# Multi-Objective Osprey Optimization Algorithm-Based Resource Allocation in Fog-IoT

Nagarjun E, Dharamendra Chouhan, Dilip Kumar S M
Department of Computer Science and Engineering-University of Visvesvaraya College of Engineering,
Bangalore University, Bengaluru, India

***Abstract*—Fog Computing (FC) paradigm offers significant potential for hosting diverse delay-sensitive Internet of Things (IoT) applications. However, the limited resources of fog devices pose significant challenges for deploying multiple applications, particularly in heterogeneous and dynamic IoT scenarios, due to the absence of effective mechanisms for resource estimation and discovery. An efficient resource allocation strategy is crucial for meeting the Quality of Service (QoS) requirements of IoT applications while enhancing overall system performance. Identifying the optimal allocation strategy for IoT applications with multiple QoS parameters is a complex and computationally intensive challenge, classified as an NP-complete problem. This paper proposes a Multi-Objective Optimization Algorithm (MOOA) for optimal resource allocation using the Osprey Optimization Algorithm (OOA) to efficiently allocate available resources. The proposed algorithm was evaluated against existing approaches, including the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), under varying task loads ranging from 100 to 500 tasks. The simulation results demonstrate significant performance improvements, including an average reduction in execution time by 12.45% compared to PSO and 22.97% compared to GA, response time by 32.57% compared to GA and 24.45% compared to PSO, and completion time by 44.39% compared to GA and 33.23% compared to PSO. These findings highlight the proposed algorithm's ability to efficiently handle task allocation in dynamic FC environments and its potential to address complex QoS requirements in real-world IoT applications.**

***Keywords*—Fog computing; IoT; resource allocation and reallocation; task allocation***

## I. Introduction

The Internet of Things (IoT) has revolutionized data generation, driving advancements in healthcare monitoring, virtual reality, industrial automation, and many other applications that demand reliable, low-latency communication and computational services [1]. These applications often impose stringent QoS requirements, such as minimizing delay and energy consumption, which pose significant challenges for IoT devices [2], [3].

Fog computing (FC) is a computing paradigm placed in the middle of a three tiered architecture that encompasses the cloud, fog nodes (FNs), and IoT devices at the edge. Cisco introduced this layer in 2012 to bring the features and capabilities of cloud computing nearer to data sources and user devices that communicate over the internet [4], [5], [6]. FC is recognized as a key solution to the limitations of cloud computing, particularly for delay-sensitive applications. However, while FC provides a decentralized architecture, the challenge of selecting suitable FNs for application modules with diverse deadline constraints remains a critical issue [7].

Resource allocation in FC entails the distribution of storage, computational, and communication resources among FNs, users, and service providers. These entities often have conflicting objectives and requirements, necessitating mechanisms that balance their interests while optimizing overall system performance [8]. Furthermore, the dynamic nature of FC characterized by fluctuations in resource availability and user demand highlights the need for adaptive allocation strategies. Effective resource allocation and reallocation in FC are particularly crucial for IoT tasks, where metrics such as resource utilization, execution time, response time, and completion time are key considerations [9].

To address these challenges, this study proposes a Multi-Objective Osprey Optimization Algorithm (MOOA) to allocate the appropriate number of fog resources corresponding to fluctuations in IoT task demands.

- Proposed a meta-heuristic MOOA for resource allocation and reallocation in FC for IoT tasks to handle the dynamic and heterogeneous requirements of IoT tasks.
- Implement the proposed algorithm and evaluate its performance against GA and PSO by varying the number of IoT tasks and exploring different parameters.
- The results demonstrate a significant reduction in average response time, execution time and completion, showcasing the efficiency of our approach.

The rest of this paper is organized as follows: Section II provides a detailed background on the problem of resource allocation improvement in FC environments, highlighting the drawbacks of existing methods. Section III presents the system model, problem statement, and objectives. Section IV describes the proposed algorithm and its key features. In Section V, we present our simulation and experimental results, comparing the performance of our algorithm with existing approaches. Finally, Section VI concludes the paper and discusses future research directions in this area.

## II. Related Works

Hussain *et al.* [10] developed the Resource Aware Prioritized Task Scheduling (RAPTS) approach for heterogeneous FC environments. The primary objective of RAPTS is to ensure the execution of tasks with strict deadlines while optimizing response time, cost, and makespan, and enhancing resource utilization within the fog layer. This approach was implemented in iFogSim and evaluated based on various performance metrics, including response time, resource utilization, task deadline compliance, cost, and makespan. Comparative analysis with advanced fog schedulers like RACE (CFP) and RACE (FOP) demonstrated that RAPTS achieved improvements of up to 29%, 53%, 15%, 11%, and 43% in resource utilization, response time, makespan, cost, and task deadline adherence, respectively. However, completion time and execution time are not addressed.

Arshed *et al.* [11] introduced the Resource Aware Cost-Efficient Scheduler (RACE) to allocate incoming application modules to fog devices. The scheduler manages to optimize resource utilization at the fog layer by minimizing the monetary cost of using cloud resources, and reduces the application execution time and bandwidth usage. It incorporates two key algorithms: the ModuleScheduler, which classifies application modules based on their computational and bandwidth requirements, and the CompareModule, which determines their placement. Simulation results indicate that RACE outperforms traditional cloud placement strategies and baseline algorithms in most scenarios. However, the approach does not specifically address response time, completion time, and execution time.

Khan *et al.* [12] introduced an improved Ripple-Induced Whale Optimization Algorithm (RWOA) for scheduling independent tasks in fog-cloud environments. The method leverages ripple effects to refine suboptimal solutions, aiming to minimize makespan and energy consumption while enhancing throughput. Despite its effectiveness, the approach does not explicitly consider metrics such as response time, completion time, and execution time. Chafi *et al.* [13] proposed a novel algorithm based on Particle Swarm Optimization (PSO) to optimize energy consumption and workflow time in heterogeneous FC environments. By utilizing the collective behavior of particles, the algorithm effectively explores the solution space and adapts to the dynamic and unpredictable nature of FC resources. Simulation results demonstrate notable enhancements in workflow completion time, energy efficiency, and resource utilization. However, the approach does not explicitly consider metrics such as response time, and execution time.

Bandopadhyay *et al.* [8] proposed a Game-Theoretic Resource Allocation and Dynamic Pricing Mechanism in FC (GTRADPMFC). The model incorporates non-cooperative competition among FNs for resource allocation and employs dynamic pricing mechanisms to promote efficient resource usage. Through theoretical evaluation and simulation experiments, the study demonstrated that GTRADPMFC enhances both resource efficiency and the overall performance of FC systems. Furthermore, the paper outlines methods to manage scenarios with insufficient data samples and provides flexibility for users who cannot meet specific completion time requirements. GTRADPMFC optimizes resource allocation by establishing pricing strategies while accounting for potential delays in task completion. The research also includes simulations, convergence analyses, complexity assessments, and guarantees for optimization. However, the model does not explicitly address performance metrics such as response time and execution time.

Mokni *et al.* [14] proposed decision-making phase that analyzes data generated by IoT devices to identify the optimal offloading strategy. To accomplish this, the TOF-NSGAII approach was introduced, aiming to reduce both energy consumption and latency during the offloading of IoT tasks in a fog-cloud environment. Each IoT device transmits tasks with distinct attributes, such as input data size, and the algorithm assigns these tasks to specific virtual machines to optimize resource utilization. By utilizing the combined resources of fog and CC, the TOF-NSGAII approach effectively meets its goals of minimizing energy usage and latency, as validated by experimental results. However, the model does not explicitly address performance metrics such as response time and execution time.

Raissouli *et al.* [15] developed an improved version of NSGA II, namely, reinforcement weighted probabilistic NSGA II, which uses weighted probabilistic mutation. This algorithm replaces random mutation with probabilistic mutation to enhance exploration of the solution space. This method uses domain-specific knowledge to improve convergence and solution quality, resulting in improved energy efficiency and reduced delay compared to traditional NSGA II and other evolutionary algorithms. However, the model does not explicitly address performance metrics such as response time, completion time and execution time.

The studies described above show that various efforts have been made to solve resource allocation and task scheduling problems in FC environments. However, the efficiency of algorithms in handling complex and dynamic FC scenarios across varying conditions has yet to be thoroughly investigated. Additionally, existing algorithms do not explicitly consider key performance metrics such as response time, completion time, and execution time. In contrast, our proposed approach addresses these limitations by incorporating these critical metrics to enhance the effectiveness of resource allocation and reallocation in FC.

## III. System Model

An IoT network consisting of IoT devices and FNs is considered, where the FNs vary in energy and processing power capabilities, making task scheduling a challenging problem. In this setup, we have $I$ IoT devices and $F$ heterogeneous FNs, represented by sets $I = \{1, 2, 3, \ldots, I\}$ and $F = \{1, 2, 3, \ldots, F\}$, respectively. The IoT devices collect data from a range of sensors and transmit it to the fog layer. It is assumed that the IoT devices offload tasks directly to the FNs without any local processing. Each FN is equipped with an agent tasked with receiving and processing data from the IoT devices. Task scheduling is assumed to occur in a distributed manner, with each FN acting as a scheduler upon receiving data from IoT devices. A task is represented as $T_t(S_t, C_t)$, where $S_t$ is the task size in KB and $C_t$ is the task complexity in Million Instructions (MI). Since the end devices often lack sufficient computational resources to process the tasks, the tasks are offloaded to nearby FNs for processing. The commonly-used symbols are delineated in Table I.

### A. Problem Statement and Objectives

The problem is defined as finding optimal resource allocation for IoT tasks in a FC environment. Each fog node serves as a resource for processing tasks, and the goal is to allocate these tasks efficiently with the following objectives:

*1) Objectives:* The objectives of the proposed work are as follows:

- Minimize completion time for processing IoT tasks.
- Minimize response time for IoT tasks.
- Minimize execution time for IoT tasks.
- Maximize the resource utilization.

TABLE I. NOTATION TABLE FOR THE MULTI-OBJECTIVE OSPREY OPTIMIZATION ALGORITHM (MOOA)

| Notation | Description |
|---|---|
| $T$ | Set of IoT tasks, where $T = \{t_1, t_2, \ldots, t_N\}$. |
| $FN$ | Set of Fog Nodes (FNs), where $FN = \{fn_1, fn_2, \ldots, fn_M\}$. |
| $N$ | Population size (number of osprey solutions in the search space). |
| $T_{max}$ | Total number of iterations for the optimization process. |
| $R$ | Population matrix representing resource allocation solutions, where $X = \{R_1, R_2, \ldots, R_N\}$. |
| $RP_i$ | A single solution in the population, representing task-to-node allocations. |
| $r_{i,j}$ | Position of the $j^{th}$ dimension of solution $R_i$, indicating resource allocation for task $j$. |
| $r_{i,j}^{P1}$ | Updated position of the $j^{th}$ dimension for osprey $i$ during optimization. |
| $r_{i,j}^{P2}$ | Updated position of the $j^{th}$ dimension for osprey $i$ during optimization. |
| $lb_j$, $ub_j$ | Lower and upper bounds for the $j^{th}$ dimension of the solution space. |
| $r_{i,j}$ | A random number generated for the $j^{th}$ dimension of osprey $i$. |
| $F(X_i)$ | Fitness value of solution $X_i$, considering multiple objectives. |
| $f$ | Objective function |
| $X^*$ | Best solution obtained after optimization. |
| $t_{new}$ | Newly arriving IoT task that requires resource reallocation. |
| $fn_{fluct}$ | Fog node with fluctuating resource availability due to dynamic conditions. |

## IV. PROPOSED WORK

In this section, the details of the proposed Multi-Objective Optimization Algorithm (MOOA) are presented. The diagram illustrating the proposed architecture is presented in Fig. 1.

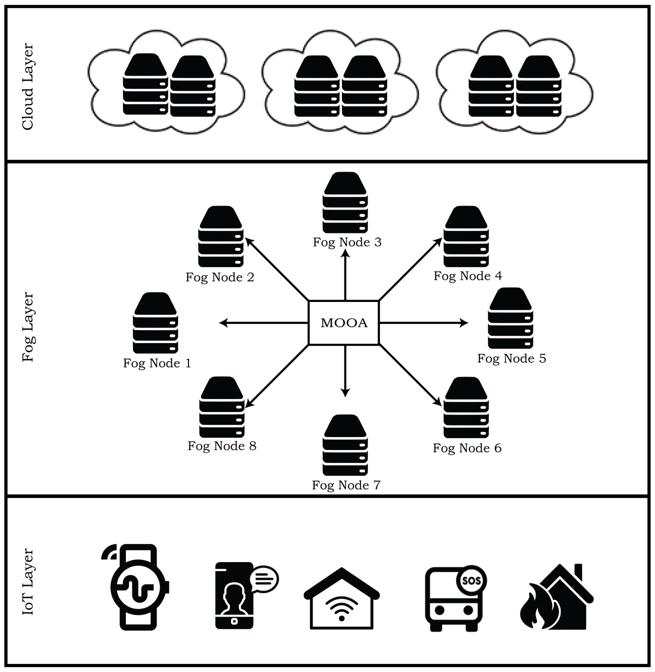

Fig. 1. Proposed MOOA architecture.

### A. OOA Overview

The Osprey Optimization Algorithm (OOA) is inspired by the hunting and behavior patterns of the osprey, a bird of prey that primarily feeds on fish. Known for its sharp vision and specialized hunting strategy, the osprey locates fish underwater from a height of 10 to 40 meters before diving to capture them. After catching its prey, the osprey takes it to a safe location to consume it. This natural behavior, marked by precision and efficiency, forms the basis of the OOA, which applies these characteristics to create an effective optimization technique [16], [17]. In the context of FC, the OOA is adapted to allocate and reallocate resources efficiently among tasks and FNs. The OOA is designed to optimize multiple objectives simultaneously, such as minimizing response time, execution time, completion time, and balancing the load among FNs. The algorithm operates in three key phases: initialization, iterative optimization, and updating.

### B. Initialization Phase

In the initialization phase, a population of $N$ ospreys is randomly generated. Each osprey in the population represents a candidate solution, where its position encodes a potential allocation of resources. The matrix representation of the population can be modeled as shown in Eq. 1. The position of the $i$-th osprey in the $j$-th dimension is represented by $r_{i,j}$, where $r_{i,j}$ corresponds to a problem variable related to resource allocation (e.g. CPU, RAM and Bandwidth). The initial positions are within the predefined bounds of the problem variables.

$$R = \begin{bmatrix} R_1 \\ \vdots \\ R_i \\ \vdots \\ R_N \end{bmatrix}_{N\times m} = \begin{bmatrix} r_{1,1} & \cdots & r_{1,j} & \cdots & r_{1,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ r_{i,1} & \cdots & r_{i,j} & \cdots & r_{i,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ r_{N,1} & \cdots & r_{N,j} & \cdots & r_{N,m} \end{bmatrix}_{N\times m}, \quad (1)$$

Where $N$ is the number of candidate solutions (ospreys), $m$ is the number of tasks to be allocated, $r_{i,j}$ represents the allocation of resources from a FN to the $j$-th task in the $i$-th candidate solution. At the beginning of the algorithm, the initial allocation matrix is generated randomly to ensure diversity. The positions of the ospreys in the resource allocation search space are initialized using Eq. 2:

$$r_{i,j} = lb_j + r_{ij} \cdot (ub_j - lb_j)\,, i = 1, 2, \ldots, N, j = 1, 2, \ldots, m, \quad (2)$$

Where $lb_j$ and $ub_j$ are the lower and upper bounds of resource availability for the $j$-th task, $r_{i,j}$ is a random number uniformly distributed in the range $[0, 1]$, ensuring a diverse initial allocation.

### C. Objective Function

The main goal of task allocation in a FC environment is to minimize the time needed for task completion, response, and execution. The values obtained from evaluating the objective function for the given problem from Eq. (4) to (10) can be expressed as a vector, as outlined in Eq. (3).

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} F(R_1) \\ \vdots \\ F(R_i) \\ \vdots \\ F(R_N) \end{bmatrix}_{N\times 1}, \tag{3}$$

1) Execution Time: The execution time $E_{i,j}$ of $task_i$ on $FN_j$ can be expressed as:

$$E_{i,j} = \frac{T_{i,j}}{C_{i,j}} \tag{4}$$

Where $T_{i,j}$ is task length for $task_i$ on $FN_j$ (number of instructions). $CC_{i,j}$ is computational capacity of $FN_j$, calculated as:

$$CC_{i,j} = P_{i,j} \times M_{i,j} \tag{5}$$

Where $P_{i,j}$ is number of processing elements (PEs) in $FN_j$. $M_{i,j}$ is MIPS rate of $FN_j$.

2) Response Time: The response time $RT_{i,j}$ for $task_i$ on $FN_j$ is calculated as:

$$RT_{i,j} = F_{i,j} - S_{i,j} \tag{6}$$

Where $F_{i,j}$ is finish time of $task_i$ on $FN_j$. $S_{i,j}$ is start time of $task_i$ on $FN_j$.
The finish time $F_{i,j}$ is given by:

$$F_{i,j} = R_{i,j} + E_{i,j} \tag{7}$$

3) Completion Time: The completion time $CT_{i,j}$ for $task_i$ on $FN_j$ is calculated as:

$$CT_{i,j} = F_{i,j} + C_{T_{i,j}} \tag{8}$$

Where $C_{T_{i,j}}$ is communication time for $task_i$ on $FN_j$. Thus, the completion time $CT_{i,j}$ is:

$$CT_{i,j} = (RT_{i,j} + E_{i,j}) + C_{T_{i,j}} \tag{9}$$

The objective function value is evaluated at each position of the osprey:

$$F_i = [F_1(E), F_2(RT), F_3(CT)]^T \tag{10}$$

## D. Phase 1: Position Identification and Resource Allocation

In fog resource allocation, resource updates can mimic osprey hunting behavior. Just as ospreys identify and target fish underwater, the algorithm identifies better resource positions in the search space to enhance exploration and escape local optima. For each resource, the positions of other resources in the search space that have better objective function values are considered optimal targets. These targets are analogous to the "fish" in the search space. The set of optimal resources for each resource is defined in Eq. (11).

$$RP_i = \{R_k \mid k \in \{1,2,\ldots,N\} \wedge R_k < F_i\} \cup \{R_{\text{best}}\} \tag{11}$$

Here, $R_k$ represents the resources with better objective values, and $R_{\text{best}}$ is the best resource found so far. This approach helps guide the allocation process towards more efficient solutions in FC.

$$r_{ij}^{P1} = r_{ij} + r_{ij} \cdot (SF_{ij} - I_{ij} \cdot r_{ij}), \tag{12a}$$

$$r_{ij}^{P_1} = \begin{cases} r_{ij}^{P_1}, lb_j \le r_{ij}^{P_1} \le ub_j; \\ lb_j, r_{ij}^{P_1} < lb_j; \\ ub_j, r_{ij}^{P_1} > ub_j. \end{cases} \tag{12b}$$

The updated resources $R_i$ are refined based on their fitness values:

$$R_i = \begin{cases} R_i^{P1}, F_i^{P_1} < F_i; \\ R_i, \text{ else }, \end{cases} \tag{13}$$

This approach ensures efficient resource allocation and optimizes performance in FC systems.

## E. Phase 2: Identifying the Suitable Resources

After identifying a suitable resource node, the FC system allocates tasks to the selected node for processing in an efficient and resource-aware manner. This phase models the process of refining resource allocation decisions to ensure that the tasks are processed in a way that minimizes costs and optimizes system performance. The refinement of allocation decisions involves making small adjustments to the assigned resources, enhancing the exploitation capabilities of the resource allocation algorithm, and converging toward better solutions near the currently identified optimal nodes.

In the design of the FC allocation algorithm, this behavior is simulated by calculating a new potential resource allocation for each task based on predefined criteria using Eq. (14a), (14b). If the new allocation improves the value of the objective function (e.g. response time, completion time, execution time, or improving resource utilization), the algorithm updates the allocation to this new configuration according to Eq. (15). This iterative refinement ensures more efficient exploitation of available resources and promotes convergence toward the most optimal allocation.

$$r_{i,j}^{p_2} = r_{i,j} + \frac{lb_j + r \cdot (ub_j - lb_j)}{t}, i = 1,2,\ldots,N, \quad j = 1,2,\ldots,m, t = 1,2,\ldots,T \tag{14a}$$

After updating the position, the new position is constrained within the bounds:

$$r_{i,j}^{P2} = \begin{cases} r_{i,j}^{P2}, lb_j \le r_{i,j}^{P2} \le ub_j; \\ lb_j, r_{i,j}^{P2} < lb_j; \\ ub_j, r_{i,j}^{P2} > ub_j, \end{cases} \tag{14b}$$

$$R_i = \begin{cases} R_i^{P2}, F_i^{P2} < F_i; \\ R_i, \text{ else }, \end{cases} \tag{15}$$

else, the osprey remains at its current position.

**Algorithm 1:** MOOA for Resource Allocation and Reallocation in FC for IoT Tasks

```
Input: Task set $T = \{t_1, t_2, ..., t_N\}$, Fog Node set $FN = \{fn_1, fn_2, ..., fn_M\}$, $T_{max}$, $N$, Resource constraints (e.g.,
    CPU, MIPS, memory, energy per unit), $f$.
Output: Optimized resource allocation and reallocation solutions for IoT tasks in the fog network.
/* Phase 1: Initial Resource Allocation */
Step 1: Initialize the Population;
Generate initial population matrix with random resource allocation positions for each task to FNs using using (1) and
  (2).
Step 2: Evaluate Initial Fitness;
foreach $X_i \in X$ do
    Compute fitness for each solution using objective function $f$ by (3)
for $t \leftarrow 1$ to $T_{max}$ do
    foreach osprey $i \in \{1, 2, ..., N\}$ do
        foreach task-to-node mapping dimension $j \in \{1, 2, ..., m\}$ do
            /* Phase 1: Position identification and Resource Allocation */
            /* Calculate New Position for Resource Allocation */
            Generate random number $r_{i,j}$;
            Update position for osprey $i$ using (11).
            Calculate new position $x_{P1_{i,j}}$ using (12a).
            Check boundary conditions for $x_{P1_{i,j}}$ using (12b).
            Update $R_i$ using (13).
        /* Phase 2: Identifying the suitable resources */
        /* Evaluate New Fitness for the Updated Allocation */
        Compute new fitness by using (3).
        Calculate new position $x_{P2_{i,j}}$ using (14a).
        Check boundary conditions for $x_{P2_{i,j}}$ using (14b).
        /* Replace if New Solution is Better */
        Update $R_i$ using (15).
Output: Initial resource allocation solution $X^*$;
/* Phase 2: Resource Reallocation */
Step 3: Monitor Task and Resource Fluctuations;
Collect real-time information about task arrivals, resource availability, and execution progress in FNs;
Step 4: Reallocate Resources Dynamically;
foreach new task $t_{new}$ or fluctuating resource $fn_{fluct}$ do
    foreach solution $X_i \in X$ do
        foreach task-to-node mapping dimension j do
            /* Phase 1: Position identification and Resource Allocation */
            /* Recompute Positions for Reallocation */
            Update resource mapping based on osprey behavior and fluctuating resource constraints by using (11).
            Calculate new position $x_{P1_{i,j}}$ using (12a).
            Check boundary conditions for $x_{P1_{i,j}}$ using (12b).
            Update $R_i$ using (13).
        /* Phase 2: Identifying the suitable resources */
        /* Reevaluate Fitness for Reallocation */
        Compute by using (3).
        Calculate new position $x_{P2_{i,j}}$ using (14a).
        Check boundary conditions for $x_{P2_{i,j}}$ using (14b).
        /* Replace Solution if Reallocation is Better */
        Update $R_i$ using (15).
Output: Final optimized solution $X^*$ for resource allocation and reallocation;
```

## *F. Termination Criterion*

The optimization continues until a termination criterion is met, such as a maximum number of iterations $T_{\text{max}}$ or a satisfactory solution is found.

## *G. Iterative Process, Flowchart, and Algorithm of MOOA*

The MOOA for resource allocation and reallocation in FC for IoT tasks is designed to handle the dynamic and heterogeneous requirements of IoT tasks. The steps for implementing the MOOA are illustrated as the flowchart in Fig. 2, and their

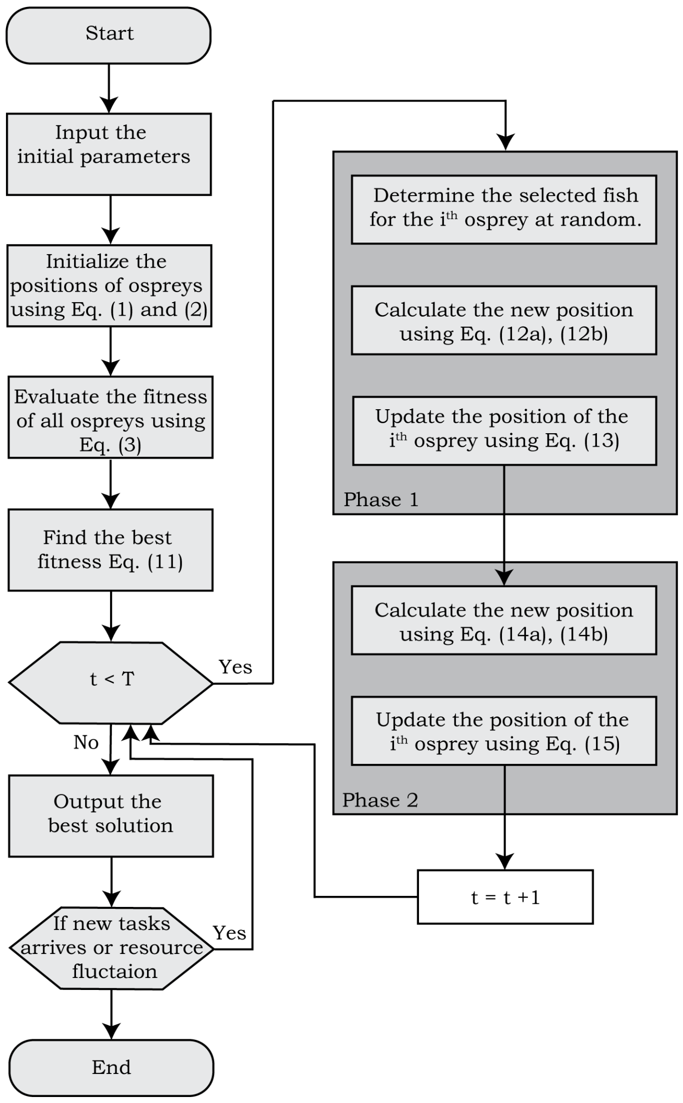
Start
Input the initial parameters
Initialize the positions of ospreys using Eq. (1) and (2)
Evaluate the fitness of all ospreys using Eq. (3)
Find the best fitness Eq. (11)
t < T
Yes
No
Output the best solution
If new tasks arrives or resource fluctaion
Yes
End
Determine the selected fish for the i^th osprey at random.
Calculate the new position using Eq. (12a), (12b)
Update the position of the i^th osprey using Eq. (13)
Phase 1
Calculate the new position using Eq. (14a), (14b)
Update the position of the i^th osprey using Eq. (15)
Phase 2
t = t +1

Fig. 2. Flowchart of MOOA.

procedural details are outlined in Algorithm I as pseudocode.

Phase 1: Initial Resource Allocation: The algorithm begins by generating an initial population matrix $X$, where each solution represents a potential mapping of tasks to fog nodes. The positions in the matrix are initialized randomly within the defined bounds for resource constraints (e.g., CPU, MIPS, memory). Each solution is evaluated using multi-objective fitness functions mentioned in the Eq. (10). For each iteration $t$ of the optimization process, every osprey (solution) adjusts its position by considering random factors ($r_{i,j}$) and the bounds of resource constraints ($lb_j, ub_j$). The new position is calculated for each dimension (task-to-node mapping), ensuring the solution stays within the defined bounds. The fitness of the updated position is evaluated for all objective functions. If the updated position improves the fitness of the solution, it replaces the existing solution. This iterative process continues until the maximum number of iterations ($T_{\max}$) is reached, producing the initial resource allocation solution $X^*$.

Phase 2: Dynamic Resource Reallocation: In the second phase, the algorithm monitors the fog network in real-time to account for fluctuations in task arrivals and resource availability. Information about new tasks, resource constraints, and ongoing execution is collected. For every new task or fluctuating resource, the algorithm adjusts the resource mapping dynamically. Each solution in the population is reevaluated by recalculating the task-to-node mappings based on osprey behavior, incorporating updated resource constraints. The positions are updated using random factors ($r_{i,j}$) while ensuring they remain within the bounds of resource availability. The fitness of each updated solution is computed, and if the reallocation improves the solution, it replaces the current one. The algorithm outputs the optimized resource allocation and reallocation solution $X^*$, balancing the objectives of equation (10). This ensures efficient and adaptive resource management in FC environments for IoT tasks.

## V. Simulation Results and Discussion

This section presents the evaluations conducted to demonstrate the effectiveness of the proposed approach. The experiments were performed using the iFogSim2 [18] simulation platform, and performance of the proposed MOOA algorithm was compared with existing algorithms, including GA [19] and PSO [20]. The simulations were run on a Windows 11 system equipped with an Intel(R) Core(TM) i5-9300H CPU operating at 2.40 GHz. The experimental parameters are related to various IoT tasks, the fog and cloud computing environment, and the configuration of the MOOA algorithm. Details of the experimental settings are provided in Table II.

TABLE II. Simulation Parameters

| **Parameters** | **Values** |
|---|---|
| Simulation tool | iFogSim2 |
| System Architecture | x86 |
| OS | Linux |
| VMM | Xen |
| No. of FNs | 50 |
| RAM | 4,000 (MB) |
| Storage | 1000000 |
| BW | 10000 |
| GA Parameters | |
| Simulation parameters | Value |
| Number of iterations | 60 |
| Population size | 10 |
| Mutation probability | 0.2s |
| PSO Parameters | |
| Number of particles | 30 |
| Iterations | 100 |
| Inertia constant | 0.85 |
| Cognitive constant | 1 |
| Social constant | 2 |
| MOOA Parameters | |
| Number of ospreys | 30 |
| Iterations | 100 |
| Experiment 1 | |
| Purpose | Heterogenous task |
| Data input parameters | 100, 200, 300, 400, 500 |
| FN parameters | 50 |
| Experiment 2 | |
| Purpose | Heterogenous nodes |
| Data input parameters | 200 |
| FN parameters | 100, 200, 300, 400, 500 |

### A. *Experiment 1*

Fig. 3 compares the execution time for varying task loads (100 to 500 tasks) among the proposed algorithm, GA, and

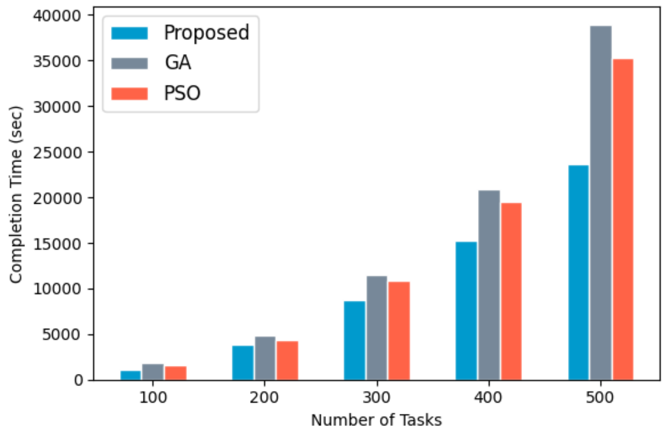

Fig. 3. Execution time of tasks with 50 FNs.

PSO. As expected, execution time increases with the number of tasks. The proposed algorithm achieved a significant reduction in execution time, showing an average improvement of 12.45% over PSO and 22.97% over GA. This underscores the proposed algorithm's ability to handle tasks efficiently in FC environments.

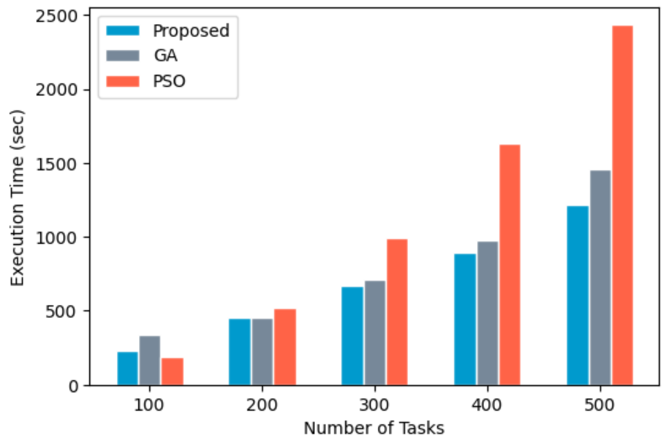

Fig. 4. Response time of tasks with 50 FNs.

Fig. 4 illustrates the response time for different task loads. The proposed algorithm consistently outperformed GA and PSO, achieving an average response time improvement of 32.57% compared to GA and 24.45% compared to PSO. This highlights the robustness of the proposed algorithm in minimizing delay and enhancing performance.

Fig. 5 depicts the completion time comparison across task loads. The proposed algorithm demonstrated superior performance, with an average reduction of 44.39% compared to GA and 33.23% compared to PSO. This confirms the proposed algorithm's scalability and effectiveness in optimizing computational processes in FC environments.

## B. Experiment 2

Fig. 6 compares the execution time for varying FNs (10 to 50 nodes) among the proposed algorithm, PSO and GA.

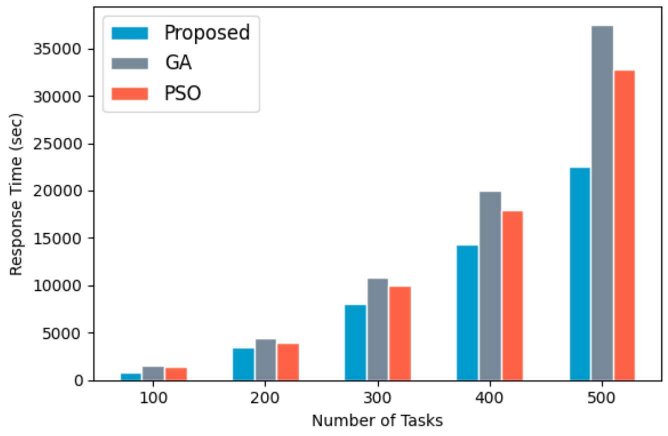

Fig. 5. Completion time of tasks with 50 FNs.

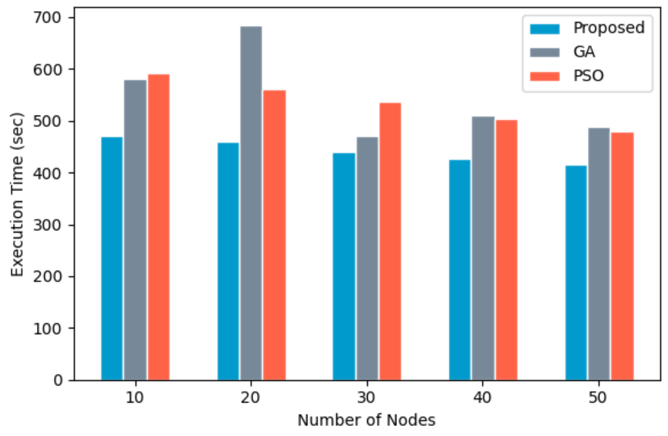

Fig. 6. Execution time of 200 tasks with different FNs.

As expected, execution time decreases with the number of FNs. The proposed algorithm achieved a significant reduction in execution time, showing an average improvement of 17.16% over PSO and 18.02% over GA. This underscores the proposed algorithm's ability to handle tasks efficiently in FC environments.

Fig. 7 illustrates the response time comparison across for varying FNs (10 to 50 nodes). The proposed algorithm consistently outperformed GA and PSO, achieving an average response time improvement of 39.65% compared to GA and 17.57% compared to PSO. This highlights the robustness of the proposed algorithm in minimizing delay and enhancing performance. Fig. 8 depicts the completion time comparison across for varying FNs (10 to 50 nodes). The proposed algorithm demonstrated superior performance, with an average reduction of 36.44% compared to GA and 16.58% compared to PSO. This confirms the proposed algorithm's scalability and effectiveness in optimizing computational processes in FC environments.

## VI. Conclusion

In this paper, we proposed a MOOA aimed at optimizing resource allocation and reallocation in FC environments for

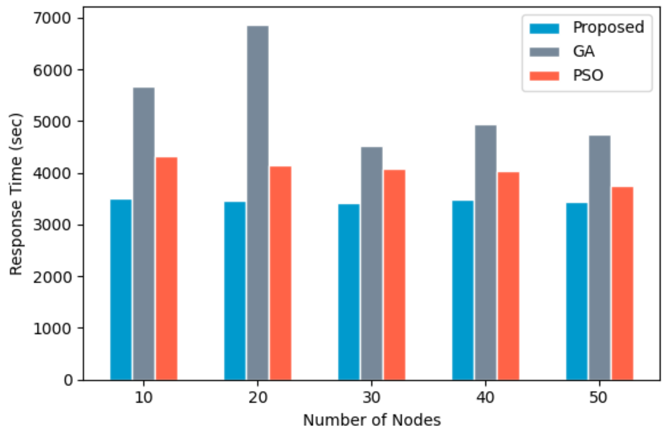

Fig. 7. Response time of 200 tasks with different FNs.

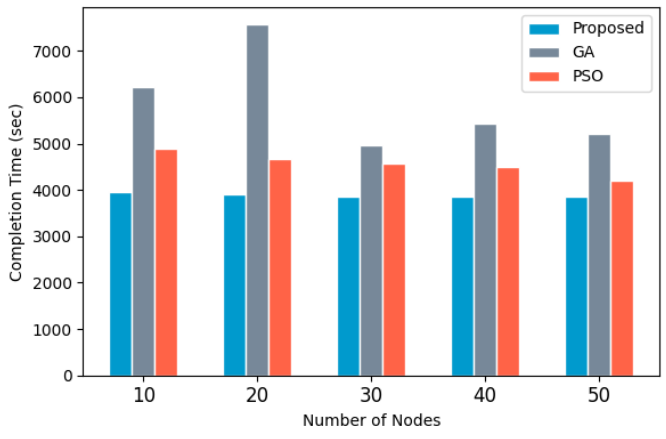

Fig. 8. Completion time of 200 tasks with different FNs.

IoT tasks. By optimizing multiple objectives such as completion time, response time, and execution time, MOOA provides an efficient solution for managing fog resources in dynamic and resource constrained environments. The simulation results demonstrated the effectiveness of the proposed algorithm in handling tasks efficiently in FC environments. The algorithm achieved a notable reduction in execution time, with an average improvement of 12.45% over PSO and 22.97% over GA. Moreover, MOOA consistently outperformed GA and PSO in terms of response time, achieving average improvements of 32.57% and 24.45%, respectively. Analysis of completion time further demonstrated the algorithm's superior performance, with an average reduction of 44.39% compared to GA and 33.23% compared to PSO. These results underline the robustness and scalability of MOOA, highlighting its effectiveness in optimizing task allocation and computational processes in FC environments. Future work will focus on extending this approach to incorporate energy efficiency, fault tolerance, and multi-objective optimization to address diverse QoS requirements in dynamic FC environments.

## References

[1] T. Arpitha, D. Chouhan, and J. Shreyas, "An efficient aco-inspired multipath routing for source location privacy with dynamic phantom node selection scheme in iot environments," *Soft Computing*, pp. 1–18, 2024.

[2] N. Srinidhi, E. Nagarjun, and S. Dilip Kumar, "Hybrid algorithm for efficient node and path in opportunistic iot network," *Journal of Information Technology Management*, vol. 13, pp. 68–91, 2021.

[3] N. Srinidhi, E. Nagarjun, J. Shreyas, S. Dilip Kumar, and D. Chouhan, "Ensuring fault tolerant connectivity in iot networks," in *Computer Communication, Networking and IoT: Proceedings of ICICC 2020*. Springer, 2021, pp. 391–400.

[4] H. K. Apat, B. Sahoo, V. Goswami, and R. K. Barik, "A hybrid meta-heuristic algorithm for multi-objective iot service placement in fog computing environments," *Decision Analytics Journal*, vol. 10, p. 100379, 2024.

[5] M. Zolghadri, P. Asghari, S. E. Dashti, and A. Hedayati, "Resource allocation in fog–cloud environments: State of the art," *Journal of Network and Computer Applications*, vol. 227, p. 103891, 2024.

[6] F. U. Khan, I. A. Shah, S. Jan, S. Ahmad, and T. Whangbo, "Machine learning-based resource management in fog computing: A systematic literature review," *Sensors*, vol. 25, no. 3, 2025. [Online]. Available: https://www.mdpi.com/1424-8220/25/3/687

[7] E. Nagarjun, D. Chouhan, I. Zabiulla, and S. D. Kumar, "Efficient resource provisioning in fog computing using agent-based contract net protocol: A smart healthcare case study," in *2024 First International Conference on Software, Systems and Information Technology (SSITCON)*. IEEE, 2024, pp. 1–6.

[8] A. Bandopadhyay, S. Swain, R. Singh, P. Sarkar, S. Bhattacharyya, and L. Mrsic, "Game-theoretic resource allocation and dynamic pricing mechanism in fog computing," *IEEE access*, 2024.

[9] D. Zhao, Q. Zou, and M. Boshkani Zadeh, "A qos-aware iot service placement mechanism in fog computing based on open-source development model," *Journal of Grid Computing*, vol. 20, no. 2, p. 12, 2022.

[10] M. Hussain, S. Nabi, and M. Hussain, "Rapts: resource aware prioritized task scheduling technique in heterogeneous fog computing environment," *Cluster Computing*, pp. 1–25, 2024.

[11] J. U. Arshed and M. Ahmed, "Race: resource aware cost-efficient scheduler for cloud fog environment," *IEEE Access*, vol. 9, pp. 65 688–65 701, 2021.

[12] Z. A. Khan and I. A. Aziz, "Ripple-induced whale optimization algorithm for independent tasks scheduling on fog computing," *IEEE Access*, 2024.

[13] S.-E. Chafi, Y. Balboul, M. Fattah, S. Mazer, and M. El Bekkali, "Novel pso-based algorithm for workflow time and energy optimization in a heterogeneous fog computing environment," *IEEE Access*, 2024.

[14] I. Mokni and S. Yassa, "A multi-objective approach for optimizing iot applications offloading in fog–cloud environments with nsga-ii," *The Journal of Supercomputing*, pp. 1–39, 2024.

[15] H. Raissouli, S. B. Belhaouari, and A. A. B. Ariffin, "Rwp-nsga ii: Reinforcement weighted probabilistic nsga ii for workload allocation in fog and internet of things environment," *International Journal of Distributed Sensor Networks*, vol. 2024, no. 1, p. 7645953, 2024.

[16] M. Dehghani and P. Trojovskỳ, "Osprey optimization algorithm: A new bio-inspired metaheuristic algorithm for solving engineering optimization problems," *Frontiers in Mechanical Engineering*, vol. 8, p. 1126450, 2023.

[17] F. Wei, X. Shi, and Y. Feng, "Improved osprey optimization algorithm based on two-color complementary mechanism for global optimization and engineering problems," *Biomimetics*, vol. 9, no. 8, p. 486, 2024.

[18] R. Mahmud, S. Pallewatta, M. Goudarzi, and R. Buyya, "Ifogsim2: An extended ifogsim simulator for mobility, clustering, and microservice management in edge and fog computing environments," *Journal of Systems and Software*, vol. 190, p. 111351, 2022.

[19] K. H. K. Reddy, A. K. Luhach, B. Pradhan, J. K. Dash, and D. S. Roy, "A genetic algorithm for energy efficient fog layer resource management in context-aware smart cities," *Sustainable Cities and Society*, vol. 63, p. 102428, 2020.

[20] I. M. Jabour and H. Al-Libawy, "An optimized approach for efficient-power and low-latency fog environment based on the pso algorithm," in *2021 2nd Information Technology To Enhance e-learning and Other Application (IT-ELA)*. IEEE, 2021, pp. 52–57.