AN ENHANCED STARFISH OPTIMIZED VIRTUAL MACHINE MIGRATION FOR IMPROVING LOAD BALANCING IN FOG-CLOUD ENVIRONMENTS

Charles Mahimainathan A[1] And Suganya M [2]

[1]Department of Computer Science, RVS College of Arts and Science, Sulur, Coimbatore, Tamil Nadu, India
[2] School of Computer Studies, RVS College of Arts and Science, Coimbatore, Tamil Nadu, India

ABSTRACT

In this age of the Internet of Things (IoT), crucial management frameworks for large, dispersed systems must include fog and cloud computing technologies. By combining IoT with fog and cloud computing, massive amounts of IoT data can be processed in real time, meeting all of your processing needs. Nonetheless, optimizing Resource Allocation (RA) and Load Balancing (LB) in dynamic and varying operating settings continues to be a critical issue. Many traditional RA-LB techniques in IoT-fog-cloud systems frequently experience dynamic and unpredictable workload variations, leading to the overloading of some Physical Machines (PMs) and the underutilization of others. This disparity might result in high energy usage and resource scarcity in fog-cloud data centres. This study presents a novel Optimized Virtual Machine (VM) Migration-based RA-LB (OVM2-RALB) technique utilizing the Enhanced Starfish Optimization Algorithm (ESOA) for IoT-fog-cloud systems. To drastically cut down on energy consumption in fog-cloud data centers, the main objective is to dynamically move VMs from overcrowded PMs to underused ones. At first, the incoming job is categorized as eitherfog-dependent or cloud-dependent based on its guaranteed ratio. This ratio is decided by available PMs and their corresponding VMs in fog and cloud. The mean load for each PM is computed, which is utilized to calculate the load balancing factor to identify overloaded and underloaded PMs. Then, the ESOA, which is an improvement upon the standard SOA by combining tent chaotic mapping and Logarithmic Spiral Reverse (LSR) learning, is adopted for the VM migration process. This ESOA seeks to choose the most suitable PM for VM migration. It also determines the most appropriate VM for migration based on migration expenses, load balancing factor, energy use, and bandwidth utilization. Furthermore, the selected VM in the overloaded PM is migrated to the chosen underloaded PM. Thus, this VM migration results in a balanced load and alleviates overload on the PM. Finally, simulation results show that this OVM2-RALB outperforms conventional RA-LB techniques in IoT-fog-cloud environments.

KEYWORDS

IoT-fog-cloud systems, Task scheduling, Resource allocation, Load balancing, VM migration, Starfish optimization

1. INTRODUCTION

to reduce latency and enhance connectivity [2,3]. Smaller enterprises regard cloud computing as an effective data management option that proficiently stores the vast datasets generated by IoT devices. Sending large amounts of different data to central cloud servers can cause delays and slower response times because of theremote connections among IoT devices as well as cloud datacentres [4].

Fog computing technologies effectively mitigate the constraints of cloud data centersand meet the latency demands of IoT devices. The utilization of network-edge services reduces the distance between IoT data sources while fog computing improves cloud computing's capabilities [5]. Fog devices are close to each other, which speeds up processing and makes applications respond faster while using less bandwidth. The computational capacity and storage of these devices are inferior to those present in cloud networks [6]. Cloud and fog computing each have their own set of pros and cons, so they can't satisfy the needs of data-intensive Internet of Things applications to their fullest extent. Hence, integration of fog and cloud models is essential for establishing an effective computing environment [7]. On the other hand, resource management between fog and cloud nodes in IoT applications is a significant challenge because of the diversity and poor coupling of fog devices. To combat this issue, many RA-LB techniques have been developed in recent years for an effective IoT-fog-cloud system [8]. For instance, LB was processed by combining Particle Swarm Optimization and Simulated Annealing [9] for distributing resources in fog-cloud systems. The LB, on the other hand, required continual monitoring of cloud resource statuses and fog, which might result in excessive energy use and communication overhead. Workload distribution between fog nodes and reduction for interaction and computational delays have been reduced by an optimization technique [10,11]. However, it consumed more energy as IoT networks expanded in scale and complexity. In [12], a One-to-One-based optimizer with a Priority and LB (O2O-PLB) algorithm was developed for resource allocation in fog-cloud environments. However, load imbalance and task failure rate remained high.

Although these techniques enhance resource distribution efficiency, they have a common issue in high energy consumption. Dynamic and unpredictable workload fluctuations often cause certain PMs to be overloaded while others remain underutilized. In fog-cloud data centers, this imbalance can cause excessive energy consumption and wasteful use of resources. Therefore, it is essential to integrate the VM migration strategy to dynamically relocate workloads from overloaded PMs to underloaded ones, which savesenergy and avoids the usage of resources unnecessarily in IoT-fog-cloud environments.

This article develops the OVM2-RALB technique using the ESOA for IoT-fog-cloud systems. Initially, it categorizes the user tasks as either fog or cloud tasks according to the task guarantee ratio to the PMs and their respective VMs. After that, the mean load for all PMs is measured, which is utilized to determine the LB factor to discover overloaded and underloaded PMs. One of the solutions for balancing load among PMs is migrating VMs from overloaded PMs to underloaded ones. The proposed VM migration strategy executes this task by using ESOA, which is an enhancement to the existing SOA by incorporating tent chaotic mapping and LSR learning. This ESOA's overarching goal is to determine, from a set of potential PMs, which one is best suited to migrate virtual machines (VMs), taking into account criteria such as migration cost, energy consumption, load balancing, and bandwidth use. In addition, the selected virtual machine in the overworked PM is moved to the ideal underutilized PM. With the VM moved to the best PM, the PM is now load-balanced and not overloaded. When the fog-cloud is neither too full nor too empty, the virtual machine migration process terminates.

Here are the sections that follow: The research that is pertinent is covered in Section 2. Section 4 proves the effectiveness of the OVM2-RALB, whereas Section 3 explains it. Section 5 summarizes the work done and describes upcoming improvements.

2. LITERATURE SURVEY

This section explores recent studies related to VM migration in IoT-fog-cloud platforms. For dynamic LB on the cloud, a multi-agent virtual machine migration was developed in [13]. However, it has a long execution time. In [14], an energy-efficient LB strategy was suggested

that relies on VM consolidation in cloud networks. To avoid unnecessary VM migrations, a mechanism for classifying load states was first employed for PMs with load irregularities, taking into consideration both current and projected loads. Then, a resource-weighted selection model for VM migration was developed, which picks suitable VMs to migrate based on multidimensional resource utilization while minimizing resource fragmentation. In addition, the optimal destination PMs for virtual machines was determined using a VM placement algorithm that takes into account resource fitness and load correlation. But there were a lot of virtual machine migrations and energy consumption.

A method for energy-efficient virtual machine allocation and migration was developed using the Smart Elastic Scheduling Algorithm (SESA) in [15]. This method was based on cosine similarity and bandwidth utilization. However, there has been no decrease in energy use. Integrating into networks on a grand scale is likewise not an easy task. For online VM replacements that take energy usage, SLA breaches, and migration frequency into consideration, a Modified Genetic-based VM Consolidation (MGVMC) technique was devised in [16]. Virtual machines were migrated to the correct PM using the Genetic Algorithm (GA). However, they failed to examine the impact on reliability and scalability when increasing the number of jobs.

The Re-initialization and Decomposition-based Whale Optimization Algorithm (RD-WOA) integrates the WOA and NSGA-II to find Pareto-optimal solutions[17]. They reduced the amount of VM migrations, which in turn reduced migration costs and energy consumption, by improving job allocation to the VM. Making span and resource consumption may have been affected if they had considered optimal task scheduling. In [18], a Predicted Mixed Integer Linear Programming (MILP) Robust Solver (PMRS) considered VM migration as a Γ-robust knapsack problem (Γ-RKP). Initially, VMs' future behaviours were anticipated. However, it is computationally intensive and has little robustness. The goal of the Modified Feeding Birds Algorithm (ModAFBA) for VM consolidation in cloud data centers is to enhance operational effectiveness and resource management ([19]). As a means to reduce VM migrations, it employed adaptive position updating rules and techniques. The decision-making process could be enhanced with the addition of data on application needs and workload trends.

In [20], a Harmonic Migration Algorithm (HMA) was developed that combines the migration algorithm with harmonic analysis to migrate VM from an overloaded to an underloaded PM and activate or disable the VM utilizing cloud switching techniques. The jobs were distributed to the appropriate VM in a round-robin method, and the VM's load was predicted using the gated recurrent unit. However, it migrated the VM when the predicted load surpassed the threshold, which is not appropriate for real-time cloud systems.

In [21], a modified BAT approach was implemented for VM migration in a cloud setting. The fitness function calculation technique used in spider monkey optimization was implemented into the normal bat algorithm to optimize each bat's fitness calculation and provide additional options for quickly selecting the fittest move. The best move allows VMs to connect with the closest PMs and complete the migration rapidly, rather than being stopped. However, the network's energy usage throughout the migration process was not taken into consideration, affecting network performance. In [22], a novel approach to LB and VM migration was developed that employs modelled agents and IoT devices to optimize resource utilization and work distribution. New Optimized Virtual Machine Migration Scheme (OVMMS) based on Squirrel Search Algorithm (SSA) for migration and search was showcased. Nevertheless, the efficiency of the network was diminished due to the disregard for energy consumption. As a means of enhancing LB efficiency within the fog-cloud paradigm, a Hybrid Markov Chain-based Dynamic Scheduling system was created in [23]. To schedule tasks, this method used the Arithmetic Optimization algorithm (AOA), and to forecast virtual machine loads, it used a hidden Markov chain. On the other hand,

load forecasting using the Markov chain was susceptible to unexpected and sudden changes in workload, which could lead to less-than-ideal job distribution and overall system underperformance.

From this literature, it is inferred that existing VM migration approaches operate independently of RA-LB techniques. This can impact energy usage, resource distribution, and execution time. Also, unwanted or redundant VM migrations can occur in dynamic and heterogeneous settings. To address this challenge, this work introduces the OVM2-RALB technique based on a new enhanced optimization algorithm, which improves resource utilization and reduces energy consumption efficiently in IoT-fog-cloud systems.

3. PROPOSED METHODOLOGY

This section briefly explains the proposed OVM2-RALB technique for IoT-fog-cloud systems. A schematic illustration of this technique is portrayed in Figure 1, comprising users who generate tasks, a task classifier, a resource distributor, a fog layer, and resources in the cloud layer. Initially, the tasks created by users are positioned in the arrival queue to classify them into fog and cloud tasks. The classified tasks are then transferred to the resource distributor for allocating resources either on the cloud or fog layer. Once the tasks are assignedto the resources, it checks if the PM is overloaded or underloaded. If the PM is overloaded, the ESOA- based VM migration strategy is employed to pick the most appropriate VMs to be migrated to the suitable PMs, thus balancing the load on PMs.

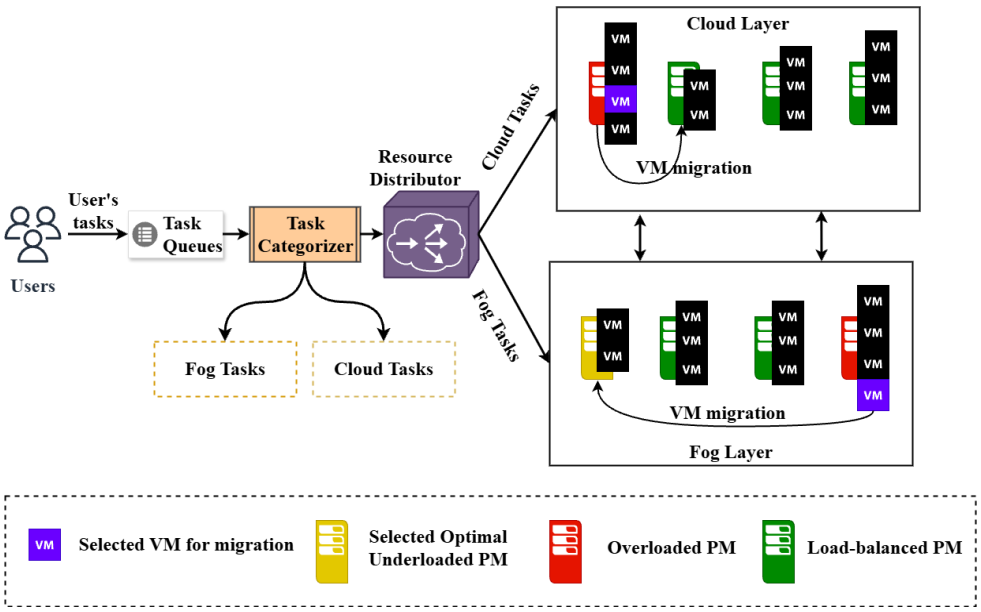

Figure 1. Schematic Illustration of the Proposed Study

3.1. IoT-Fog-Cloud Network Model

The suggested system paradigm achieves effective data handling from IoT devices by utilizing a hierarchically structured IoT-fog-cloud environment. Figure 2 shows the three main layers of the model: cloud, fog, and edge. In this Internet of Things (IoT) fog-cloud design, each layer makes use of its own set of capabilities, making it easy to oversee the complex IoT system's operations. Reduced delays and optimized resource consumption allow for more efficient data processing. The sensors, actuators, and wearables that make up the edge layer request tasks from the fog or

cloud layer, depending on how much processing power is needed. Fog nodes are temporary servers or other devices placed at the perimeter of a network to allow for local processing and storage. The fog layer has gateways that provide connections among edge devices, fog nodes, and cloud data centers. A vast number of data centers on the cloud layer can provide substantial computing power, extensive storage capacity, and sophisticated data processing capabilities. When an edge device produces tasks, they are sent from the task categorizer to the resource distributor to ascertain the suitable processing layer and provide resources to tasks, respectively. Time-sensitive jobs are allocated to the fog queue, whereas time-insensitive tasks are designated for the cloud queue.

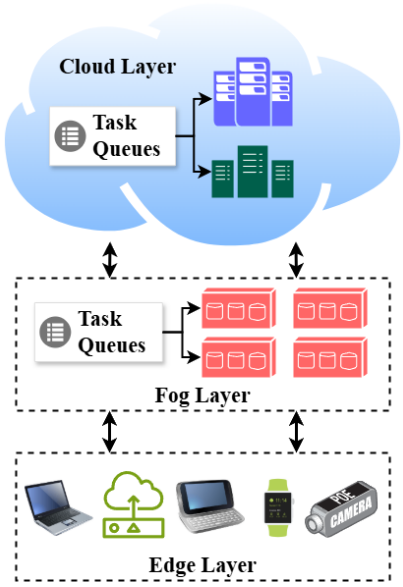

Figure 2. IoT-Fog-Cloud Network Model

3.2. Task Classification

In fog- cloud environment, PMs heterogenous because they have various quantity of VMs.The available PMs in fog and cloud datacentres is represented as $PM = \{PM_1, \dots, PM_p\}$, where p is the total count of PMs and the available VMs is represented as $VM = \{VM_1, \dots, VM_q\}$, here q represents total available VMs. The arrival queue in fog and cloud layers has a collection of k tasks, denoted by $T = \{T_1, \dots, T_k\}$. The i^{th} task in the queue is each task $T_i, i \in [1,2, \dots, k]$. The goal of allocating resources is to finish a work and deliver its results to the user before the job's deadline passes. A job is deemed successful if it is returned to the user ahead of schedule; otherwise, it is seen as a failure. So, the tasks from IoT devices are sorted as either fog or cloud tasks based on their task guarantee ratio, which is determined by the processing requirements of each task [10].The task scheduled to each VM is represented by $T_v = \{T_{v_1}, T_{v_2}, \dots, T_{v_n}\}$, where v_n is the number of tasks scheduled to each VM. The fundamental goal of the RALB method in an IoT-fog-cloud network is to reduce power consumption and maximize resource utilization by distributing the load evenly among all PMs. Therefore, the standard variance (σ) for each PM is initially determined to identify overloaded and underloaded PMs. To achieve this, it is crucial to compute the load of each PM according to the scheduled tasks (i.e. workload) of deployed VMs. The average load of j-th VM in i-th PM $(\bar{L}_j^i)$ is calculated as:

$$\bar{L}_j^i = CPU_j^i + MU_j^i + BU_j^i \quad (1)$$

In Eq. (1), CPU_j^i, MU_j^i, and BU_j^i denote the consumption of CPU, memory, and bandwidth of j-th VM in i-th PM, respectively. So, the average load of i-th PM $(\bar{L}^i)$ and its standard variance (σ^i) is determined as:

$$\bar{L}^i = \frac{\sum_{j=1}^{q} \bar{L}_j^i}{q}, \forall j \in PM_i \quad (2)$$

$$\sigma^i = \sqrt{\frac{\sum_{i=1}^{p} (\bar{L} - \bar{L}^i)^2}{p}} \quad (3)$$

$$\text{Where } \bar{L} = \frac{\sum_{i=1}^{p} \bar{L}^i}{p} \quad (4)$$

The average load of all PMs is represented by $\bar{L}$ in Eq. (3). The i-th PM is considered underloaded when σ^i is less than the minimum threshold value and overloaded whenσ^i is larger than the maximum threshold value. Among all PMs, the lowest possible threshold value is $\bar{L}^i$ and this value is equivalent to the maximum threshold value. As a result, the PMs that are either overworked or understaffed are located.

3.3. Fitness Function for VM Migration

To optimize the VM migration, the following factors are considered to formulate the fitness function for ESOA:

- ***Migration Cost:*** It determines the percentage of overall migrations done to the amount of migrations attained.

$$MC = \frac{1}{p} \sum_{i=1}^{p} \left(\frac{\tau}{\epsilon \times q} \right) \quad (5)$$

In Eq. (5), τ is the total VM migrations, and ϵ is a migration constant.

- ***Energy Utilization:*** It measures how much energy is consumed by each VM to execute the tasks.

$$EC = \frac{1}{T} \sum_{t=1}^{T} U \quad (6)$$

$$\text{Where } U = s \times U_{max} + (1 - s) \times U_{max} \times RU \quad (7)$$

In Eqns. (6)-(7), T is the total interval and U is the energy consumed, s refers to a scaling factor, U_{max} is the highest energy utilized and RU is the resource usage.

- ***Bandwidth Factor:*** It is calculated by

$$A = \sum_{i=1}^{p} \sum_{j=1}^{q} \left(B_j^i + B_j \right) \quad (8)$$

In Eq. (8), B_j^i denotes the bandwidth of j-th VM in i-th PM, and B_j indicates the bandwidth utilized for j-th VM migration.

- ***Load Balancing Factor:*** It is determined as:

$$\tau = \frac{\max_{j\in\{1,2,\dots,q\}} \bar{L}_j^i - \min_{j\in\{1,2,\dots,q\}} \bar{L}_j^i}{\sum_{j=1}^{q} \bar{L}_j^i / q} \quad (9)$$

Based on these factors, the fitness function of the ESOA for VM migration problem (f_{VM}) is modeled as:

$$\mathcal{F}_{VM} = \left(\omega_1 \frac{1}{MC}\right) + \left(\omega_2 \frac{1}{EC}\right) + \left(\omega_3 \frac{1}{A}\right) + (\omega_4 \tau) \quad (10)$$

In Eq. (10), ω_1, ω_2, ω_3, and ω_4 are the weight parameters such that $\omega_1 + \omega_2 + \omega_3 + \omega_4 = 1$.

3.4. VM Migration Using Enhanced Starfish Optimization Algorithm

This study uses ESOA for the VM migration job, treating each starfish as a separate VM migration configuration (i.e., candidate VMs to be relocated from overloaded PMs to underloaded PMs) in the IoT-fog-cloud context. The starfish populations that the SOA works with stand in for various solutions. The optimization process repeatedly updates all starfishes to discover the optimal VM and PM for VM migration. The design parameter encompasses the total number of VMs and PMs, each with unique minimum and maximum resource requirements, commonly referred to as margins.

3.4.1. Initialization Using Tent Chaotic Map

Initially, the positions of starfish (i.e., a particular VM and PM) are generated at random as:

$$S = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1D} \\ s_{21} & s_{22} & \cdots & s_{2D} \\ \vdots & \ddots & \iddots & \vdots \\ s_{N1} & s_{N2} & \cdots & s_{ND} \end{bmatrix}_{N\times D} \quad (11)$$

In Eq. (11), N is the population count and D represents the total of the design parameters,S is the matrix that accumulates the positions of the starfish along a $N \times D$ dimension. In Eq. (12), the position of each starfish is predicted by

$$s_{ij} = l_j + r(u_j - l_j), i = 1, \dots, N; j = 1, \dots, D \quad (12)$$

In Eq. (12), s_{ij} is the j^{th} location of the i^{th} starfish, r is an arbitrary integer ranging $(0,1)$, and u_j and l_j are j^{th} dimension size lower and upper boundaries of design parameters. The random initialization of location impacts both population diversity and algorithm resilience. The SOA can only verify the unpredictability of the population location during the initialization stage; nevertheless, randomness does not imply uniformity. To increase the variety of this initial population, this study uses tent chaotic mapping, which creates random integers with a uniform distribution between 0 and 1. The qualities and unpredictability of this map can significantly increase performance by altering the starfish's beginning placements. Thus, the initial locations are modified by Eq. (13) to enhance the initialization population diversity.

$$s_{ij}^{new} = \begin{cases} \frac{s_{ij}^{old}}{a}, & s_{ij}^{old} < a \\ \frac{1 - s_{ij}^{old}}{1 - a} & s_{ij}^{old} \geq a \end{cases} \quad (13)$$

In Eq. (13), s_{ij}^{new} is the new location of starfish after tent chaotic mapping, s_{ij}^{old} is the current location of starfish (i.e., randomly initialized location), and a is assigned to 0.7.After the initial starfish positions are created, the fitness values of each starfish are calculated as Eq. (10) and stored as:

$$\mathcal{F} = \begin{bmatrix} \mathcal{F}(S_1) \\ \mathcal{F}(S_2) \\ \vdots \\ \mathcal{F}(S_N) \end{bmatrix}_{N\times 1} \tag{14}$$

ESOA initiates the exploration and exploitation phases subsequent to the generation of initial positions of starfish by comparing the random number ranging $(0,1)$ with the dimension of G_k. If the number is not greater than G_k, the exploration step is started; or else, the exploitation step is begun.

3.4.2. Exploration

To represent the starfish's exploratory behaviour, the ESOA's exploration phase replicates the starfish's five arms' capacity to search by inserting eyes at the end of each arm. A novel search pattern is proposed during the ESOA exploration phase, which integrates the unidimensional search pattern for $D \leq 5$ and the 5D search pattern for $D > 5$. The threshold of dimension is determined by the five limbs (or eyeballs) of a starfish. The optimization problem's search space is extensive if $D > 5$, which requires starfish to move all five limbs to investigate their environments. Additionally, starfish limbs require information regarding the optimal search agent position to guide their movement. This behaviour is modelled as:

$$Y_{i,k}^{T} = \begin{cases} s_{i,k}^{T} + a_1\left(s_{best,k}^{T} - s_{i,k}^{T}\right)\cos\theta\,, & rand \leq 0.5 \\ s_{i,k}^{T} + a_1\left(s_{best,k}^{T} - s_{i,k}^{T}\right)\sin\theta & rand > 0.5 \end{cases} \tag{15}$$

Where $a_1 = (2 \times rand - 1)\pi$ (16)

$$\theta = \frac{\pi}{2} \times \frac{T}{T_{max}} \tag{17}$$

In Eq. (15), $Y_{i,k}^{T}$ and $s_{i,k}^{T}$ are the obtained and present locations of a starfish, correspondingly, $s_{best,k}^{T}$ is k-dimensional of the present ideal location, k denotes 5 arbitrarily chosen dimensions among D dimensions, $rand \in (0,1)$, T is the present iteration, and T_{max} is the maximum iteration. Also, the ability of starfish appendages to rotate left or right to approach food with the same probability is defined by sin and cos. In this stage, a_1 is arbitrarily generated to update the location of each candidate in each iteration, and θ is a value in the range of 0 to π/2. These factors can be used to determine the effect of the distance between the ideal and current locations in the selected updating dimension. In the event that $D > 5$ in an optimization problem, Eq. (15) employs the 5D search pattern to update only 5 dimensions of the locations to assure search capacity and improve search efficiency in comparison to the vectorial search pattern. The limbs then tend to remain in the previous location rather than moving to the new location when the updated location is outside the design parameters. This is symbolized by

$$s_{i,k}^{T+1} = \begin{cases} Y_{i,k}^{T}, & l_{b,k} \leq Y_{i,k}^{T} \leq u_{b,k} \\ s_{i,k}^{T} & orelse \end{cases} \tag{18}$$

The location is updated during the exploration stage if $D \leq 5$, using the unidimensional search pattern. In this case, a starfish employs the location data of other starfish to move only one limb in pursuit of the food source. The location has been revised as follows:

$$Y_{i,k}^{T} = E_t s_{i,k}^{T} + A_1\left(s_{x_1,k}^{T} - s_{i,k}^{T}\right) + A_2\left(s_{x_2,k}^{T} - s_{i,k}^{T}\right) \quad (19)$$

In Eq. (19), $s_{x_1,k}^{T}$ and $s_{x_2,k}^{T}$ are the k-dimensional locations from 2 arbitrarily chosen starfish, correspondingly, A_1 and A_2 are 2 arbitrary integers between $(-1,1)$, k is the arbitrarily chosen integer in the D dimensions, and E_t defines the energy of the starfish, which is determined by

$$E_t = \frac{T_{max}-T}{T_{max}} \cos\theta \quad (20)$$

In Eq. (20), θ is determined by Eq. (17). Similar to the previous updating rule, if a starfish's acquired position falls beyond the margin, it is more likely to stay in its prior place rather than move to the updated one.

3.4.3. Exploitation

Two update schemes are developed during the ESOA exploitation phase, considering predatory and regeneration behaviors to identify global solutions. ESOA mimics the predatory phase of starfish by employing a parallel 2-directional search approach that requires the usage of information from other starfish as well as the population's current ideal position. Using the parallel 2-directional search approach, the five distances between the ideal location and other starfish are first determined, and two distances are randomly chosen as an acknowledgement to update each starfish's location. Distances are calculated by

$$d_m = \left(s_{best}^{T} - s_{m_k}^{T}\right), m = 1, \ldots, 5 \quad (21)$$

In Eq. (21), d_m are five obtained distances between the global best and other starfish, and m_k are 5 arbitrarily chosen starfish. As a result, the updating rule for each starfish in the preying behavior is modeled as:

$$Y_i^{T} = s_i^{T} + rand_1 d_{m1} + rand_2 d_{m2} \quad (22)$$

In Eq. (22), $rand_1$ and $rand_2$ are arbitrary integers ranging $(0,1)$, d_{m1} and d_{m2} are arbitrarily chosen in d_m. According to the parallel two-directional search technique, some starfish candidates journey backward in the same iteration while others move toward the superior guiding solution. Thus, the candidates' ability to overcome local optima is comparable.

Furthermore, because starfish move slowly, they are susceptible to other predators during predation. To escape being captured by a predator, a starfish may chop off an arm. As a result, only the last starfish in the population $(i = N)$ passes through the ESOA regeneration phase. The starfish travels slowly because regeneration takes several months. During regeneration, the updating rule is modeled, and the position is updated by:

When it comes to the ESOA regeneration phase, only the last of the starfish throughout the population $(i = N)$ makes it through. The starfish moves at a snail's pace because of the many months it takes to regenerate. The exact location is updated by modeling the updating algorithm during regeneration.

$$Y_i^{T} = e^{\left(-T \times N / T_{max}\right)} s_i^{T} \quad (23)$$

The position of starfish is calculated using Eq. (22) or Eq. (23) exceeds the design parameters' margins, the location is allocated as:

$$s_i^{T+1} = \begin{cases} Y_i^T, & l_b \le Y_i^T \le u_b \\ l_b, & Y_i^T < l_b \\ u_b, & Y_i^T > u_b \end{cases} \quad (24)$$

The regeneration stage only includes a small amount of evaluations of fitness (estimated twice every iteration), in contrast to the exploitation step which is crucial for avoiding local solutions and increasing global convergence).

3.4.4. LSR Learning

LSR learning emphasizes the inverse determination of the ideal person within the constraints. As the iteration advances, the location of the ideal individual continuously shifts, and the spiral reverse learning search space of opposition-based learning becomes increasingly diminished. On the other hand, this innovative method is very useful in later rounds when population diversity decreases since it operates on individuals that vary within a narrow space, especially those close the optimal solution. This ensures that the algorithm does not converge too quickly and that the ideal global level is not disregarded. Here is how this conduct is described:

$$Y_{op}^T = \left(rand_1 \times u_j\right) + \left(rand_1 \times l_j\right) - Y_{best}^T \quad (25)$$

$$s_{i,k}^{T+1} = \left|Y_{best}^T - Y_{op}^T\right| \times e^{bl} \times \cos(2\pi l) + Y_{best}^T \quad (26)$$

In Eq. (25)-(26), Y_{op}^T is the place of starfish after LSR learning, l is values generated between -1 and 1 randomly, which determines how much $s_{i,k}^{T+1}$ is close to the Y_{best}^T, and a spiral shape is created for $b = 1$. The pseudocode and flowchart for this ESOA based OVM2-RALB technique is presented in Algorithm 1 and Figure 3, respectively.

Algorithm 1: OVM2-RALB Using ESOA

Input: Tasks $\mathcal{T}$ and resources R

Output: Optimal VM and PM for VM migration

1. **Begin**
2. $\boldsymbol{for}(each\ incoming\ task)$
3. Classify the tasks into fog and cloud based on their guarantee ratio;
4. Schedule the classified tasks into their respective layers;
5. $\boldsymbol{end\ for}$
6. Compute the average load of each VM in each PM as Eq. (1);
7. Determine the mean load of each PM and its standard variance by Eqns. (2)-(3);
8. Identify overloaded and underloaded PMs;
9. Compute the migration cost, energy utilization, bandwidth factor, and LB factor using Eqns. (5)-(9);
10. Formulate the fitness function as Eq. (10);
11. Initialize the number of starfish population N, maximum iteration T_{max}, u_j and l_j;
12. Create initial locations arbitrarily as Eq. (12) and modify them based on the tent chaotic mapping as Eq. (13);

13. Evaluate fitness values of each starfish and store it as Eq. (14);
14. $\boldsymbol{while}(T \leq T_{max})$
15. $\boldsymbol{if}(rand > 0.5)$
16. Determine θ and E_t as Eqns. (17) and (20), respectively;
17. $\boldsymbol{for}(each\ starfish)$
18. $\boldsymbol{if}(D > 5)$
19. Create p-dimensions arbitrary from D;
20. $\boldsymbol{for}(j = 1{:}5)$
21. Determine a_1 as Eq. (16);
22. Update the starfish location as Eq. (15);
23. $\boldsymbol{end\ for}$
24. Verify the location boundary as Eq. (18);
25. $\boldsymbol{else}$
26. Generate A_1, A_2, and p;
27. Update the p-dimension of location as Eq. (19);
28. Verify the location boundary as Eq. (18);
29. $\boldsymbol{end\ if}$
30. $\boldsymbol{end\ for}$
31. $\boldsymbol{else}$
32. Calculate the optimal distances to starfish from various locations.
33. $\boldsymbol{for}(each\ starfish)$
34. Create arbitrary integer r_k and update the location as Eq. (22);
35. $\boldsymbol{if}(i = N)$
36. Change the site as Eq. (23);
37. $\boldsymbol{end\ if}$
38. Verify the location boundary as Eq. (24);
39. $\boldsymbol{end\ for}$
40. $\boldsymbol{end\ if}$
41. Determine the fitness values for each starfish;
42. Find the global best solution;
43. Update the location as Eq. (26);
44. $\boldsymbol{end\ while}$
45. **Return** the global solution (i.e., optimal VM to be migrated and optimal PM for VM migration);
46. **End**

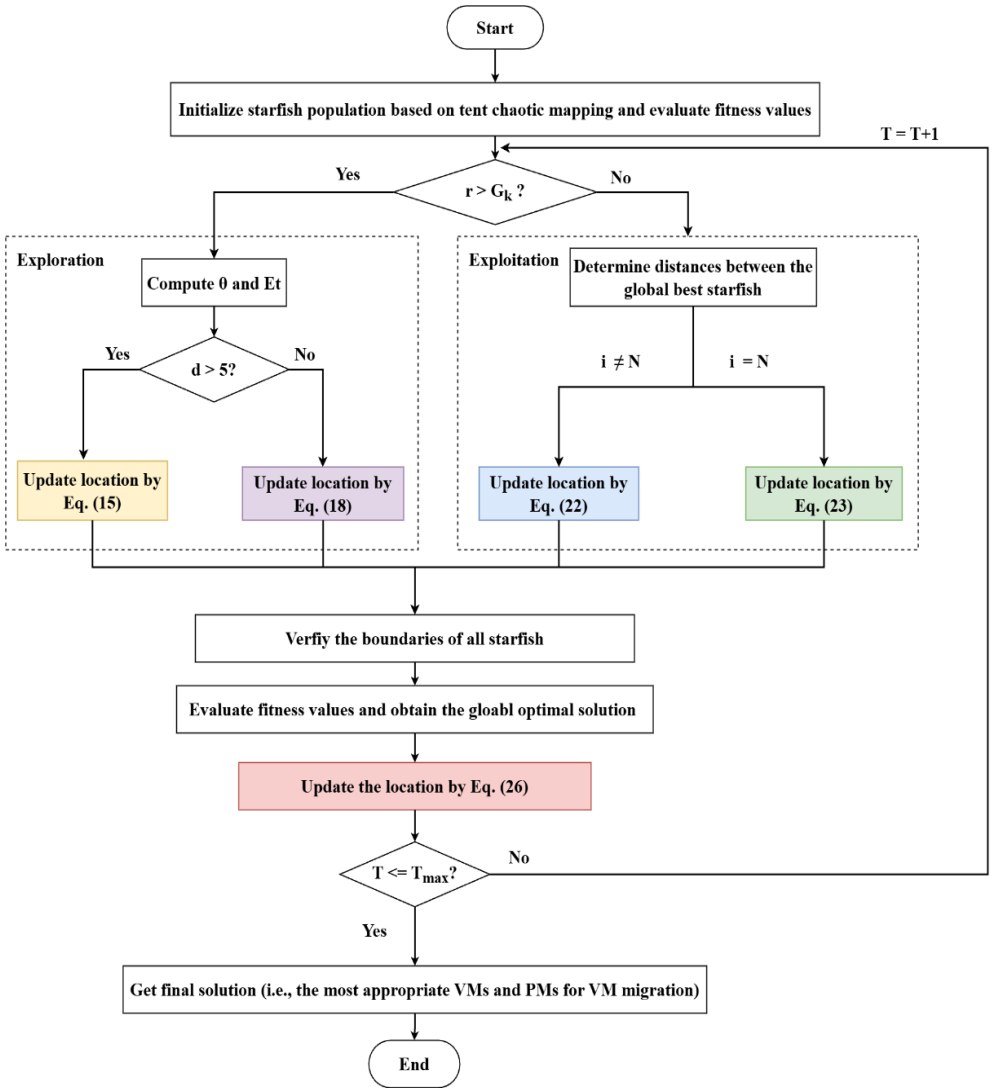

Figure 3. Flowchart of ESOA for Optimal VM Migration

4. SIMULATION RESULTS

In an IoT-fog-cloud network, this section assesses how well the suggested OVM2-RALB method works. The iFogSim simulator is used to test different setup scenarios. The iFogSim toolbox models fog-cloud resources, which are then used to develop and evaluate the TCRA-LB method. The initial phase involves generating various types of fog and cloud resources to establish a fog-cloud paradigm and implement the OVM2-RALB technique. A hierarchical structure is created by connecting resources to their respective levels. Each layer has resources with features for data processing, storage, and information transmission within the system. Table 1 lists the key characteristics of fog and cloud resources, such as CPU, memory, latency, and bandwidth capabilities.

Table 1. Resource and Task Attributes

Resource category	Node counts	CPU (MIPS)	Memory (GB)	Bandwidth (Mbps)	Latency (ms)
Fog	40	500-2000	1-8	1-10	1-10

Cloud	10	5000-10000	16-32	50-100	20-50
Task category	**Task counts**	**CPU requirements (MIPS)**	**Memory requirements (MB)**	**Bandwidth requirements (Mbps)**	**Latency constraints (ms)**
Time-sensitive	500	100-1000	300-500	2-8	2-8
Time-insensitive	500	2000-5000	500-1000	20-70	20-40

The evaluation of the OVM2-RALB technique is based on a uniform distribution of user workloads to simulate the continuous deployment of IoT devices at network edges. It also consists of the characteristics of time-sensitive and time-insensitive jobs, specifying CPU, memory, bandwidth requirements, and latency constraints. Time-sensitive jobs require minimal CPU, low memory, and bandwidth, with stricter latency limits compared to time-insensitive tasks. Edge nodes can generate 500 tasks per class, which are then batched and sent to resource distributors in the fog and cloud levels for allocation based on QoS requirements and resource availability in each tier. There will be a maximum of 50 iterations for ESOA and a population size of 50.

4.1. Performance Analysis Of Proposed And Existing VM Migration Techniques

This section evaluates the OVM2-RALB's efficiency compared to the existing techniques, such as O2O-LB [12], SESA [15], MGVMC [16], OVMMS [22], and HMCDS [23], based on various metrics. The comparative results are discussed in the subsections below.

4.1.1. Response Time

It controls how long it takes for the fog-cloud network as a whole to complete each job, as:

$$Responsetime = Transmissiontime + Processingtime \quad (27)$$

In Eq. (27), transmission time refers to the duration required for data to be sent from edge devices to fog nodes.

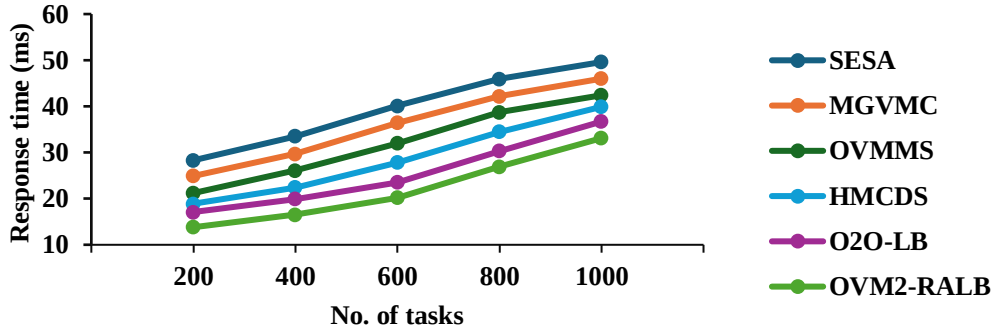

Figure 4. Comparison of Response Time for Proposed OVM2-RALB and Existing Techniques

The processing time refers to the duration required by the fog nodes to execute the incoming data. Responsetimes for several RALB techniques under varying quantities of tasks are compared in Figure 4. It is observed that the proposed OVM2-RALB reduced response time significantly compared to the SESA, MGVMC, OVMMS, HMCDS, and O2O-LB for different task counts.

Compared to these techniques, the OVM2-RALB with 1000 tasks reduces the response time by 33.27%, 28.04%, 21.93%, 17.04%, and 9.81%, respectively.

4.1.2. Latency

It calculates the overall processing and data transfer delays encountered by the IoT-fog-cloud system, as

$$Latency = \sum Computationallatency + \sum Transmissionlatency \quad (28)$$

In Figure 5, the latency results of various RALB techniques under varying quantities of tasks are evaluated. It is addressed that the proposed OVM2-RALB lessened latency compared to the SESA, MGVMC, OVMMS, HMCDS, and O2O-LB for varying task quantities in an IoT-fog-cloud environment. For 1000 tasks, the OVM2-RALB decreases the latency by 34.07%, 29.64%, 25.68%, 18.16%, and 9.64%, respectively, compared to the same existing techniques.

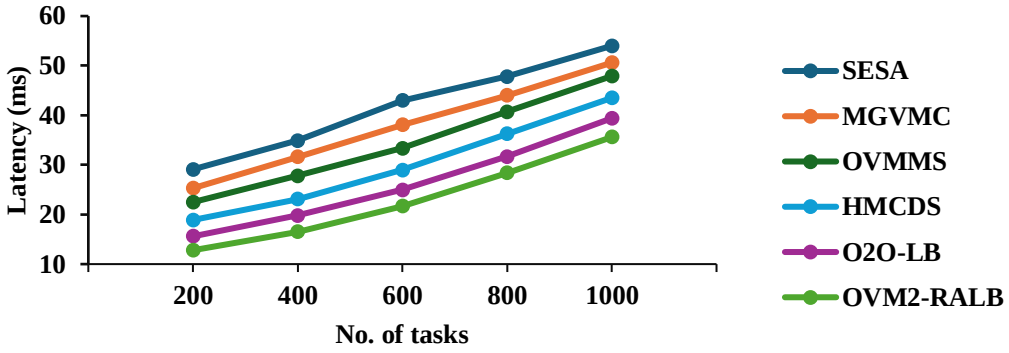

Figure 5. Comparison of Latency for Proposed OVM2-RALB and Existing Techniques

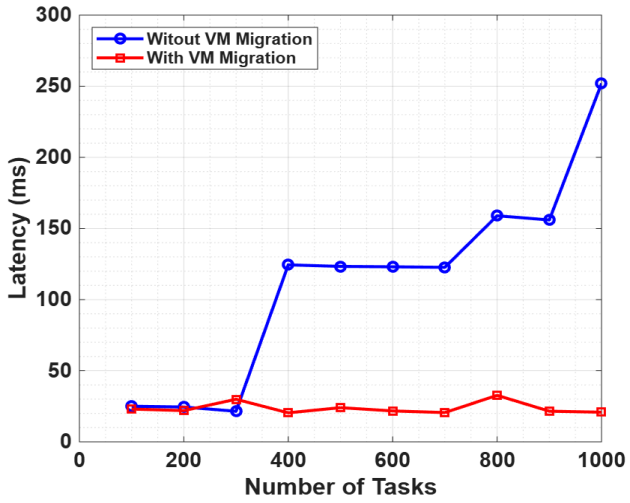

Figure 6. Comparison of Latency with and without VM Migration

Figure 6 depicts the latency resulting from two scenarios, with and without VM migration, for a varying number of tasks The latency curve of both cases initially exhibits similar behaviors since, early in the beginning of the task's execution, the user connects to the VM, a hosted service in the first linked cloudlet. In up to 1000 tasks, in the absence of VM migration, latency rises drastically

from around 20 ms to approximately 125 ms because of the user handoff occurrences as the VM still remains in the former cloudlet. This leads to increased routes in communication and transmission delays. Conversely, the latency during the execution of all 1000 tasks in the case of VM migration is around 20 ms when the proposed technique is implemented, and the VM is migrated with the user to the closest cloudlet. This shows that VM migration is an efficient operation with low latency for advancing workloads.

4.1.3. Load Imbalance

It establishes the inequitable allocation of work among cloud and fog resources, as shown in Eq. (9).In Figure 7, the load imbalance results for the proposed and existing RALB techniques are shown under varying task quantities. It is noted that the proposed OVM2-RALB efficiently decreases load imbalance compared to SESA, MGVMC, OVMMS, HMCDS, and O2O-LB for varying task quantities. For 1000 tasks, the OVM2-RALB lessens the load imbalance by 25%, 19.64%, 15.09%, 10%, and 4.26%, respectively, compared to the same existing techniques.

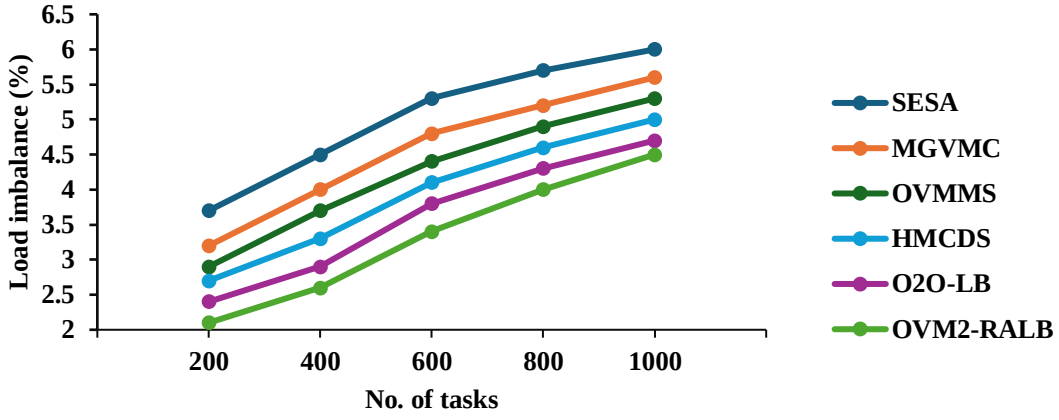

Figure 7. Comparison of Load Imbalance for Proposed OVM2-RALB and Existing Techniques

4.1.4. Task Failure Rate

This metric measures the percentage of tasks that are not completed efficiently because of limitations in resources and systems.

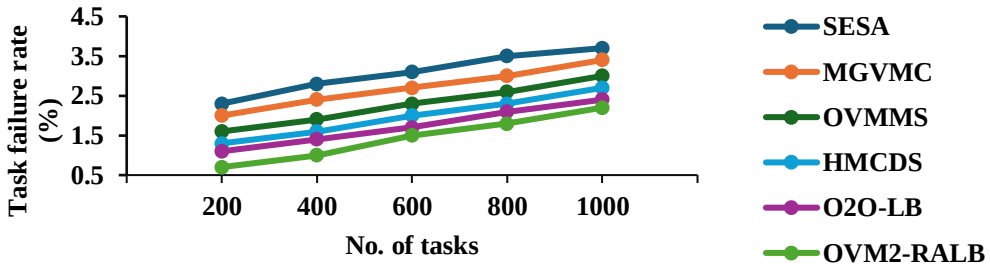

Figure 8. Comparison of Task Failure Rate for Proposed OVM2-RALB and Existing Techniques

Figure 8 illustrates the task failure rate results for the proposed and existing RALB techniques under varying task quantities. It is noted that the proposed OVM2-RALB efficiently decreases

the task failure rate compared to SESA, MGVMC, OVMMS, HMCDS, and O2O-LB for varying task quantities. For 1000 tasks, the OVM2-RALB lessens the task failure rate by 40.54%, 35.29%, 26.67%, 18.52%, and 8.33%, respectively, compared to the same existing techniques.

4.1.5. Makespan

This refers to the duration of the initial job commencing and concluding the last one on VMs.

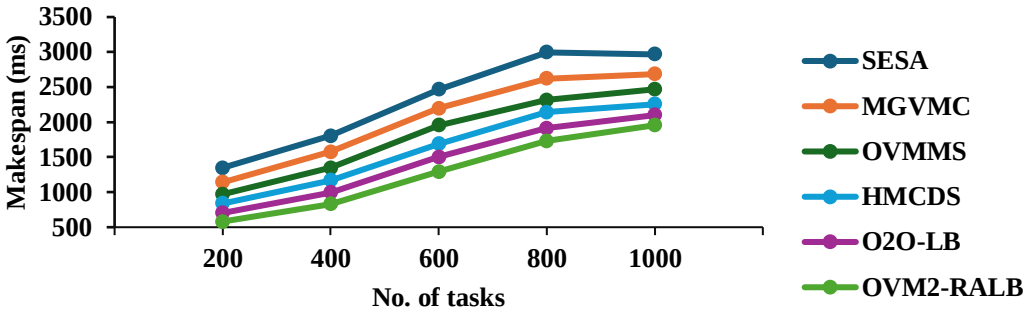

Figure 9. Comparison of Makespan for Proposed OVM2-RALB and Existing Techniques

Figure 9 illustrates the makespan results for the proposed and existing RALB techniques under varying task quantities. It is noted that the proposed OVM2-RALB efficiently decreases the makespan compared to SESA, MGVMC, OVMMS, HMCDS, and O2O-LB for varying task quantities. For 1000 tasks, the OVM2-RALB lessens the makespan by 34.14%, 27.15%, 20.78%, 13.22%, and 6.86%, respectively, compared to the same existing techniques.

4.1.6. Energy Utilization

It is the amount of power that the system's resources in the cloud and fog layer use

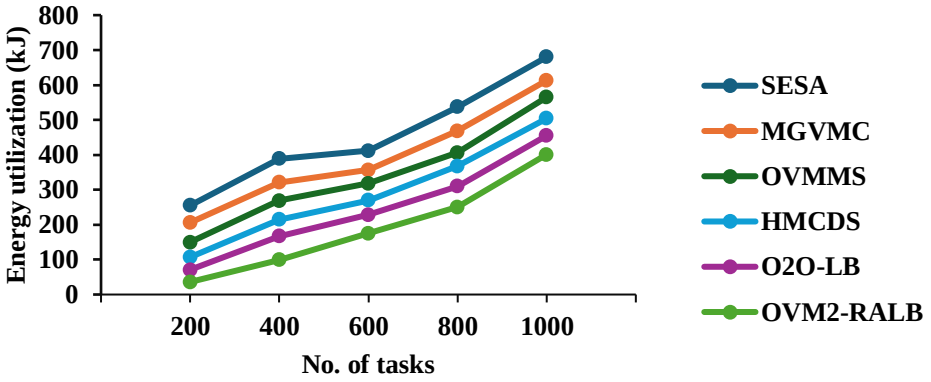

Figure 10. Comparison of Energy Consumption for Proposed OVM2-RALB and Existing Techniques

Figure 10 portrays the energy utilization results for the proposed and existing RALB techniques under varying task quantities. It is noted that the proposed OVM2-RALB efficiently decreases the energy usage compared to SESA, MGVMC, OVMMS, HMCDS, and O2O-LB for varying

task quantities. For 1000 tasks, the OVM2-RALB lessens the energy usage by 41.03%, 34.58%, 29.03%, 20.44%, and 11.87%, respectively, compared to the same existing techniques.

4.1.7. Execution Time

The overall duration required to execute a certain job. It encompasses the duration required for task processing, scheduling, and execution.

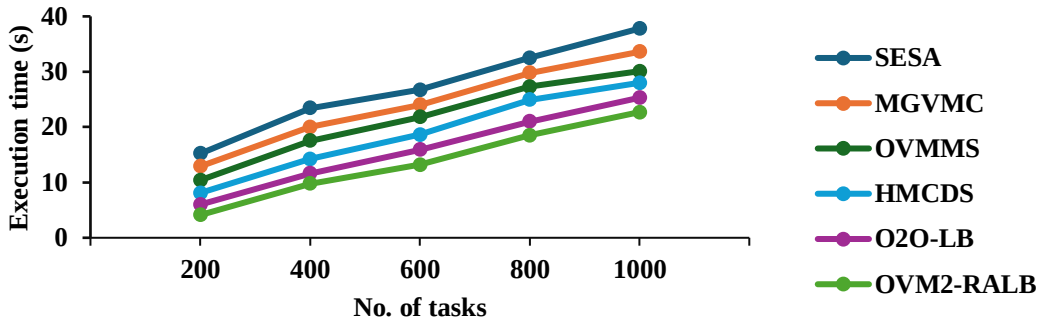

Figure 11. Comparison of Execution Time for Proposed OVM2-RALB and Existing Techniques

Figure 11 portrays the execution time results for the proposed and existing RALB techniques under varying task quantities. It is noted that the proposed OVM2-RALB efficiently decreases the execution time compared to SESA, MGVMC, OVMMS, HMCDS, and O2O-LB for varying task quantities. For 1000 tasks, the OVM2-RALB decreases the execution time by 39.95%, 32.44%, 24.58%, 18.93%, and 10.28%, respectively, compared to the same existing techniques.

4.2. Discussion

From these analyses, it is inferred that the suggested OVM2-RALB demonstrates a reliable performance increase on all metrics due to adaptable VM migration under a multi-objective optimization strategy. In contrast to current algorithms that address RA, LB, and VM migration as partially independent processes, OVM2-RALB integrates them into one framework. The proposed approach allows for avoiding both resource overloading and idle resource waste by accurately identifying overloaded and underloaded PMs through the load balancing aspect and dynamically relocating the most suitable VMs. This process is further enhanced by the use of ESOA, which optimizes migration cost, energy, bandwidth, and load balance simultaneously to perform only beneficial and minimal VM migrations. This approach brings task execution closer to optimal resources, decreasing response time, latency, makespan, execution time, task failure rate, and energy usage. Therefore, the key factor in the overall performance has been the optimized VM migration strategy that ensures equitable resource utilization at fog and cloud layers during dynamic workloads.

5. CONCLUSION

This paper developed the OVM2-RALB approach for IoT-fog-cloud systems using ESOA. The arriving tasks were initially classified as fog tasks or cloud tasks according to the task guarantee ratio. To determine if PMs were overloaded or underloaded, the mean load for each PM was calculated. The fitness function was defined by calculating the LB factor, energy use, bandwidth use, and migration cost. Depending on the specified fitness function, ESOA was used to choose the best PM and VMs for VM migration. The chosen VM was then migrated from the overloaded

PM to the underloaded PM. Finally, results from simulations showed that compared to earlier RA-LB methods in IoT-fog-cloud settings, this OVM2-RALB got the following: 33.1 ms reaction time, 35.6 ms latency, 4.5% load imbalance, 2.2% task failure rate, 1956 ms makespan, 401 kJ energy consumption, and 22.7 s execution time for 1000 tasks.

CONFLICTION OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] Behnke, I. & Austad, H. (2023) "Real-time performance of industrial IoT communication technologies: a review," *IEEE Internet of Things Journal*, Vol. 11, No. 5, pp.7399–7410.

[2] Andriulo, F. C., Fiore, M., Mongiello, M., Traversa, E. & Zizzo, V. (2024) "Edge computing and cloud computing for internet of things: a review," *Informatics*, Vol. 11, No. 4, p. 71.

[3] Gad-Elrab, A. A., Alsharkawy, A. S., Embabi, M. E., Sobhi, A., & Emara, F. A. (2024) "Adaptive multi-criteria-based load balancing technique for resource allocation in fog-cloud environments," *International Journal of Computer Networks & Communications (IJCNC)*, Vol. 16, No. 1.

[4] Ahmed, S. F., Shawon, S. S., Afrin, S., Rafa, S. J., Hoque, M. &Gandomi, A. H. (2025) "Optimising internet of things (IoT) performance through cloud, fog and edge computing architecture," *IET Wireless Sensor Systems*, Vol. 15, No. 1, e70016.

[5] Al-Atawi, A. A. (2024) "Enhancing data management and real-time decision making with IoT, cloud, and fog computing," *IET Wireless Sensor Systems*, Vol. 14, No. 6, pp.539–562.

[6] Srirama, S. N. (2024) "A decade of research in fog computing: Relevance, challenges, and future directions," *Software: Practice and Experience*, Vol. 54, No. 1, pp.3–23.

[7] Ahlawat, C. & Krishnamurthi, R. (2023) "Towards smart technologies with integration of the internet of things, cloud computing, and fog computing," *International Journal of Networking and Virtual Organisations*, Vol. 29, No. 1, pp.73–124.

[8] Vergara, J., Botero, J. &Fletscher, L. (2023) "A comprehensive survey on resource allocation strategies in fog/cloud environments," *Sensors*, Vol. 23, No. 9, p. 4413.

[9] Shaik, M. B., Reddy, K. S., Chokkanathan, K., Biabani, S. A. A., Shanmugaraja, P. & Brabin, D. D. (2024) "A hybrid particle swarm optimization and simulated annealing with load balancing mechanism for resource allocation in fog-cloud environments," *IEEE Access*, Vol. 12, pp.172439–172450.

[10] Ala'anzy, M. A., Zhanuzak, R., Akhmedov, R., Mohamed, N. & Al-Jaroodi, J. (2024) "Dynamic load balancing for enhanced network performance in IoT-enabled smart healthcare with fog computing," *IEEE Access*, Vol. 12, pp.188957–188975.

[11] Le, H. N., & Tran, H. C. (2022). "Ita: The improved throttled algorithm of load balancing on cloud computing", *International Journal of Computer Networks & Communications (IJCNC)*, Vol. 14.

[12] Bharathi, V. C., Abuthahir, S. S., Ayyavaraiah, M., Arunkumar, G., Abdurrahman, U. &Biabani, S. A. A. (2025) "O2O-PLB: a one-to-one-based optimizer with priority and load balancing mechanism for resource allocation in fog-cloud environments," *IEEE Access*, Vol. 13, pp.22146–22155.

[13] Swarnakar, S., Banerjee, C., Basu, J. & Saha, D. (2023) "A multi-agent-based VM migration for dynamic load balancing in cloud computing cloud environment," *International Journal of Cloud Applications and Computing*, Vol. 13, No. 1, pp.1–14.

[14] Yao, W., Wang, Z., Hou, Y., Zhu, X., Li, X. & Xia, Y. (2023) "An energy-efficient load balance strategy based on virtual machine consolidation in cloud environment," *Future Generation Computer Systems*, Vol. 146, pp.222–233.

[15] Kaur, A., Kumar, S., Gupta, D., Hamid, Y., Hamdi, M., Ksibi, A. & Saini, S. (2023) "Algorithmic approach to virtual machine migration in cloud computing with updated SESA algorithm," *Sensors*, Vol. 23, No. 13, p. 6117.

[16] Radi, M., Alwan, A. A. & Gulzar, Y. (2023) "Genetic-based virtual machines consolidation strategy with efficient energy consumption in cloud environment," *IEEE Access*, Vol. 11, pp.48022–48032.

[17] Singh, S. & Singh, D. (2023) "A bio-inspired VM migration using re-initialization and decomposition based-whale optimization," *ICT Express*, Vol. 9, No. 1, pp.92–99.

[18] Wu, J., Yang, W., Han, X., Qiu, Y., Gudkov, A. & Song, J. (2024) "Hotspot resolution in cloud computing: a Γ-robust knapsack approach for virtual machine migration," *Journal of Parallel and Distributed Computing*, Vol. 186, p. 104817.
[19] Alsadie, D. &Alsulami, M. (2024) "Efficient resource management in cloud environments: a modified feeding birds algorithm for VM consolidation," *Mathematics*, Vol. 12, No. 12, p. 1845.
[20] Siruvoru, V. & Aparna, S. (2025) "Harmonic migration algorithm for virtual machine migration and switching strategy in cloud computing," *Concurrency and Computation: Practice and Experience*, Vol. 37, No. 1, e8320.
[21] Archana & Kumar, N. (2025) "A modified bat mechanism for virtual machine migration in a cloud environment," *SN Computer Science*, Vol. 6, No. 1, p. 74.
[22] Liu, C., Ma, L., Zhang, M. & Long, H. (2025) "Optimizing cloud resource management with an IoT-enabled optimized virtual machine migration scheme for improved efficiency," *Journal of Network and Computer Applications*, p. 104137.
[23] Khaledian, N., Razzaghzadeh, S., Haghbayan, Z. &Völp, M. (2025) "Hybrid Markov chain-based dynamic scheduling to improve load balancing performance in fog-cloud environment," *Sustainable Computing: Informatics and Systems*, Vol. 45, p. 101077.

AUTHORS

Mr. A. Charles Mahimainathan is an Assistant Professor of Computer Science. With over 7.5 years of academic experience, he is currently pursuing his Ph.D. in Computer Science at Bharathiar University, Coimbatore. His research interests include advanced computer networking. He has authored several research articles in reputed journals and has presented papers in various national and international conferences. He is an active member of the International Association of Engineers (IAENG), with involvement in the societies of Bioinformatics, Software Engineering, and Information Systems Engineering.

Dr. M. Suganya is currently serving as Associate Professor and Head of the Department of B.Sc. Information Technology at RVS College of Arts and Science, Coimbatore, Tamil Nadu, India. She holds a Ph.D. in Computer Science from Bharathiar University and has over 18 years of teaching experience and 14 years of research experience. Her research interests include computer networks, network security, machine learning, privacy-preserving techniques, visual cryptography, and cloud security. Dr. Suganya has authored and co-authored numerous research papers in Scopus, Web of Science, and UGC CARE-indexed journals. She is also an approved Ph.D. supervisor at Bharathiar University and currently guides both full-time and part-time doctoral scholars. She is a member of the International Association of Engineers (IAENG) and actively participates in academic committees and curriculum development. Dr. Suganya has been recognized with multiple teaching awards and continues to contribute to the academic and research community through publications, conferences, and faculty development initiatives.