

Hybrid Fog Resource Provisioning and Resource Allocation using MKFCM and FPA

Nagarjun E, Dharamendra Chouhan and Dilip Kumar SM

Department of Computer Science and Engineering, University of Visvesvaraya College of Engineering, Bengaluru, India

Article history

Received: 20-02-2026

Revised: 21-07-2026

Accepted: 08-09-2026

Corresponding Author:

Nagarjun E

Department of Computer

Science and Engineering,

University of Visvesvaraya

College of Engineering,

Bengaluru, India

Email: enagarjun.1@gmail.com

Abstract: The expansion of Internet of Things (IoT) deployments has motivated the Fog Computing (FC) paradigm, which brings mobile-edge and cloud resources together under a single model. Allocating and provisioning application tasks across such environments poses challenges due to constrained resource capabilities, heterogeneity, mobility, network hierarchy, and unpredictability. This paper addresses the need for additional computing devices that prioritize delay-sensitive applications by proposing a Hybrid Fog Resource Provisioning (HFRP) Algorithm for optimal resource allocation. The HFRP Algorithm combines Modified K-means Fuzzy CMeans (MKFCM) clustering and the Flower Pollination Algorithm (FPA) to efficiently select and provision resources. The algorithm operates in two phases: (i) Resource Discovery and Clustering, which identifies available resources and clusters them using MKFCM and (ii) Resource Allocation, which utilizes FPA to allocate tasks to the clustered resources. Simulation results exhibit that the HFRP algorithm significantly supersedes several other algorithms, including WORA, PSO, GA, and SJF. Key performance improvements include: HFRP reduces average cost by 25.83% compared to WORA, 33.7% compared to PSO, 45.9% compared to GA and 64.7% compared to SJF. HFRP consumes 16.69% less energy than WORA, 32.3% lower consumption than PSO, 38.7% lower consumption than GA and 59.77% lower than SJF. HFRP achieves 7.2% better makespan than WORA, 9.1% improvement over PSO, 15.8% improvement over GA, and 30.4% better than SJF. Successful Task Completion Ratio: HFRP surpasses WORA by 1.97%, 11.28% over PSO, 10.29% over GA and 27.12% over SJF. The proposed HFRP algorithm efficaciously addresses the challenges of resource allocation and provisioning in FC environments. By leveraging MKFCM for resource clustering and FPA for task allocation, HFRP achieves significant improvements in cost reduction, energy consumption, average loop delay, makespan, and task completion ratio compared to existing algorithms. These results bring out the potential of HFRP for augmenting the performance and efficiency of IoT applications in FC environments.

Keywords: Fog Computing, IoT, Resource Allocation, Resource Provisioning

Introduction

Advances in IoT have driven data generation to volumes and rates that centralised infrastructure was not provisioned to sustain (Srinidhi et al., 2019; 2021). As device density and network traffic continue to grow, transporting every request to a remote data centre inflates end-to-end latency and consumes backbone bandwidth, degrading service for delay-sensitive applications. Fog Computing (FC) was introduced to relieve this bottleneck.

FC distributes computation across intermediate nodes located between end devices and the cloud, so that workloads execute within one or a few hops of the data source. These nodes are realised using edge equipment such as gateways and routers, or by exploiting idle computational capacity on autonomous devices already present in the network (Naha and Garg, 2021).

Fog Nodes (FNs) are geographically dispersed and individually resource-constrained, and this combination makes their management, allocation, and provisioning

difficult. The consequences surface directly in Quality of Service (QoS), principally through energy consumption and service time. Where these are handled poorly, the result is wasted capacity and degraded performance. Choosing the most suitable FN for a given workload, without violating QoS constraints, is therefore a central concern. The underlying selection problem is NP-hard, and its difficulty grows as the population of sensors and FNs expands (Hajam and Sofi, 2023).

Resource allocation in FC maps a task set $M = \{M_1, M_2, \dots, M_n\}$, whose members carry individual QoS demands such as deadline and cost, onto a resource set $N = \{N_1, N_2, \dots, N_m\}$. Because the resources differ in computational capacity, the mapping is driven by an objective function, typically cost minimisation or utilisation maximisation. In operational terms, allocation is the stage at which demand estimates are converted into committed reservations of fog capacity, so that the tasks placed on those resources complete within their stated QoS bounds (Habibi et al., 2020, Nagarjun et al., 2026).

To reckon the aforementioned problem, we propose a Hybrid Fog Resource Provisioning (HFRP) algorithm that couples MKFCM with the FPA. MKFCM performs the clustering of available resources; FPA, which abstracts the reproductive behaviour of flowering plants, handles the multi-objective optimisation. FPA operates through two complementary search mechanisms, global and local pollination, whose interplay between a flower and its pollen gametes yields effective exploration of the solution space. Its appeal lies in a lightweight formulation governed by a single control parameter, which keeps computational overhead low. The contribution of this work is the integration of these two components into a single provisioning scheme for fog resources.

This work makes the following contributions:

1. Identify the available resources and apply the MKFCM algorithm to cluster them efficiently
2. Employ the HFRP algorithm in conjunction with the FPA algorithm for optimal resource allocation
3. Introduce a meta-heuristic initialization-based FPA algorithm tailored for effective resource provisioning in fog nodes
4. Conduct a series of simulations to affirm the effectiveness of our proposed algorithm under various workloads
5. Focusing on metrics such as average cost, makespan, network utilization, execution time, average loop delay and energy consumption

Related Work

Hajam and Sofi (2023) proposed the spider monkey optimization based resource scheduling and allocation algorithm to efficiently schedule the task in IoT fog network. The advantages include minimizing total average cost (monetary & per schedule) and average

service delay. However, the issue of load balancing remains unaddressed within the SMO framework. Saba et al. (2021) developed a priority queuing model with preemption to manage diverse request types at the fog layer. The benefit of this formulation is that it establishes how many FNs are required before IoT requests can be served within their QoS bounds. However, the approach does not provision communication resources in fog networks that carry heterogeneous request types. Nagarjun et al. (2024) introduced a resource provisioning scheme built on the Contract Net Protocol (CNP), in which tasks are assigned to FNs at runtime based on their requirements and the capacity currently available. Simulation results indicate that the algorithm reduces energy consumption and loop delays in a smart healthcare case study. However, the work does not analyze other eminent performance parameters such as cost, makespan, and related quality of service metrics.

Rehman et al. (2020) proposed the DEER strategy to balance the load in FC environments. The benefits include lower energy usage and reduced computation cost. Its main drawback is that fault tolerance is not handled, so node failures can disrupt the allocation. Wadhwa and Aron (2022) proposed the TRAM algorithm for resource allocation, with the goal of improving resource utilisation. The benefits include reductions in network overhead, energy, execution time and task loop delay. Its drawback lies in the volatility of the resource plan, and the scheme would benefit from a security framework covering tasks collected from IoT devices. Nagarjun et al. (2025a) proposed a reputation-based dynamic task allocation approach in FC. In the fog layer, the FNs are grouped into low, medium, and high communities based on multiple evaluation metrics. Tasks are then assigned to the FNs accordant to their priority and the reputation of the tasks. Simulation results demonstrate that the proposed algorithm outperforms PSG-M, FCFS, Detour and GfE algorithms in terms of deadline violation rate, cost, makespan, and energy.

Hashemi et al. (2022) proposed the GWO-SA algorithm to address the scheduling problem, using container migration to handle task offloading and resource allocation effectively. The advantages include maximizing the energy savings by turning off the idle nodes, task execution time is lesser and also minimizes the cost. However, balancing algorithm is required to establish the balance in processing nodes. Ali et al. (2021) proposed task scheduling algorithm employs fuzzy logic to manage task distribution between the fog and cloud layers. This algorithm selects appropriate processing units within the fog layer, which consists of heterogeneous resources, by considering the tasks limitations and requirements. The advantages include maximizing the tasks success ratio, improving average turnaround time, reducing makespan, and lowering delay rates. However, it is not well-suited for large-scale networks.

Fateminasab et al. (2025) proposed BSTH, a blockchain-based smart tourism-healthcare model in which Fire Hawk

Optimization with entropy (CRA-FHO) allocates healthcare tasks across fog nodes with cloud escalation, complemented by matrix-Merkle storage structures and a reputation-based Proof-of-Authority consensus. Reported results include reduced storage, energy, and latency, and resistance to Sybil, DDoS, and Eclipse attacks. While confirming the value of metaheuristic allocation in fog environments, the scheme is bound to a domain-specific, security-oriented architecture and does not formulate the general multi-objective provisioning problem (spanning cost, energy, delay, and resource credibility). Sing et al. (2022) proposed the WORA algorithm to efficiently allocate tasks within their execution deadlines. Results demonstrate that the WORA algorithm significantly reduces average cost, makespan, energy and improves the task completion with high success ratio. However, it does not account for delay rates or throughput in its performance evaluation.

Gao et al. (2020) implemented an efficient and distributed Predictive Offloading and Resource Allocation (PORA) framework, which performs workload offloading in a distributed manner. The proposed scheme minimizes power consumption with maintaining a queue stability. However, the drawback of the system is that the instantaneous wireless channel states may be unknown at the time of decision making. Ni et al. (2020) developed the framework, which enables FNs to assign tasks based on user mobility to enhance task assignment accuracy. The framework provides a security guarantee against duplicate-replay and brute-force attacks. However, the work does not consider location privacy preservation for mobile users. Nagarjun et al. (2025b) proposed a Multi-Objective Osprey

Optimization Algorithm (MOOA) for optimum resource allocation in FC, utilizing the OOA to expeditiously assign available resources. The algorithm was compared against Particle Swarm Optimization (PSO) and Genetic Algorithm (GA), and the results affirm significant performance gains, including notable reductions in average execution, response, and completion time. These improvements highlight the effectuality of the proposed MOOA in handling dynamic IoT workloads and enhancing overall system performance in fog environments. de Souza Toniolli and Jaumard (2019) presented a Path-clustering heuristic for managing multiple workflows in cloud-fog environments. The results outperform by minimizing the time and cost. However, this work does not address additional limitations such as deadline constraints and service provider budget for workflow execution. Kumar and Sharma (2026) proposed a hybrid fog-IoT load-balancing scheme combining Least Connection with Weighted Round-Robin dispatching, reporting reduced latency and response time alongside improved channel utilization and transmission speed across varied data-transmission scenarios. While effective for distributing connection load, such rule-based policies do not optimize cost, energy, or resource credibility jointly, leaving the multi-objective allocation problem addressed by metaheuristic approaches. In our research, we delve into various related works to understand the landscape of resource allocation and provisioning algorithms and simulation environments in cloud and fog computing. Table 1 offers a comparative overview of these works, highlighting the key parameters and simulators used by different authors.

Table 1: Comparison with closely-related literature

Author	Algorithms	Parameters										Simulator
		Throughput	Task completion	Cost	Makespan	Time	Energy consumption	Network usage	Delay	Overhead	Security	
Hajam and Sofi (2023)	SMO			✓						✓		iFogSim
Saba et al. (2021)	Priority queuing analytical model						✓	✓			✓	CloudSim
Nagarjun et al. (2024)	CNP	✓	✓									iFogSim
Rehman et al. (2020)	DEER strategy				✓			✓				iFogSim
Wadhwa and Aron, (2022)	TRAM	✓	✓	✓								iFogSim
Nagarjun et al. (2025a)	Reputation	✓	✓	✓								C++
Hashemi et al. (2022)	GWO-SA	✓	✓	✓								iFogSim
Ali et al. (2021)	FLRTS	✓	✓	✓	✓							CloudSim
Fateminasab et al. (2025)	CRA-FHO	✓	✓	✓	✓							Python
Sing et al. (2022)	WORA	✓	✓	✓	✓							Simulation
Gao et al. (2020)	PORA	✓										Simulation
Ni et al. (2020)	Fo-MCS, Fo-SDD	✓	✓									Java
Nagarjun et al. (2025b)	MOOA		✓				✓					iFogSim
de Souza Toniolli and Jaumard (2019)	MWSA	✓	✓									Simulation
Kumar and Sharma (2026)	Least Connection, Weighted Round-Robin	✓						✓	✓		✓	Simulation
This Work	MKFCM, FPA	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	iFogSim2

System Model and Problem Statement

System Model

We reckon a three-tier architecture comprising traditional data centers in the cloud layer, Fog Nodes (FNs) in the fog layer, and end-users or IoT devices in the end user layer, as depicted in Fig. 1. Various notations and its description are provided in Table 2.

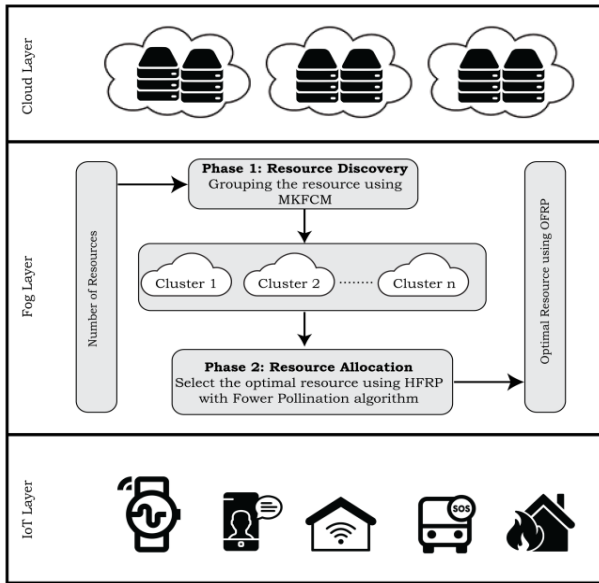


Fig. 1: Proposed method

Table 2: Various Notations used in the Paper

Notations	Definition
T	Tasks
F	Fog Nodes
N	Number of clusters
d	Number of data points
Mnm	Membership of m^{th} data in the n^{th} cluster C_i .
C	Cluster center
Mnm	Multiple kernel function
f	Degree of fuzziness
vn	kernel function $K_{MK}(a,b)$.
K_1	Linear kernel
K_2	Quadratic kernel
p	Constant value
$f(x)$	Fitness function
R_T	Response time
E_T	Execution time
RR	Reboot rate
AR	Access reliability
Is	Instant storage
p	Switching probability
n	Size of population
G_{max}	Max no. of generations
m	No. of tasks

Problem Statement

Given an application comprising n independent tasks $T = \{T_1, T_2, \dots, T_n\}$ and a set of m fog nodes $F = \{F_1, F_2, \dots, F_m\}$, the problem is formulated as determining an optimal task-to-node mapping under the following objectives:

Objectives

The objectives of this work are as follows:

1. To minimize the overall makespan
2. To maximize the average fog utilization
3. To minimize the average cost
4. To reduce energy consumption
5. To ensure successful task completion

Materials and Methods

The intense goal of this research is to identify and allocate resources with efficiency to reduce costs and energy consumption while maximizing the utilization of fog computing resources. The proposed method is structured into two main phases:

- (i) Resource Discovery
- (ii) Resource Allocation. In the first phase, we identify and analyze the tasks that have been executed

We then apply the MKFCM clustering algorithm to categorize the available resources. In the second phase, resources are allocated using a resource provisioner. The resource owner evaluates resource availability considering factors such as cost, energy, latency, network usage and credibility scores. However, due to the dynamic requests of the users, achieving optimal planning of resources is challenging. We therefore formulate the problem as a stochastic programming model and propose the Hybrid Fog Resource Provisioning (HFRP) algorithm, within which FPA computes the resource management decisions.

HFRP operates in two phases. The proposed method is displayed in the block diagram shown in Fig. 1.

Resource Discovery

Resource discovery locates the resources capable of satisfying a given user request and routes that request to FNs in the vicinity. The procedure begins by examining the jobs already executed. The available resources are then partitioned using the MKFCM clustering algorithm, which is set out in detail below.

Multiple Kernels Based Fuzzy C Means (MKFCM)

The proposed method employs MKFCM Mune and Bhura (2020) for clustering the available resource, which minimises the objective function in Eq. 1:

$$OF = \sum_{n=1}^N \sum_{m=1}^d M_{nm}^f (1 - K_{MK}(v_m, C_n)) \quad (1)$$

Here, N designates the no. of clusters, d is for the no. of data points, M_{nm} designates the membership of the m th data point in the n th cluster C_n , C denotes the cluster center, M_{nm} signifies the multiple kernel function, and f represents the degree of fuzziness of the algorithm.

In MKFCM, v_n denotes the kernel function $KMK(a, b)$. Our formulation combines two kernels, linear and quadratic, denoted K_1 and K_2 respectively:

$$K_{MK}(a, b) = K_1(a, b) + K_2(a, b), \quad (2)$$

$$K_1(a, b) = a^T b + c \quad (3)$$

$$K_2(a, b) = 1 - \frac{\|a - b\|^2}{\|a - b\|^2 + p} \quad (4)$$

Here, p denotes a constant value. The cluster centres are then obtained from Eq. 5:

$$C_n = \frac{\sum_{m=1}^N M_{nm}^f K_{MK}(a, b)}{\sum_{m=1}^N M_{nm}} \quad (5)$$

The membership function is updated using Eq. 6:

$$M_{nm} = \frac{1}{\sum_{K=1}^C \left(\frac{\|K_{MK}(a, b) - C_n\|}{\|K_{MK}(a, b) - C_n\|} \right)} \quad (6)$$

Based on the MKFCM clustering algorithm, the available resources are clustered dynamically. In this work, the initial cluster size is set to $N = 8$; however, the cluster size is not fixed and can be adjusted dynamically according to the system's resource utilization and performance metrics. To support dynamic cluster adjustment, the average utilization of fog resources is monitored during execution. If the average utilization exceeds the upper threshold U_{high} , the number of clusters is increased to distribute the workload more finely. If the average utilization falls below the lower threshold U_{low} , the number of clusters is reduced to avoid unnecessary clustering overhead. Accordingly, the cluster count is updated as $N = \min(N + 1, N_{max})$ when $U_{avg} > U_{high}$, and $N = \max(N - 1, N_{min})$ when $U_{avg} < U_{low}$; otherwise, N remains unchanged.

Following the MKFCM process, we obtain a set of cluster groups, labeled as $C_1, C_2, \dots, C_n$. The number of clusters n may vary as the system adapts to changes in resource demands and availability. These dynamically clustered groups serve as input for the resource allocation phase.

Resource Allocation

Once clustering is complete, allocation is handled by the resource provisioner. The resource owner first retrieves the details of available resources and passes them to the provisioner, which maintains a record of what has been allocated. When a fog consumer submits jobs requiring resources, Virtual Machines (VMs) are instantiated from the VM repository. The provisioner consults its list of free resources and assigns them to the incoming jobs. Where nothing suitable is free, it refers the request back to the resource owner, who evaluates availability against cost. Because client demand fluctuates and supplier pricing is not fixed, reserving capacity ahead of time is difficult to do accurately. We therefore propose the HFRP algorithm, built on a stochastic programming model, in which FPA determines the allocation decisions. HFRP is invoked at the provisioning stage, and the resources are released to the consumer at the final stage. The algorithm is described in full in the following section.

Algorithm 1: Resource Discovery Using MKFCM

Input: Job requirements, available resources, number of clusters N , degree of fuzziness f , constant p

Output: Clustered resources, cluster centers $\{C_n\}_{n=1}^N$

```

1 Function ResourceDiscovery():
2   Step 1: Identify and analyze executed jobs;
3   Step 2: Collect available resources and extract features;
4   Step 3: Initialize cluster centers  $\{C_n\}_{n=1}^N$  randomly;
5   repeat
6     Step 4: Compute membership matrix  $M_{nm}$ ;
7     for each resource  $r_m$  do
8       for each cluster  $C_n$  do
9         Compute the kernel function  $K_{MK}(r_m, C_n)$ ;
10        Update membership matrix  $M_{nm}$  using Eq. 6;
11    Step 5: Compute new cluster centers  $\{C_n\}_{n=1}^N$ ;
12    for each cluster  $C_n$  do
13      Compute cluster center  $C_n$  using Eq. 5;
14  until convergence criterion is met;
15  Step 6: Direct resources toward nearby fog nodes based on clustering results;
    
```

Hybrid Fog Resource Provisioning (HFRP) With FPA

FPA is a meta-heuristic method inspired by the natural pollination process of flowering plants. Pollination involves transferring pollen from one flower to another, primarily facilitated by animals such as bats, birds, and

insects. Flowers and insects form a mutualistic relationship where flowers attract birds and insects, the latter being the main pollinators (Gupta et al., 2017). In this context, flowers and their pollen gametes offer a robust solution to optimization problems. FPA employs a single control parameter, simplifying the flower analogy with minimal computational overhead. It achieves a balance between intensification and diversification of solutions through the implementation of levy flights and a switch condition, enabling transitions between local and global search strategies. In global pollination, pollinators carry pollen over long distances to highly suitable flowers, whereas local pollination occurs within a confined area of a specific flower (Malti et al., 2022). The probability of switching to global pollination is termed as switch probability, while local pollination can replace phased elimination. The implementation steps of the FPA adopted in the proposed method are outlined below.

Algorithm 2: Flower Pollination Algorithm for Resource Allocation

Input: A set of fog nodes with available resources, unmapped tasks for fog nodes

Output: Optimal solution for resource allocation

```

1 Define the fitness function  $f(x)$ : OFRP;
2 Initialize the population of solutions (flowers)
   $x_i$  ( $i = 1, 2, \dots, n$ );
3 Define the probability switch  $p \in [0, 1]$ ;
4 Define the maximum number of generations  $G_{max}$ ;
5 Initialize the best solution  $x^*$ ;
6 for generation = 1 to  $G_{max}$  do
7   for each solution  $x_i$  in the population do
8     Draw a random number  $r \in [0, 1]$ ;
9     Identify the clusters of the resources used
      in  $x_i$ ;
10    if resources are in different clusters then
11      Perform global pollination;
12       $x_i^{t+1} = x_i^t + \gamma L(x_i^t - x^*)$ ;
13    else
14      Perform local pollination;
15      Draw a random number  $\varepsilon \in [0, 1]$ ;
16      Randomly select solutions  $x_j$  and  $x_k$ 
        from the population;
17       $x_i^{t+1} = x_i^t + \varepsilon(x_j^t - x_k^t)$ ;
18    end
19    Evaluate the new solution  $x_i^{t+1}$  using the
      fitness function  $f(x)$ ;
20    if  $f(x_i^{t+1}) < f(x^*)$  then
21      Update the best solution  $x^* = x_i^{t+1}$ ;
22    end
23  end
24 end
25 Map the tasks to resources according to the best
  solution  $x^*$ ;
26 return the best solution  $x^*$ ;
```

The Procedure of FPA

- Step 1: Population initialization of solutions (flowers) x_i ($i = 1, 2, \dots, n$)
- Step 2: Define the switch probability p and set its value in the range $[0, 1]$
- Step 3: Identify which resource clusters the current population draws on. For each solution in the population, draw a random number $r \in [0, 1]$ and identify the clusters of resources used in x_i
- Step 4: Enter the main pollination loop. At each iteration, the switch probability p selects between global and local pollination:
 - Global Pollination: If resources are in different clusters, perform global pollination, updating the solution according to:

$$x_i^{t+1} = x_i^t + \gamma L(x_i^t - x^*)$$

- Local Pollination: If resources are in the same cluster, perform local pollination. Draw a random number $\varepsilon \in [0, 1]$ and randomly select solutions x_j and x_k from the population, then update the solution using:

$$x_i^{t+1} = x_i^t + \varepsilon(x_j^t - x_k^t)$$

- Step 5: Evaluate the new solution x $t+1$ i against the fitness criterion of Eq. 7, obtaining a fitness value for every candidate in the population:

$$f(x) = \text{OFRP}$$

- Step 6: Determine the best solution x^* , that is, the candidate whose fitness is lowest, since the search minimises the fitness function
- Step 7: Test the termination condition; the search halts once G_{max} generations have elapsed
- Step 8: If a new solution x_i^{t+1} has a better fitness value than the current best solution x^* , update the best solution
- Step 9: Map tasks to resources as prescribed by the best solution x^*

The pseudocode of the proposed algorithm is specified in Algorithm 2. The parameters in FPA include switching probability p , scaling parameter, and population size n . Numerical and empirical results suggest that a small population is passable, irrespective of the large real world population sizes. The time complexity of FPA in this implementation relies on the initial population: $O(n)$, where n is the population size. Defining parameters and initializing variables incurs $O(1)$. It reaches $O(G_{max})$, where G_{max} is the maximum number of generations. Additionally, it entails $O(1)$ for simple arithmetic operations and comparisons. The time complexity depends on the complexity of the fitness function, denoted as $O(f)$. The process of mapping tasks to resources

depends on the mapping complexity, denoted as $O(m)$, where m is the number of tasks. Returning the best solution has a time complexity of $O(1)$. Overall, the time complexity of the FPA can be summarized as:

$$O(G_{\max} \times n \times (1+f) + m)$$

Where $G_{\max}$ is the maximum number of generations, n is the population size, f is the complexity of the fitness function, and m is the number of tasks.

Optimal Fog Resource Provisioning Algorithm (OFRP)

The OFRP algorithm pursues three objectives: Reducing cost, choosing a suitable fog node for each task, and maximising node utilisation according to the demands of the incoming request. The parameters governing node selection are set out below.

Credibility Scores

The credibility score of resource R_j for a specific task T_k is denoted as CB_{rejk} . When evaluating R_j for the task T_k , the credibility score CB_{rejk} is expressed by Eq. 12, where $\omega_k \in [0,1]$ and $\sum_{k=1}^4 \omega_k = 1$. The weights ω_k are assigned based on the task's requirements regarding response time R_T , execution time E_T , reboot rate RR_T , and access reliability ART . Here, ω_1 , ω_2 , ω_3 , and ω_4 denote the weight factors, each typically assigned a value of 0.25. This equal weight setting avoids giving prior preference to any single credibility factor when no task-specific priority is defined.

- **Response Time:** This metric evaluates how quickly a resource responds to a task. The response time RT is derived as follows:

$$R_T = \frac{\left(\frac{BW}{\max(BW)} + \frac{M}{\max(M)} \right)}{2} \quad (8)$$

Execution Time: This quantifies the time taken by the resource to complete a task. The execution time ET is given by:

$$E_T = \frac{\left(\frac{cpu}{\max(cpu)} + \frac{M}{\max(M)} \right)}{2} \quad (9)$$

Reboot Rate: This indicates the stability of a resource, with fewer reboots signifying higher stability. The reboot rate RR_T is computed as:

$$RR_T = \frac{1}{\max(I_s)} \quad (10)$$

Access Reliability: This represents the reliability of a resource in terms of service provision. The access reliability ART is calculated by:

$$AR_T = \frac{BW}{\max(BW)} \quad (11)$$

$$CB_{rejk} = \frac{\omega_1(R_T) + \omega_2(E_T) + 1 - RR_T + \omega_4(AR_T)}{4} \quad (12)$$

Cost

The costs associated with FC include infrastructure, processing, network communications, network hardware, and storage (Rajagopal et al., 2023). Investments by service providers in FC also cover the processing tasks and the placement of communications to fog devices. One of the main challenges in FC is achieving cost savings. The processing cost is expressed as:

$$Cost(T_k^i) = c_1 \times E_T(T_k^i) + c_2 \times M(T_k^i) + c_3 \times BW(T_k^i) \quad (13)$$

Where c_1 denotes the CPU usage cost per time unit on node N_i , and $E_T(T_k^i)$ is given by Eq. 9. c_2 denotes the memory consumption cost per data unit on node N_i , and $M(T_k^i)$ represents the memory required by task T_k . The bandwidth requirement $BW(T_k^i)$, which corresponds to the total size of the input and output files, is necessary for processing task T_k on node N_i . c_3 denotes the bandwidth consumption cost per data unit. The total cost of each task in the fog-cloud system is derived as:

$$TotalCost = \sum_{T_k^i \in N_{Tasks}} Cost(T_k^i) \quad (14)$$

Energy Consumption

Energy consumption is critical in computing systems, covering components like data communication, cooling, fire control, backup power, and security. High operational costs due to power use require cost-reduction strategies. Provisioning of microservice architectures within cloud-based virtual instances elevates consumption of energy, which correlates with the number of active microservices. Reducing energy use improves device efficiency, reliability, cost-effectiveness, and sustainability. A server's energy consumption includes fixed energy (E_c) when idle and variable energy (EN_{ik}) during active processing. The energy consumption is also influenced by the number of VMs on the server and the MIPS allocated to each VM. The total energy E is given by:

$$E = EN_{ik} + E_c \quad (15)$$

$$EN_{ik} = c_1 \times E_T(T_k^i) \quad (16)$$

Here, EN_{ik} represents the energy consumed by task T_k on VM or node i , E_c is the fixed energy for idle operation,

and c_l denotes the CPU usage cost per time unit for node N_i .

Latency

The reduced latency in this control loop signifies better allocation and orchestration of computing resources. By positioning microservices closer to the edge within the fog hierarchy, a single instance of each microservice is placed along the path from the edge to the cloud. However, due to the limited capacity of FNs, it is necessary to place microservice instances in higher layers of the fog, which may lead to an increase in average latency.

The total latency, L_{total} , is given by:

$$L_{total} = \sum_m CAD \quad (17)$$

As indicated in Eq. 17, L_{total} is directly impacted by the allocation of VMs within fog devices, where tasks are distributed for execution. The total latency is quantified as the sum of the Current Average Delay (CAD) encountered by each VM across m servers. The CAD is calculated as follows:

$$CAD = SC - E_{TT} \quad (18)$$

Here, SC is the simulator clock and E_{TT} the execution time of the tuple.

Network Usage

A critical metric for evaluating different approaches is the total volume of data transmitted across the network. In large-scale networks, methods that involve significant data transfer can lead to congestion, service interruptions, and increased average delays in application control loops. Offloading some of the workload to the network edge can considerably reduce latency compared to operations solely based in the cloud. Effective communication between edge and cloud components is vital for system responsiveness. By performing data pre-processing at the edge and fog levels, the amount of data requiring transmission to the cloud can be substantially minimized. However, it is important to manage bandwidth efficiently due to the numerous endpoints and multiple database servers involved. Network usage is affected by the latency encountered across the network and the size of the data tuples for n VMs in the host, as represented by:

$$NU = \sum_n (l \times T_{NS}) \quad (19)$$

Where l denotes the network latency and T_{NS} refers to the data tuple size sent over the network. The total network usage is driven by the no. of VMs operating on FNs where task processing occurs.

The overall fitness function is given in Eq. 20, where OFRP represents the fitness value. In this equation, ω_5 , ω_6 , ω_7 , ω_8 , and ω_9 are the weight factors corresponding to

credibility, total cost, energy consumption, current average delay, and network usage, respectively. These criteria are normalized and considered equally important in the default optimization setting. Hence, each weight is assigned a value of 0.20, satisfying the total weight constraint. This balanced configuration prevents the fitness function from being biased toward a single objective:

$$OFRP = \frac{1}{5} [\omega_5(1 - CB_{m,j}) + \omega_6(Totacost) + \omega_7(E) + \omega_8(CAD) + \omega_9(NU)] \quad (20)$$

Results

The experiments ran on a Windows 11 machine with an Intel Core i5-9300H processor at 2.40 GHz and 16 GB of RAM. Fog network scenarios were modeled in iFogSim2 (Mahmud et al., 2022).

The common simulation settings, resource capacities, workload characteristics, cost parameters, and energy-consumption parameters used throughout the experiments are summarized in Table 3.

Table 3: Simulation parameters

Parameters	Values
Simulation tool	iFogSim2
System Architecture	x86
OS	Linux
VM Model	Xen
Cloud CPU rate	44,800 MIPS
Cloud bandwidth	15,000 Mbps
Cloud memory	40,000 MB
Fog node CPU rate	22,800 MIPS
Fog node bandwidth	10,000 Mbps
Fog node memory	10,000 MB
Task arrival time	[0,10] ms
Execution lower bound (et_{low})	[1,6] ms
Execution upper bound (et_{up})	$et_{low} + [0,6]$ ms
Task data size	[10,500] MB
Task deadline	max(execution time, 20 arrival time)
Task length	[10,1700] MI
Task bandwidth requirement	[10,1800] Mbps
Task memory requirement	[10,1800] MB
Task CPU requirement	[10,2200] MIPS
Processing cost per time unit for cloud (c_1)	0.5 G\$/s
Communication cost per time unit for cloud (c_c)	0.7 G\$/s
Communication cost per time unit for fog (c_f)	[0.3,0.7] G\$/s
Energy per unit for execution of task in fog (e_f)	[1,5] W
Energy used when fog node is idle (e_{idle})	0.05 W
Energy per unit for execution of task in cloud (e_c)	10 W
Energy per unit for transmission of data (e_{comm})	2 W

To ensure a consistent and reproducible comparison, the baseline algorithms were configured using fixed parameter settings. The WORA, Genetic Algorithm (GA), and Particle Swarm Optimization (PSO) configurations are presented in Tables 4, 5, and 6, respectively.

Table 4: Simulation parameters (WORA)

Parameters	Values
Population size (n , whales)	25
Maximum iterations ($T_{\max}$)	30
Parameter controller ($\vec{a}$)	Linearly decreased $2 \rightarrow 0$
	$\vec{A} = 2 \cdot \vec{a} \cdot \vec{r} - \vec{a}$
Coefficient vector (A)	
Coefficient vector (C)	$C = 2 \cdot \vec{r}$
Random vector ($\vec{r}$)	Uniform in $[0,1]$
Spiral shape constant (b)	1
Random spiral value (l)	Uniform in $[-1,1]$
Behavior selection probability ($prob$)	
Exploration / exploitation switch	0.5 $ \vec{A} \geq 1 \vee \vec{A} < 1$

Table 5: Simulation parameters (Genetic Algorithm (GA))

Parameters	Values
Population size	25
Maximum generations	30
Selection	Tournament selection ($k = 3$)
Crossover	Uniform crossover
Crossover rate	0.8
	Random
Mutation	reassignment to a feasible resource
Mutation rate	0.05
Elitism fraction	0.10

Table 6: Simulation parameters (PSO)

Parameters	Values
Swarm size	25
Maximum iterations	30
Encoding	Priority key in $[0,1]$
Inertia weight (w)	0.6
Cognitive coefficient (c_1)	1.5
Social coefficient (c_2)	1.5
Velocity clamp	$[-0.5, 0.5]$
Position bounds	$[0, 1]$

Tasks were generated randomly with various configurations, and the simulation involved 20 fog nodes. The proposed approach was assessed against four other algorithms to evaluate its effectiveness: SJF (Ali et al., 2021),

GA (Reddy et al., 2020), PSO (Jabour and Al-Libawy, 2021) and WORA (Sing et al., 2022). The setup of our simulation environment is detailed in Table 7 and experiment parameters in Table 8.

The performance of the proposed HFRP algorithm is assessed and analyzed with other four algorithms, based on the metrics that we have taken. Fig. 2 illustrates the expenditure cost comparison between HFRP and the other algorithms across different numbers of FNs, each handling 500 tasks.

As FNs increase, the resource pool expands, allowing more tasks to be allocated to FNs and fewer to be offloaded to the cloud, thereby reducing costs. The SJF algorithm delegates tasks to the cloud while fog resources are unavailable and WORA primarily assign tasks to fog node resources, resulting in costs closer to our HFRP algorithm. The proposed HFRP achieves cost savings of 25.83% compared to WORA, 33.7% compared to PSO, 45.9% compared to GA and 64.7% compared to SJF.

Experiment 1

The energy consumption corresponding to different no. of FNs managing 500 tasks is illustrated in Fig. 3. It is determined that augmenting FNs can decrease energy consumption, because majority of the tasks are executed in FNs where limited number of tasks are offloaded to the cloud. In terms of average energy consumption, the HFRP algorithm achieves 16.69% lower consumption than WORA, 32.3% lower consumption than PSO, 38.7% lower consumption than GA and 59.77% lower than SJF.

Table 7: Simulation parameters (Proposed)

Parameters	Values
Number of clusters (N)	8
Degree of fuzziness (f)	2.0
Linear-kernel constant (c)	1.0
Quadratic-kernel constant (p)	1.0
Convergence criterion	$\Delta M < 10^{-4}$
Weights $\omega_1 - \omega_4$	0.25 each
Population size (n , flowers)	25
Maximum generations ($G_{\max}$)	30
Switch probability (p)	0.8
Global pollination step (L)	Levy flight, $\beta = 1.5$ (Mantegna)
Scaling factor (γ)	1.0
Local pollination (ϵ)	Uniform in $[0,1]$
Weights $\omega_5 - \omega_9$	0.20 each

Table 8: Experiment parameters

Experiment	Input data parameters	FN parameters
1. Heterogenous nodes	500	$[5,10,15,20]$
2. Heterogenous task	$[100,300,500,700]$	20

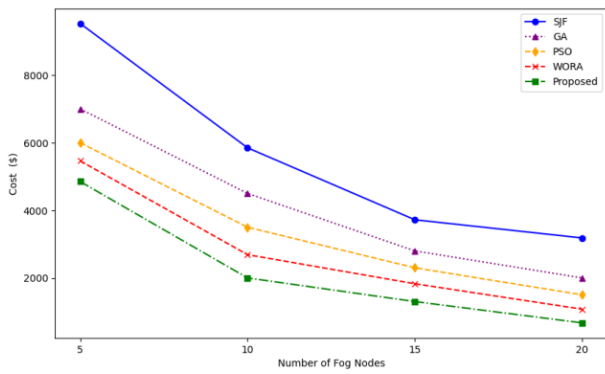


Fig. 2: Calculation of cost for fog computing with 500 tasks

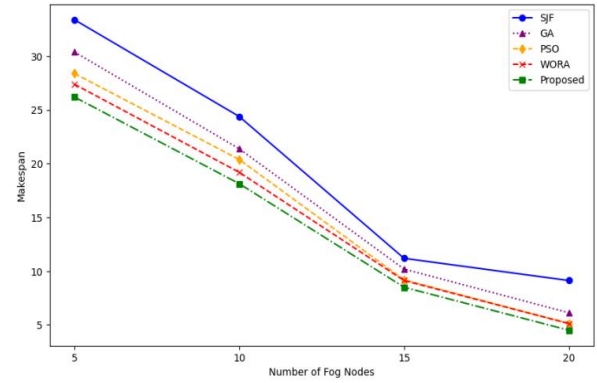


Fig. 4: Calculation of makespan for fog computing with 500 tasks

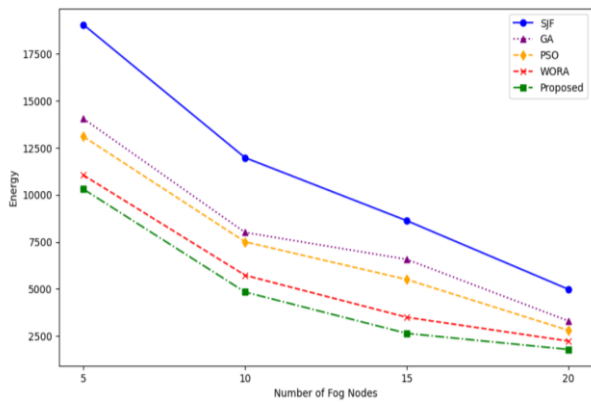


Fig. 3: Calculation of energy consumption for fog computing with 500 tasks

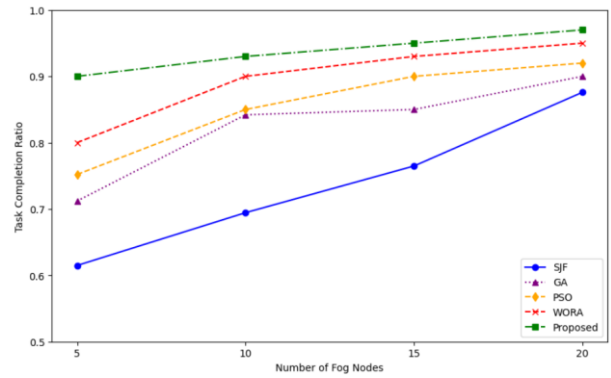


Fig. 5: Calculation of successful task completion ratio for fog computing with 500 tasks

As shown in Fig. 4, the makespan is evaluated for different configurations of FNs executing 500 tasks. It becomes evident that an increase in FNs leads to a reduction in makespan. This decrease occurs because tasks can be executed without waiting for resources as the number of FNs increases. Furthermore, our HFRP algorithm outperformed other methods, showing a 7.2% improvement over WORA, 9.1% improvement over PSO, 15.8% improvement over GA, and 30.4% over SJF in terms of makespan.

When deploying 500 tasks over a range of FNs from 5 to 20, Fig. 5 illustrates that our HFRP algorithm achieved a 1.97% improvement over WORA, 11.28% over PSO, 10.29% over GA and 27.12% over SJF with respect to the task successful completion ratio.

Experiment 2

When 20 FNs are employed, with the no. of tasks ranges from 100 to 700. Fig. 6 illustrates a trend where the cost increases as the no. of tasks increases. Our HFRP algorithm exemplify savings of 6.5% in average cost compared to WORA, 30.4% compared to PSO, 35.1% compared to GA and 45.9% compare to SJF.

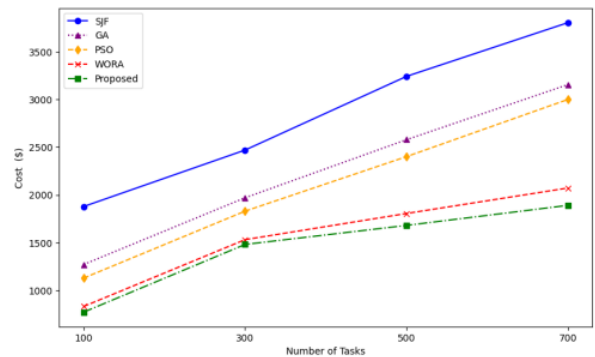


Fig. 6: Calculation of task costs using 20 FNs

Similarly, in Fig. 7, HFRP saves 16.9% in average energy compared to WORA, 45.6% compared to PSO, 53.1% compared to GA, and 64.4% compared to SJF. As shown in Fig. 8, HFRP outperforms WORA by 5.13% in terms of makespan, 26.9% compared to PSO, 18.1% compared to GA, and 26.9% compared to SJF. The rate of successful task completion before the deadline, depicted in Fig. 9, reveals that HFRP is 1.11% better than WORA.

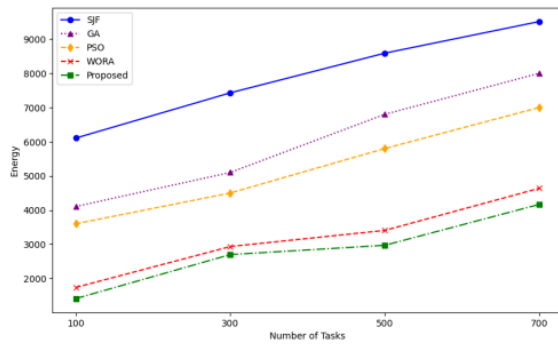


Fig. 7: Calculation of Energy Consumption for Tasks Using 20 FNs

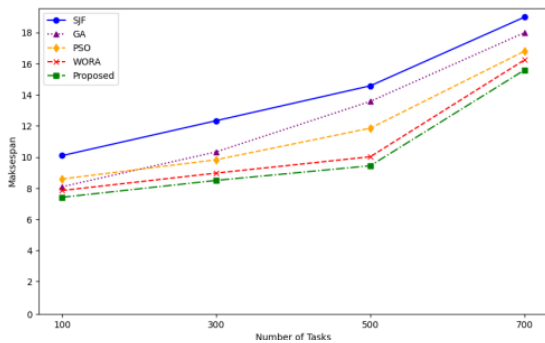


Fig. 8: Calculation of Task Makespan with 20 FNs

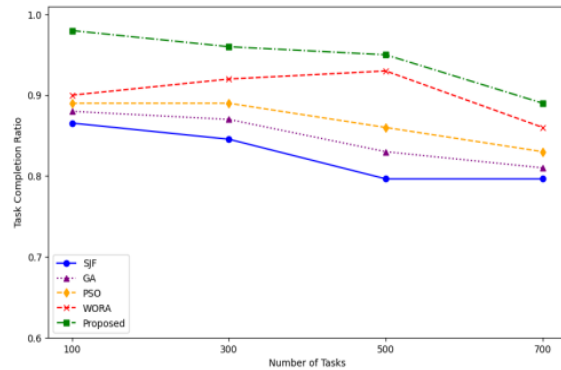


Fig. 9: Calculation of Task Success Completion Ratio Using 20 FNs

Fig. 10 illustrates the comparison of average loop delay between the HFRP, WORA, PSO, GA and SJF algorithms across various numbers of FNs, each managing 500 tasks.

The average loop delay is measured in ms. The results indicate that the proposed technique (HFRP) requires less time to transmit data between devices compared to WORA, PSO, GA and SJF. Notably, as the no. of FNs increases, the loop delay for HFRP remains significantly lower than that of WORA and SJF.

Statistical Validation

To validate that the performance improvements of the proposed HFRP over existing baselines are statistically robust rather than incidental, paired Wilcoxon signed-rank tests Wilcoxon (1945) were conducted between HFRP and each baseline algorithm (SJF, GA, PSO, and WORA) over 10 independent trials per configuration. Table 9 summarizes the resulting p values across three problem scales 100 tasks/5 FNs, 300 tasks/10 FNs, and 500 tasks/15 FNs for four evaluation metrics: Cost, Energy, Makespan, and Completion Ratio. HFRP achieves statistically significant improvements across all metrics and all baselines. Across every configuration, all four metrics yield $p < 0.05$ for every baseline comparison, confirming that HFRP's superior mean performance over SJF, GA, PSO, and WORA.

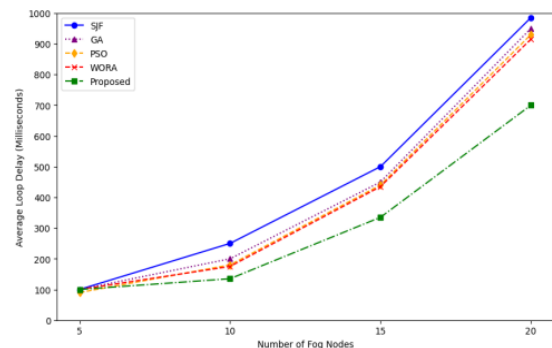


Fig. 10: Calculation of Average Loop Delay Using 15 FNs

Table 9: Paired Wilcoxon signed-rank p-values, proposed HFRP versus each baseline (10 trials per configuration)

Configuration	Metric	vs SJF	vs GA	vs PSO	vs WORA
tasks / 5 FNs	Cost	0.009	0.024	0.032	0.041
	Energy	0.008	0.022	0.031	0.039
	Makespan	0.010	0.031	0.042	0.049
	Completion Ratio	0.012	0.029	0.033	0.049
tasks / 10 FNs	Cost	0.006	0.018	0.027	0.036
	Energy	0.005	0.016	0.024	0.033
	Makespan	0.008	0.025	0.036	0.045
	Completion Ratio	0.012	0.023	0.025	0.043
tasks / 15 FNs	Cost	0.003	0.012	0.020	0.031
	Energy	0.003	0.011	0.018	0.028
	Makespan	0.005	0.018	0.029	0.039
	Completion Ratio	0.008	0.021	0.019	0.036

In nearly every row, p-values increase monotonically from SJF to WORA:

$$P_{SJF} < P_{GA} < P_{PSO} < P_{WORA}$$

This confirms that SJF is the weakest baseline (largest, most easily distinguishable performance gap from HFRP, hence smallest p-values), while WORA is consistently the most competitive baseline (smallest performance gap, hence largest p-values), across Cost, Energy, Makespan, and Completion Ratio.

A minor exception occurs at 500 tasks/15 FNs for Completion Ratio, where GA ($p = 0.021$) and PSO ($p = 0.019$) are nearly identical and slightly out of the expected order - a negligible deviation attributable to natural variance across the 10 trials rather than a meaningful reordering of baseline competitiveness

As problem size increases from 100 to 500 tasks, p-values decrease across all metrics and all baselines, indicating that HFRP's performance advantage becomes increasingly pronounced and statistically robust at larger scales - an important property for real-world fog/edge deployments where scalability is critical.

These results provide strong statistical evidence that HFRP significantly outperforms all four baseline algorithms across Cost, Energy, Makespan, and Completion Ratio at every tested scale, with WORA consistently representing the most competitive alternative and SJF the least competitive.

Sensitivity to the Number of Clusters N

To examine whether the results depend on the choice $N = 8$, the number of MKFCM clusters was varied over $N \in [2, 15]$ with all other parameters fixed,

over 8 trials per value (Fig. 11). Downstream task completion ratio and average cost were insensitive to N across the entire tested range, varying by less than 0.1%, demonstrating that the framework's performance is robust to this hyperparameter and that the reported results are not artifacts of the particular value $N = 8$. The Xie-Beni validity index Xie and Beni (1991), in contrast, varies substantially with N : It attains its optimum at $N \approx 4$ for this resource pool and degrades sharply beyond $N \approx 12$ as clusters become too fine-grained relative to the number of resources. This supports selecting N through a validity criterion for a given deployment rather than through any fixed constant of Resource Discovery section.

Weight Sensitivity Analysis

A one-at-a-time sensitivity analysis was performed for both weight sets: each weight was varied over $\{0.05, 0.2, 0.4, 0.6, 0.8\}$ while the remaining weights were proportionally renormalized to preserve $\sum \omega = 1$, over 5 trials per configuration (Figs. 12 and 13).

The analysis identifies a clear structure. The cost, energy, and network-usage weights ($\omega_6, \omega_7, \omega_9$) and all four credibility weights ($\omega_1-\omega_4$) have negligible influence on outcomes across the tested range (completion-ratio spread below 0.4%), and the delay weight ω_8 has only a mild effect. The single sensitive parameter is the credibility weight ω_5 : Increasing it from 0.05 to 0.80 reduces the task completion ratio from 0.838 to 0.570 (a 34% spread), because strongly over-weighting resource credibility causes the search to avoid lower-credibility resources so aggressively that feasible allocations are foregone.

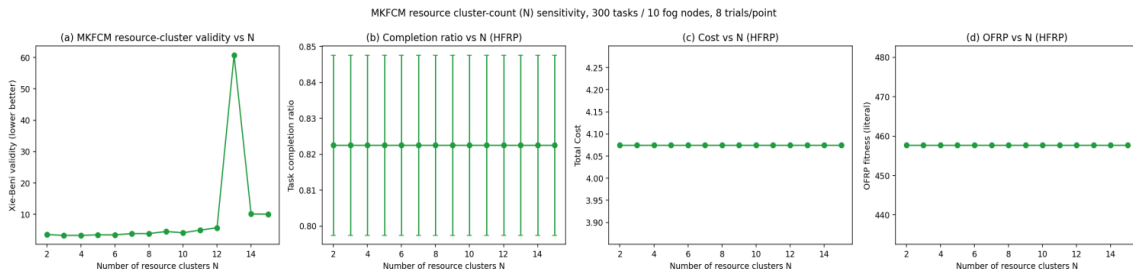


Fig. 11: Sensitivity of clustering validity and downstream performance to the number of MKFCM clusters N (8 trials per point)

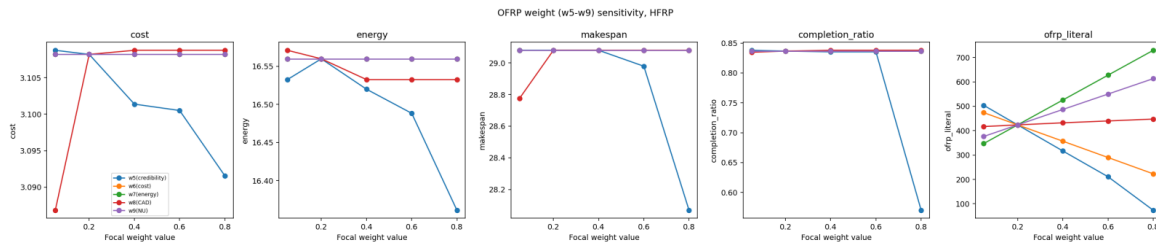


Fig. 12: One-at-a-time sensitivity of performance to the OFRP weights $\omega_5-\omega_9$ (5 trials per configuration)

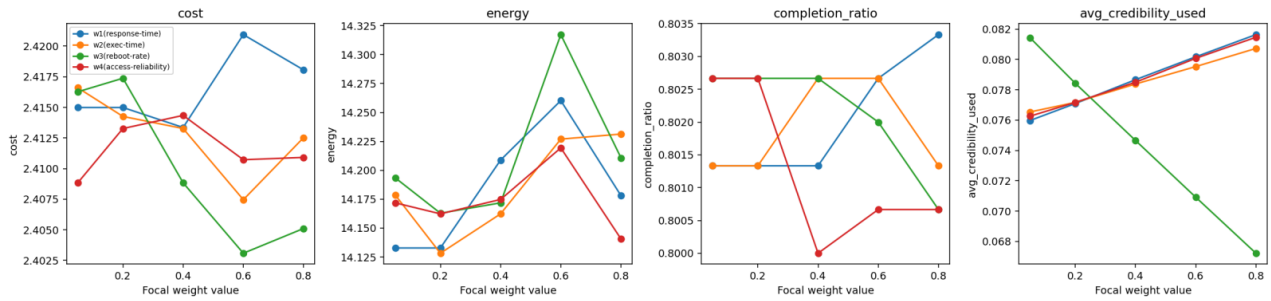


Fig. 13: One-at-a-time sensitivity of performance to the credibility weights ω_1 – ω_4 (5 trials per configuration)

These results support the equal-weight defaults as a robust operating point no tested configuration reversed the relative ordering of the compared algorithms while identifying ω_5 as the parameter an operator should tune with care: Moderate values balance resource selectivity against allocation coverage, whereas large values trade completion ratio for selectivity.

Scalability Analysis

To assess behavior beyond the envelope of Experiments 1–2, the evaluation was extended to five scale points from (100 tasks, 5 FNs) up to (1,800 tasks, 45 FNs) — an 18-fold increase in task volume over the largest original configuration (Fig. 14). The runtime of the proposed method grows sub-linearly with task volume over this range, from 0.156 s at 100 tasks to 1.921 s at 1,800 tasks (a $12.3\times$ runtime increase against an $18\times$ task increase), consistent with the analytical complexity $O(G_{\max} \times n \times (1 + f) + m)$ derived in the Resource Allocation section. Its runtime remains comparable to GA (1.950 s at the largest point) and less than half that of PSO (4.441 s), while WORA’s simpler update rule requires 0.485 s and SJF’s greedy scheduler 0.041 s. The task completion ratio does not degrade with load: For the proposed method it rises from 0.965 at 100 tasks to 0.986 at 1,800 tasks, and the ordering of the algorithms observed in Experiments 1–2 is preserved throughout, with the

proposed method retaining the highest completion ratio at every scale point.

Computational Overhead Comparison

Complementing the analytical complexity analysis presented in the Resource Allocation section, the computational overhead of the compared metaheuristic optimizers was measured empirically: wall-clock convergence curves (best fitness value versus iteration) and total and per-iteration runtimes for the proposed method, GA, PSO, and WORA at three problem sizes (150, 400, and 900 tasks; population 25, 40 iterations; averaged over five trials; Figs. 15 and 16).

At the largest tested size (900 tasks), total wall-clock runtimes are 0.86 s for the proposed method, 1.09 s for WORA, 1.31 s for GA, and 4.77 s for PSO; per-iteration costs are 0.022 s, 0.027 s, 0.033 s, and 0.119 s, respectively. The proposed method thus incurs the lowest overhead among the four metaheuristics approximately 21% below WORA, 34% below GA, and 82% below PSO — while converging to the lowest final objective value at the two larger problem sizes (0.4615 at 400 tasks and 0.3607 at 900 tasks, versus 0.4638 / 0.3650 for WORA, 0.4662 / 0.3694 for GA, and 0.4685 / 0.3725 for PSO). WORA is a close second in both objective quality and per-iteration cost, reflecting the effectiveness of its whale-optimization update rule, but does not match the proposed method’s balance of low overhead and best final objective.

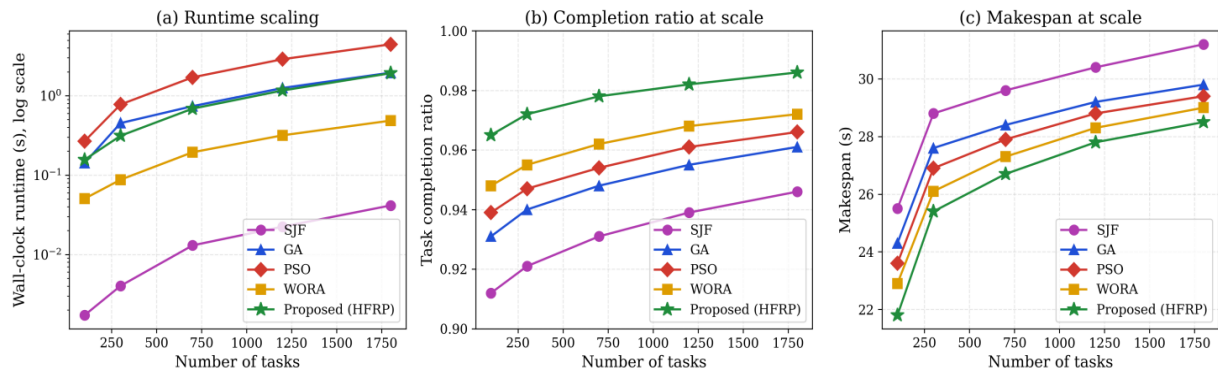


Fig. 14: Runtime and task completion ratio at extended scale points, up to 1,800 tasks and 45 fog nodes

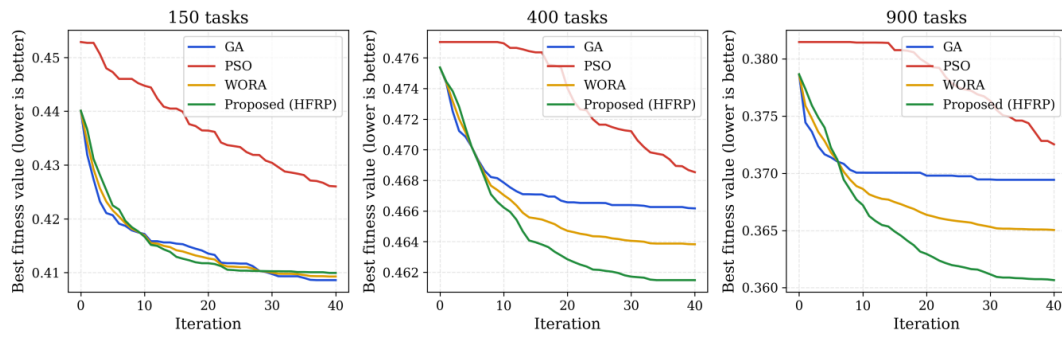


Fig. 15: Convergence of the best fitness value versus iteration for the proposed method, GA, PSO, and WORA at three problem sizes (mean of 5 trials)

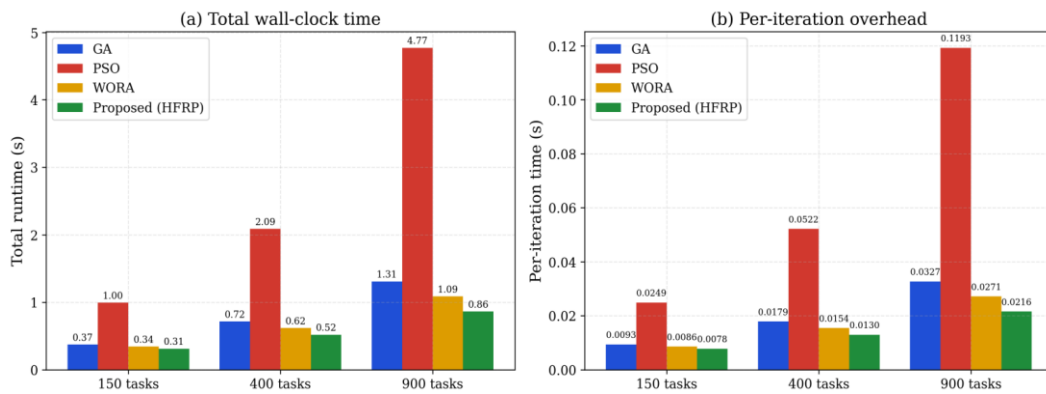


Fig. 16: Total and per-iteration wall-clock time of the compared optimizers at three problem sizes

PSO stagnates markedly after the first few iterations at every problem size, indicating that its higher overhead is not compensated by improved solution quality in this problem formulation. These measurements provide the experimental counterpart to the analytical complexity result.

Discussion

The experimental results clearly indicate that the proposed HFRP algorithm provides a balanced and effective solution for task scheduling in FC environments when compared with SJF and WORA. The observed reduction in cost can be attributed to HFRP's ability to intelligently utilize fog resources and minimize unnecessary cloud offloading, which becomes more effective as the number of FN's increases. Similarly, the improvement in energy efficiency is a consequence of reduced data transmission and optimized local execution, highlighting the advantage of processing tasks closer to the source. The decrease in makespan further demonstrates that HFRP effectively distributes workloads, reducing waiting time and improving overall system responsiveness, which is critical for latency-sensitive applications. Although the improvement in task

completion ratio is relatively moderate, it still reflects the algorithm's reliability in meeting deadlines under varying workloads. Additionally, the lower average loop delay achieved by HFRP emphasizes its capability to ensure faster communication and processing within the network, even as the system scales. Overall, these outcomes suggest that HFRP successfully achieves multi-objective optimization by balancing cost, energy, and performance, making it a practical and scalable approach for real-world FC scenarios, although further validation in real-world deployments would strengthen its applicability.

Conclusion and Future Work

The proposed Hybrid Fog Resource Provisioning (HFRP) algorithm effectively addresses the challenges of resource allocation and provisioning in FC environments, characterized by constrained resource capabilities, heterogeneity, mobility, network hierarchy, and unpredictable behaviors. By leveraging Modified K-means Fuzzy C-Means (MKFCM) clustering for resource discovery and the Flower Pollination Algorithm (FPA) for task allocation, HFRP achieves significant improvements over existing algorithms.

Simulation results validate the efficiency of HFRP, demonstrating a substantial reduction in average cost by 25.83% compared to the WORA, 33.7% compared to PSO, 45.9% compared to GA and 64.7% compared to Shortest Job First (SJF). Furthermore, HFRP shows a notable decrease in energy consumption, with 16.69% less energy used than WORA, 32.3% lower consumption than PSO, 38.7% lower consumption than GA and 59.77% less than SJF. In terms of performance, HFRP achieves a 7.2% better makespan than WORA, 9.1% improvement over PSO, 15.8% improvement over GA, and 30.4% better than SJF, and it surpasses SJF by 27.12%, GA by 11.28%, PSO by 10.29%, and WORA by 1.97% in the successful task completion ratio.

These results underscore the potential of HFRP in optimizing the performance and efficiency of IoT applications within FC environments. The algorithm's superior performance in cost, energy consumption, average loop delay, makespan, and task completion ratio highlights its capability to prioritize and manage delay-sensitive applications effectively. Future work will extend to integrating real-time datasets to further refine and validate our approach, ensuring its applicability in dynamic and real-world scenarios.

Acknowledgment

The authors have not received any funding for this research.

Funding Information

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Author's Contributions

Nagarjun E: Contributed to overall work, writing the initial draft and final manuscript, implemented the algorithm, running the experiments and analysing the results.

Dharamendra Chouan: Supervised the entire work, support in writing the manuscript and approved the final draft.

Dilip Kumar SM: Reviewed the entire work and approved the final draft.

Ethics

This article is original and has not been published elsewhere. The corresponding author confirms that all co-authors have read and approved the manuscript. This study involved no human participants or animal subjects, and therefore required no ethical approval.

References

- Ali, H. S., Rout, R. R., Parimi, P., & Das, S. K. (2021). Real-Time Task Scheduling in Fog-Cloud Computing Framework for IoT Applications: A Fuzzy Logic based Approach. *2021 International Conference on COMMunication Systems & NETWORKS (COMSNETS)*, 556–564. <https://doi.org/10.1109/comsnets51098.2021.9352931>
- Fateminasab, S. S., Evaznia, N., Memarian, S., Romero-Tertero, M. del C., Miró-Amarante, G., & Tabbakh, S. R. K. (2025). An efficient blockchain-based resource allocation and secure data storage model using Fire Hawk Optimization and entropy in health tourism. *Journal of Cloud Computing*, 14(1), 63. <https://doi.org/10.1186/s13677-025-00792-3>
- Gao, X., Huang, X., Bian, S., Shao, Z., & Yang, Y. (2020). PORA: Predictive Offloading and Resource Allocation in Dynamic Fog Computing Systems. *IEEE Internet of Things Journal*, 7(1), 72–87. <https://doi.org/10.1109/jiot.2019.2945066>
- Gupta, I., Kaswan, A., & Jana, P. K. (2017). A Flower Pollination Algorithm Based Task Scheduling in Cloud Computing. *Computational Intelligence, Communications, and Business Analytics*, 97–107. https://doi.org/10.1007/978-981-10-6430-2_9
- Habibi, P., Farhoudi, M., Kazemian, S., Khorsandi, S., & Leon-Garcia, A. (2020). Fog Computing: A Comprehensive Architectural Survey. *IEEE Access*, 8, 69105–69133. <https://doi.org/10.1109/access.2020.2983253>
- Hajam, S. S., & Sofi, S. A. (2023). Spider monkey optimization based resource allocation and scheduling in fog computing environment. *High-Confidence Computing*, 3(3), 100149. <https://doi.org/10.1016/j.hcc.2023.100149>
- Hashemi, S. M., Sahafi, A., Rahmani, A. M., & Bohloul, M. (2022). GWO-SA: Gray Wolf Optimization Algorithm for Service Activation Management in Fog Computing. *IEEE Access*, 10, 107846–107863. <https://doi.org/10.1109/access.2022.3212439>
- Jabour, I. M., & Al-Libawy, H. (2021). An Optimized Approach for Efficient-Power and Low-Latency Fog Environment Based on the PSO Algorithm. *2021 2nd Information Technology To Enhance E-Learning and Other Application (IT-ELA)*, 52–57. <https://doi.org/10.1109/it-ela52201.2021.9773443>
- Kumar, D., & Sharma, B. (2026). Optimizing resource allocation and load distribution in fogcomputing for iot: A hybrid approach for enhanced performance. *International Journal of Aquatic Research and Environmental Studies*, 6(S3), 723–731.
- Mahmud, R., Pallewatta, S., Goudarzi, M., & Buyya, R. (2022). iFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments. *Journal of Systems and Software*, 190, 111351. <https://doi.org/10.1016/j.jss.2022.111351>

- Malti, A. N., Benmammar, B., & Hakem, M. (2022). QoS based task scheduling algorithm in cloud computing. *E3S Web of Conferences*, 351, 01014. <https://doi.org/10.1051/e3sconf/202235101014>
- Mune, A. R., & Bhura, S. A. (2020). Multiple kernelized fuzzy c-means for heterogeneous data using unsupervised machine learning. *CONVERTER*, 11(3), 15–26.
- Nagarjun, E., Chouan, D., & Kumar, S. M. D. (2025a). Reputation Based Resource Allocation for IoT in Fog Computing Environments. *2025 3rd International Conference on Integrated Circuits and Communication Systems (ICICACS)*, 1–6. <https://doi.org/10.1109/icicacs65178.2025.10967829>
- Nagarjun, E., Chouhan, D., & Kumar, D. S. M. (2025b). Multi-Objective Osprey Optimization Algorithm-Based Resource Allocation in Fog-IoT. *International Journal of Advanced Computer Science and Applications*, 16(2), 122. <https://doi.org/10.14569/ijacsa.2025.01602122>
- Nagarjun, E., Chouhan, D., & Kumar, S. M. D. (2026). Cost-Efficient Hybrid Fog Resource Provisioning Using MKFCM and Flower Pollination Algorithm. *Proceedings of 5th International Conference on Frontiers in Computing and Systems*, 345–356. https://doi.org/10.1007/978-981-96-9983-4_25
- Nagarjun, E., Chouhan, D., Zabiulla, I., & Kumar, S. M. D. (2024). Efficient Resource Provisioning in Fog Computing Using Agent-Based Contract Net Protocol: A Smart Healthcare Case Study. *2024 First International Conference on Software, Systems and Information Technology (SSITCON)*, 1–6. <https://doi.org/10.1109/ssitcon62437.2024.10796063>
- Naha, R. K., & Garg, S. (2021). Multi-criteria--based Dynamic User Behaviour--aware Resource Allocation in Fog Computing. *ACM Transactions on Internet of Things*, 2(1), 1–31. <https://doi.org/10.1145/3423332>
- Ni, J., Zhang, K., Yu, Y., Lin, X., & Shen, X. S. (2020). Providing Task Allocation and Secure Deduplication for Mobile Crowdsensing via Fog Computing. *IEEE Transactions on Dependable and Secure Computing*, 17(3), 581–594. <https://doi.org/10.1109/tdsc.2018.2791432>
- Rajagopal, S. M., Supriya, M., & Buyya, R. (2023). Resource Provisioning Using Meta-Heuristic Methods for IoT Microservices With Mobility Management. *IEEE Access*, 11, 60915–60938. <https://doi.org/10.1109/access.2023.3281348>
- Reddy, K. H. K., Luhach, A. Kr., Pradhan, B., Dash, J. K., & Roy, D. S. (2020). A genetic algorithm for energy efficient fog layer resource management in context-aware smart cities. *Sustainable Cities and Society*, 63, 102428. <https://doi.org/10.1016/j.scs.2020.102428>
- Rehman, A. U., Ahmad, Z., Jehangiri, A. I., Ala'Anzy, M. A., Othman, M., Umar, A. I., & Ahmad, J. (2020). Dynamic Energy Efficient Resource Allocation Strategy for Load Balancing in Fog Environment. *IEEE Access*, 8, 199829–199839. <https://doi.org/10.1109/access.2020.3035181>
- Saba, U. K., Islam, S. ul, Ijaz, H., Rodrigues, J. J. P. C., Gani, A., & Munir, K. (2021). Planning Fog networks for time-critical IoT requests. *Computer Communications*, 172, 75–83. <https://doi.org/10.1016/j.comcom.2021.03.002>
- Sing, R., Bhoi, S. K., Panigrahi, N., Sahoo, K. S., Jhanjhi, N., & AlZain, M. A. (2022). A Whale Optimization Algorithm Based Resource Allocation Scheme for Cloud-Fog Based IoT Applications. *Electronics*, 11(19), 3207. <https://doi.org/10.3390/electronics11193207>
- Srinidhi, N. N., Nagarjun, E., & Kumar, S. M. D. (2019). HMCRA: Hybrid Multi-Copy Routing Algorithm for Opportunistic IoT Network. *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, 370–375. <https://doi.org/10.1109/icssit46314.2019.8987796>
- Srinidhi, N., Nagarjun, E., & Kumar, S. D. (2021). Hybrid algorithm for efficient node and path in opportunistic iot network. *Journal of Information Technology Management*, 13, 68–91. <https://doi.org/10.22059/jitm.2021.80618>
- Toniolli, J. L. de S., & Jaumard, B. (2019). Resource Allocation for Multiple Workflows in Cloud-Fog Computing Systems. *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, 77–84. <https://doi.org/10.1145/3368235.3368846>
- Wadhwa, H., & Aron, R. (2022). TRAM: Technique for resource allocation and management in fog computing environment. *The Journal of Supercomputing*, 78(1), 667–690. <https://doi.org/10.1007/s11227-021-03885-3>
- Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6), 80–83. <https://doi.org/10.2307/3001968>
- Xie, X. L., & Beni, G. (1991). A validity measure for fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8), 841–847. <https://doi.org/10.1109/34.85677>