IRJEMS International Research Journal of Economics and Management Studies
Published by Eternal Scientific Publications
ISSN: 2583 – 5238 / Volume 4 Issue 3 March 2025 / Pg. No: 266-281
Paper Id: IRJEMS-V4I3P130, Doi: 10.56472/25835238/IRJEMS-V4I3P130

Original Article

Dependent Tasks Verification-Aware Task Scheduling and Resource Allocation in Cloud Devops Using Obl-Fuzzy and Smrnn

[1]Sai Sandeep Ogety
[1]Cloud Computing & DevOps Specialist, Independent Researcher, Raleigh, NC, USA.

Received Date: 04 March 2025 Revised Date: 22 March 2025 Accepted Date: 25 March 2025 Published Date: 27 March 2025

Abstract: *In Cloud Computing (CC), Task Scheduling (TS) involves assigning tasks to suitable resources, while Resource Allocation (RA) focuses on efficiently distributing cloud resources for optimal utilization. Nevertheless, the prevailing studies didn't perform dependent task verification, which might increase the Continuous Integration/Continuous Deployment (CI/CD) cycles. Thus, Offset Broken Line– Fuzzy (OBL-Fuzzy) and Sinusoidal Maxsig Recurrent Neural Network (SMRNN)-enabled dependent tasks verification-aware TS and RA in cloud Development Operations (DevOps) are presented in this paper. Primarily, the graph is constructed for the DevOps application by using Exponential Coffman Kahn's Directed Acyclic Graph (ECK-DAG). Then, workflow assignment, attribute extraction, clustering by Elbow Cohen's Density-Based Spatial Clustering of Applications with Noise (ECDBSCAN), workflow fragmentation, and Load balancing by Addax Rastrigrin Optimization Algorithm (AROA) are performed. Further, the Merkle tree is constructed for the fragmented workflows using HEX-Gini-BLOOM Merkle-Tree (HGB-MT). Afterward, the constructed Merkle tree is checked with the constructed graphs. The tasks are scheduled if it is verified; otherwise, the task request is rejected. Then, the dependent tasks' status is checked in the HGB-MT; if any dependent tasks are pending, then the least priority is given to that task. Next, by employing OBL-Fuzzy, the tasks are scheduled. Finally, by using AROA, the resources are allocated. As per the results, the proposed model achieved a high accuracy of 99%.*

Keywords: *Task scheduling, Resource Allocation, Development Operations (DevOps), Continuous Integration/Continuous Deployment (CI/CD), Virtual Machine (VM) load prediction, Elbow Cohen's Density-Based Spatial Clustering of Applications with Noise (ECDBSCAN), and Load balancing.*

I. INTRODUCTION

Currently, CC is gaining major attention to handle dynamic computing tasks and streamline the automation of software development/DevOps (Buttar et al., 2023). Here, the TS process assigns tasks to available Virtual Machine (VM) regarding priority (Dreibholz & Mazumdar, 2023) (Sharma et al., 2022). Similarly, RA concentrates on distributing cloud resources based on the demands of various applications (Liu et al., 2023). Nevertheless, due to poor RA, resource underutilization may occur. Thus, many innovative techniques are introduced to schedule the tasks and allocate the resources (Shafiq et al., 2021).

A hybrid particle swarm algorithm, Genetic Algorithm (GA), and Ant Colony Optimization (ACO) were introduced for TS in existing studies (Fu et al., 2021) (Houssein et al., 2021). Similarly, Actor-Critic Deep Reinforcement Learning was introduced for efficient RA (Chen et al., 2021a). Similarly, for RA, Nondominated Sorting GA with the Elite Strategy (NSGA-II) and Whale Optimization Algorithm (WOA) were utilized (Chen et al., 2021) (Jayaprakash et al., 2021). Nevertheless, the existing studies didn't perform dependent task verification, leading to increased CI/CD cycles. Thus, this paper proposes a dependent task verification-aware TS and RA in cloud DevOps.

A) Problem Statement

- None of the prevailing works concentrated on dependent task verification, which might increase the CI/CD cycles.
- In the existing (Shi & Lin, 2022), RA was performed without considering the load of the VM.
- Due to the huge number of task requests, the collision might present in VMs (Singh et al., 2021).
- Most prevailing works didn't differentiate the task before assigning it to the VM.

B) Objectives

- Dependent task status is checked in HGB-MT; if any dependent tasks are pending, then the task is given the least priority.
- SMRNN is introduced to predict the load of VMs.
- AROA is established to load balance the workflow requests.

- ECDBSCAN is utilized to cluster similar workflows to avoid mismatched RA.

This remaining part is arranged as follows: the existing literature is illustrated in Section 2, the proposed methodology is discussed in Section 3, the result is conveyed in Section 4, and lastly, the proposed model is concluded with future work in Section 5.

II. LITERATURE SURVEY

(Shi & Lin, 2022) established VM-RA model in CC. Here, for RA, the multi-objective optimization GA was utilized. The model attained improved computational performance. Nevertheless, the load of the VM was not focused. Therefore, the waiting time for tasks was increased.

(Singh et al., 2021) Recommended RA and TS framework in CC. The swarm-centric ant colony optimization was utilized for TS and RA. Regarding Quality of Service (QoS), the research performed better. However, owing to the huge number of task requests, the collision might present in VMs.

(Belgacem et al., 2022) Discovered the RA model in CC. The Intelligent Multi-Agent system and Reinforcement Learning Method (IMARM) were allocated resources here. The model obtained superior outcomes regarding fault tolerance. Still, for unpredictable cloud environments, this work provided poor performance.

(Kaur Walia et al., 2021) Presented the TS approach in the cloud. Here, a hybrid GA and Flower Pollination-centric Algorithm (FPA) were used for TS. Low energy consumption was attained in this study. Nevertheless, the research had higher computational overhead and longer decision-making time.

(Pirozmand et al., 2021) Offered TS model in CC. Here, for TS, the Energy-Conscious Scheduling Heuristic- GA was employed. The model consumed less energy. The model might struggle to maintain scalability as the number of tasks increases.

III. PROPOSED FRAMEWORK

Here, to allocate the resources, the AROA is introduced. In Figure 1, the diagrammatic layout of the proposed model is displayed.

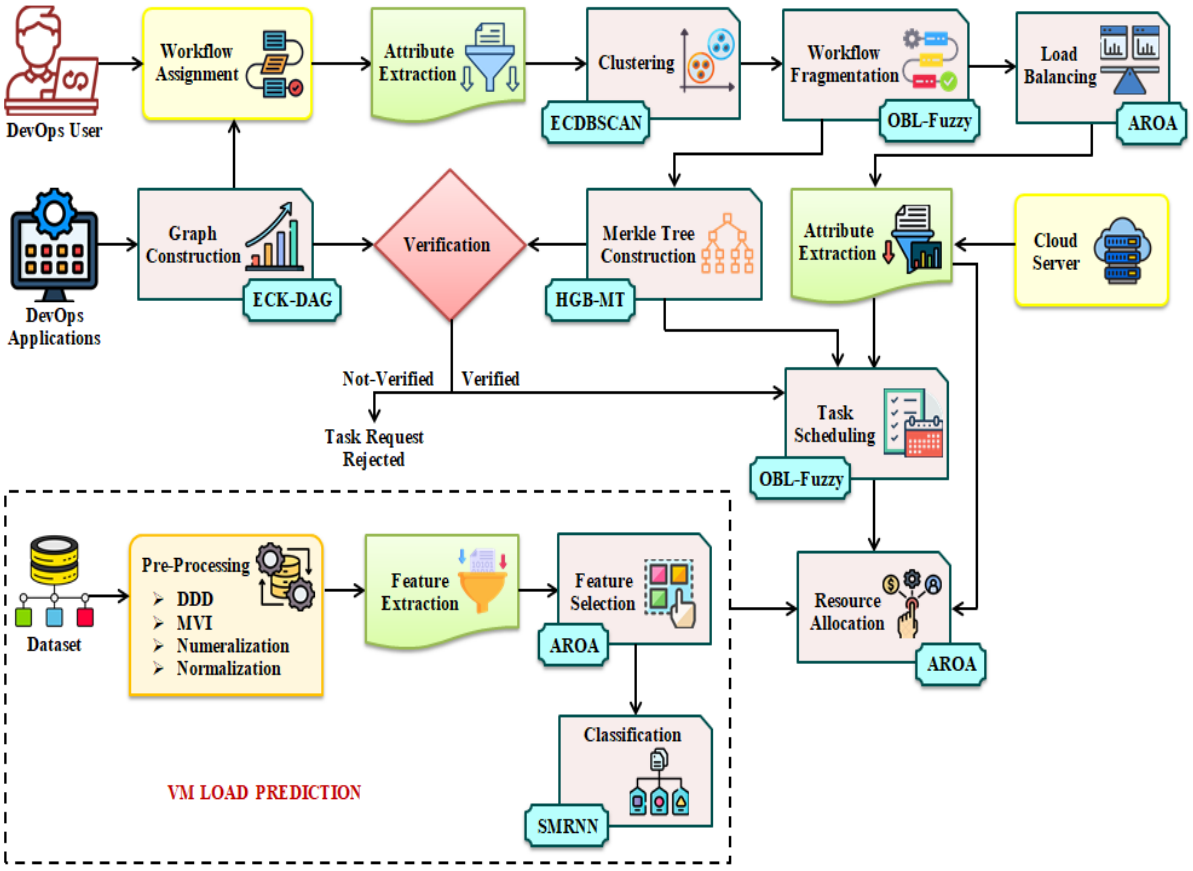


Figure 1: Diagrammatic layout of the proposed model

A) DevOps Application

Initially, for RA, the tasks of DevOps applications are provided. It is defined as ∂o .

B) Graph Construction

Next, by using ECK-DAG, the graph is constructed for ∂o . Directed Acyclic Graph (DAG) efficiently models the data flow in DevOps applications. The computational complexity of DAG is increased if the graph has a huge number of nodes and edges. Hence, Exponential Coffman Kahn's (ECK) technique is used.

In DAG, the DevOps application modules are considered nodes (γ_g), and the links (χ_e) between the DevOps application modules are indicated as edges (ξ).

$$\gamma_g = (\gamma_1, \gamma_2, \gamma_3, \ldots, \gamma_H) \quad where \quad g = (1,2,\ldots,H) \tag{1}$$

$$\xi = \{(\gamma_1, \gamma_2), (\gamma_1, \gamma_3), (\gamma_2, \gamma_4), (\gamma_3, \gamma_4), (\gamma_4, \gamma_5), \ldots..\} \tag{2}$$

Here, $g = (1,2,\ldots,H)$ implies the number of nodes. The topological sorting is performed based on ECK. Coffman Kahn's technique identifies and processes nodes with an in-degree 0. Here, the exponential function is used to manage the complex scenarios, thus achieving the desired topological ordering (Top_{od}).

$$Top_{od} = (\gamma_1, \gamma_3, \gamma_2, \gamma_4, \gamma_5) \tag{3}$$

Next, the adjacency matrix (Adj) is estimated as,

$$Adj = \begin{bmatrix} & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 \\ \gamma_1 & 0 & \chi_1 & \chi_2 & 0 & 0 \\ \gamma_2 & 0 & 0 & 0 & \chi_3 & 0 \\ \gamma_3 & 0 & 0 & 0 & \chi_4 & 0 \\ \gamma_4 & 0 & 0 & 0 & 0 & \chi_5 \\ \gamma_5 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4}$$

The constructed graph is indicated as G_τ .

C) Workflow Assignment

The workflows (ωf) are assigned to the DevOps users G_τ . The assigned workflows are specified as α_κ .

D) Attribute Extraction

The attributes, such as Job ID, Namespace, File size distributions, and so on, are extracted from the α_κ . The extracted attributes are signified as (ξx_a).

E) Clustering

The (ωf) are clustered by using ECDBSCAN based on (ξx_a). Density-Based Spatial Clustering of Applications with Noise (DBSCAN) excellently clusters the workflow regarding the density of connections. Yet, improper calculation of epsilon and minpts leads to under or over-clustering in DBSCAN. Consequently, the Elbow technique and Cohen's technique are used.

Mostly, based on the Elbow technique, the epsilon parameter (β) is calculated.

$$\beta = \sum_{d=1}^{V} \sum_{\xi x_a} \left\| \xi x_a - u_d \right\| \tag{5}$$

Here, V indicates the number of clusters and u_d designates the centroid. Then, the clusters are formed based on the minpts (M), which are calculated by Cohen's technique.

$$M = \frac{\overline{\xi x}_1 - \overline{\xi x}_2}{\sqrt{\frac{St_1^3 + St_2^3}{2}}} \tag{6}$$

Where, $\overline{\xi x}$ describes the mean value of (ξx_a), and St states the standard deviation. Afterward, for identifying the clusters, the core point (C) is computed.

$$C = \beta \times M \tag{7}$$

Regarding (C), (β), and (M), similar workflows (ωf) are clustered. The clustered workflows are signified as Θ_ϕ.

Pseudocode for ECDBSCAN

Input: Workflow (ωf)

Output: Clustered workflow (Θ_ϕ)

Begin

Initialize $(\omega f), (\xi x_a)$

For each (ωf)

Compute epsilon

$$\beta = \sum_{d=1}^{V} \sum_{\xi x_a} \left\| \xi x_a - u_d \right\|$$

Form

$$M = \frac{\overline{\xi x}_1 - \overline{\xi x}_2}{\sqrt{\frac{St_1^3 + St_2^3}{2}}}$$

Determine clusters by (C)

$$C = \beta \times M$$

Continue until convergence

End For

Obtain (Θ_ϕ)

End

Then, the workflow fragmentation is performed.

F) Workflow Fragmentation

Further, by employing OBL-Fuzzy, they (Θ_ϕ) are fragmented. Fuzzy is capable of giving the most proficient solutions. However, Fuzzy presents the tuning difficulty of the membership function. Thus, the Offset Broken Line membership function is utilized.

a. Rule Generation

creates fuzzy rules $(\Im)$, If-Then rules are employed.

$$\Im \xrightarrow{\Theta_\phi} \begin{cases} if \;\; T == P & then \;\; concurrent\, process \\ if \;\; T == S & then \;\; repetitive\, process \\ if \;\; T == I & then \;\; alternative\, process \\ if \;\; T == \partial if & then \;\; vertical\, fragmentation\, and\, orthogonal\, fragmentation \end{cases} \tag{8}$$

Here, T implies the tasks, P reveals parallel workflow, S denotes the similar process workflow, I represents the independent workflow, and ∂if states a different task and similar process workflow.

b. Membership Function

Next, the Offset Broken Line membership function is computed.

$$\Omega = \begin{cases} 0 & \Theta_\phi \le c \\ \dfrac{\Theta_\phi - c + \eta}{d - c + \eta} & c < \Theta_\phi \le d \\ 1 & d < \Theta_\phi \le p \\ \dfrac{q - \Theta_\phi + \eta}{q - p + \eta} & p < \Theta_\phi \le q \\ 0 & \Theta_\phi > q \end{cases} \tag{9}$$

Here, η states offset or center, and c, d, p, and q are constant values. The crisp data $(\in)$ are converted into $(\Im)$ Fuzzification (Fuz).

$$Fuz = (\in \to \Im) \tag{10}$$

Moreover, the $(\Im)$ are converted into $(\in)$ in the defuzzification (Def).

$$Def = (\Im \to \in) \tag{11}$$

The fragmented workflows are indicated as F_h.

G) Load Balancing

Afterward, by using the AROA, the workflow requests $(\Re)$ are load-balanced. Addax Optimization Algorithm (AOA) aims to strike a balance between exploring the search space for possible solutions. Nonetheless, AOA has local optima issues. Thus, the Rastrigrin chaotic map is utilized.

The Addaxes population (Y) is initialized. Here, the workflow requests are considered as the initialized population.

$$Y \xrightarrow{\Re} \begin{bmatrix} Y_1 \\ \vdots \\ Y_r \\ \vdots \\ Y_N \end{bmatrix}_{N\times m} = \begin{bmatrix} y_{1,1} & \cdots & y_{1,s} & \cdots & y_{1,m} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ y_{r,1} & \cdots & y_{r,s} & \cdots & y_{r,m} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ y_{N,1} & \cdots & y_{N,s} & \cdots & y_{N,m} \end{bmatrix}_{N\times m} \tag{12}$$

$$y_{r,s} = low_s + \Gamma_{r,s} \bullet (upper_s - low_s) \tag{13}$$

Here, Y_r denotes the r^{th} addax, $y_{r,s}$ represents the s^{th} dimension in the search space, $\Gamma_{r,s}$ implies the random number, and low_s and $upper_s$ specify the lower and upper bounds, respectively. Then, the fitness function $(\Im t)$ is estimated, which considers minimum response time $(\min(res))$ as fitness.

$$\Im t = \min(res) * \begin{bmatrix} \Im t_1 \\ \vdots \\ \Im t_r \\ \vdots \\ \Im t_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} \Im t(Y_1) \\ \vdots \\ \Im t(Y_r) \\ \vdots \\ \Im t(Y_N) \end{bmatrix}_{N\times 1} \tag{14}$$

Then, to locate the food sources, the addaxes conduct extensive searches. The new position $\left(y_{r,s}^{pos1}\right)$ of addaxes is changed as,

$$y_{r,s}^{pos1} = y_{r,s} + \Gamma_{r,s} \cdot \left(SE_{r,s} - \exists_{r,s} \cdot y_{r,s}\right) \tag{15}$$

$$Y_r = \begin{cases} if\ \Im t_{r,s}^{pos1} \le \Im t_{r,s} & y_{r,s}^{pos1} \\ else & y_{r,s} \end{cases} \tag{16}$$

Where $SE_{r,s}$ signifies the selected area, $\exists_{r,s}$ outlines the random value, and $\Im t_{r,s}^{pos1}$ implies fitness $\left(y_{r,s}^{pos1}\right)$.

Next, the addaxes begin digging in shady areas. Here, the Rastrigrin chaotic map $\left(\Phi_{r,s}\right)$ is used. The new position $\left(y_{r,s}^{pos2}\right)$ is determined as,

$$\Phi_{r,s} = \sum \left[y_{r,s}^2 - 10\cos\left(2\pi y_{r,s}\right) + 10\right] \tag{17}$$

$$y_{r,s}^{pos2} = y_{r,s} + \left(1 - 2\,\Phi_{r,s}\right) \cdot \frac{upper_s - low_s}{it} \tag{18}$$

$$Y_r = \begin{cases} if\ \Im t_{r,s}^{pos2} \le \Im t_{r,s} & y_{r,s}^{pos2} \\ else & y_{r,s} \end{cases} \tag{19}$$

Here, it indicates the current iteration and $\Im t_{r,s}^{pos2}$ signifies the fitness $\left(y_{r,s}^{pos2}\right)$. The load balanced workflow requests are signified as L_i.

H) Attribute Extraction

Then L_i, the attributes like job id, file size distributions, and runtime distribution are extracted. Similarly, the attributes, such as Memory, Speed, Million Instructions Per Second (MIPS), etc, are extracted from the cloud server $\left(Cl_{ser}\right)$. The extracted attributes are defined as $\left(Atr_{\varepsilon}\right)$.

I) Merkle Tree Construction

Similarly, by using HGB-MT, the Merkle tree is constructed for F_h. Merkle Tree (MT) enables fast verification of large data. Yet, for verifying all data, MT needs L network access. Thus, the HEX-Gini-BLOOM technique is included in MT.

The hashcode $\left(K\right)$ is generated for each F_h by employing HEX-Gini-BLOOM $\left(W\right)$.

$$W = \left\{\ell_{\kappa'}\left(1 - \sum \exp\left(\left(F_h\right)^2 \times 16^{h-1}\right)\right)\right\} \rightarrow \left(K_1, K_2, K_3, \ldots\ldots, K_x\right) \tag{20}$$

Here, $\ell_{\kappa'}$ indicates the hash function value and K_x the number of generated hash. Then, F_h they are separated into leaf hash $\left(K_{la}\right)$, branch hash $\left(K_{bn}\right)$, and root hash $\left(K_{ro}\right)$.

$$K_{la} = \left\{\left(K_1 + K_2\right), \left(K_3 + K_4\right), \ldots\ldots, \left(K_{(x-1)} + K_x\right)\right\} \tag{21}$$

$$K_{bn} = \left\{\left(K_1 + K_2 + K_3 + K_4\right), \ldots\ldots, \left(K_{(x-3)} + K_{(x-2)} + K_{(x-1)} + K_x\right)\right\} \tag{22}$$

$$K_{ro} = \left\{\left(K_1 + K_2 + K_3 + K_4 +, \ldots\ldots, +K_{(x-1)} + K_x\right)\right\} \tag{23}$$

The constructed Merkle tree is represented as Ω_{MT}. Then, the Ω_{MT} is checked with the G_τ.

$$VO = \begin{cases} if\ \Omega_{MT} \rightarrow exists\ in\ G_\tau & Performtask\ scheduling \\ if\ \Omega_{MT} \rightarrow not\ exists\ in\ G_\tau & Task\ request\ is\ rejected \end{cases} \tag{24}$$

Here, VO indicates the verification outcomes. Next, dependent task verification $\left(DV\right)$ is done; here, the dependent tasks status $\left(\partial ft\right)$ is checked in the Ω_{MT}; if any dependent tasks are pending, then the least priority is given to that task.

J) Task Scheduling

The tasks are scheduled by using OBL-Fuzzy to arrange the tasks based on priority based on (DV), (VO), and (Atr_{ε}) . In Section 3.6, the process of OBL-Fuzzy is explained. Here, the fuzzy rules (Q) are generated as,

$$Q=\begin{cases} if\ \partial ft = completed\ \&\&\quad D = high\quad \&\& \\ MIPS = high\ \&\&\quad spd = high\quad \&\&\quad X = high & high\ priority \\ if\ \partial ft = in\ process\ \&\&\quad D = medium\quad \&\& \\ \quad MIPS = medium\ \&\&\ spd = medium\quad \&\& \\ \quad X = medium & medium\ priority \\ if\ \partial ft = pending\ \&\&\quad D = low\quad \&\& \\ MIPS = low\ \&\&\quad spd = low\quad \&\&\quad X = low & low\ priority \end{cases} \tag{25}$$

Here, D indicates file size, spd implies speed, and X specifies memory. The scheduled tasks are represented as ζ_{σ} follows.

K) VM Load Prediction

Likewise, to reduce the waiting time for tasks, VM load is predicted.

L) Dataset

Mainly, to train the VM load prediction system, the "VM Workload Predictor" dataset is gathered and is denoted as $\tilde{\vartheta}_{\varphi}$.

M) Pre-Processing

Next, they $\tilde{\vartheta}_{\varphi}$ are pre-processed; initially, the duplicate values $\tilde{\vartheta}_{\varphi}$ are removed, and it is specified as DD_{up} . Then, missing values DD_{up} are imputed (MI) and are written as,

$$MI = \frac{\sum DD_{up}}{k} \tag{26}$$

Here k is the number of DD_{up} . Afterward, they (MI) are converted into numerical values (Nu_{ll}) . Next, by using the Z-score formula, they Nu_{ll} are normalized. The normalized data $(\dddot{\Theta}_{w})$ are given as,

$$\dddot{\Theta}_{w} = \frac{Nu_{ll} - \mu}{\ddot{\sigma}} \tag{27}$$

Here, μ the mean and standard deviation are designated, respectively, and the pre-processed data is denoted as $\wp_{l}$.

N) Feature Extraction

Then $\wp_{l}$, the features, such as timestamp, Central Processing Unit (CPU) core, CPU usage, memory usage, and so on, are extracted and indicated Ψ_{fet} .

O) Feature Selection

Optimal features are selected Ψ_{fet} by employing AROA. Here, maximum classification accuracy is considered as a fitness function. The selected features are defined as ς_{v} follows.

P) Classification

The VM load is predicted based on ς_{v} employing an SMRNN. Recurrent Neural Networks (RNNs) efficiently learn from past experiences. Nevertheless, the RNNs suffer from vanishing gradient problems and low learning efficiency. Thus, in RNN, the Sinusoidal initialization technique and Maxsig activation function are utilized. In Figure 2, the SMRNN classifier diagram is displayed.

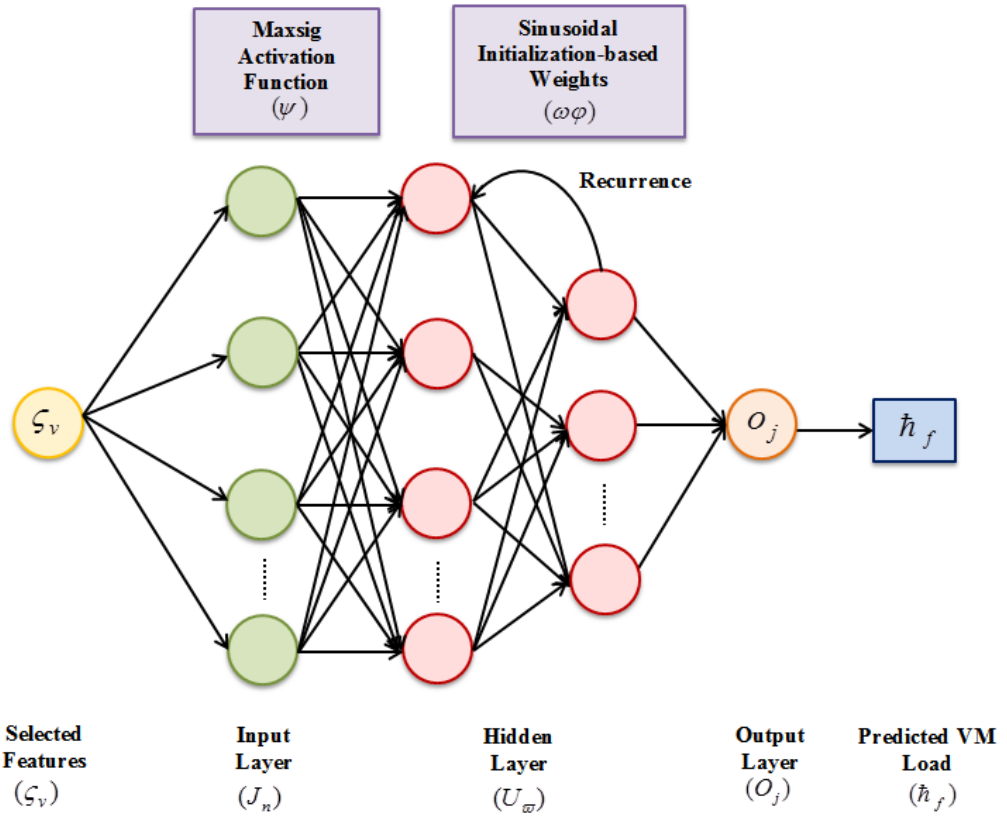


Figure 2: SMRNN classifier

- ***Input Layer***

  The ς_v is fed to the input layer. Later, the input layer operation outcomes (J_n) are subjected to the hidden layer.

- ***Hidden Layer***

  The hidden layer (U_ϖ) is updated based on the current input and previous hidden state.

$$U_\varpi = \psi \cdot \left[(U_{\varpi-1}, J_n) * \omega\varphi + B\right] \tag{28}$$

Here, $U_{\varpi-1}$ specifies the previous hidden state, ψ signifies the Maxsig activation function, B signifies the bias term, and $\omega\varphi$ determines Sinusoidal initialization-based weights.

$$\psi = \max\left(J_n, \rho\left(J_n\right)\right) \tag{29}$$

$$\omega\varphi \xrightarrow{J_n} Am \cdot \sin\left(2\pi \cdot fq \cdot R + \tilde{\phi}\right) \tag{30}$$

Here, ρ designates the sigmoid function Am and fq indicates the amplitude and frequency, respectively, R implies the random value, and $\tilde{\phi}$ represents the phase offset.

- ***Output Layer***

  The output layer (O_j) provides VM load prediction outcomes.

$$O_j = \psi\left(\omega\varphi * U_\varpi + B\right) \tag{31}$$

The predicted VM load is signified as $\hbar_f$.

Pseudocode for SMRNN

Input: Selected Features (ς_v)

Output: Predicted VM load $(\hbar_f)$

Begin

Initialize $(\varsigma_v),(B)$

For each (ς_v)

Perform input layer (J_n)

Discover

$$U_{\varpi} = \psi \cdot \left[(U_{\varpi-1}, J_n) * \omega\varphi + B\right]$$

Compute

$$\psi = \max\left(J_n, \rho(J_n)\right)$$

Find

$$\omega\varphi \xrightarrow{J_n} Am \cdot \sin\left(2\pi \cdot fq \cdot R + \tilde{\phi}\right)$$

Implement

$$O_j = \psi\left(\omega\varphi * U_{\varpi} + B\right)$$

End For

Obtain $(\hbar_f)$

End

The SMRNN excellently predicted the VM loads.

Q) Resource Allocation

The resources are allocated for the corresponding ζ_σ by using AROA based on $(\hbar_f)$ and (Atr_ε). Here, a high utilization rate is considered as the fitness function. The allocated resources are defined as A_o. The proposed model excellently scheduled the tasks and allocated the resources in cloud DevOps.

IV. RESULT AND DISCUSSION

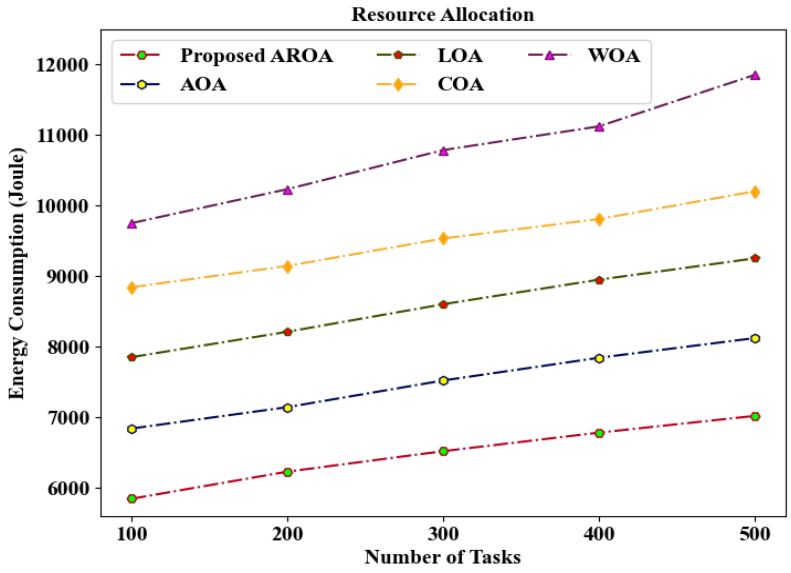
Here, the proposed method's performance evaluation is done. Likewise, the proposed model is implemented in the working platform of PYTHON.

A) Dataset Description

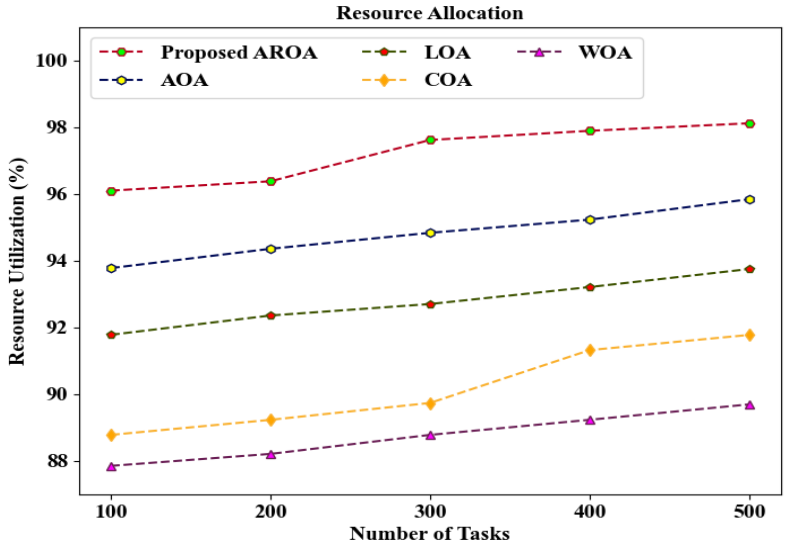
The proposed model is assessed by employing the "VM Workload Predictor" dataset. The dataset link is given in the reference section. Likewise, this dataset consists of 869458 numbers of data. Among that, 80% (6,95,566) and 20% (1,73,891) of data are used for training and testing, respectively.

B) Performance Validation

Here, the proposed techniques are compared with prevailing methods.



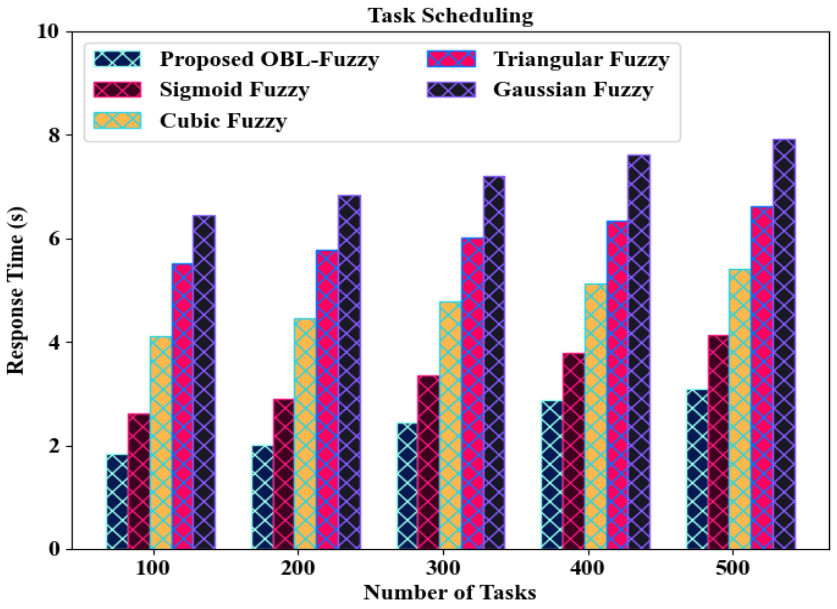
(a)



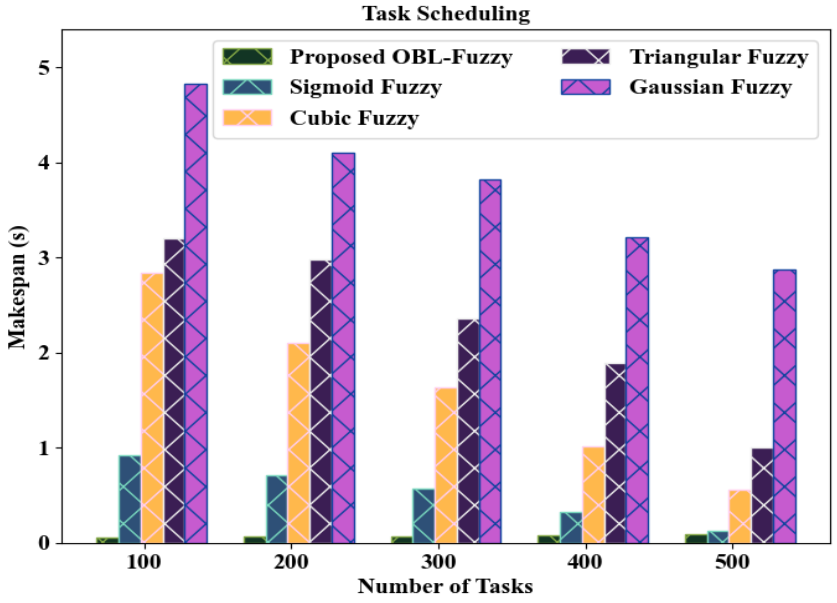
(b)

Figure 3: Graphical representation regarding (a) energy consumption and (b) resource utilization

The graphical representation of the proposed and existing techniques' displayed in Figure 3. Here, the proposed AROA consumed less energy of 7021 Joule and utilized a huge resource of 98.12% for 500 tasks. Nevertheless, limited performance was attained by the existing techniques, such as AOA, Lion Optimization Algorithm (LOA), Coyote Optimization Algorithm (COA), and Whale Optimization Algorithm (WOA).



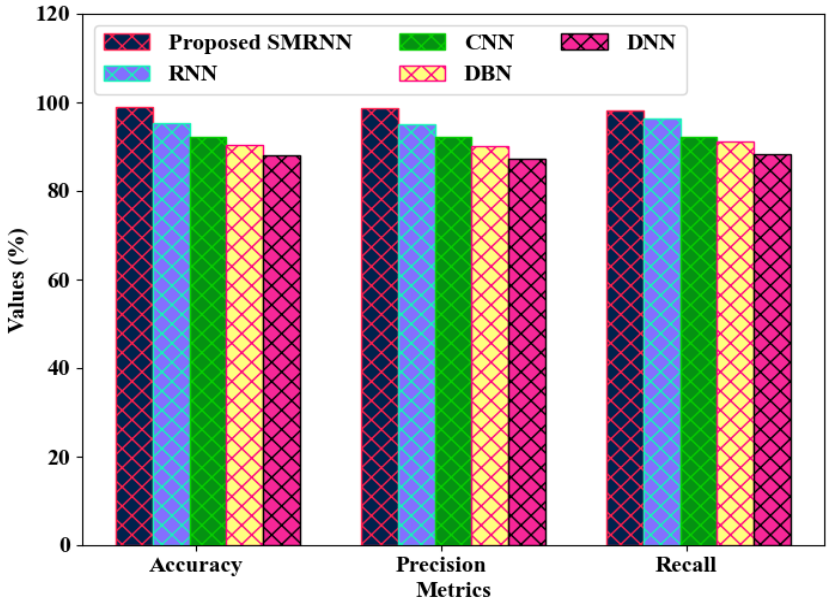
(a)



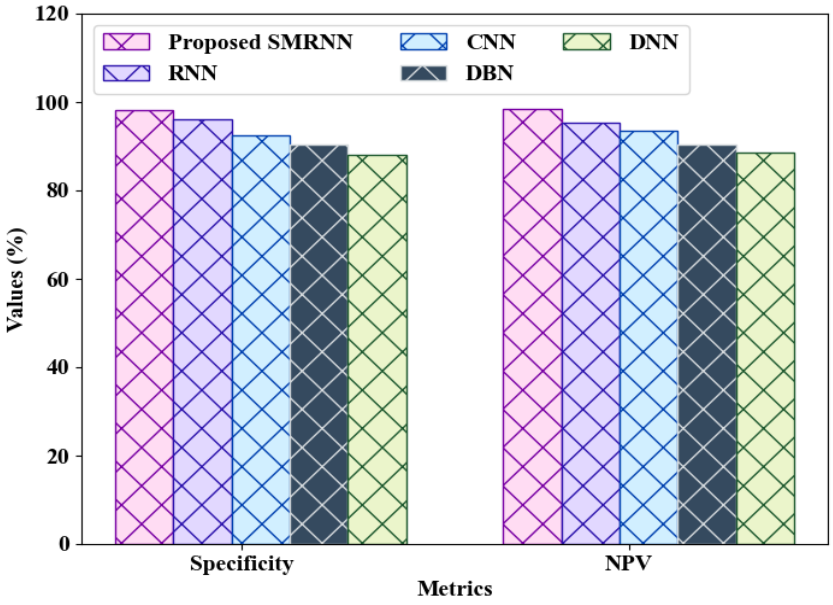
(b)

Figure 4: Performance assessment regarding (a) response time and (b) makespan

Figure 4 displays a performance assessment of the proposed and prevailing techniques. For 400 numbers of tasks, the proposed OBL-Fuzzy obtained a low response time of 2.8754s and a makespan of 0.0854s, while the prevailing Sigmoid Fuzzy, Cubic Fuzzy, Triangular Fuzzy, and Gaussian Fuzzy attained poor performance.



(a)



(b)

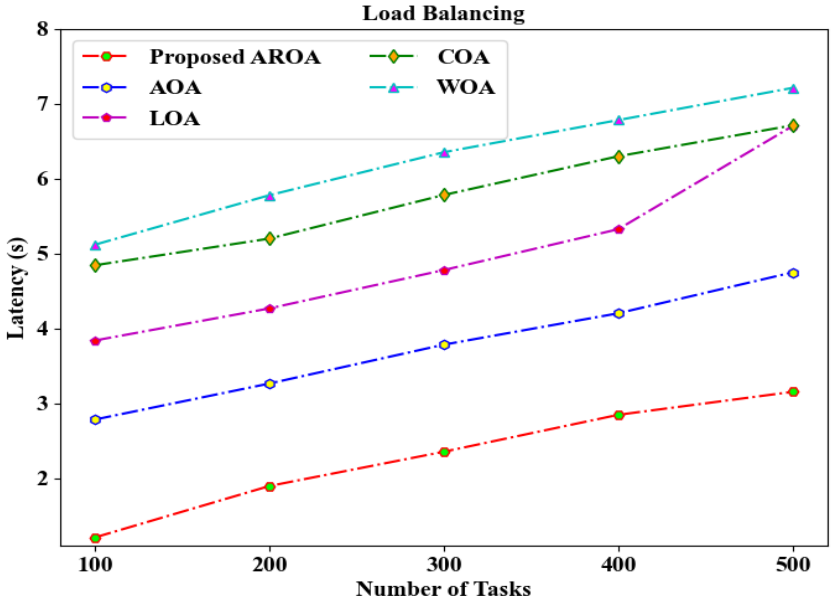
Figure 5: Comparative evaluation regarding (a) accuracy, precision, recall, (b) specificity, NPV

The comparative evaluation of the proposed and conventional methods is depicted in Figure 5. Regarding accuracy, precision, recall, specificity, and Negative Predictive Value (NPV), the proposed SMRNN achieved 98.85%, 98.63%, 98.12%, 98.23%, and 98.56%; yet, the prevailing techniques like RNN, Convolutional Neural Network (CNN), Deep Belief Network (DBN), and Deep Neural Network (DNN) attained limited performance.

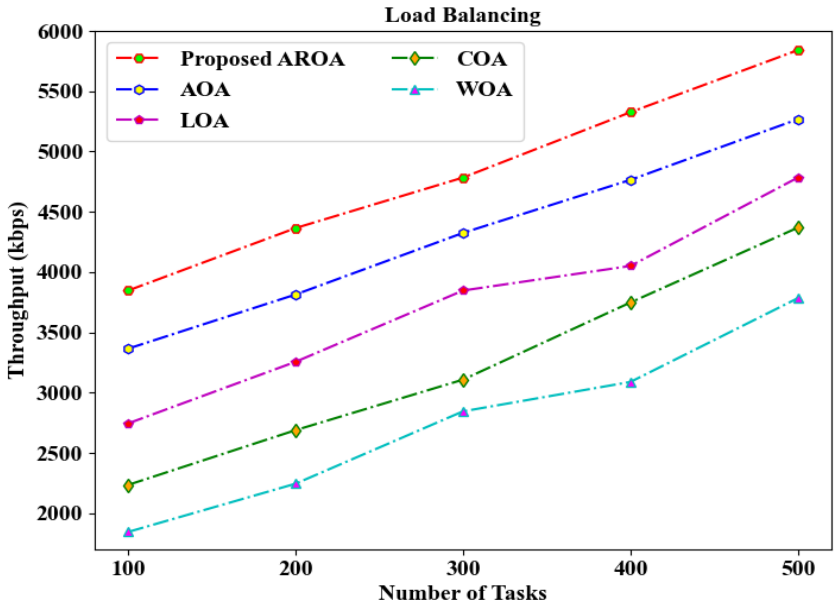
Table 1: Rule generation time

Techniques	Rule Generation Time (ms)
Proposed OBL-Fuzzy	987
Sigmoid Fuzzy	1245
Cubic Fuzzy	1754
Triangular Fuzzy	2086
Gaussian Fuzzy	2369

In Table 1, the rule generation time of the proposed and existing techniques is shown. The proposed OBL-Fuzzy took less than 987ms for rule generation, whereas the existing Sigmoid Fuzzy, Cubic Fuzzy, Triangular Fuzzy, and Gaussian Fuzzy took a huge amount of time.



(a)



(b)

Figure 6: Performance analysis regarding (a) latency, (b) throughput

The performance analysis of the proposed AROA and prevailing methods are depicted in Figure 6. For 100 tasks, the proposed AROA obtained a low latency of 1.2101s and a high throughput of 3847kbps, while the prevailing techniques attained poor performance.

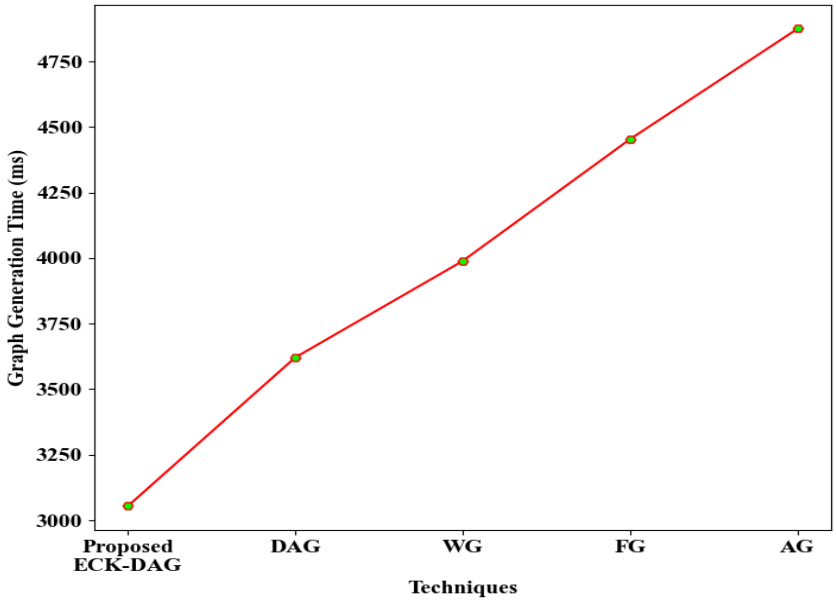


Figure 7: Graph generation time

Figure 7 shows the graph generation time of the proposed and existing techniques. Regarding graph generation time, the proposed ECK-DAG attained a low value of 3054ms, whereas the existing DAG, Weighted Graph (WG), Finite Graph (FG), and Acyclic Graph (AG) had high time complexity.

Table 2: Clustering time

Methods	Clustering Time (ms)
Proposed ECDBSCAN	24108
DBSCAN	28956
K-Means	33658
CLARA	38054
FCM	43625

The clustering times of the proposed and prevailing techniques are depicted in Table 2. For clustering, the proposed ECDBSCAN took a less time of 24108ms, while the prevailing DBSCAN, K-Means, Clustering Large Applications (CLARA), and Fuzzy C-Means (FCM) obtained an average clustering time of 36073.25ms.

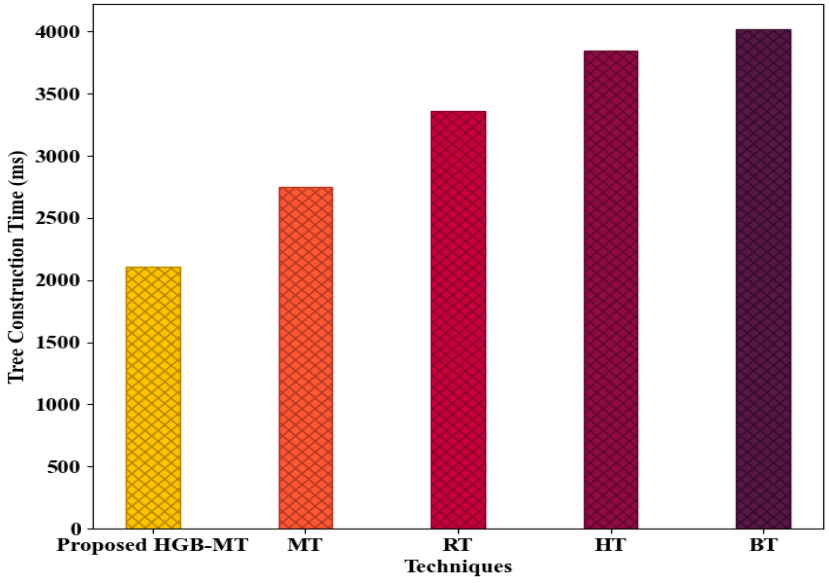


Figure 8: Tree construction time

The tree construction time of the proposed and existing methods are displayed in Figure 8. Regarding tree construction time, the proposed HGB-MT obtained less than 2108ms, while the prevailing MT, RT, HT, and BT attained maximum time.

C) Comparative Assessment

A comparative assessment is carried out for the proposed and related works.

Table 3: Comparative Analysis

Authors' name	Techniques	Response time (s)	Makespan (s)	Energy Consumption (Joule)	Resource Utilization (%)
Proposed Model	OBL-Fuzzy and AROA	1.8472	0.0584	5847	96.10
(Ben Alla et al., 2021)	Dynamic Priority-Queue (DPQ)	1.95	0.181	-	-
(Kruekaew & Kimpan, 2022)	Artificial Bee Colony (ABC) with Q-learning	-	7.678	-	-
(Nabi et al., 2021)	Resource-aware Dynamic TS Approach (RDTSA)	7.25	32.6	-	-
(Fathalla et al., 2021)	Best K-First-Fit (Best-KFF)	-	-	-	53.18
(Goyal et al., 2021)	WOA	-	-	8165.603	-

Table 3 displays the comparative assessment; here, the proposed OBL-Fuzzy attained a low response time of 1.8472s and a low makespan of 0.181s. The proposed AROA obtained a low energy consumption of 5847Joule and a high resource utilization of 96.10%. Nevertheless, poor performance was attained by the existing DPQ, ABC with Q-learning, RDTSA, Best-KFF, and WOA.

V. CONCLUSION

Here, OBL-Fuzzy and SMRNN-based TS and RA in cloud DevOps are presented. Here, this study performed significant processes, such as TS and RA. For 500 tasks, the proposed AROA consumed less energy of 7021Joule. Similarly, the proposed SMRNN achieved a high accuracy and precision of 98.85% and 98.63%, respectively. Thus, the proposed model had high efficiency. The proposed model failed to preserve workflow information and user authentication even though it concentrated on improving the efficiency of TS and RA.

Future Work

Multi-factor authentication techniques will be introduced in the future to preserve the workflow information and perform user authentication.

VI. REFERENCES

[1] **Dataset link:** https://github.com/kwananth/VMWorkloadPredictor

[2] Belgacem, A., Mahmoudi, S., & Kihl, M. (2022). Intelligent multi-agent reinforcement learning model for resource allocation in cloud computing. *Journal of King Saud University - Computer and Information Sciences*, *34*(6), 2391–2404. https://doi.org/10.1016/j.jksuci.2022.03.016

[3] Ben Alla, H., Ben Alla, S., Ezzati, A., & Touhafi, A. (2021). A novel multiclass priority algorithm for task scheduling in cloud computing. In *Journal of Supercomputing* (Vol. 77, Issue 10). Springer US. https://doi.org/10.1007/s11227-021-03741-4

[4] Buttar, A. M., Khalid, A., Alenezi, M., Akbar, M. A., Rafi, S., Gumaei, A. H., & Riaz, M. T. (2023). Optimization of DevOps Transformation for Cloud-Based Applications. *Electronics (Switzerland)*, *12*(2), 1–15. https://doi.org/10.3390/electronics12020357

[5] Chen, J., Wang, Y., & Liu, T. (2021). A proactive resource allocation method based on adaptive prediction of resource requests in cloud computing. *Eurasip Journal on Wireless Communications and Networking*, *2021*(1), 1–20. https://doi.org/10.1186/s13638-021-01912-8

[6] Chen, Z., Hu, J., Min, G., Luo, C., & El-Ghazawi, T. (2021 a). Adaptive and Efficient Resource Allocation in Cloud Datacenters Using Actor-Critic Deep Reinforcement Learning. *IEEE Transactions on Parallel and Distributed Systems*, 1–14. https://doi.org/10.1109/TPDS.2021.3132422

[7] Dreibholz, T., & Mazumdar, S. (2023). Towards a lightweight task scheduling framework for cloud and edge platforms. *Internet of Things (Netherlands)*, *21*, 1–16. https://doi.org/10.1016/j.iot.2022.100651

[8] Fathalla, A., Li, K., & Salah, A. (2021). Best-KFF: a multi-objective preemptive resource allocation policy for cloud computing systems. *Cluster Computing*, *25*(1), 1–17. https://doi.org/10.1007/s10586-021-03407-z

[9] Fu, X., Sun, Y., Wang, H., & Li, H. (2021). Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm. *Cluster Computing*, 1–10. https://doi.org/10.1007/s10586-020-03221-z

[10] Goyal, S., Bhushan, S., Kumar, Y., Rana, A. U. H. S., Bhutta, M. R., Ijaz, M. F., & Son, Y. (2021). An optimized framework for energy-resource allocation in a cloud environment based on the whale optimization algorithm. *Sensors*, *21*(5), 1–20. https://doi.org/10.3390/s21051583

[11] Houssein, E. H., Gad, A. G., Wazery, Y. M., & Suganthan, P. N. (2021). Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends. *Swarm and Evolutionary Computation*, *62*, 1–41. https://doi.org/10.1016/j.swevo.2021.100841

[12] Jayaprakash, S., Nagarajan, M. D., Prado, R. P. de, Subramanian, S., & Divakarachari, P. B. (2021). A systematic review of energy management strategies for resource allocation in the cloud: Clustering, optimization and machine learning. *Energies*, *14*(17), 1–18. https://doi.org/10.3390/en14175322

[13] Kaur Walia, N., Kaur, N., Alowaidi, M., Bhatia, K. S., Mishra, S., Sharma, N. K., Sharma, S. K., & Kaur, H. (2021). An Energy-Efficient Hybrid Scheduling Algorithm for Task Scheduling in the Cloud Computing Environments. *IEEE Access*, *9*, 1–13. https://doi.org/10.1109/ACCESS.2021.3105727

[14] Kruekaew, B., & Kimpan, W. (2022). Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid

Artificial Bee Colony Algorithm with Reinforcement Learning. *IEEE Access*, *10*, 17803–17818. https://doi.org/10.1109/ACCESS.2022.3149955

[15] Liu, H., Chen, P., Ouyang, X., Gao, H., Yan, B., Grosso, P., & Zhao, Z. (2023). Robustness challenges in Reinforcement Learning based time-critical cloud resource scheduling: A Meta-Learning based solution. *Future Generation Computer Systems*, *146*, 18–33. https://doi.org/10.1016/j.future.2023.03.029

[16] Nabi, S., Ibrahim, M., & Jimenez, J. M. (2021). DRALBA: Dynamic and Resource Aware Load Balanced Scheduling Approach for Cloud Computing. *IEEE Access*, *9*, 61283–61297. https://doi.org/10.1109/ACCESS.2021.3074145

[17] Pirozmand, P., Hosseinabadi, A. A. R., Farrokhzad, M., Sadeghilalimi, M., Mirkamali, S., & Slowik, A. (2021). Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing. *Neural Computing and Applications*, *33*(19), 1–14. https://doi.org/10.1007/s00521-021-06002-w

[18] Shafiq, D. A., Jhanjhi, N. Z., Abdullah, A., & Alzain, M. A. (2021). A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications. *IEEE Access*, *9*, 41731–41744. https://doi.org/10.1109/ACCESS.2021.3065308

[19] Sharma, M., Kumar, M., & Samriya, J. K. (2022). An optimistic approach for task scheduling in cloud computing. *International Journal of Information Technology (Singapore)*, *14*(6), 2951–2961. https://doi.org/10.1007/s41870-022-01045-1

[20] Shi, F., & Lin, J. (2022). Virtual Machine Resource Allocation Optimization in Cloud Computing Based on Multi-objective Genetic Algorithm. *Computational Intelligence and Neuroscience*, *2022*, 1–10. https://doi.org/10.1155/2022/7873131

[21] Singh, H., Bhasin, A., & Kaveri, P. R. (2021). QRAS: efficient resource allocation for task scheduling in cloud computing. *SN Applied Sciences*, *3*(4), 1–7. https://doi.org/10.1007/s42452-021-04489-5