on Supercomputing , pp. 1-12.
33. Shen S., V. van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," In 15th IEEE/ACM International Symposium on Cluster, Cloud and GridComputing, 2015, pp. 465–474.
34. Supriya, S. ., & Dhanalakshmi, K. . (2023). 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, 12(8s), 160–172.