*International Journal on Recent and Innovation Trends in Computing and Communication*
***ISSN: 2321-8169 Volume: 11 Issue: 10***
***Article Received:** 15 August 2023 **Revised:** 25 September 2023 **Accepted:** 06 October 2023*

# Load Balancer using Whale-Earthworm Optimization for Efficient Resource Scheduling in the IoT-Fog-Cloud Framework

**[1,2]Gaurav Goel, [1]Dr. Shamik Tiwari, [3]Dr. Rajeev Tiwari**
School of Computer Science
[1]University of Petroleum and Energy Studies,
Dehradun, India.
[2]Department of Computer Science and Engineering
Chandigarh Engineering College-CGC,
Landran (Mohali), India
gaurav.goel9@gmail.com, shamik.tiwari@ddn.upes.ac.in.
[3]School of Computer Science and Engineering,
IILM University,
Greater Noida, Uttar Pradesh, India
errajeev.tiwari@gmail.com

**Abstract**— Cloud-Fog environment is useful in offering optimized services to customers in their daily routine tasks. With the exponential usage of IoT devices, a huge scale of data is generated. Different service providers use optimization scheduling approaches to optimally allocate the scarce resources in the Fog computing environment to meet job deadlines. This study introduces the Whale-EarthWorm Optimization method (WEOA), a powerful hybrid optimization method for improving resource management in the Cloud-Fog environment. Striking a balance between exploration and exploitation of these approaches is difficult, if only Earthworm or Whale optimization methods are used. Earthworm technique can result in inefficiency due to its investigations and additional overhead, whereas Whale algorithm, may leave scope for improvement in finding the optimal solutions using its exploitation. This research introduces an efficient task allocation method as a novel load balancer. It leverages an enhanced exploration phase inspired by the Earthworm algorithm and an improved exploitation phase inspired by the Whale algorithm to manage the optimization process. It shows a notable performance enhancement, with a 6% reduction in response time, a 2% decrease in cost, and a 2% improvement in makespan over EEOA. Furthermore, when compared to other approaches like h-DEWOA, CSDEO, CSPSO, and BLEMO, the proposed method achieves remarkable results, with response time reductions of up to 82%, cost reductions of up to 75%, and makespan improvements of up to 80%.

**Keywords**- Fog Computing; Resource Management; Task Scheduling; Whale optimization; Load Balancer.

## I. Introduction

The Internet of Things (IoT) significantly influences network and computing technologies in Industry 5.0 and succeeding generations. IoT gives access to a wide number of applications, including but not limited to healthcare, traffic management, and self-driving cars, through its integrated components like sensors and network connectivity to a wide range of gadgets and household objects. [1][2]. To keep up with the pace of technology, smart devices generate a wide range of data that must be processed and computed quickly for clients, and end users. By reducing latency, Fog computing promotes IoT and complements cloud computing [3] [4]. Fog nodes extend the cloud to the network's edge, bringing it closer to the source of IoT data. To address issues such as latency, energy efficiency, and security, data is processed at Fog nodes before being transmitted to the cloud for high storage and compute requirements. Cloud and Fog evaluation paradigms are required for all types of IoT data. The Fog-Cloud paradigm is now one of the best solutions available for increased Quality of Service (QoS) requirements [5][6]. Unlike a cloud system, a Fog system is constantly constrained by processing capability. Many IoT applications are latency-sensitive and require different levels of makespan and response time. Due to latency issues, scheduling, and management of these types of applications become difficult and unacceptable in industries like healthcare, autonomous driving, and real-time data processing requirements in the gaming industry. Resource scheduling is one of the solutions to the issues of performance parameters in Fog-Cloud systems [7][8]. The use of cloud platforms raises the cost of communication and computation for the IoT applications. Thus, a balanced tradeoff needs to be maintained among the challenges of such systems.

Cloud-Fog based platform service companies offer a variety of services at varying prices. Cloud services have their own costs based on user demand, with different levels of constraint for consumers. The cost of adopting a Fog-Cloud system may be affected by two factors: computation time and storage. Many present strategies are incapable of balancing cost and QoS factors. Therefore, this study investigates the significance of tasks in a Fog-Cloud system concerning both QoS and cost. We introduce a hybrid optimization algorithm that combines the Whale optimization algorithm [9] with the Earthworm optimization algorithm [10] for the efficient scheduling of tasks. The suggested technique is intended to achieve job scheduling with cost savings and efficient QoS achievement.

The primary contribution of this study can be summarized as follows:

1. A hybrid optimization algorithm, called WEOA (Whale-Earthworm Optimization Algorithm), designed to enhance resource management within the Fog-Cloud environment efficiently is introduced.
2. The exploration phase and convergence speed are improved for optimal resource allocations.
3. Proposing and designing autonomous “Load-balancer based task Allocation Framework” as per the three-layer architecture of Fog environment.

This work is organized into several sections: Section 2 provides a summary of previous resource management techniques in fog computing, while Section 3 outlines the System Architecture for this domain. Section 4 delves into the problem formulation within the Fog environment, and Section 5 elaborates on the problem-to-solution transition, introducing the proposed algorithm, and Section 6 presents the results. Finally, in Section 7, the study concludes.

## II. Literature Review

**Kumar et al. [10]** have proposed a hybrid electric earthworm optimization algorithm to tackle the problem of efficiently scheduling jobs in a Fog-cloud environment to improve QoS for IoT applications in the IoT-Fog-cloud framework. The Electric fish and earthworm optimization method is used by researchers to reduce system latency and energy consumption. Both active and passive electrolocation methods are utilized to enhance the position-updating process, thus improving the overall efficiency of the suggested approach. This approach applies to real-world smart cities and offers valuable utility in vehicle network management. The authors compared cost, makespan, execution time, and energy consumption and found that their method outperformed previous procedures. The proposed approach is tested on real-world workloads from CEA-CURIE and HPC2N. This work lacks an offloading mechanism and a workload management task. **Sarrafzade et al. [11]** have suggested a genetic form algorithm for service placement in the Fog environment. The proposed method is a penalty-based method for determining latency and time usage in cloud installations. The writers also took the proximity of applications into account. A priority value is identified by the chromosome-selection procedure, which estimates the proximity of the dependent component. The proposed technique successfully reduced costs, network utilization, and energy consumption. The work's restriction is that time and cost measurements may be incorporated into this research. **Yadav et al. [12]** have developed a modified fireworks algorithm for reducing makespan and cost. Researchers employed opposition-based learning and differential evaluation methodologies with a modified firework algorithm. For the efficiency of an algorithm, researchers have concentrated on the exploration and exploitation phases. Con-vergence can be increased by incorporating both phases and algorithms cannot become trapped in local optima. The proposed strategy is compared to the existing meta-heuristic technique on parameter cost and makespan, and its importance is verified. Work limitations the makespan metric still needs improvement. The Hidden Markov Model was explored by **Javaheri et al. [13]** for anticipating available Fog nodes for incoming requests, workflows that have missed the deadline, and offloading jobs from the Fog to the cloud. The Baum-Welch algorithm is utilized for HMM training, and the Viterbi technique is used to discover accessible Fog nodes. DO-HHO (Discrete-opposition-Harris Hawks optimization algorithm) has been presented by researchers for effective workflow scheduling with the parameters cost, deadline, and energy. The proposed technique successfully decreases delay, workload offloading, and SLA violations. The research's shortcoming is the high amount of energy squandered in this procedure. **Abu-Amssimir et al. [14]** have suggested a greedy-edge-placement strategy for latency-sensitive IoT applications that minimize delay. The proposed technique efficiently maximizes throughput while minimizing delay. Researchers are working on the greedy-delay-minimizing applications selection and placement step for the suggested strategy. The selection stage achieved lower end-to-end latency, as did the placement stage, which used the DFS algorithm to pick Fog nodes for module application placement. The suggested approach successfully provides high-quality services while lowering net-work bandwidth, latency, and energy usage. The work's weakness is that reaction time may need to be calculated to demonstrate network efficiency. **Maatoug et al. [15]** developed a Discrete-event-system-specification mechanism for smart building energy management. Different sub-models are developed in the suggested technique by connecting multiple objects in the building that create the Fog environment. The developed technique effectively reduces latency and energy usage. The proposed model can be implemented by eliminating or adding a sub-model that aids in the achievement of efficient

QOS in a Fog environment. Some essential QoS metrics, such as makespan and reaction time, are absent and insufficient to ensure QoS. **Ogundoyin et al. [16]** have described a hybrid optimization algorithm based on the Particle Swarm Optimization and the Firefly algorithm. Researchers discussed the issue of sojourn rate and trust, energy usage, and node capacity. The liner-sum-weight approach is employed to consolidate an individual objective function, and the best-worst method is utilized to establish the weight vector for the aggregate function. The proposed technique surpasses traditional methods in performance. **Hussain et al. [17]** presented a novel vehicular fog computation approach. The authors considered the delay issue as well as energy-sensitive automobile applications. Researchers concentrated on data offloading from automotive devices to RSUs and BSs (base stations). The authors suggested a multi-objective optimization problem named Swarm-Optimized Non-dominated sorting-Genetic algorithm (SONG) in this paper. When compared to the NSGA-2 and SMPSO approaches, the new approach gives higher quality than previous strategies. The network's response time improves just somewhat over previous solutions. **Ghobaei-Arani et al. [18]** have developed a meta-heuristic approach for enhanced service allocation. The authors presented a whale optimization method for improved resource management in the Cloud-Fog system, and they used throughput and energy consumption as objective functions to determine the system's efficiency. The simulation findings show that the proposed strategy minimizes delay and energy consumption while improving resource utilization. Some drawbacks in the currently used methodologies are observed and are detailed below on various parameters examined:

1. A lack of variety in the workload: To evaluate task scheduling algorithms' efficacy and flexibility, a variety of workloads and application kinds should be used [10] [17]. However, some studies could concentrate on workload characteristics or neglect to take a variety of application requirements into account.
2. Little Attention Paid to Communication Overhead: In cloud-fog environments, effective task scheduling must consider communication overhead between fog nodes and the cloud as well as between fog nodes. Some publications, however, might not sufficiently handle this issue or might use oversimplifying suppositions that don't account for actual communication limitations [11][14][15][17].
3. Inadequate Resource Management Methodologies: Effective task scheduling should include resource management techniques like load balancing, fault tolerance, and scalability in addition to job allocation. Some works might ignore these factors or offer scant guidance on how to manage them in a cloud-fog architecture [11][12][14][15][16].
4. Cost and energy efficiency trade-offs: Finding the ideal balance between cost and energy efficiency can be difficult, although both are crucial considerations in task scheduling. Studies that place too much emphasis on one feature while disregarding the other may produce less-than-ideal results [13][15] [16][17][18].

We present the notations used in our suggested solution as seen in Table 1.

TABLE I. NOTATION AND DEFINITIONS

| Notation | Definition |
|---|---|
| $F_{ri}^{time}$ | Finishing time of the Task |
| $S_{ri}^{time}$ | Starting time of the Task |
| $Wt_{time}$ | Waiting time |
| T | Current iteration |
| $\overline{D}$ | Distance |
| Tmax | Maximum iteration |
| NP | Earthworm Population Size |
| nKEW | Number of earthworms |
| $\psi_{Comp\text{-}cost}$ | Computation Cost |
| $\psi_{Comm\text{-}cost}$ | Communication Cost |
| ψcost(m) | Cost of Memory |
| ψcost(p) | Cost of CPU |
| ψcost(b) | Cost of Bandwidth |
| MS | Makespan |
| RT | Response time |
| Wt time | Waiting Time |
| ART | Average Response time |

## III. SYSTEM ARCHITECTURE

The three-tier framework of a cloud-Fog-IoT computing system is depicted in Figure 1. The first-tier layer is made up of (D1, D2,... Dn) IoT devices, and it contains data that will be transferred to the top layer for processing. The intermediate Fog layer (f1, f2...fn) has mini servers with limited storage and processing capability [19][20]. Each Fog node completed operations that were atomic and unrelated to one another. Tasks that cannot be completed or handled by Fog devices are routed to the cloud layer. The third cloud layer is outfitted with high-end servers with high execution capability and high storage capacity virtual machines (VM1, VM2,....VMn).

***Article Received:** 15 August 2023 **Revised:** 25 September 2023 **Accepted:** 06 October 2023*

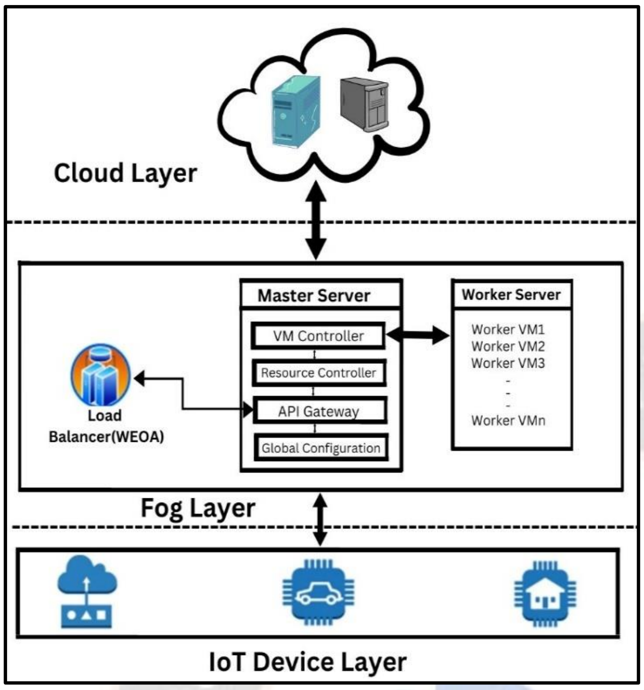

Figure 1. Three-tier framework of a cloud-Fog-IoT computing system with improved Load Balanced – WEOA.

The WEOA is implemented via the Load Balancer [21], [22] in Figure 1. The tasks are initially directed to the Load Balancer, when the product of the number of tasks and instructions exceeds a certain threshold. When this limit is exceeded, the load balancer contacts the API gateway, which then instructs the VM Controller to generate more worker virtual machines [23]. All machine parameters are retrieved by the API gateway from the VM Controller and then sent to the load balancer. Using the WEOA approach, the load balancer chooses the best computer, assigns tasks to the selected machine, and sends them to the resource controller over the API Gateway. The "Global Configuration" stores the metadata of the optimal machine and makes it accessible to subsequent cycles. All tasks are distributed to worker VMs by the "Resource Controller". To save energy, the "VM Controller" effectively removes inactive VMs. The "Resource Controller" then sends data from finished jobs to the cloud layer. Depending on the size of the device, tasks may be distributed to either Fog or Cloud devices. As tasks are executed, task offloading may occur, either from Fog to Cloud or from Cloud to Fog [24].

## Problem Formulation

Figure 1 depicts the task scheduling system, where Task T=(t1,t2,....tn) represented the jobs received from IoT devices. {FN1, FN2,....FNn} are the fog nodes, and (CN1........CNn) are the cloud nodes, which are responsible for task execution. However, there is an issue with the efficient assignment of tasks to these nodes in the Fog-Cloud scenario. The overall number of server nodes in the system is the sum of Fog and Cloud nodes. These nodes are all heterogeneous in terms of bandwidth (Nb) and processing speed (Nps). The Fog nodes process the tasks first, followed by the cloud nodes. The primary motivation for this research is to determine how to efficiently assign jobs to these nodes to minimize cost, time, and response time. When allocating jobs to nodes, our load balancer algorithm will consider all these limits.

The problem of vehicular Fog computing is considered in this paper, where autonomous cars face traffic management issues in some nations. A car with no drivers collects data from RSU (roadside units) about traffic, potholes, and humps, among other things. In this case, real-time data is essential so that a braking system can function properly and avert any accidents. We are dealing with response time, cost, and makespan parameters in this study because supplying real-time information to automatic cars requires a high response time.

### *A. Objective Function*

The motivation of this work is to minimize the cost, makespan, and response time. These three parameters are considered in this work since most applications, such as self-driving cars, healthcare, gaming, and media, require the quickest response time, the largest throughput, and the lowest operational cost. The WOA is based on the bubble net attack tactic used by humpback whales. To update the positions of search agents, one of three strategies is used: random search, shrinking encircling, or spiral feeding.

### *B. Cost*

Nodes, whether Fog nodes or cloud nodes, have an inherent cost connected with the tasks they perform in a system [25]. Costs for task processing and data transfer are constant factors in the system. By examining the costs related to the nodes' consumption of memory, computing power, and bandwidth, the computation costs may be calculated. On the other hand, communication cost includes the time needed for resource allocation, which may be measured by taking into account network latency, the time needed to run resource management algorithms, and the number of devices competing for resources. Combining the expenses of processing and communication results in the overall network cost, which is represented by Equation 1.

$$\text{Total-Cost} = \psi_{\text{Comp-cost}} + \psi_{\text{Comm-cost}} \quad (1)$$

### *C. Computation Cost*

The cost of memory, processing, and bandwidth utilized by the nodes can be used to calculate computation costs.

$$\psi_{\text{Comp-cost}} = \psi\text{cost}(m) + \psi\text{cost}\ (p) + \psi\text{cost}(b) \quad (2)$$

The cost of memory utilization, the cost of CPU, and the cost of bandwidth as described in Equation 2 can be used to calculate the Computation Cost of a system.

$$\psi\text{cost}(m)=\sum_{i=1}^{m} c_i^m \qquad (3)$$

The cost of memory taken by nodes Ni during job completion is described in Equation 3. It is calculated using the cumulative memory used by fog nodes and clouds.

$$\psi\text{cost}(p)=\sum_{j=1}^{n} c_j^p \qquad (4)$$

Similarly, Equation 4 above describes the cost of using the CPU by nodes Ni during task processing. Total CPU cost is the sum of cloud and fog costs, as shown in Equation 5.

$$Total_{CPU}^{Cost} = C_{CPU}^{cloud} \cup C_{CPU}^{Fog} \qquad (5)$$

The cost of bandwidth must also be factored into the overall system cost. Bandwidth utilization on Fog nodes and the cloud may be the same. The total bandwidth utilized by both fog and cloud during the system's working period can be used to calculate bandwidth cost, as shown in Equation 6.

$$\psi\text{cost}(b)=\sum_{k=1}^{l} c_k^b \qquad (6)$$

### *D. Communication Cost*

The time required to allocate resources is known as the communication cost, and it may be calculated by taking into account system network latency, the amount of time needed to run resource management algorithms, and the number of devices actively looking for resources.

$$\psi_{\text{Comm-cost}} = T_{\text{net-latency}} + T_{\text{algo-exec}} + N_{\text{Dev}} \qquad (7)$$

Network latency ($T_{\text{net-latency}}$), as shown in the Equation 7, has a significant impact on how long resources are allocated between Fog nodes and devices. Reduced latency, measured in seconds (s), results in faster resource allocation. Additionally, the time used ($T_{\text{algo-exec}}$), which is likewise measured in seconds, is strongly impacted by the resource management algorithm's effectiveness in the fog environment. An algorithm that has been carefully optimized can quickly find available resources and distribute them more effectively. The total number of devices ($N_{\text{Dev}}$) linked to the Fog environment might also affect how quickly resources are allocated. More time may be needed for resource allocation and management if there are more devices to handle.

Total cost in a system can be calculated using equations 2 and 7 as shown in Equation 8.

$$\text{Total-Cost} = (\psi\text{cost}(m) + \psi\text{cost}(p) + \psi\text{cost}(b)) + (\text{Tnet-latency} + \text{Talgo-exec} + \text{NDev}) \qquad (8)$$

### *E. Makespan*

Makespan can be calculated by subtracting the time of job completion from the time of task start. It is the time it takes to complete a task from start to end. Makespan [26][27] is calculated as shown in Equation 9:

$$\text{MS}= \min \{F_{ti}^{time} - S_{ti}^{time}\}, \text{ where ti } \varepsilon \text{ T, where ti } \varepsilon \text{ T} \quad (9)$$

Here, $F_{ti}^{time}$ is the task's completion time, and $S_{ti}^{time}$, is the task's start time. Makespan must be kept to a minimum for effective job scheduling.

### *F. Response-Time*

The machine's response time to any inquiry is defined as response time. Response times [28, 29] can be estimated for n concurrent user processes and m requests.

$$\text{RT}= n/m + Wt_{\text{time}} \qquad (10)$$

In the preceding Equation 10, numerous inquiries will be raised by many users n, and all of these inquiries m will be responded to by machines at any given time. Waiting time ($Wt_{\text{time}}$) between two jobs might be included to calculate response time accurately. To process any inquiry, a system may have a long response time. The average response time is the time it takes machines to complete n tasks, where (Ti=t1, t2,...tn). The average response time can be calculated using the previously mentioned equation 10a.

$$\text{ART}= \sum_{i=1}^{m} Ti \Big/ n \qquad (10a)$$

Ti is the machine's individual time for task completion, and n is the number of tasks completed.

### *G. objective-Function*

According to the preceding explanation, the objective function of our task, as stated in Equation 11, is to minimize cost, time, and response time.

**Objective-function**= we1 * ψcost + we2* MS + we3 * RT (11)

Where we1, we2, and we3 are the relevant metric weights. The weightage of cost, time, and response time must be equal to one, as described in Equation 12.

$$\sum_{i=1}^{3} wei = 1 \qquad (12)$$

## IV. Proposed Algorithm

This section discusses the algorithm we suggest. WEOA is a hybrid of whale optimization and Earthworm optimization algorithms. The advantages of both optimization techniques are used to improve the system's efficiency.

In our suggested strategy, we particularly use the reproduction 2 method from the original Earthworm Algorithm. This method makes use of the crossover operation in the Earthworm Algorithm to make multiple machine replication easier. Additionally, we use the Whale Optimization Algorithm to hunt down and encircle tasks, or "prey," in this context. In the end, this combination strategy aids in the finding of the best solutions for resource management within the Fog-Cloud system by speeding up the exploration phase and convergence speed.

### *A. Initialization*

The arrangement of the whales represents a practical response to an optimization challenge. This project's main goal is to evaluate how well tasks and machines are mapped out. The earthworms and whales are both initialized in the first step of this process. The prevalence of earthworms contributes to the ease of whale reproduction. To find the most optimal solution within the search space, a maximum number of whale positions is randomly chosen.

**Algorithm: The WEOA pseudo-code**

```
Input: $Max_{iteration}$, $Threshold_{Task-Size}$.
Output: Efficient Depict of tasks by reducing ψcost, MS, RT.
Procedure: Start
    1.  Initialize population W of whales and Earthworms.
    // Count of Machine M and Number of Tasks.
    2.  Evaluate the Fitness Function.
    3.  While $current_{itr}$ <= $Max_{iteration}$ do
    4.  if prey > TaskThreshold then
    5.  Evaluated the fitness of Whales.
          // Check Task-Size based on the instructions.
    6.  Cross-over reproduction from fitness and append to
        (W) as equation 13.
          // Initializing heterogeneous machines from fitness.
    7.  elif $Encircling_{prob}$ > 0.5 then
    8.  Update the whale position using encircling equation 14.
//Update global configuration with the optimal machine.
            else do
    9.  Update the whales position randomly.
  //Update global configuration with an optimal machine.
    10. End if
    11. End if
    12. End-While
    13. Search for an optimal solution from $global_{config}$
    14. Optimal Solution Found.
    15. End Procedure
```

**Iteration Process and Fitness Function**

The Fitness function and subsequent algorithmic stages are focused on iteration, which is essential for determining the best search agent within the search space. The number of iterations that can be performed, known as the maximum iteration value or $Max_{iteration}$, determines how many iterations can be performed. The task size threshold value is used to define the job size for each iteration. The provided equation is used to evaluate the machine's fitness when the job size exceeds this limit. The Earthworm Optimization Algorithm is used to establish the reproduction plan if the machine's fitness no longer matches the threshold value. The optimization algorithm progresses through both the exploration and exploitation stages during each iteration.

### *B. Exploration phase based on Earthworm strategy.*

The Earthworm optimization technique is used in the exploration phase of this algo-rithm. Machine reproduction or replication will proceed according to equation 13:

$$Xy(i) = \begin{cases} XA(i), \ 1 \le i < k \\ XB(i-k), \ k < i \le n \end{cases} \quad (13)$$

Let A and B be the two machines; we will reproduce the new machine Y using A and B. Let XA represent machine A's feature vector, XB represent machine B's feature vector, and Xy represents machine Y's feature vector. Because we are using crossover reproduction (EOA), the offspring will have some random characteristics from A and some from B. The above equation demonstrates how the traits in Y are inherited from A and B.

According to the preceding equation, the "ith" feature of Y will be an ith feature from the parent A if $1 \leqslant i < k$, and the "(i - k)th" feature from B if $k < i \leqslant n$. where "k" is the cross-over point and is a random integer between 1 and n, and "n" is the length of A and B's feature vectors. As a result, the offspring Y will be a hybrid species with random features of A and B, indicating cross-over reproduction.

### *C. Encircling the prey*

In this algorithm, the exploration phase is based on the Earthworm optimization technique. Reproduction or replication of machines will occur as per equation 14:

$$\begin{gathered} X(t+1) = X(t) + A * \cos(a * t) * \cos(b * t) \\ Y(t+1) = Y(t) + A * \cos(a * t) * \sin(b * t) \qquad (14) \\ Z(t+1) = Z(t) + A * \sin(a * t) \\ \vec{X}(t+1) = X(t+1)\,\hat{\imath} + Y(t+1)\,\hat{\jmath} + Z(t+1)\,\hat{k} \end{gathered}$$

The encircling motion of the whales is represented by the preceding equation, and the functions x(t), y(t), and z(t) give the x, y, and z coordinates of the whale in 3-dimensional space, respectively. The amplitude of the whale's spiral movement is represented by A in the above equation, while a and b are two whale-specific angles. The value of t varies from 0 to 1, indicating the progression of the spiral movement through time. The final equation is used to convert vectors from coordinates

*International Journal on Recent and Innovation Trends in Computing and Communication*
***ISSN: 2321-8169 Volume: 11 Issue: 10***
***Article Received:** 15 August 2023 **Revised:** 25 September 2023 **Accepted:** 06 October 2023*

returned by the functions. As a result, X is the whale's position vector.

### *D. The exploitation phase is based on the Whale attacking mechanism.*

The exploitation phase aids in the hunt for prey; in our suggested approach, the exploitation phase is carried out using the whale optimization algorithm, as shown in Equation 15:

$$\vec{X}(t+1) = m.\,(\vec{P} - \vec{X}(t)) + \vec{X}(t) \qquad (15)$$

Let $\vec{P}$ be the prey's location vector and X(t) be the whale's position vector at time t. The above equation represents the whale's attacking character. The movement factor "m" here is between [0, 1]. It is determined by things such as the machine's specifications or the behavior of other whales in the population. The operation continues until the maximum number of iterations is reached. The optimal solution is derived from the global configuration upon completing the maximum iterations. If the final ideal solution is identified from the global configuration, the procedure concludes.

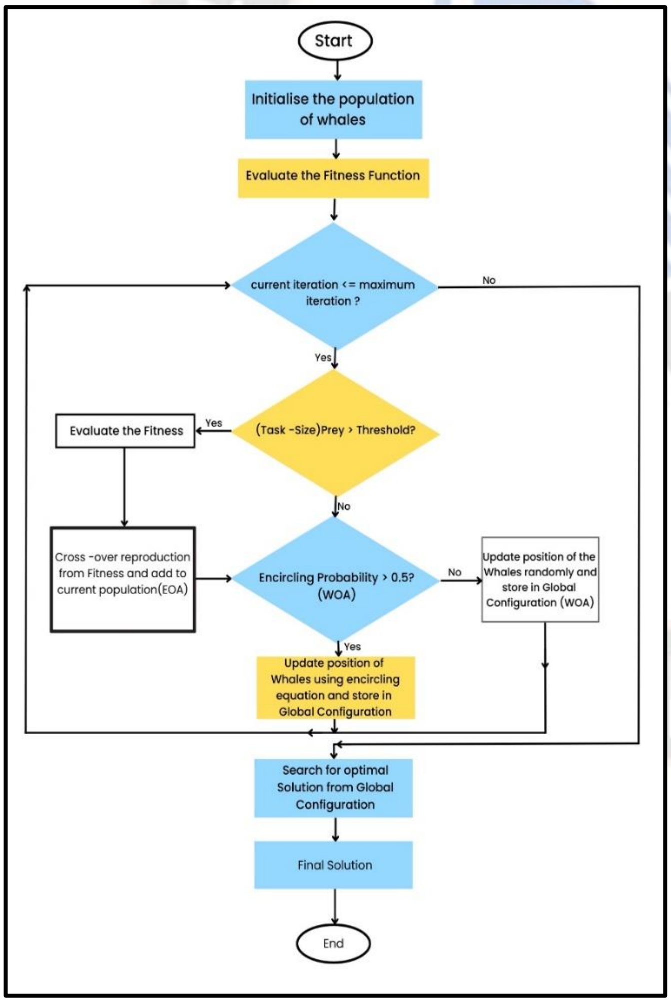
Start
Initialise the population of whales
Evaluate the Fitness Function
current iteration <= maximum iteration ?
No
Yes
Evaluate the Fitness
Yes
(Task -Size)Prey > Threshold?
No
Cross -over reproduction from Fitness and add to current population(EOA)
Encircling Probability > 0.5? (WOA)
No
Update position of the Whales randomly and store in Global Configuration (WOA)
Yes
Update position of Whales using encircling equation and store in Global Configuration
Search for optimal Solution from Global Configuration
Final Solution
End

Figure 2. The proposed Whale Earthworm Optimization Algorithm Flowchart.

Figure 2 depicts the proposed WEOA process. The flowchart begins with the initialization phases, which include the initialization of whales and earthworms. The method is continued until an optimal solution is identified based on the iteration number specified. The fitness function of the machines is estimated using Equation 11, which considers cost, makespan, and response time. The entire system is employed in vehicular Fog computing situations where automatic automobiles are operating on roadways and real-time data without delay is required for automatic vehicles to run. According to the Performance improvement rate stated in the result section for the vehicular Fog computing environment, response time is weighted (we1) at (0.60), makespan is weighted (we2) at (0.20), and the cost is weighted (we3) at (0.20). Response-time weightage is substantially higher since data on traffic bottlenecks, humps, and when to apply breaks are required on a high priority without delay for making decisions and avoiding an accident.

## V. RESULT AND DISCUSSION

This section delves into the experimental setup and contrasts it with conventional procedures. All tests are conducted using the iFogSim toolkit [37] on a system configured with an Intel Core i7 processor, Windows 10 operating system, and 16GB of RAM.

### *A. Experiment setup and dataset statistics*

The data workflows used to assess the performance of the proposed approach are extracted from real-world datasets, namely, HPC2N and CEA-CURIE (source: https://www.cs.huji.ac.il/labs/parallel/workload, accessed on October 20, 2023). These workload logs encompass execution traces resulting from concurrent HPC workload processing. Table 2 depicts the actual workloads employed in this work. In a real-world scenario, HPC2N and CEA-Curie have many workloads, although we only used ten in this experiment.

TABLE II. ILLUSTRATION OF REAL WORKLOADS

| Workload Log | Parallel Tasks | CPUs | Users | Filename |
|---|---|---|---|---|
| HPC2N | 202871 | 240 | 257 | HPC2N-2002-2.2-cln.swf |
| CEA Curie | 312826 | 93312 | 582 | CEA-Curie-2011-2.1-cln.swf |

Both Fog and cloud nodes are used for simulation processing. The experimental setup was evaluated using the parameters Cost, makespan, and response-time. Each machine on Fog and

Cloud has a different amount of bandwidth, processing power, and RAM. In comparison to the cloud, Fog nodes have a lower range of bandwidth, CPU frequencies, and RAM utilization. The cost is expressed in terms of (Grid$). Table 3 describes the Cloud and Fog scenario's configuration details.

TABLE III. CONFIGURATION OF CLOUD AND FOG SCENARIOS.

| Parameters | Fog | Cloud | Units |
|---|---|---|---|
| Processing Speed | [1000:2000] | [3000:5000] | MIPS |
| Bandwidth | [128:1024] | [512:4096] | Mbps |
| RAM | [250:5000] | [5000:20000] | MB |
| Cost | [0.2:0.5] | [0.6:1.0] | G$ |
| VMs Numbers | [15,20,35] | [10,15,20] | VM |

## *B. Simulation results*

For simulation results evaluation, the suggested strategy is compared to an existing technique that was implemented utilizing optimization algorithms such as whale optimization, earthworm, cuckoo search, and so on. We have compared our proposed approach with five existing techniques including h-DEWOA (hybrid-differential-evolution-enabled whale-optimization algorithm) [38], cuckoo-search- differential algorithm (CSDEO) [39], Cuckoo-search-particle-swarm-optimization algorithm (CSPSO) [40], blacklist matrix-based-multi-objective algorithm (BLEMO) [41], and EEOA (Electric-earthworm-optimization algorithm) [10]. All comparisons are performed over 30 iterations across 10 workloads from the HPC2N and CEA-CURIE datasets.

## *C. Results for CEA-Curie workload*

Figure 3 depicts the performance metric makespan comparative results for the suggested strategy with all five strategies discussed. According to the comparison data, CSPSO did not outperform the other techniques; however, the value of makespan for CSPSO is significantly larger than the other strategies. The differential evolution technique is integrated with the whale algorithm in the h-DEWOA approach since the whale algorithm has the problem of remaining in local optima and having a slow convergence speed. h-DEWOA also has poor makespan results. On the performance measure makespan, the suggested strategy beats all the mentioned techniques. The approach benefits from dynamic machine allocation and a load balancer approach, which constantly replicates the machine if jobs with higher threshold values come during iterations. For earlier workloads (WE01-WE05) proposed approach performance is much better in comparison to other techniques due to independent tasks and availability of resources & VMs. in later workloads (WE06-WE10), tasks are highly dependent and resource availability get lesser, that’s why makespan value difference among proposed and traditional technique is noticed less.

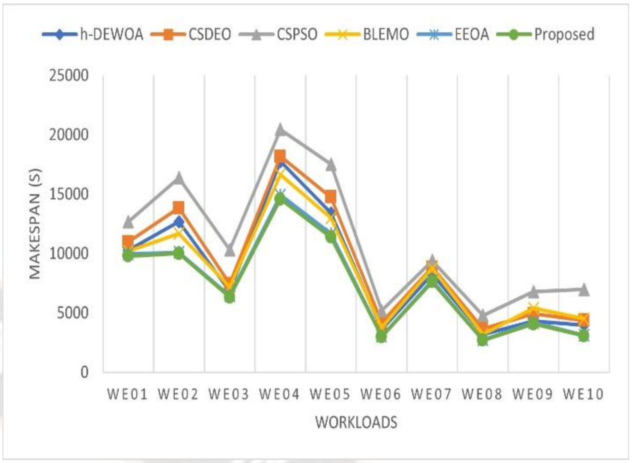

Figure 3. Best Makespan for CEA-Curie workload

As indicated in Table 4, the performance improvement rate is calculated to determine the percentage of improvement of the suggested method using all described techniques as shown in Equation 16.

$$\frac{p\,(pre) - p\,(pro)}{p\,(pro)} \qquad (16)$$

Here, $p_{pre}$ is the old approach's makespan value, and $p_{pro}$ is the suggested approach's makespan value. As illustrated in Table 4, the proposed strategy outperforms the techniques. WE01 best case makespan value of EEOA is 9987.45 for workload, and 9788.07 for the recommended technique, according to the PIR equation $\frac{9987.45 - 9788.07}{9788.07} \approx 2.04$ %. As a result, the proposed technique outperforms the EEOA by 2%.

TABLE IV. THE PERCENTAGE OF PERFORMANCE IMPROVEMENT FOR THE BEST MAKESPAN IS INCLUSIVE OF THE CEA-CURIE WORKLOAD IN THE PROPOSED APPROACH.

| PIR (%) of the proposed approach with all existing techniques | | | | | |
|---|---|---|---|---|---|
| Workloads | Proposed Vs h-DEWOA (%) | Proposed Vs CSDEO (%) | Proposed Vs CSPSO (%) | Proposed Vs BLEMO (%) | Proposed Vs EEOA (%) |
| WE01 | 4.98 | 12.07 | 29.50 | 4.21 | 2.04 |
| WE02 | 26.83 | 38.54 | 63.81 | 17.03 | 1.05 |
| WE03 | 7.82 | 17.50 | 62.80 | 13.68 | 1.94 |
| WE04 | 21.78 | 24.61 | 40.16 | 14.14 | 2.38 |
| WE05 | 17.80 | 29.58 | 53.86 | 14.19 | 2.49 |
| WE06 | 20.92 | 40.36 | 73.97 | 26.87 | 1.75 |
| WE07 | 8.55 | 15.97 | 23.43 | 15.03 | 1.15 |
| WE08 | 18.28 | 34.71 | 76.54 | 17.25 | 1.38 |
| WE09 | 5.33 | 20.53 | 65.84 | 32.03 | 2.69 |
| WE10 | 30.08 | 42.46 | 127.87 | 46.76 | 1.83 |

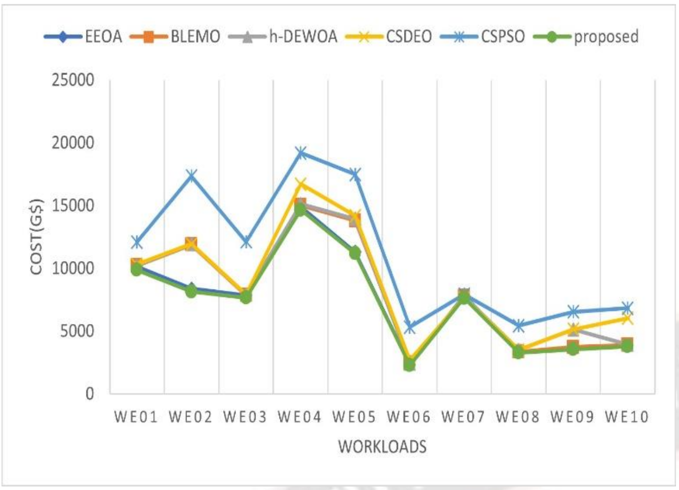

Figure 4. Cost comparison for CEA-Curie workload

Figure 4 depicts a cost comparison of the CEA-Curie workload, with CSPSO having a greater cost in the system. For workloads WE01-WE08, BLEMO, h-DEWOA, and CSDEO have roughly the same cost. Our proposed method outperforms the EEOA technique and improves performance by 2% on the CEA-curie workload. Because VMs allocate properly, the suggested technique operates efficiently and at a lower cost value. The network is managed jointly by the load balancer and the resource controller. If the load balancer sends an inquiry about task mapping via the API gateway, the resource controller will verify the available resources on each machine. Because no idle machines will remain in the network, the proposed technique will be less expensive.

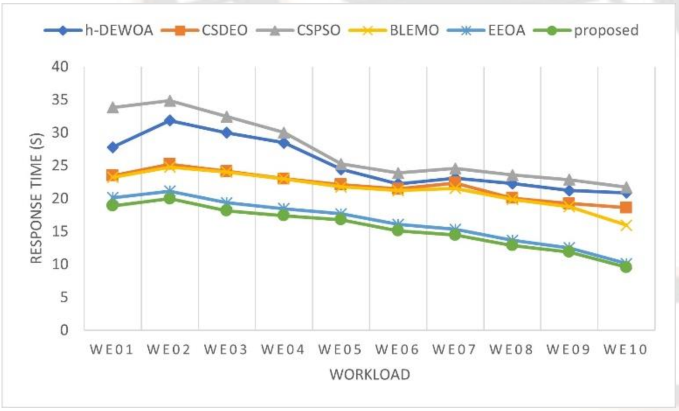

Figure 5. Response-time comparison for CEA-Curie workload

On the CEA-Curie workload, Figure 5 depicts a response-time comparison with existing approaches. In the system, response time is calculated by combining communication and think time. Because the CSPSO approach has a longer delay, its RT is higher when compared to other techniques. CSDEO and BLEMO task mapping are nearly identical. Both techniques are vector-based, with a vector maintained for mapping tasks to VMs. When compared to existing approaches, the proposed methodology has a low RT. The whale optimization exploitation phase aids in responding quickly to jobs. In comparison, our algorithm performed admirably across all workloads. In terms of performance, the proposed strategy outperforms the techniques. The EEOA WE01 RT value for the workload is 20.14, and the proposed method is 18.89, according to the PIR calculation, $\frac{20.14-18.98}{18.98} \approx$ 6.11 %. As a result, the proposed technique outperforms the EEOA by 6%.

## *D. Results for HPC2N Workload*

Figure 6 compares the suggested strategy to all other techniques in terms of performance metric makespan. Figure 3 shows that the CSPSO approach performs poorly on the HPC2N workload for the Performance metric makespan. CSPSO's task mapping technique is unable to operate on a high-performance network. On 10 workloads, the h-DEWOA and BLEMO methods perform equally well after 30 iterations. Our proposed strategy outperformed existing techniques and performed better. Even for the High-performance network dataset, VMs in the proposed technique are correctly handled by the VM controller and resource controller. Here, the Exploitation phase, which is controlled by the whale optimization algorithm, is doing admirably.

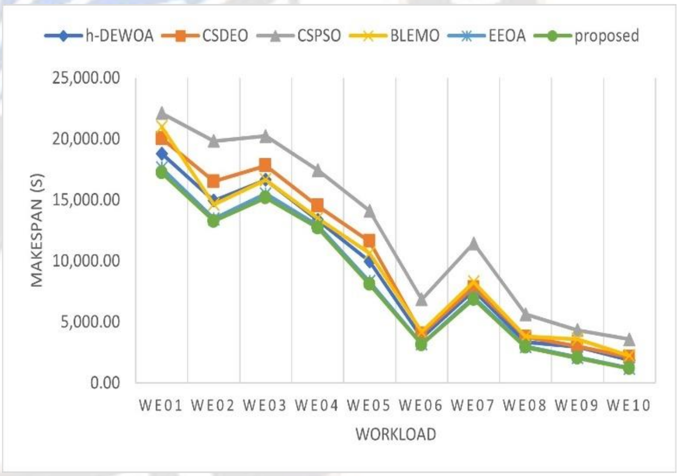

Figure 6. Best Makespan for HPC2N Workload

The proposed technique's performance increase rate is also determined for the HPC2N workload in comparison to previous strategies using Equation 16, as shown in Table 5. According to the PIR equation, the best case makespan value of EEOA for workload is 17612.19 for WE01 and 17225.72 for the proposed approach, $\frac{17612.19-17225.72}{17225.72} \approx$ 2.24%. Because of this, the proposed technique outperforms EEOA by 2%.

TABLE V. THE PERCENTAGE OF PERFORMANCE IMPROVEMENT FOR THE BEST MAKESPAN IS ACHIEVED WITH THE INCLUSION OF THE HPC2N WORKLOAD IN THE PROPOSED APPROACH.

| **PIR (%) of the proposed approach with all existing techniques** | | | | | |
|---|---|---|---|---|---|
| **Workloads** | **Proposed Vs h-DEWOA (%)** | **Proposed Vs CSDEO (%)** | **Proposed Vs CSPSO (%)** | **Proposed Vs BLEMO (%)** | **Proposed Vs EEOA (%)** |
| WE01 | 6.75 | 13.76 | 25.53 | 19.66 | 2.25 |
| WE02 | 11.16 | 22.83 | 47.36 | 8.99 | 1.45 |
| WE03 | 7.32 | 14.81 | 30.45 | 7.49 | 2.22 |
| WE04 | 3.53 | 12.42 | 35.17 | 4.81 | 1.58 |
| WE05 | 19.73 | 39.48 | 69.51 | 28.71 | 2.99 |
| WE06 | 15.48 | 25.24 | 78.33 | 31.58 | 2.12 |
| WE07 | 7.80 | 11.31 | 62.77 | 18.25 | 2.25 |
| WE08 | 11.71 | 26.43 | 78.17 | 28.53 | 1.93 |
| WE09 | 41.37 | 43.13 | 79.44 | 73.08 | 2.40 |
| WE10 | 54.80 | 75.91 | 80.66 | 78.36 | 1.88 |

Figure *7* compares the overall cost for the HPC2N workload, where cost is expressed in G$. The cost factor is defined by the bandwidth, CPU, and RAM of the parameter.

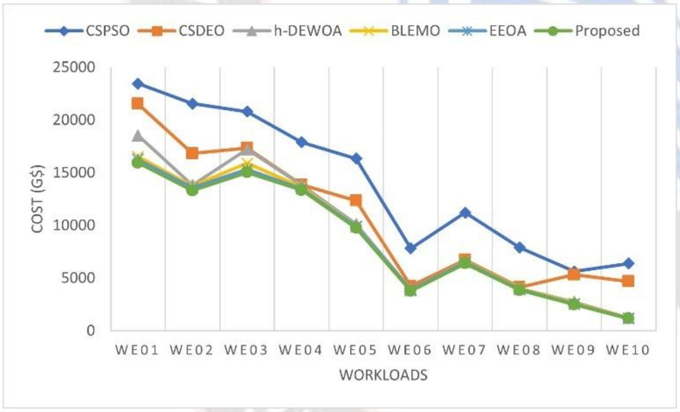

Figure 7. Cost comparison for HPC2N workload

According to Figure 7, the cost of CSPSO is substantially higher than the cost of BLEMO and h-DEWOA. All techniques' costs for workloads WE04 and WE05 have been raised. In comparison to EEOA and other approaches, the proposed technique is less expensive. Over the EEOA strategy, the proposed approach outperforms it by 2%. The proposed solution reduces the cost of employing VMs by generating them dynamically based on the requirements [42]. However, in the case of the CSPSO approach, all VMs are initialized without regard for network load. As a result, the CSPSO technique is substantially more expensive than other procedures.

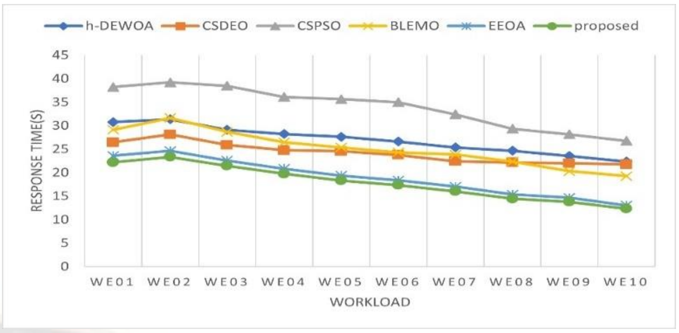

Figure 8. Response-time comparison for HPC2N workload.

Figure 8 depicts a comparison of Response time on the HPC2N workload with existing approaches. Similarly, for the HPC2N task, the delay in the CSPSO approach is greater, hence the RT of the CSPSO technique is greater when compared to all other techniques. In contrast to existing approaches, our proposed methodology exhibits a reduced response time (RT). Notably, our algorithm consistently performed exceptionally well across all workloads, outperforming existing techniques in terms of overall performance. Specifically, for the workload, the EEOA yielded a WE01 RT value of 23.58, whereas our proposed technique achieved an improved RT of 22.18. Consequently, based on the PIR calculation, there is a notable improvement of approximately 6.31%. Thus, our proposed technique surpasses the EEOA by 6% in terms of performance.

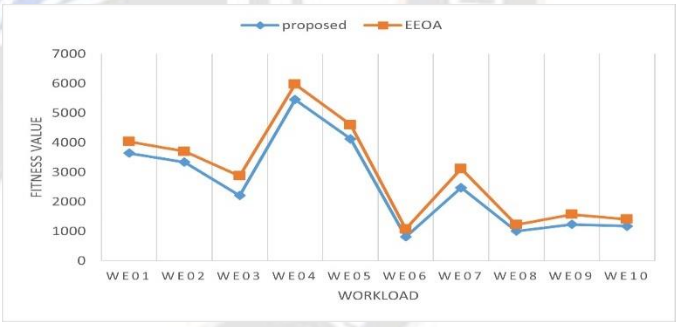

Figure 9. Objective Function value throughout a single run of the proposed algorithm for CEA-Curie workload.

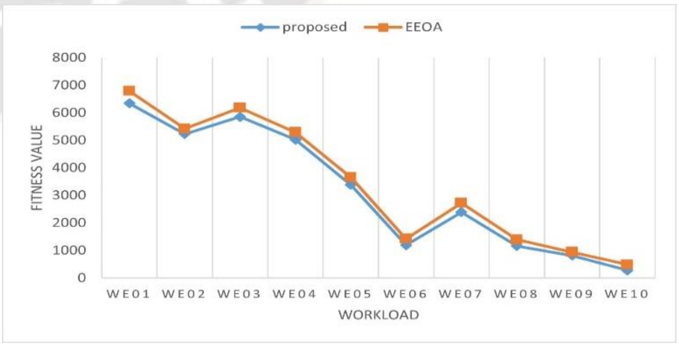

Figure 10. Objective Function value throughout a single run of the proposed algorithm for HPC2N workload.

Figures 9 and 10 display the objective function values throughout a single run iteration for both the CEA-Curie and HPC2N algorithms. As per Equation 11, the fitness value for both the proposed and EEOA algorithms is calculated, considering the values of We1 and We2, which are 0.2 and 0.6, respectively. The suggested WEOA algorithm exhibits quicker convergence compared to the EEOA algorithm. The results indicate that, when contrasted with the EEOA algorithm, the suggested WEOA algorithm effectively explored the solution space and reached the optimal value within a comparable timeframe.

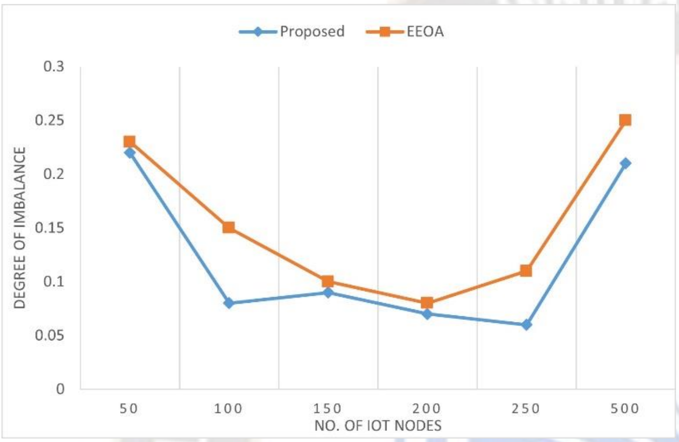

Figure 11. Degree of imbalance of the offloaded the tasks.

Figure 11 depicts the degree of imbalance for the EEOA algorithm and the proposed approach using the same previous parameter settings as the number of IoT nodes increases. The graph demonstrates that the suggested WEOA algorithm maintains less values. This suggests that the proposed approach successfully distributes the workload across the Fog nodes.

TABLE VI. ANOVA STATISTICAL ANALYSIS: TWO FACTOR WITHOUT REPLICATION FOR DEGREE OF IMBALANCE

| Source of Variation | SS | df | MS | F | P-value | F crit |
|---|---|---|---|---|---|---|
| Rows | 0.050375 | 5 | 0.010075 | 30.68528 | 0.000929 | 5.050329 |
| Columns | 0.003008 | 1 | 0.003008 | 9.162437 | 0.029183 | 6.607891 |
| Error | 0.001642 | 5 | 0.000328 | | | |
| | | | | | | |
| **Total** | **0.055025** | **11** | | | | |

Table 6 shows the results of an ANOVA statistical analysis of two factors without replication on the degree of imbalance. The proposed algorithms outperform traditional algorithms in terms of analytical efficiency. In statistical analysis, a p-value less than 0.05 and an F crit value less than F demonstrate the capability of the suggested approach. According to this analysis, the proposed approach is superior for handling optimization challenges on the IoT-Fog-Cloud system.

## VI. CONCLUSION AND FUTURE WORK

The novel WEOA is a hybrid algorithm that combines elements of the Whale Optimization and Earthworm Optimization algorithms within the load balancer which effectively strikes a balance between exploration and exploitation in these approaches. an effective task allocation technique is introduced via the load balancer. The proposed algorithms are compared against five competing optimization techniques, considering key parameters such as cost, makespan, and response time using the HPC2N and CEA-CURIE datasets. The approach using the load balancer outperforms h-DEWOA, CSDEO, CSPSO, BLEMO, and EEOA by up to 6% in response time, 2% in cost, and 2% in makespan compared to EEOA. Furthermore, it achieves remarkable results, surpassing h-DEWOA, CSDEO, CSPSO, and BLEMO by up to 82% in response time, up to 75% in cost, and up to 80% in makespan. In the future, task offloading could incorporate machine learning models to anticipate and predict job arrivals and placement order, potentially enhancing Quality of Service (QoS) optimization in fog-cloud-based systems.

## REFERENCES

[1] N. EA, D. Tamilarasi, S. Sasikala, R. R. Nair, and K. . Uma, “An Efficient Food Quality Analysis Model (EFQAM) using the Internet of Things (IoT) Technologies,” Microprocess. Microsyst., p. 103972, 2021, doi: 10.1016/j.micpro.2021.103972.

[2] Y. Ramzanpoor, M. Hosseini Shirvani, and M. Golsorkhtabaramiri, Multi-objective fault-tolerant optimization algorithm for deployment of IoT applications on fog computing infrastructure, vol. 8, no. 1. Springer International Publishing, 2022. doi: 10.1007/s40747-021-00368-z.

[3] M. Gorlatova, H. Inaltekin, and M. Chiang, “Characterizing task completion latencies in multi-point multi-quality fog computing systems,” Comput. Networks, vol. 181, no. December 2019, p. 107526, 2020, doi: 10.1016/j.comnet.2020.107526.

[4] R. and S. N. and S. P. Tiwari Rajeev and Sille, “Utilization and Energy Consumption Optimization for Cloud Computing Environment,” in Cyber Security and Digital Forensics , 2022, pp. 609–619.

[5] M. Mehmood, N. J. B, and J. Akram, Efficient Resource Distribution in Cloud, vol. 1. Springer International Publishing, 2019. doi: 10.1007/978-3-319-98530-5.

[6] M. S. Qureshi et al., “A comparative analysis of resource allocation schemes for real-time services in high-performance computing systems,” Int. J. Distrib. Sens. Networks, vol. 16, no. 8, 2020, doi: 10.1177/1550147720932750.

[7] M. S. Aslanpour, S. S. Gill, and A. N. Toosi, “Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research,” Internet of Things, vol. 12, p. 100273, 2020, doi: 10.1016/j.iot.2020.100273.

[8] A. Abouaomar, S. Cherkaoui, A. Kobbane, and O. A. Dambri, "A resources representation for resource allocation in fog computing networks," 2019 IEEE Glob. Commun. Conf. GLOBECOM 2019 - Proc., 2019, doi: 10.1109/GLOBECOM38437.2019.9014146.

[9] P. Albert and M. Nanjappan, "WHOA: Hybrid Based Task Scheduling in Cloud Computing Environment," Wirel. Pers. Commun., vol. 121, no. 3, pp. 2327–2345, 2021, doi: 10.1007/s11277-021-08825-1.

[10] C. Framework, "EEOA : Cost and Energy Efficient Task Scheduling in a Cloud-Fog Framework," 2023.

[11] N. Sarrafzade, R. Entezari-Maleki, and L. Sousa, "A genetic-based approach for service placement in fog computing," J. Supercomput., vol. 78, no. 8, pp. 10854–10875, 2022, doi: 10.1007/s11227-021-04254-w.

[12] A. M. Yadav, K. N. Tripathi, and S. C. Sharma, "An enhanced multi-objective fireworks algorithm for task scheduling in fog computing environment," Cluster Comput., vol. 25, no. 2, pp. 983–998, 2022, doi: 10.1007/s10586-021-03481-3.

[13] D. Javaheri, S. Gorgin, J. A. Lee, and M. Masdari, "An improved discrete harris hawk optimization algorithm for efficient workflow scheduling in multi-fog computing," Sustain. Comput. Informatics Syst., vol. 36, p. 100787, Dec. 2022, doi: 10.1016/J.SUSCOM.2022.100787.

[14] N. Abu-Amssimir and A. Al-Haj, "A QoS-aware resource management scheme over fog computing infrastructures in IoT systems," Multimed. Tools Appl., 2023, doi: 10.1007/s11042-023-14856-6.

[15] A. Maatoug, G. Belalem, and S. Mahmoudi, "A location-based fog computing optimization of energy management in smart buildings: DEVS modeling and design of connected objects," Front. Comput. Sci., vol. 17, no. 2, p. 172501, 2022, doi: 10.1007/s11704-021-0375-z.

[16] S. O. Ogundoyin and I. A. Kamil, "Optimal fog node selection based on hybrid particle swarm optimization and firefly algorithm in dynamic fog computing services," Eng. Appl. Artif. Intell., vol. 121, p. 105998, 2023, doi: https://doi.org/10.1016/j.engappai.2023.105998.

[17] M. M. Hussain et al., "SONG: A Multi-Objective Evolutionary Algorithm for Delay and Energy Aware Facility Location in Vehicular Fog Networks," Sensors, vol. 23, no. 2, 2023, doi: 10.3390/s23020667.

[18] M. Ghobaei-Arani and A. Shahidinejad, "A cost-efficient IoT service placement approach using whale optimization algorithm in fog computing environment," Expert Syst. Appl., vol. 200, no. May 2021, p. 117012, 2022, doi: 10.1016/j.eswa.2022.117012.

[19] S. Kumar and R. Tiwari, "Dynamic popularity window and distance-based efficient caching for fast content delivery applications in CCN," Eng. Sci. Technol. an Int. J., vol. 24, no. 3, pp. 829–837, 2021, doi: 10.1016/j.jestch.2020.12.018.

[20] S. Kaur, Y. Kumar, A. Koul, and S. Kumar Kamboj, "A Systematic Review on Metaheuristic Optimization Techniques for Feature Selections in Disease Diagnosis: Open Issues and Challenges," Arch. Comput. Methods Eng., vol. 30, no. 3, pp. 1863–1895, 2023, doi: 10.1007/s11831-022-09853-1.

[21] K. Kaur, S. Garg, G. Kaddoum, F. Gagnon, and D. N. K. Jayakody, "EnLoB: Energy and load balancing-driven container placement strategy for data centers," 2019 IEEE Globecom Work. GC Wkshps 2019 - Proc., pp. 1–6, 2019, doi: 10.1109/GCWkshps45667.2019.9024592.

[22] M. and G. S. and K. S. Tiwari Rajeevand Mittal, "Energy-Aware Resource Scheduling in FoG Environment for IoT-Based Applications," in Energy Conservation Solutions for Fog-Edge Computing Paradigms, M. and G. L. M. Tiwari Rajeevand Mittal, Ed. Singapore: Springer Singapore, 2022, pp. 1–19. doi: 10.1007/978-981-16-3448-2_1.

[23] L. Abualigah and A. Diabat, "A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments," Cluster Comput., vol. 24, no. 1, pp. 205–223, 2021, doi: 10.1007/s10586-020-03075-5.

[24] A. Kishor and C. Chakarbarty, "Task Offloading in Fog Computing for Using Smart Ant Colony Optimization," Wirel. Pers. Commun., no. 0123456789, 2021, doi: 10.1007/s11277-021-08714-7.

[25] M. Haghi Kashani, A. M. Rahmani, and N. Jafari Navimipour, "Quality of service-aware approaches in fog computing," Int. J. Commun. Syst., vol. 33, no. 8, pp. 1–34, 2020, doi: 10.1002/dac.4340.

[26] A. M. Senthil Kumar and B. Kasireddi, "An efficient task scheduling method in a cloud computing environment using firefly crow search algorithm (FF-CSA)," Int. J. Sci. Technol. Res., vol. 8, no. 12, pp. 623–627, 2019.

[27] T. and T. V. and T. R. Lal Gunjanand Goel, "Performance Tuning Approach for Cloud Environment," in Intelligent Systems Technologies and Applications 2016, 2016, pp. 317–326.

[28] F. M. Talaat, "Effective prediction and resource allocation method (EPRAM) in fog computing environment for smart healthcare system," Multimed. Tools Appl., vol. 81, no. 6, pp. 8235–8258, 2022, doi: 10.1007/s11042-022-12223-5.

[29] G. Goel and R. Tiwari, "Resource scheduling in fog environment using optimization algorithms for 6G networks," Int. J. Softw. Sci. Comput. Intell., vol. 14, no. 1, pp. 1–24, 2022.

[30] M. R. Alizadeh, V. Khajehvand, A. M. Rahmani, and E. Akbari, "Task scheduling approaches in fog computing: A systematic review," Int. J. Commun. Syst., vol. 33, no. 16, pp. 1–36, 2020, doi: 10.1002/dac.4583.

[31] R. and A. A. and K. S. Goel Gauravand Tiwari, "Workflow Scheduling Using Optimization Algorithm in Fog Computing," in International Conference on Innovative Computing and Communications, 2022, pp. 379–390.

[32] D. Rahbari and M. Nickray, "Low-latency and energy-efficient scheduling in fog-based IoT applications," Turkish J. Electr. Eng. Comput. Sci., vol. 27, no. 2, pp. 1406–1427, 2019, doi: 10.3906/elk-1810-47.

[33] M. Abdel-Basset, R. Mohamed, R. K. Chakrabortty, and M. J. Ryan, "IEGA: An improved elitism-based genetic algorithm for task scheduling problem in fog computing," Int. J. Intell. Syst., vol. 36, no. 9, pp. 4592–4631, 2021, doi: 10.1002/int.22470.

[34] M. Shen, B. Ma, L. Zhu, R. Mijumbi, X. Du, and J. Hu, “Cloud-Based Approximate Constrained Shortest Distance Queries over Encrypted Graphs with Privacy Protection,” IEEE Trans. Inf. Forensics Secur., vol. 13, no. 4, pp. 940–953, 2018, doi: 10.1109/TIFS.2017.2774451.

[35] M. I. Naas, L. Lemarchand, P. Raipin, and J. Boukhobza, “IoT Data Replication and Consistency Management in Fog Computing,” J. Grid Comput., vol. 19, no. 3, p. 33, 2021, doi: 10.1007/s10723-021-09571-1.

[36] M. S. Hassan, W. G. Aref, and A. M. Aly, “Graph indexing for shortest-path finding over dynamic sub-graphs,” Proc. ACM SIGMOD Int. Conf. Manag. Data, vol. 26-June-20, pp. 1183–1197, 2016, doi: 10.1145/2882903.2882933.

[37] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, “iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments,” Softw. Pract. Exp., vol. 47, no. 9, pp. 1275–1296, 2017.

[38] A. Chhabra, S. K. Sahana, N. S. Sani, A. Mohammadzadeh, and H. A. Omar, “Energy-Aware Bag-Of-Tasks Scheduling in the Cloud Computing System Using Hybrid Oppositional Differential Evolution-Enabled Whale Optimization Algorithm,” Energies, vol. 15, no. 13, 2022, doi: 10.3390/en15134571.

[39] A. Chhabra, G. Singh, and K. S. Kahlon, “Multi-criteria HPC task scheduling on IaaS cloud infrastructures using meta-heuristics,” Cluster Comput., vol. 24, pp. 885–918, 2021.

[40] A. Chhabra, G. Singh, and K. S. Kahlon, “QoS-aware energy-efficient task scheduling on HPC cloud infrastructures using swarm-intelligence meta-heuristics,” Comput. Mater. Contin, vol. 64, pp. 813–834, 2020.

[41] S. Vila, F. Guirado, J. L. Lerida, and F. Cores, “Energy-saving scheduling on IaaS HPC cloud environments based on a multi-objective genetic algorithm,” J. Supercomput., vol. 75, no. 3, pp. 1483–1495, 2019.

[42] Sharma, T., & Rattan, D. (2021). Malicious application detection in android—a systematic literature review. Computer Science Review, 40, 100373.