Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7778249

AN ENERGY-EFFICIENT DYNAMIC TASK SCHEDULING IN THE CLOUD WITH R-EDF AND LC-GTO BASED ON VM AVAILABILITY IDENTIFICATION

Yogaraja GSR
Assistant Professor, Dept. of Information Science & Engineering, SJC Institute of Technology, Chickaballapur

Dr. Thippeswamy MN
Professor & HOD, Dept. of Computer Science and Engineering, (AI&ML and Cyber Security), Ramaiah Institute of Technology Bangalore

Dr. Venkatesh K
Professor, Dept. of Computer Science & Engineering, NITTE Meenakshi Institute of technology, Bangalore

Abstract: For providing balanced resources to the Internet of Things (IoT) cloud users, Task Scheduling (TS) plays an essential role. But, energy consumption and execution timeare increased by scheduling tasks on the cloud resources without proper analysis of task strategy.For overcoming this setback, this paper proposes the Chessboard-K-Prototype Algorithm(Ch-KPA)-based task grouping with the RICE-based Earliest Deadline First (R-EDF) scheduling. Primarily, the bag of tasks is taken as input from which attributes are extracted. Next, using fuzzy, the tasks are classified as small, large, and medium based on the task length. Subsequently, based on the task attributes, the medium and large tasks are grouped as sequential and parallel using Ch-KPA. Next, based on first in first out, the tasks are added to the queue. Subsequently, the Logistic Chaotic map-based Giant Trevally Optimization (LC-GTO) selects the single optimal Virtual Machine (VM) for the small task. Likewise, the optimal container and VMs of multi-cloud are selected for sequential and parallel tasks. Meanwhile,the availability of selected VM in the VM monitoring layer is determined by Drop-connect-Random Translation Multi-Layer Perceptron (DRT-MLP) utilizing a feature updated table. If the VM is not in the updating state, the task is mapped to that VM.Lastly, the R-EDF scheduler dynamically schedules the task to the selected VM centered on the deadline. The proposed approach's efficiency is proved by the experimental outcomes.

Keywords: *Virtual Machine (VM),Chessboard-K-Prototype Algorithm (Ch-KPA), Drop-connect-Random Translation Multi-Layer Perceptron (DRT-MLP), task scheduling, Earliest Deadline First (EDF).*

1. INTRODUCTION

A foundation of modern technology is cloud computing, which offers scalable and cost-effective solutions to a variety of diverse issues (Iftikhar et al., 2023).Ultimately, for managing and handling IoT devices in cloud platforms, diverse resources are required (Ali et al., 2021).For example, Multimedia application providers could rent VMs from cloud providers for rendering their users with diverse services. In this way, users request and receive their favourite applications from the service provider (Rawas, 2021). Nevertheless, such under-

usage and over-usage of VM resources cause energy constraint issues as cloud servers have high bandwidth latency and communication waste (Zhang et al. 2022). Hence, for reducing energy consumption, enhancing resource utilization by balancing the number of on-run hosts' tasks in VMs is one potentially viable strategy (Marahatta et al., 2021). Therefore, a research named as task scheduling in the cloud is performed for balancing the tasks on VMs and for minimizing energy consumption.

An approach that assigns users' tasks to VMs for execution is named task scheduling. From the customers' view, their requested tasks should be run in the shortest amount of time on the VM by a suitable scheduling algorithm(Ebadifard et al., 2020).In TS, for mapping a proper set of resources to a particular task, a schedule is generated(Lu et al., 2019); this is performed by the component called Scheduler that mainly considers response time, latency, and throughput (Shyalika et al., 2020). Static and dynamic scheduling strategies are the two most fundamental forms of TS approaches in the cloud environment(Murad et al., 2022).Static scheduling utilizes public information and schedule tasks to the cloud service providers. Tasks whose information is unknown are dealt with the latter, which has more overhead compared to the former scheduling approach (Velliangiri et al., 2021). But, to manage and distribute many services over the cloud, an effective scheduler should render a dynamic scheduling approach (Bal et al., 2022). Conversely, only when the workloads have a small variation, the static TS algorithms are utilized(Houssein et al., 2021); this is not effective for the real-time scenario. Hence, in most of the works, developing energy-efficient dynamic TS in the cloud environment is concentrated.

Centered on the optimization algorithms, namely the Whale Optimization algorithm (WOA) (Chen et al., 2020), Ant Colony Optimization (ACO) algorithm (Wei, 2020), et cetera, some of the developed dynamic task schedulers are designed. But, missing deadlines cause performance losses or even complete failures of the real-time systems depending on the types of systems(Wang et al., 2020).In addition, when the VMs are handled improperly during the migration, the device state error occurs; this results in improper allocation of tasks to the VM in update/migration. Hence, for solving this issue, this work proposes the R-EDF task scheduling based on DRT-MLP with the LC-GTO virtual machine selection.

1.1 Problem statements

The downsides in existing research works for TS and VM selection are given below:

- In TS, asignificantproblem of device state errors occurring during the VM backup or migration is not considered in prevailing works. This error affects the balanced schedulers' performance.
- The energy consumption and cost consumption increased when the prevailing models assigned the task to the VM without considering the entire task strategy.
- In existingapproaches, as the overloaded VMs acquired resources from the neighboring VMs, the energy of other VMs in a cloud got depleted when a VM was overloaded with tasks.
- In prevailingstudies, based on the less response time, distance,et cetera, the target VM was selected. However, in these works, improper scheduling may take place as vital VM factors, namely supported Ethernet and protocols were neglected.

By considering these limitations, developing an energy-efficient and load-balanced dynamic TS approach in the IoT cloud is the proposed system's objective. The proposed system's contributions are given below:

- The availability of the VM is monitored using the DRP-MLP technique to minimize the device state error problem.
- To provide energy-efficient TS, the Ch-KPA-based task grouping is done, where the sequential and parallel strategy tasks are scheduled simultaneously. This assures energy efficiency and a balanced task load in the proposed dynamic TS model.
- For scheduling and load balancing the tasks to the VM resources, the R-EDF scheduler with LC-GTO is proposed to avoid overloading of the VMs.
- To find the accurate resource-providing VM, the LC-GTO is proposed forchoosing the optimal VMs centered on the cost and waiting time, along with the supporting Ethernet protocol.

This paper's remaining part is arranged as: the proposed dynamic TS model's related studies are explained in section 2, the proposed methodology for energy-efficient TS is explicated in section 3, the proposed system's experimental outcomes in contrast to the prevailing and conventional techniques are given in section 4, and the paper is concluded with a future work suggestion in section 5.

2. RELATED WORKS

(Ding et al., 2020)propounded a Q-learning-centric dynamic TS for energy-effective cloud computing. Initially, for the queuing model, a centralized task dispatcher was utilized.Then, grounded on the priority of task laxity and task lifetime, Q-learning scheduled the tasks. The experimental analysis exhibited the presentedsystem's superior performance. Still, in the presented model, the fresh actions of task strategy couldn't be identified accurately with the Q-Learning.

(Alsadie, 2021)implemented a Meta-heuristic technique for Dynamic Virtual Machine Allocation (MDVMA) with optimized TS in cloud data centers. To develop a TS module, the crossover and mutation operation grounded on the genetic algorithm was established.As per the outcomes, the implemented algorithm's energy usage was minimized than the prevailing approaches. However, without considering the structure, the tasks were scheduled; this resulted in less load-balancing efficiency.

(Zhu et al., 2021) explored a self-adapted TS algorithm for container cloud grounded on learning automata. Through the scheduling experience accumulation, the algorithm improved the correlation between tasks and the running environment. Centered on the resource residual degree, the system's effectiveness was proved by the experimental outcomes. The experimental outcomes proved the system's effectiveness grounded on the resource residual degree. However, the developed system couldn't render optimal TS as the cloud resources' heterogeneity was neglected.

(Dubey & Sharma, 2021)explained a TS algorithm with deadline constraints in cloud computing. The hybridized Chemical Reaction optimization and Particle Swarm Optimization (CR-PSO) were leveraged for the TS. The CR-PSO system's efficiency was exposed by the experimental outcomes. The system's energy efficiency was decreased since the task strategy wasn't considered.

(Tuli et al., 2020)recommended a shared data-aware dynamic TS scheme for data-intensive applications on a hybrid cloud. Centered on the Aneka dynamic resource provisioning service, the dynamic tasks were scheduled. The suggested system proved its efficiency centered on the execution time. However, for large-scale applications, the system wasn't suitable.

(Teylo et al., 2021)propounded a dynamic task scheduler for resisting multiple hibernations in the cloud environment. It developed hibernation-aware static and dynamic schedulers with deadline constraints. Two modules, such as the primary scheduling heuristic module and the dynamic scheduler module were comprised in the Hibernation Aware Dynamic Scheduler. The outcomes proved the propounded Hibernation Aware Dynamic Scheduler model's effectiveness regarding minimized monetary costs. But, a work-stealing procedure mitigated the overloaded tasks; this drained the corresponding resource-stolen VM's energy.

(Jing et al., 2021) suggested a Quality-of-Service (QoS)-aware TS scheme for medical cloud platforms. For the scheduling of the task in the cloud environment, the discrete Particle Swarm Optimization (PSO) algorithm was used.According to the experimental outcomes, the propounded approach achieved the expected QoS. But, the recommended system couldn't schedule the tasks generated periodically.

(Muniswamy & Vignesh, 2022)established a hybrid optimal and deep learning algorithm for dynamic scalable TS in the cloud environment. The Modified Pigeon-inspired optimization assured priority-centric scheduling. Subsequently, the rapid adaptive feedback Recurrent Neural Network was developed for pre-virtual allocation. The outcomes exposed the established system's efficiency centered on the resource imbalance degree. Nevertheless, for attaining diverse objectives, separate classifier models were developed; this elevated the established model's response time.

(Sanaj & Joe Prathap, 2020) presented a TS scheme in the cloud environment. Grounded on the Enhanced version of the Round Robin (ERR) algorithm, the TS was done. As per the implementation outcomes, other prevailing algorithms were surpassed by the ERR algorithm regarding execution time. But, the ERR doesn't concentrate on the deadline of the tasks by which deadline-crossed tasks couldn't be scheduled efficiently.

(Guo, 2021)established a fuzzy self-defense algorithm for TS optimization in cloud computing. Centered on multi-objective TS, the objective function for the fuzzy self-defense algorithm was developed. As per the outcome, the developed algorithm utilized fewer resources of the VM. But, since the supporting factors of the VM like supported Ethernet and protocol were not considered, the scheduling efficiency was reduced.

3. PROPOSED METHODOLOGY FOR DYNAMIC TASK SCHEDULING IN THE CLOUD

TS plays a vital role in maintaining the cloud resources' energy. But, the prevailing models failed for developing energy-effective scheduling without taking into account the input task strategy. Hence, this study proposes an R-EDF scheduler with the Ch-KPA task strategy identification.

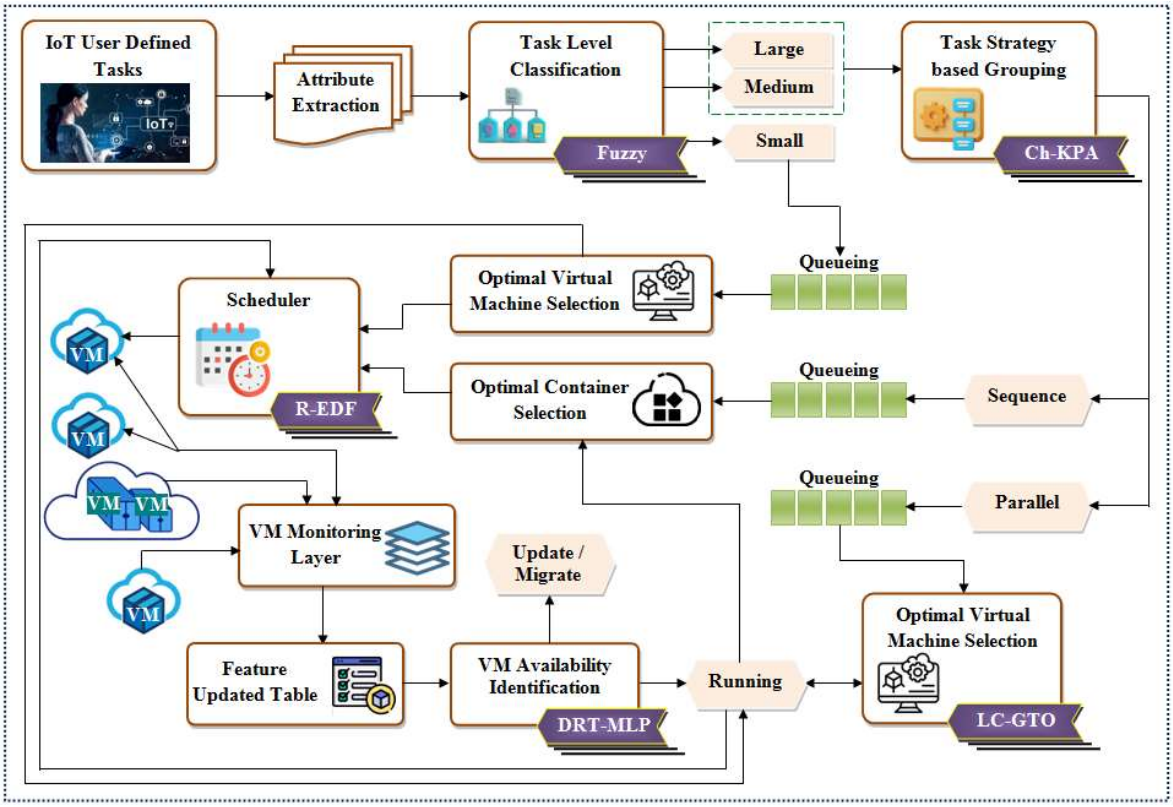

Figure 1: Architecture of the proposed model

3.1 Input data

The proposed system's input is the Bag of Tasks (BOT) (T_n) assigned by the IoT user to the cloud platform, which contains many VMs. This is mathematically expressed as,

$$T_n = \{T_1, T_2, ..., T_z\} \tag{1}$$

Here, T_z definesthe z^{th} subtask.

3.2 Attribute extraction

Next, task attributes like task length (T_l), independence, granularity, locality, and device attributes like bandwidth, power, battery information, and storage are extracted from the tasks (T_n). The extracted features are signified as,

$$F_x = \{F_1, F_2, ..., F_m\} \tag{2}$$

Here, the m^{th} feature extracted is given as F_m.

3.3 Task-level classification

After the attributes are extracted, based on the length of the task (T_l), the given task $(T_n)$is classified. For the classification of the tasks using the trapezoidal membership function, fuzzy rules are utilized.

Using the trapezoidal membership function, the fuzzifierfuzzifies the crisp input (T_l) as,

$$small = \begin{cases} 1 & if\left(T_l \le p\right) \\ \dfrac{T_l - q}{p - q} & if\left(p < T_l < q\right) \end{cases} \tag{3}$$

$$medium = \begin{cases} 1 & if\left(q \le T_l \le r\right) \\ \dfrac{T_l - s}{s - r} & if\left(r < T_l < s\right) \end{cases} \tag{4}$$

$$l\arg e = \begin{cases} 1 & if\left(T_l \ge r\right) \\ \dfrac{T_l - r}{s - r} & if\left(r < T_l < s\right) \end{cases} \tag{5}$$

Here, the task sizes are depicted as $small, medium, l\arg e$, the bound values of the fuzzy membership are signified as p, q, r, s, which are the threshold value to map the Degree Of Membership (DOM) of small, medium, and large tasks.

After the DOM is acquired, $IF-THEN$ statements aggregate the tasks. The $\max$ centered aggregation $\left(\aleph_{agg}\right)$ is done, which is expressed as,

$$\aleph_{agg} = \max\left(small, medium, l\arg e\right) \tag{6}$$

Lastly, the aggregated values are defuzzified, and the crisp output c_{agg}, which is small, medium, or large task class, is acquired.

The task is allotted to a single VM's queue if the given c_{agg} is a small task.

3.4 Task strategy-based clustering

However,if the classified task is a medium or large task, the tasks are grouped into sequential and parallel tasks grounded on the strategy and subsequently scheduled. In this, the Ch-KPA is leveraged for the grouping. The K-Prototype Algorithm (KPA) is chosen owing to its efficiency in grouping categorical and numerical data. But, hamming distance can't accurately express the degree of dissimilarity between the categorical attribute and the center. Hence, for solving this issue, this algorithm introduces the chessboard distance.

In this, for grouping the sequential and parallel tasks, the attribute set F_x is given to theCh-KPA.

Cluster-center selection: Thenumber of clusters (p) with $k-$ cluster centers is initialized for the given input F_x. The $k-$ cluster centers are randomly initialized in Ch-KPA. The initialized cluster centroids are expressed as,

$$C_n = C_i^{cat}, C_i^{num} \tag{7}$$

Here, the random cluster centers for the categorical and numerical attributes are indicated as C_i^{cat}, C_i^{num}.

Distance estimation: After the centroids are chosen,the distance (D) between the centroid point $\left(C_i^{cat}\right)$ and the other data points $\left(d_j \in F_x\right)$ iscalculated as,

$$D_{i,j} = \sqrt{\sum_n \left(C_i^{num} - \hat{d}_j\right)^2} \tag{8}$$

Here, the data points' mean value is signified as $\hat{d}_j$.

Dissimilarity estimation: Subsequently, the dissimilarity difference for the categorical data (α_j) is calculatedutilizing the Chessboard distance as,

$$\Im_{i,j} = \max_n \left| C_i^{cat} - \alpha_j \right| \tag{9}$$

Assign clusters: Lastly, $D_{i,j} + \Im_{i,j}$ is calculated with every $k-$clusters, and the object is assigned to a cluster that contains a low overall difference. Hence, the final cluster acquired is indicated as,

$$\beta_k = \left[\beta_{seq}, \beta_{par}\right] \tag{10}$$

Here, the sequential and parallel task clusters are symbolized as β_{seq}, β_{par}.

3.5 Queuing

As each task is assigned to different kinds of resources, the tasks are added to the different queues after the tasks are grouped. The queued tasks are given as,

$$Q_{sm} = \left[t_1, t_2, ..., t_\ell\right] \tag{11}$$

$$Q_{sq} = \left[\tau_1, \tau_2, ..., \tau_q\right] \tag{12}$$

$$Q_{pl} = \left[\varsigma_1, \varsigma_2, ..., \varsigma_{\aleph}\right] \tag{13}$$

Here, the queues of small, sequential, and parallel tasks are signified as Q_{sm}, Q_{sq}, Q_{pl}. The final tasks added to the queue are depicted as t_ℓ, τ_q, and $\varsigma_{\aleph}$.

3.6 Optimal VM selection

Next,the optimal VM selection takes place for mapping the task to the optimal resources. In this, a single optimal VM is chosen for the small tasks as the small process can be completed in a single VM. Here,LC-GTO is utilized for the VM selection. The GTO is chosenowing to its efficiency to providea global optimal solution. However, the random coefficient in the search decreases the exploration efficiency, which induces slow convergence. Hence,in the GTO, aLogistic Chaotic map (LC) is introduced.

Population Initialization: In the CSP, the population of the giant trevally is considered as the VMs, which is indicated as,

$$G = \begin{bmatrix} G_1 \\ \vdots \\ G_\delta \\ \vdots \\ G_q \end{bmatrix} = \begin{bmatrix} g_{1,1} & \cdots & g_{1,\wp} & \cdots & g_{1,\aleph} \\ \vdots & & \vdots & & \vdots \\ g_{\delta,1} & \cdots & g_{\delta,\wp} & \cdots & g_{\delta,\aleph} \\ \vdots & & \vdots & & \vdots \\ g_{q,1} & \cdots & g_{q,\wp} & \cdots & g_{q,\aleph} \end{bmatrix} \tag{14}$$

Here,the solution for LC-GTO is signified as G, the size of the population and the dimension are indicated as $q, \aleph$, and the value of the $\wp^{th}$ task signified by the δ^{th} candidate solution is depicted as $g_{\delta,\wp}$. Next, the position to each trevally is assigned by,

$$G_{(\delta,\wp)} = B_{\wp}^{low} + \left[B_{\wp}^{up} - B_{\wp}^{low}\right] \times \gamma \tag{15}$$

Here, the lowest and highest value that a task can have is signified as $B_{\wp}^{low}, B_{\wp}^{up}$. A random number is depicted as γ.

Fitness estimation: After the positions are defined, the trevalliesare stored in a vector as,

$$\Psi = \begin{bmatrix} \Psi_1 \\ \vdots \\ \Psi_{\delta} \\ \vdots \\ \Psi_q \end{bmatrix} = \begin{bmatrix} r(G_1) \\ \vdots \\ r(G_{\delta}) \\ \vdots \\ r(G_q) \end{bmatrix} \tag{16}$$

Here, the δ^{th} member fitness is indicated as $\Psi_{\delta} = r(G_{\delta})$, and the fitness vector is given as Ψ, which is expressed as,

$$\Psi = \langle \min(wt, tcc), \max(dlb), eq(et, pr) \rangle \tag{17}$$

Where, the fitness is considered as minimum waiting time (wt), minimum task completion task (tcc), the maximum degree of load balance (dlb), equivalent Ethernet (et), and protocol (pr). Primarily, the prey searching strategy utilizing levy flight takes place, by which their movement is updated as,

$$G(\lambda+1) = G^{best} \times \gamma + \left[\left(B^{up} - B^{lw}\right) \times \gamma + B^{low}\right] * \left(0.01 \times \frac{\upsilon \times \mu}{\sqrt[\omega]{|u|}}\right) \tag{18}$$

Here, the best position is signified as G^{best}, the iteration is given as λ, and the levy flight distribution function is indicated as ω. υ, u are the numbers that are determined utilizing LCand μ is the variance, which is given as,

$$u\log_{(\lambda+1)} = b \times u\log_{(\lambda)}\left(1 - u\log_{(\lambda)}\right) \tag{19}$$

$$\mu = \frac{\Re(1+\omega) \times \sin\left(\frac{\omega\pi}{2}\right)}{\Re\left(\frac{1+\omega}{2}\right) \times \omega \times 2^{\left(\frac{\omega-1}{2}\right)}} \tag{20}$$

Here, the chaotic map variable is depicted as b, and the control parameter to control the levy flight is signified as $\Re$.

Next, during the right area of hunting, the trevally determines the best area for its prey and food, which is indicated as,

$$G(\lambda+1) = G^{best} \times \gamma \times W + I_{mean} - G_{\delta}(\lambda) \times \gamma \tag{21}$$

Here, the position vector of the giant trevallies in the successive iteration is signified as $G(\lambda+1)$, the position change controlling parameter is depicted as W, andthe previous information's average is given as I_{mean}, which is indicated as,

$$I_{mean} = \frac{1}{q}\sum_{\delta=1}^{q} G_{\delta}(\lambda) \tag{22}$$

Subsequently,the position with the best fitness value is selected as the optimal solution (i.e. VM) by calculating the fitness of the position $G(\lambda+1)$ of trevallies, and it is depicted as V_o. Likewise, the optimal VM in multiple clouds is chosenfor the parallel tasks in the queue (Q_{pl}) and is signified as $MV_{\varphi}, \varphi = 1,2,...g$.

Pseudocode of LC-GTO

Input: VMs in the cloud
Output: Optimally selected VM
Begin

Initialize population, γ, u, υ, maximum iteration $(\lambda_{\max})$
Set iteration $\lambda = 1$
While $(\lambda \le \lambda_{\max})$ **do**
    Calculate fitness Ψ
    Perform Prey searching
    Update position using $G^{best}, u\log_{(\lambda+1)}, \mu$
    Update position to best food area $G^{best} \times \gamma \times W + I_{mean} - G_{\delta}(\lambda) \times \gamma$
    If $(\Psi(G(\lambda+1)) > \Psi(G(\lambda)))${
        Return $G(\lambda+1)$ as optimal solution
    } **Else** {
        Repeat while
    End If
End While
$\lambda = \lambda + 1$
Return output V_o
End

3.7 Optimal container selection

For the sequential tasks in the queue (Q_{sq}), based on the fitness values of $\min(wt, tcc), \max(dlb)$, the optimal containers are selected.In this, the population initialized isexpressed as,

$$\ell_j = \{\ell_1, \ell_2, ..., \ell_v\} \tag{23}$$

Where, the v^{th} cloud container is considered as ℓ_v, and the given population goes through the steps from (14) to (22) for finding the optimal container; this is indicated as Ω_o. In this, for sequential tasks, the container is selected sincethe container maps the tasks in the VMs in it if a sequential is assigned for a single cloud container that contains multiple VMs. Similarly, the waiting time for tasks is minimized.

3.8 VM availability identification

For the energy-efficient TS, the checking availability of the VM takes an important place as the selected VM might be in the updation or migration state that led to the device state error. In this, the DRT-MLP is used for checking the availability of the VM. The Multi-Layer Perceptron (MLP) is chosen owing to its large data handling efficiency. However, overfitting and vanishing gradient issues are led by a greater number of layers. Hence, in the MLP, the Drop-connect regularization layer and Random translation activation are introduced.

Input: The feature-updated table (T) that is provided by the cloud brokers' VM monitoring layer is the input of the DRT-MLP. T contains the features of the VM in the cloud,namely state (i.e., active, running, updated), supported Ethernet, supported protocol, bandwidth, memory, and type of resource available. These features are depicted as,

$$R_c = \{R_1, R_2, ..., R_y\} \tag{24}$$

Here, the y^{th} feature in the feature table is signified as R_y. These feature values are dynamically updated in the tablecentered on which the DRT-MLP predicts whether the VM is in the running state or under updation or migration. Figure 2 shows the proposed DRT-MLP network architecture.

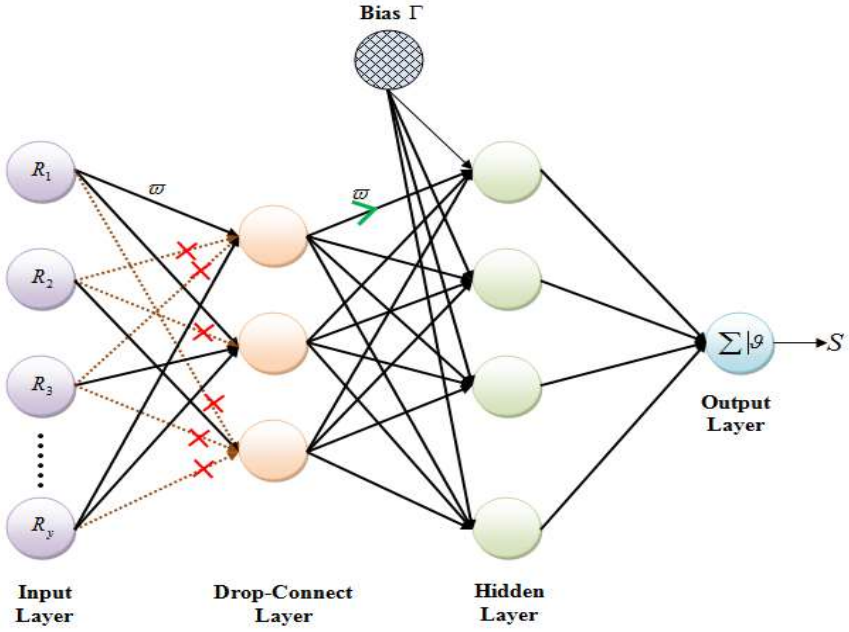

Figure 2: Neural architecture of DRT-MLP

Input layer: In this, the feature-updated table R_c is given to the input layer. Here, the input is prepared by the input layer for further processing by the other layers.

Drop-connect layer: After the input layers process the features of R_c, for the regularization of the input, the results are given to the drop-connect layer. The drop connect layer process is expressed as,

$$\eta_c = \vartheta((E \times \varpi) R_c) \tag{25}$$

In this, the binary matrix coding encoding the connection information is signified as E, the weight value of the neurons is indicated as ϖ, and theRandom translation activation is given as ϑ, which is depicted as,

$$\vartheta(R_c)=\begin{cases}R_c+h & if\ (R_c+h)>0\\ 0 & else\end{cases} \tag{26}$$

Here, the random translation constant is depicted as h.

Hidden layers: From the drop-connect layer, the prepared η_c is given to hidden layers (Z_c), where hidden neurons process the features as,

$$\zeta_c=\vartheta\left(\sum_{\rho=1}^{\varepsilon}\eta_c^{\rho}\varpi+\Gamma\right) \tag{27}$$

Where, the hidden ρ^{th} neuron of the c^{th} hidden layer is symbolized as η_c^{ρ}, the bias value is signified as Γ, and the total number of hidden neurons is given as ε. The activation of hidden neuronsis expressed as,

$$\vartheta(\zeta_c)=\begin{cases}\zeta_c+h & if\ (\zeta_c+h)>0\\ 0 & else\end{cases} \tag{28}$$

Output layer: The processed features from the hidden neurons are summed up and given to the output layer for predictingthe corresponding input's output class (S). The DRT-MLP's output is expressed as,

$$S=\vartheta\left(\sum_{c=1}^{y}\varpi\zeta_c+\Gamma\right) \tag{29}$$

After the output value S is predicted, for finding the error in the classifier, the loss function (L) is calculated as,

$$L=\frac{1}{y}\sum_{c=1}^{y}(O_{out}-S) \tag{30}$$

Where, the threshold output is depicted as O_{out}. The output class will be considered if the estimated error value is less than or equal to the threshold (ψ), else the training continues by changing the weight values.

Pseudocode of DRT-MLP

Input: Feature updated table
Output: Identified status of VM
Begin
Initialize number of layers, parameters Γ, ϖ
If $(L\le\psi)$ {
Return final output S
} **Else** {
Update weights ϖ
Perform drop-connectfunction $\vartheta((E\times\varpi)R_c)$

Activate neurons with Random translation activation

$$\zeta_c = \vartheta\left(\sum_{\rho=1}^{\varepsilon}\eta_c^{\rho}\varpi + \Gamma\right)$$

Perform

Compute output S

}

Return final output S

End if

End

Whether the VM is in the process of task migration or updation or else in the active running state is predicted by the output class. The task is added to the queue of the selected VM if only the VM is in the running state. In this queue, the tasks are dynamically scheduled regarding the earliest deadline by the scheduler. The tasks added to the Queue of the VM V_o is given as,

$$X_V = [X_1, X_2, ..., X_U] \tag{31}$$

Here, the U^{th} task assigned to the VM V_o or MV_{φ} is depicted as X_U.

3.9 Dynamic scheduler

The earliest deadline task should be given high priority after the tasks are allotted to the VM. Hence, the R-EDF is proposed. In this, the Earliest Deadline First (EDF) scheduler is chosensince it is scheduledgrounded on the deadline. However, the EDF suffers from a priority inversion problem sometimes. Hence,for the correct prioritization of tasks, the Rice technique is introduced in the EDF.

In the RICE-EDF, the RICE technique gives the priority of the tasks in the queue X_V, which is expressed as,

$$\chi = (\mathrm{Re}.\mathrm{Im}.Co) \times Ef \tag{32}$$

Here, the Reach, Impact, Confidence, and Effort (RICE) scoresof the tasks in the queue are signified as $\mathrm{Re}, \mathrm{Im}, Co, Ef$. Next, the task with the maximum χ is arranged in a single set, and the remaining tasks are arranged in another set. Subsequently, by checking the criteria given below, the tasks are scheduled.

$$\Phi_V = \sum_{K=1}^{nn}\frac{Y_K}{J_K} + \chi \le 1 \tag{33}$$

Here, the earliest deadline task is depicted as Φ_V, and the worst-case computation time of nn processes and their relative deadlines are symbolized as Y_K, J_K.

Therefore, energy-effective and load-balanced dynamic TS is done by doing this process in the cloud simultaneously. The subsequent section experimentally proves this time efficiency and load balancing results.

4. RESULTS AND DISCUSSIONS

Here, the proposed system's performance is experimentally assessed in comparison to the prevailing approaches for proving its efficiency. On the working platform of JAVA, the experiments are done with the data collected based on real-time network deployment.

4.1 Performance analysis of the dynamic scheduler

Here, regarding acceptance rate, execution efficiency, and makespan, the proposed dynamic scheduler R-EDF's performance is experimentally investigated and verified in comparison with prevailing EDF, Max-Min, Shortest Job Next (SJN), and Round Robin (RR) schedulers.

Table 1: Makespan results of the proposed model

Techniques	Number of tasks				
	100	**200**	**300**	**400**	**500**
Proposed R-EDF	2568	3875	6358	5284	4258
EDF	4236	5986	8475	7456	6579
Max-Min	6124	7485	9658	9541	8423
SJN	8744	10874	12658	11847	10514
RR	9547	11549	12579	13658	12548

Table 1 displays the proposed R-EDF's comparative measure. For 100 tasks, the proposed R-EDF achieved a makespan of 2568, while the conventional EDF attained only 4236,which is comparatively higher than the proposed approach. Similarly, when compared with the proposed one, the makespan of the other techniques like Max-Min, SJN, and RR is also higher. The proposed technique's makespan increases as the number of tasks increases; but, it is not higher than the baseline techniques. Hence, in the dynamic TS in the cloud resources, the proposed R-EDF is better in performance than the other techniques.

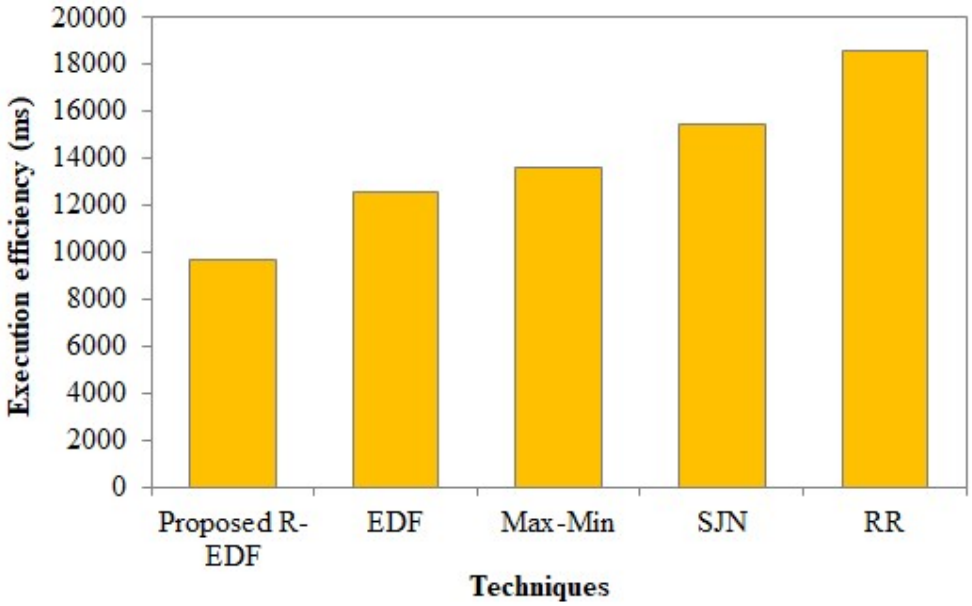

Figure 3: Execution efficiency analysis

Figure 3 reveals the proposed system's execution efficiency centered on execution time. In this, the proposed model takes an execution time of 12547ms with the EDF algorithm, which is

comparatively lesser than the traditional Max-Min, SJN, and RR schedulers. But, the proposed scheduler takes less than 9628ms, which is lower than the other baseline schedulers.This proves that the dynamic scheduling of tasks before the deadline completion is performed effectively with the R-EDF scheduler.

Table 2: Acceptance rate outcomes

Techniques	Acceptance rate (%)
Proposed R-EDF	98.32
EDF	94.12
Max-Min	89
SJN	87.49
RR	84.97

From Table 2, it is evident that the acceptance rate is enhanced by separately assigning tasks to multiple VMs simultaneously based on the earliest deadline first priority. In this, the acceptance rate for the proposed model is 98.32%, which is comparatively higher when analogized with the conventional EDF, Max-Min, SJN, and RR schedulers. This exhibits that the resources in the VMs are occupied without being overloaded, and avoided the device state error.

4.2 Performance analysis of optimal VM and container selection

Here, for proving the load balancing, fitness, throughput, energy consumption, latency, resource utilization, response time, and average waiting time efficiency, the selection algorithm LC-GTO is comparatively examined with the prevailing selection algorithms, namely GTO, ACO, PSO, and Reptile Search Algorithm (RSA).

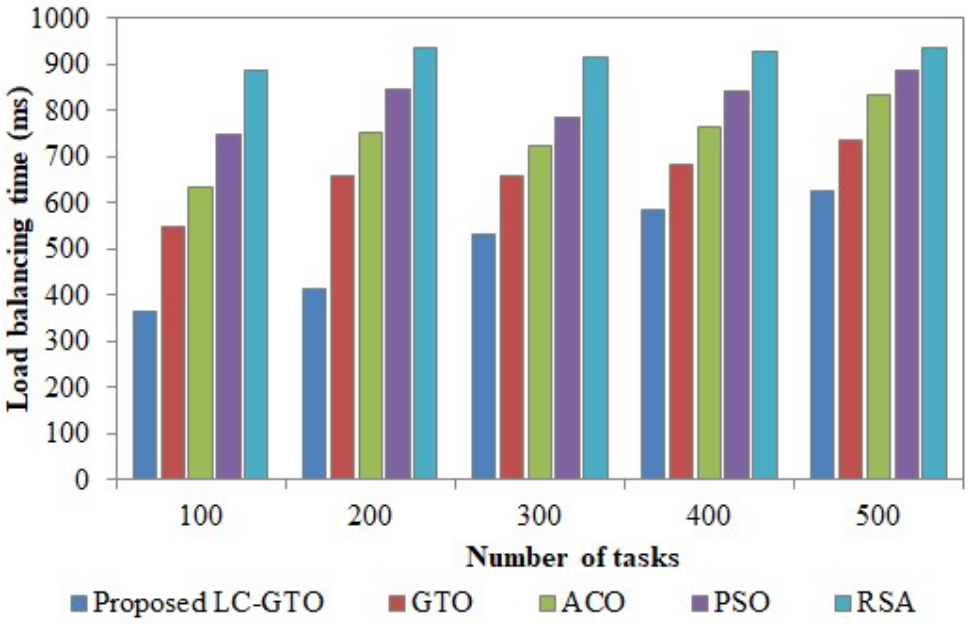

Figure 4: Performance analysis based on load balancing time

Figure 4 displays the time taken by the proposed and prevailing systems for balancing the task. For the 500 tasks, the proposed technique's load balancing time is 625.4ms. Similarly, for

balancing 100 to 400 tasks, the proposed model also obtained a lower time. On the other hand, for the existing methods like GTO, the load balancing time elevates in the range of 547.7ms to 735.1ms as the number of tasks elevates. Likewise, for other existing techniques also, the load balancing time varies (higher). This displays the system's superiority in managing the load of the tasks mapped to the VM resources.

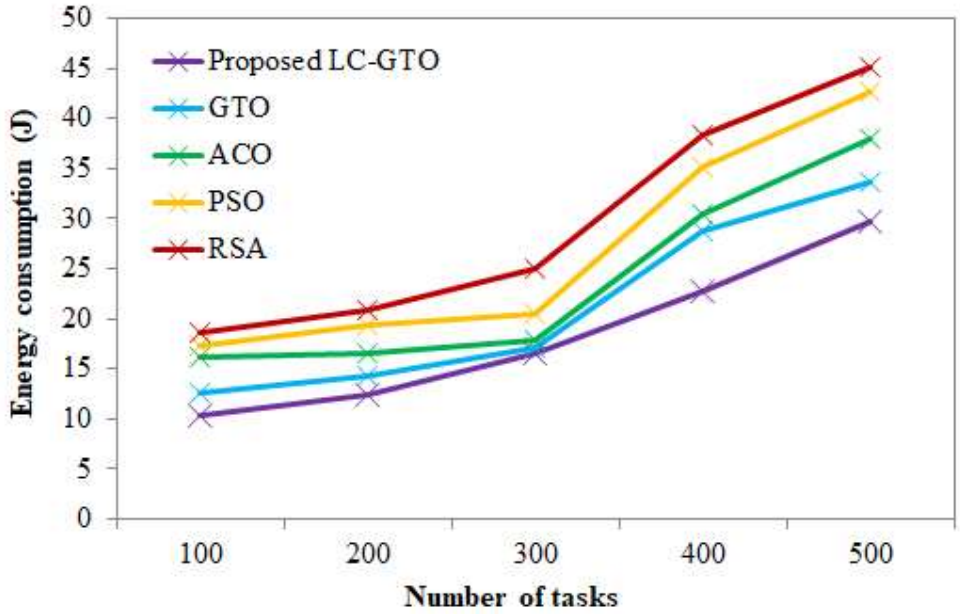

Figure 5: Energy consumption in the task scheduling

Figure 5 shows the energy consumed by various models during TS. In this, the proposed LC-GTO consumed 29.7J of energy for 500 tasks, while the GTO, ACO, PSO, and RSA approaches consumed 33.7J, 38J, 42.6J, and 45.2J of energy. Likewise, when compared to the prevailing algorithms, the proposed technique takes less energy for the other number of tasks also. This reveals that for scheduling the tasks, low energy is consumed with the LC-GTO technique.

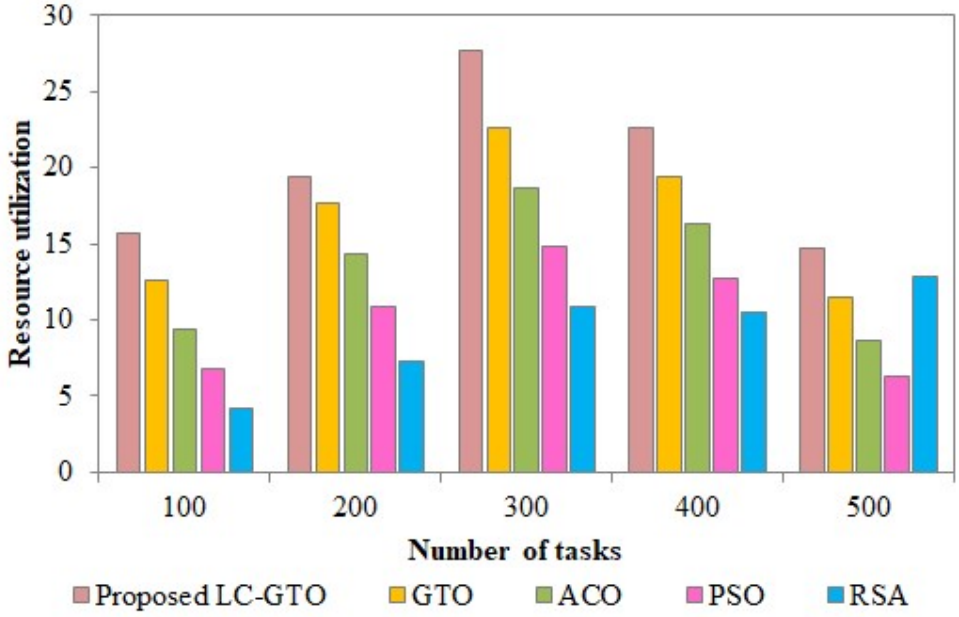

Figure 6: Resource utilization by existing and proposed techniques

The VMs' resource utilization by the proposed system is examined by analogizing it with existing scheduling algorithms. Figure 6 shows that when compared with conventional approaches for scheduling, the GTO used fewer resources. However, for the 100, 200, 300, 400, and 500 tasks, the LC approach in the GTO technique reduced the utilization to 4.23, 7.25, 10.84, 10.48, and 12.87, correspondingly. Hence, it is proved that the proposed scheduling system used fewer resources by which energy efficiency is attained.

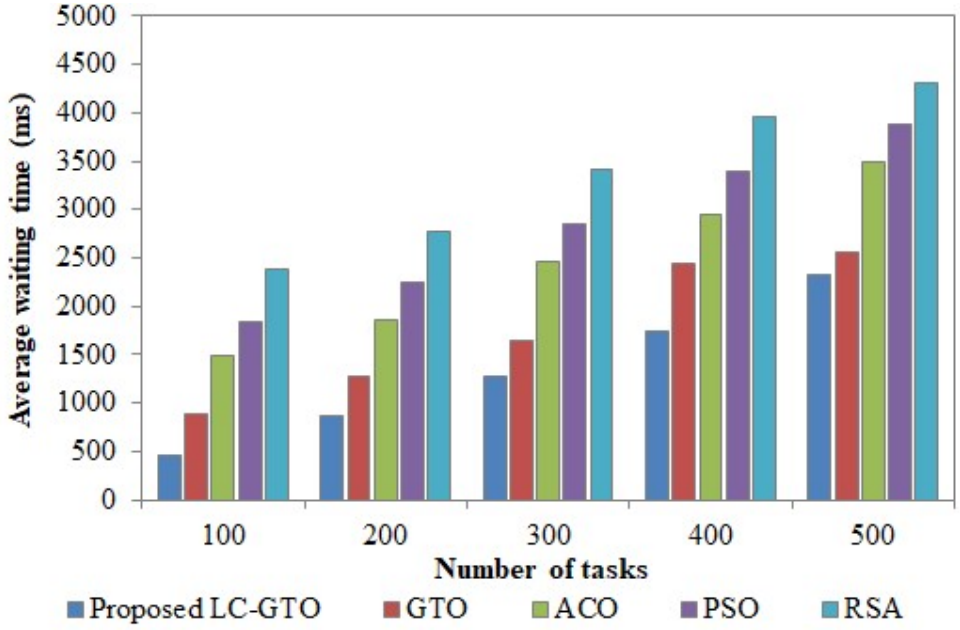

Figure 7: Average waiting time analysis

The proposed LC-GTO's average waiting time for scheduling various tasks is displayed in Figure 7. In this, for scheduling 500 tasks, the proposed approach waits only for 2317ms, while the prevailing GTO, ACO, PSO, and RSA wait for 2548ms, 3485ms, 3874ms, and 4312ms to assign the task to VM, correspondingly. This exhibits that when contrasted with prevailing approaches, the proposed model's average waiting time is very much lower. The above discussion shows that the proposed model with logistic chaotic mapping schedules the task more quickly than the prevailing approaches.

Table 3: Fitness Vs Iteration

Iteration	Fitness				
	RSA	PSO	ACO	GTO	Proposed LC-GTO
10	53	74	89	113	138
20	73	92	110	128	149
30	91	109	123	143	163
40	108	113	146	163	204

50	121	132	160	181	279

LC-GTO technique's performance regarding fitness vs. iteration is displayed in Table 3. For the proposed model, the fitness value is 138 and 149 while the no.of iteration is 10 and 20, respectively, and lastly, the fitness is 279 for 50; this exhibits the increasing capacity of the LC-GTO's performance. In addition, even a gradual increase is not exhibited by the prevailing works, which render low-range values for diverse iterations. This displays that the convergence issues are avoided by the usage of the logistic chaotic mapping in the proposed model; hence, the proposed study attains peak performance than the baseline approaches.

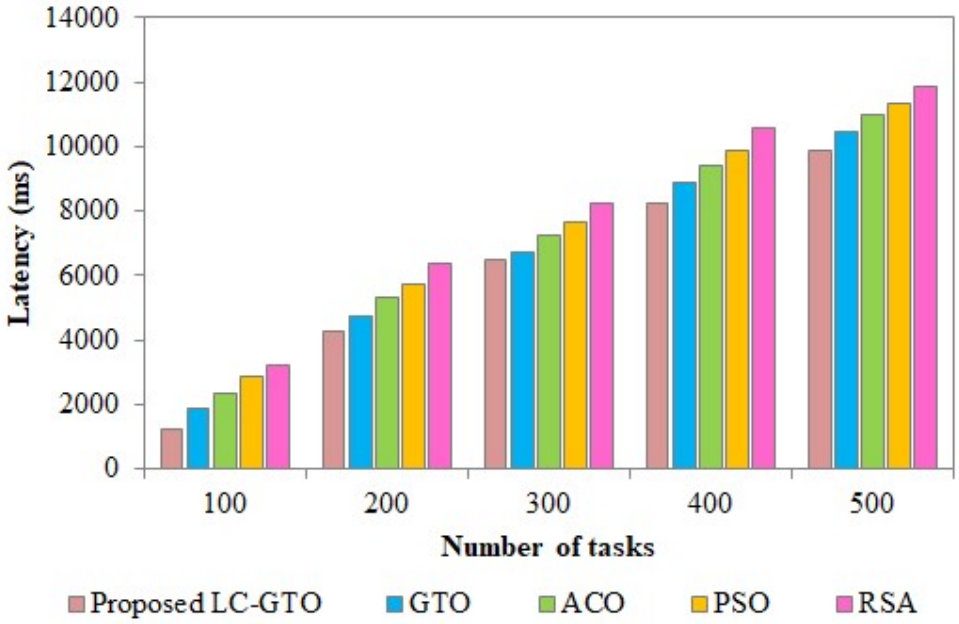

Figure 8: Experimental analysis based on latency

From Figure 8, it is clear that the proposed LC-GTO model exhibits a lower latency. For the 100 number of tasks, the proposed approach's latency is 1254ms. For the same 100 tasks, the prevailing approaches say GTO has a higher latency (1875ms). For the remaining number of tasks, the latency increases as the number of tasks increases; however, it is lower than the prevailing systems' latency. Hence, it is proved that when analogized with the prevailing approaches, the LC-GTO technique's performance is well.

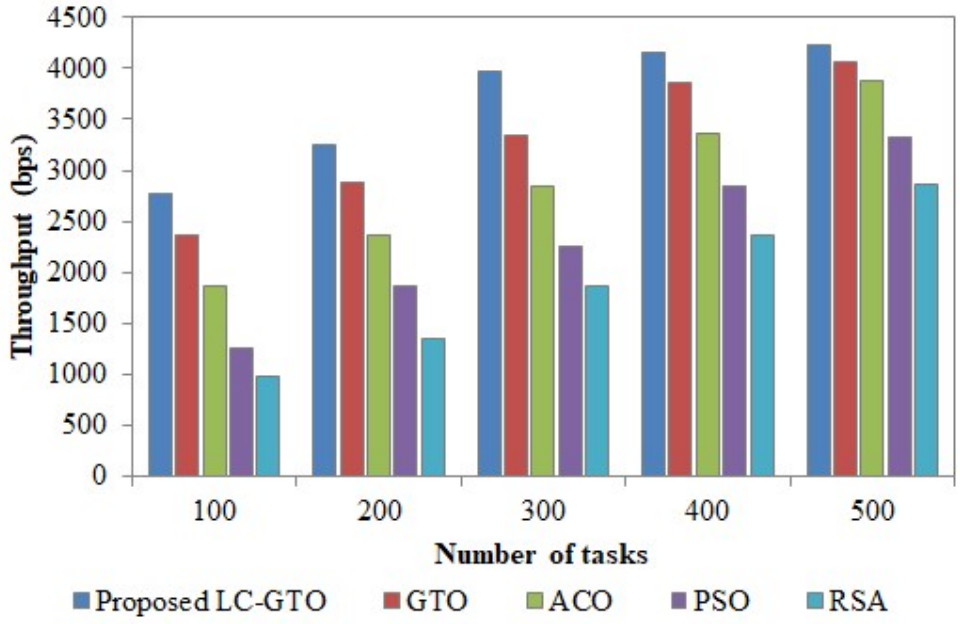

Figure 9: Throughput analysis

Figure 9 shows that when analogized with other prevailing approaches, the proposed system has higher throughput. For 100 tasks, the prevailing approaches, namely GTO (4065bps), ACO (3874bps), PSO (3326bps), and RSA (2859bps) acquired lower values, while the proposed LC-GTO's throughput is 4238bps. Similarly, the proposed approach's throughput is higher than the prevailing system for the remaining number of tasks.Hence, the overall analysis shows that in the selection of optimal VM selection for TS, the proposed model attains superior outcomes.

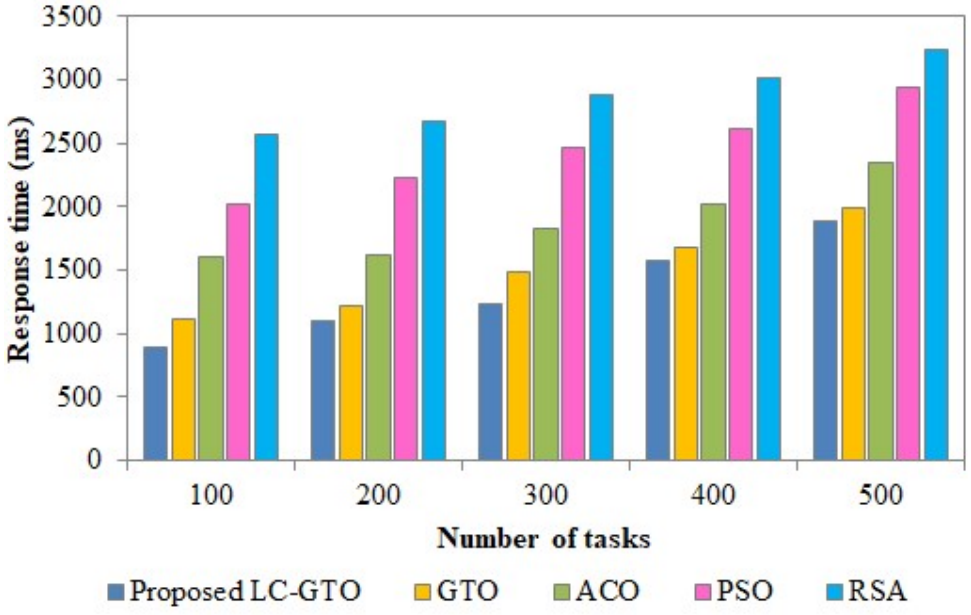

Figure 10: Response time for selecting the VM

Figure 10 calculates the proposed LC-GTO technique's response time for scheduling under a diverse number of tasks. The proposed LC-GTO's response time varies between 896ms and 1097ms for 100 to 200 tasks, while the prevailing GTO has a higher (1108ms to 1221ms) response time for scheduling the same 100 to 200 tasks. Similarly, for the rest of the approaches

also, the response time increases. This lower response time is owing to the modification of LC mapping to conventional optimization. Hence, when compared to the prevailing approaches, the proposed system attains a superior task-scheduling VM.

4.3 Performance analysis of grouping

Here, the proposed sequential and parallel task partition algorithm Ch-KPA approach is comparatively examined with the KPA, K-Means, K-Mode, and K-Medoid approachesfor proving its time efficiency.

Table 4: Comparative measure of proposed Ch-KPA

Techniques	Grouping time(ms)
Proposed Ch-KPA	1258
KPA	1937
K-Means	2054
K-Mode	2375
K-Medoid	2764

The grouping time taken by the proposed approach is depicted in Table 4. For grouping, the time taken by the proposed technique is only 1258ms (lower), while the time taken by the prevailing KPA(1937ms), K-Means (2054ms), K-Mode (2375ms), and K-Medoid (2764ms) are higher than the proposed HEM-KPA. Hence, it is proved that the sequential and parallel tasks are grouped in less time with the Ch-KPA approach by which the tasks can be efficiently assigned to multiple VM sources.

4.4 Performance analysis of VM availability recognition

Here, regarding the recognition error, recognition time, and Mean Square Error (MSE), the proposed DRT-MLP algorithm is comparatively analyzed and verified.

Table 5: Comparative measure of proposed DRT-MLP

Techniques	Recognition Error(RE)	MSE	Recognition time
Proposed DRT-MLP	0.656	0.4345	5334
MLP	1.578	0.6732	7274
CNN	2.786	0.7125	9468
DNN	5.786	0.7963	13527
ANN	8.567	0.9872	15647

The comparative measure of the proposed R-EDF is elucidated in Table 5. In this, by attaining lower error rates of the order of 0.656 (RE) and 0.4345 (MSE), the proposed approach performs better. Similarly, the error values attained by traditional techniques, namely MLP, Convolutional Neural Network (CNN), Deep Neural Network (DNN), and Artificial Neural Network (ANN) are higher and range between 1.578-8.567 (RE) and 0.6732-0.9872 (MSE). In addition, the VM's availability is examined in a lesser time (5334ms) than the other contrasted algorithms. Hence, it is clear that in the proposed system, the VM status is identified with less error and within less time.

4.5 Comparative analysis

Here, the proposed model's efficiency is verified by analogizing the energy efficiency with the prevailing works of (Ding et al., 2020), (Alsadie, 2021), and (Dubey & Sharma, 2021).

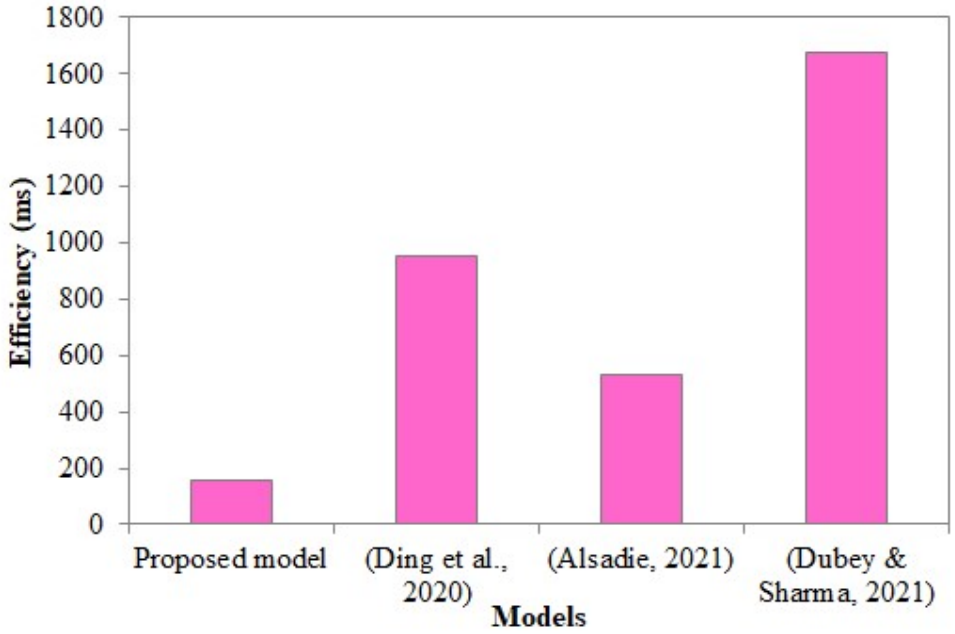

Figure 11: Comparative analysis

In the proposed system, the tasks are separated centered on their strategy, and then scheduled to the optimally selected cloud resources simultaneously. Hence, when compared with the prevailing works, the proposed system's time efficiency (154.64ms) is better. As per the proposed model's time efficiency, the tasks are dynamically scheduled to the cloud resources in load load-balanced and energy-efficient manner. This displays the proposed model's superiority over the other models for dynamic TS.

5. CONCLUSION

In this work, centered on the recognition of the availability of the VM status, a dynamic task scheduling in the optimally selected cloud resources is proposed. In this, utilizing the Ch-KPA technique, the strategy is analyzed and grouped as sequential and parallel tasks for energy-efficient TS. Next, the R-EDF technique with LC-GTO-based optimal VM and container selection takes place for efficient scheduling. Lastly, the proposed algorithms' performance is experimentally investigated and verified. During analysis, the proposed technique took less energy (29.7J) and resource utilization (14.65). Moreover, the proposed technique's reliability for dynamic TS is exhibited by the acceptance rate, load balancing time, and recognition error. Lastly, in the comparative study, the proposed system's dominance over prevailing systems regarding time efficiency is displayed. This study proposes the dynamic scheduling of the tasks based on the task strategy. However, the security issues in the IoT cloud are not considered in this work. Hence, in the future, for secure TS in the cloud, the security paradigms can be introduced together with the proposed system.

REFERENCES

Ali, A., Iqbal, M. M., Jamil, H., Qayyum, F., Jabbar, S., Cheikhrouhou, O., Baz, M., & Jamil, F. (2021). An efficient dynamic-decision based task scheduler for task offloading optimization

and energy management in mobile cloud computing. *Sensors*, *21*(13), 1-20. https://doi.org/10.3390/s21134527

Alsadie, D. (2021). A metaheuristic framework for dynamic virtual machine allocation with optimized task scheduling in cloud data centers. *IEEE Access*, *9*, 74218–74233. https://doi.org/10.1109/ACCESS.2021.3077901

Bal, P. K., Mohapatra, S. K., Das, T. K., Srinivasan, K., & Hu, Y. C. (2022). A Joint resource allocation, security with efficient task scheduling in cloud computing using hybrid machine learning Techniques. *Sensors*, *22*(3), 1-16. https://doi.org/10.3390/s22031242

Chen, X., Cheng, L., Liu, C., Liu, Q., Liu, J., Mao, Y., & Murphy, J. (2020). A WOA-Based Optimization Approach for Task Scheduling in Cloud Computing Systems. *IEEE Systems Journal*, *14*(3), 3117–3128. https://doi.org/10.1109/JSYST.2019.2960088

Ding, D., Fan, X., Zhao, Y., Kang, K., Yin, Q., & Zeng, J. (2020). Q-learning based dynamic task scheduling for energy-efficient cloud computing. *Future Generation Computer Systems*, *108*, 361–371. https://doi.org/10.1016/j.future.2020.02.018

Dubey, K., & Sharma, S. C. (2021). A novel multi-objective CR-PSO task scheduling algorithm with deadline constraint in cloud computing. *Sustainable Computing: Informatics and Systems*, *32*, 100605. https://doi.org/10.1016/j.suscom.2021.100605

Ebadifard, F., Babamir, S. M., & Barani, S. (2020). A dynamic task scheduling algorithm improved by load balancing in cloud computing. *6thInternational Conference on Web Research, ICWR 2020*, 177–183. https://doi.org/10.1109/ICWR49608.2020.9122287

Guo, X. (2021). Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm. *Alexandria Engineering Journal*, *60*(6), 5603–5609. https://doi.org/10.1016/j.aej.2021.04.051

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*, 100841. https://doi.org/10.1016/j.swevo.2021.100841

Iftikhar, S., Ahmad, M. M. M., Tuli, S., Chowdhury, D., Xu, M., Gill, S. S., & Uhlig, S. (2023). HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments. *Internet of Things*, *21*, 100667. https://doi.org/10.1016/j.iot.2022.100667

Jing, W., Zhao, C., Miao, Q., Song, H., & Chen, G. (2021). QoS-DPSO: QoS-aware Task Scheduling for Cloud Computing System. *Journal of Network and Systems Management*, *29*(1), 1–29. https://doi.org/10.1007/s10922-020-09573-6

Lu, W., Li, B., & Wu, B. (2019). Overhead aware task scheduling for cloud container services. *Proceedings of the 2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design, CSCWD 2019*, 380–385. https://doi.org/10.1109/CSCWD.2019.8791871

Harshavardhana Doddamani, Dr. Shantakumar B. Patil , Dr. Premjyoti Patil ,"A Framework For Design And Development Of Message Sharing Using Open-Source Software" (ISSN: 1006-3080) Journal of East China University of Science and Technology, 65(3),741-759. Page No:741-759, Retrieved from https://hdigdxxb.info/index.php/JE_CUST/article/pdf/741.pdf, Volume 65, Issue 3, 2022 (ISSN: 1006-3080)

Marahatta, A., Pirbhulal, S., Zhang, F., Parizi, R. M., Choo, K. K. R., & Liu, Z. (2021). Classification-based and energy-efficient dynamic task scheduling scheme for virtualized

cloud data center. *IEEE Transactions on Cloud Computing*, *9*(4), 1376–1390. https://doi.org/10.1109/TCC.2019.2918226

Muniswamy, S., & Vignesh, R. (2022). DSTS: A hybrid optimal and deep learning for dynamic scalable task scheduling on container cloud environment. *Journal of Cloud Computing*, *11*(1), 1-19. https://doi.org/10.1186/s13677-022-00304-7

Murad, S. A., Muzahid, A. J. M., Azmi, Z. R. M., Hoque, M. I., & Kowsher, M. (2022). A review on job scheduling technique in cloud computing and priority rule based intelligent framework. *Journal of King Saud University - Computer and Information Sciences*, *34*(6), 2309–2331. https://doi.org/10.1016/j.jksuci.2022.03.027

Rawas, S. (2021). Energy, network, and application-aware virtual machine placement model in SDN-enabled large scale cloud data centers. *Multimedia Tools and Applications*, *80*(10), 1-23, 15541–15562. https://doi.org/10.1007/s11042-021-10616-6

Sanaj, M. S., & Joe Prathap, P. M. (2020). An enhanced round robin (ERR) algorithm for Effective and Efficient Task Scheduling in cloud environment. *Proceedings - 2020 Advanced Computing and Communication Technologies for High Performance Applications, ACCTHPA 2020*, 107–110. https://doi.org/10.1109/ACCTHPA49271.2020.9213198

Shyalika, C., Silva, T., & Karunananda, A. (2020). Reinforcement learning in dynamic task scheduling: A review. *SN Computer Science*, *1*(6), 1-17. https://doi.org/10.1007/s42979-020-00326-5

Teylo, L., Arantes, L., Sens, P., & Drummond, L. M. A. (2021). A dynamic task scheduler tolerant to multiple hibernations in cloud environments. *Cluster Computing*, *24*(2), 1051–1073. https://doi.org/10.1007/s10586-020-03175-2

Harshavardhana Doddamani, Dr. Shantakumar B. Patil, Dr. Premjyoti Patil , "Ensemble Machine Learning Model for the Risk Prediction in Open Source Software Development Lifecycle", NeuroQuantology|August 2022|Volume20|Issue10|Page 8249-8258|doi:10.14704/nq.2022.20.10.NQ55810
https://www.neuroquantology.com/media/article_pdfs/8249-8258.pdf

Tuli, S., Sandhu, R., & Buyya, R. (2020). Shared data-aware dynamic resource provisioning and task scheduling for data intensive applications on hybrid clouds using Aneka. *Future Generation Computer Systems*, *106*, 595–606. https://doi.org/10.1016/j.future.2020.01.038

Velliangiri, S., Karthikeyan, P., Arul Xavier, V. M., & Baswaraj, D. (2021). Hybrid electro search with genetic algorithm for task scheduling in cloud computing. *Ain Shams Engineering Journal*, *12*(1), 631–639. https://doi.org/10.1016/j.asej.2020.07.003

Wang, Y., Zhou, K., Wang, Z., Li, M., Chen, N., Li, B., & Tian, H. (2020). Research on real-time embedded software scheduling model based on EDF. *IEEE Access*, *8*, 20058–20066. https://doi.org/10.1109/ACCESS.2020.2969229

Wei, X. (2020). Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing. *Journal of Ambient Intelligence and Humanized Computing*, 1-12. https://doi.org/10.1007/s12652-020-02614-7

Zhang, Y., Tang, B., Luo, J., & Zhang, J. (2022) Deadline-Aware Dynamic Task Scheduling in Edge–Cloud Collaborative Computing, *Electronics,11*(15), 1-24. https://doi.org/10.3390/electronics11152464

Zhu, L., Huang, K., Hu, Y., & Tai, X. (2021). A self-adapting task scheduling algorithm for container cloud using learning automata. *IEEE Access*, *9,* 81236-8252. https://doi.org/10.1109/ACCESS.2021.3078773