PORT SAID ENGINEERING RESEARCH JOURNAL
Faculty of Engineering - Port Said University
Volume 29 No. 2 pp: 49:61

Enhanced Hybrid Spider Monkey Optimization with Particle Swarm Optimization for Resource Management in Fog Computing Environment

*Radwa Attia[1, *], Eman Saleh[2], Heba Nashaat[3]*

[1]*Electrical Engineering Department, Faculty of Engineering, Port Said University, Port Said, Egypt, email: radwa_yousef@eng.psu.edu.eg*

[2]*Electrical Engineering Department, Faculty of Engineering, Port Said University, Port Said, Egypt, email: me.e9188@gmail.com*

[3]*Electrical Engineering Department, Faculty of Engineering, Port Said University, Port Said, Egypt, email: hebanashaat@eng.psu.edu.eg.*

[*]*Corresponding author,* *DOI: 10.21608/pserj.2025.352182.1389.*

Received 11-1-2025
Revised 24-1-2025
Accepted 28-1-2025

ABSTRACT

The rapid expansion of Internet of Things (IoT) applications has driven the widespread adoption of fog computing, which places computational resources closer to data sources. However, achieving efficient resource management in fog computing environments remains a significant challenge due to diverse resource constraints, dynamic workload variations, and stringent Quality of Service (QoS) requirements. This paper introduces a novel Enhanced Hybrid Spider Monkey Optimization (EH-SMO) algorithm with Particle Swarm Optimization (PSO) for task scheduling in fog computing, inspired by the social foraging behavior of spider monkeys and enhanced by PSO's efficient local search capabilities. The proposed EH-SMO algorithm dynamically allocates computational resources, balances workloads across fog nodes, and optimizes makespan, energy consumption, and resource utilization objectives through adaptive parameter control for energy efficient and low latency performance. Through comprehensive simulation experiments comparing EH-SMO with MCT-SMO and MPSO baseline schemes, the proposed algorithm achieves substantial improvements in scheduling efficiency, energy consumption, and resource allocation by achieving 32% and 17% reduction in makespan, reducing energy consumption by up to 33% and 45%, while improving resource utilization by 18% and 37%, respectively. EH-SMO exhibits robust adaptability to dynamic workload variations and demonstrates superior scalability across different fog computing scenarios, making it a promising solution for real-world IoT applications.

Keywords: IoT, QoS, Optimization, Fog Computing, Resource Management.

1 INTRODUCTION

Internet of Things (IoT) is a network of interconnected devices, sensors, and systems that communicate and exchange data over the Internet [1]. The rapid growth of IoT devices has led to significant advances in various critical fields [2], as shown in Figure 1. However, these IoT devices have limited computing power, storage, and energy resources, making them unsuitable for complex data analysis [3]. The anticipated growth of IoT devices poses a major challenge in managing the massive amounts of generated data. This exponential growth not only intensifies existing challenges in resource management but also introduces complexities in maintaining Quality of Service (QoS) and energy consumption requirements [4]. Fog computing has emerged as an innovative decentralized approach that connects edge devices with cloud infrastructure [5]. Despite offering a lot of processing power, the cloud's centralization causes problems with latency, bandwidth, and network congestion [6-7]. Fog computing resolves these issues, optimizing resource usage and allowing real-time processing by bringing computing, storage, and analysis capabilities closer to the data sources [8]. Combining fog computing with IoT systems faces critical challenges in resource management and task scheduling [9-11], including efficient load balancing,

optimizing energy consumption, and achieving scalable performance across increasingly complex distributed computing environments.

Figure 1: Sophisticated IoT communication paradigms

These challenges demand innovative algorithmic approaches that can intelligently navigate the intricate computational landscape of interconnected devices and intermediate processing nodes. Traditional/Deterministic mechanisms [12] form the foundation, utilizing fixed rules and mathematically proven methods such as First Come First Serve (FCFS), round robin, and priority scheduling. They offer predictable outcomes and straightforward implementation however, their rigid nature limits effectiveness in dynamic scenarios. Learning-based mechanisms [13] are the latest advancement in scheduling innovation, leveraging artificial intelligence through reinforcement learning, deep learning, and neural networks to adapt to changing conditions and improve performance over time, particularly valuable in environments with growing workload patterns. Heuristic-based mechanisms [14] address complex scheduling requirements through problem-specific rules, with approaches like Min-Min, Max-Min, and earliest deadline first algorithms efficiently finding near-optimal solutions while balancing computational overhead for time-sensitive applications. Meta-heuristic mechanisms [15] introduce sophisticated optimization strategies inspired by natural phenomena, with algorithms like genetic algorithms, Particle Swarm Optimization (PSO), and ant colony optimization excelling in handling multiple objectives simultaneously, demonstrating particular effectiveness in scenarios requiring balance across multiple performance metrics.

The optimization of task scheduling in fog computing presents a complex multi-objective challenge that demands sophisticated solution approaches. Given the inherent NP-hard nature of scheduling problems and the limitations of conventional methods, meta-heuristic algorithms emerge as particularly promising solutions. These approaches overcome the computational intractability faced by traditional deterministic methods and the local optima limitations of simple heuristics, while avoiding the extensive training data requirements of learning-based systems [16-17]. To advance this field, an Enhanced Hybrid Spider Monkey Optimization (EH-SMO) algorithm with PSO is proposed to enhance the traditional Spider Monkey Optimization (SMO) through a hybrid architecture that combines global exploration through spider monkey social behavior with PSO-driven local search optimization. The theoretical foundations, system model, and problem formulation of EH-SMO are presented, setting the stage for a detailed exploration of methodology and performance evaluation. The main contributions of this paper are summarized as follows:

A. The system model and optimization framework are developed with a focus on makespan, energy consumption, and cost metrics in fog computing environments. The EH-SMO algorithm is proposed to manage dynamic task scheduling by:
   - Integrating PSO's local search capabilities with SMO's swarm intelligence to enhance convergence speed and solution quality through a novel hybrid position update mechanism.
   - Developing an advanced selection operator that combines velocity-based search efficiency with distance-based resource matching for optimal task-fog node pairing.
   - Improving convergence speed and solution quality through innovative adaptive mechanisms.
   - Achieving efficient resource utilization through hierarchical group organization and adaptive parameter control mechanisms.
   - Optimizing overall system performance by balancing makespan, energy consumption, and cost objectives.

B. Evaluating EH-SMO algorithm's convergence characteristics and analyzing parameter impacts on makespan, energy and utilization weighted optimization objectives to demonstrate algorithm's effectiveness in finding near-optimal scheduling solutions across diverse fog computing scenarios.

The remainder of this paper is organized as follows: Section 2 presents the related works of various resource management and scheduling approaches. Section 3 explains the system model and problem formulation for the proposed EH-SMO algorithm. Section 4 explains the design of the algorithm. Section 5 displays the evaluation methodology and simulation results. Finally, Section 6 draws the conclusion.

2 RELATED WORK

Prior research in fog computing has produced diverse algorithmic solutions to address the inherent complexities of resource management and performance optimization. Several notable traditional scheduling approaches have been proposed.

The Matrix-based Task-Fog Pairing (MTFP), proposed in [18], introduces a dual-matrix architecture that utilizes compatibility and execution time matrices for task allocation. The compatibility matrix establishes viable task-node pairings, while the execution time matrix handles processing requirement calculations. This approach significantly improves processing delay and energy efficiency through systematic workflow prioritization based on deadlines and resource demands. However, it faces limitations in large-scale deployments due to continuous matrix maintenance overhead and shows reduced adaptability in dynamic scenarios due to static resource assumptions. A Resource Aware Prioritized Task Scheduling (RAPTS) algorithm is presented in [19]. It uses a minimum completion time algorithm to manage resource allocation and prioritizes tasks into three priority queues based on deadlines and virtual machine processing delays. This systematic prioritization mechanism optimizes response time, cost, makespan, and resource utilization. However, managing task dependencies and adjusting to dynamic workload variations present the biggest challenge, especially in intricate fog computing scenarios.

Learning-based mechanisms leverage artificial intelligence and machine learning techniques to create adaptive scheduling solutions in fog environments. In [20], a Multi-Objective Fuzzy Approach (MOFA) is introduced, a sophisticated distributed computing algorithm utilizing a multi-agent system with contractual, fog, and cloud agents, enhanced by a fuzzy inference system that intelligently combines genetic algorithms for optimization and fuzzy logic for scheduling evaluation. By dynamically partitioning workflows based on computational intensity and latency sensitivity, the method offers a nuanced strategy for resource allocation across different computational layers. Despite its innovative design, the approach confronts significant challenges in agent coordination and rule-base maintenance, potentially limiting its scalability and effectiveness in highly dynamic computing environments. Region Aware Dynamic Scheduling (RADISH) is introduced in [21] to integrate a bi-class neural network for task classification, MOFA algorithm for QoS-aware scheduling, and soft Actor-Critic algorithm machine learning techniques for load balancing. The model successfully employs virtual machine monitoring with potential field clustering and three-tiered repositories for task distribution. However, it faces challenges in maintaining multiple repositories and coordinating various learning components, particularly in resource-constrained environments requiring rapid adaptation. In [22], the Fuzzy-GEC algorithm merges fuzzy logic and genetic algorithms to create an intelligent task allocation method that simultaneously optimizes computational efficiency and energy consumption. The approach seeks to intelligently distribute computational workloads by employing evolutionary techniques and fuzzy-based encoding. However, the method grapples with technical complexities, including substantial computational demands and the intricate challenge of balancing multiple algorithmic components to achieve peak performance. Optimized Task Scheduling and Preemption (OSCAR) is introduced in [23] to implement a four-phase solution combining expectation-maximization clustering, modified heap-based optimizer, categorical deep Q network, and dynamic preemption strategy. While this comprehensive approach successfully enhances QoS in fog-assisted IoT environments, it faces significant challenges in coordinating multiple mechanisms and adapting to rapid workload changes.

Heuristic-based scheduling mechanisms employ practical rules to achieve near-optimal solutions for fog computing environments. The deadline-aware task scheduling challenge in fog computing is addressed in [24] through a Fuzzy Priority-aware Semi-Greedy (FuzzyPSG) algorithm by incorporating fuzzy logic to handle the inherent uncertainty in fog environments while optimizing deadline satisfaction and energy consumption. However, it faces challenges in managing task dependencies, concurrent execution scenarios, and computational overhead from the multistate procedure, particularly affecting real-time scheduling performance. In [25], the Task-Resource Adaptive Pairing (TRAP) algorithm introduces a sophisticated task-resource allocation approach featuring a three-component architecture designed to optimize computational resource management. By incorporating a batch system for task categorization, a dynamic ranking mechanism based on processing and network metrics, and a priority method considering multiple task attributes, the algorithm aims to enhance load balancing and reduce search space complexity. However, the method encounters significant challenges, particularly in generating accurate initial priority assignments and maintaining scalability in rapidly evolving computational environments that demand frequent batch system reconfiguration. Data-Locality Aware Job Scheduling in Fog-Cloud (DLJSF) algorithm is introduced in [26]. It employs a three-layer architecture and formulates scheduling as an integer linear programming optimization model. It prioritizes local task execution while enabling strategic data migration, effectively minimizing makespan through data locality awareness. However, it encounters significant challenges in environments with dynamic data generation patterns and requires careful management of data replication strategies, particularly in resource-constrained fog nodes.

Meta-heuristic approaches provide sophisticated nature-inspired optimization strategies for complex scheduling problems in fog computing. A heuristic initialization-based Spider Monkey Optimization (MCT-SMO) algorithm is proposed in [27]. It enhances the traditional SMO algorithm by incorporating MCT heuristic initialization for the initial population, explicitly targeting cost optimization through the

combined consideration of service time and monetary expenses.

While demonstrating superior performance compared to other initialization methods and PSO, particularly in average cost and service time metrics, the algorithm faces limitations in addressing load balancing and fault tolerance aspects. This work further validates the potential of enhancing meta-heuristic approaches with heuristic initialization strategies, though it highlights the need for more comprehensive solutions in large-scale fog computing environments. In [28], Latency-Aware Multi-objective Multi-Rank (LAMOM) incorporates makespan, energy consumption, cost, reliability, and workload balance optimization objectives. It employs edge-merge and comp-matching task clustering schemes within a three-tier fog architecture, utilizing multiple task-ranking schemes and parallel sub-schedules. It faces challenges in computational complexity for large task sets and performance consistency across diverse workload patterns. In [29], a Modified Particle Swarm Optimization (MPSO) algorithm is proposed. It presents a dual-phase optimization for application module placement and task allocation using population-based iterative refinement. It demonstrates notable improvements over traditional FCFS and basic PSO approaches regarding resource utilization and system performance. However, it encounters inherent challenges in computational complexity and parameter tuning optimization. Chaotic Opposition-based Differential Evolution Algorithm (CODA) [29] combines chaotic maps and opposition-based learning to create a hybrid optimization strategy. It can effectively prevent premature convergence and avoid local optima while simultaneously optimizing makespan, energy consumption, and cost. However, it struggles to balance the exploration and exploitation phases. It requires careful parameter tuning for optimal performance in different fog computing scenarios.

These scheduling approaches highlight the remarkable potential of meta-heuristic and nature-inspired algorithms in addressing multi-objective optimization challenges in fog computing environments. Although these algorithms have made significant contributions in optimizing makespan, energy consumption, and cost, there remains a need for a comprehensive solution that effectively balances exploration and exploitation while maintaining computational efficiency. This is where the Enhanced Hybrid Spider Monkey Optimization (EH-SMO) algorithm comes in, offering a novel perspective on task scheduling optimization in fog computing through enhanced search strategies and adaptive parameter control.

EH-SMO utilizes enhanced spider monkey foraging behavior to provide a flexible and adaptive solution to the task scheduling challenge in fog computing environments. Unlike many previous approaches, EH-SMO incorporates both PSO-based local search capabilities and adaptive parameter control in its optimization process, allowing it to better balance exploration and utilization in dynamic fog computing scenarios. The key components of existing scheduling mechanisms are identified and compared in Table 1 regarding their scheduling type, optimization objectives, and evaluation environments, establishing the foundation for EH-SMO solution.

3 SYSTEM MODEL AND PROBLEM FORMULATION

The primary objective of EH-SMO algorithm is to optimize task scheduling decisions in fog computing environments by determining optimal task-to-fog node mappings while minimizing energy consumption, makespan, and cost. EH-SMO hybrid model combines Spider Monkey Optimization with Particle Swarm Optimization, followed by a multi-objective problem formulation. The system model incorporates a three-layer architecture with mathematical expressions for time components, energy consumption, and cost factors.

3.1 System Model

The fog computing environment consists of three main layers, as shown in Figure 2. IoT devices create tasks, and the fog processes them and the cloud layer. Each fog node can be represented as $F = \{X, P_o\}$ where $X = \{x_1, x_2, \dots, x_n\}$ represents the n number of fog nodes, P_o represents processing power in Million Instructions Per Second (MIPS). The tasks generated by IoT devices are represented as $T = \{\tau, S, T_{lengt}\}$, where $\tau = \{\tau_1, \tau_2, \dots, \tau_m\}$ represents m number of tasks, S is the size of tasks, and T_{length} represents task length. Various notations of EH-SMO algorithm are shown in Table 2.

3.2 Transmission Time

Transmission time, execution time, and result delivery time are the three primary components that make up each task's completion time when it is assigned to fog nodes. For each task assigned to the fog node, the three-time components are calculated by (1) to (3); respectively.

$$T_{up} = \frac{S}{d_{tu}} \tag{1}$$

$$T_{exe} = \frac{T_{length}}{Po} \tag{2}$$

$$T_{down} = \frac{R_s}{d_{td}} \tag{3}$$

Where d_{tu} is the uplink data rate, R_s represents result size and d_{tf} is the downlink data rate.

The makespan ₸ represents the maximum completion time across all tasks as follows:

$$T_{total(i,j)} = T_{up(i,j)} + T_{exe(i,j)} + T_{down(i,j)} \tag{4}$$

$$₸ = \max\left(T_{total(i,j)}\right), \tau_i \in T, f_j \in F \tag{5}$$

Table 1. Comparative analysis of different task offloading algorithms

Approach	Task Scheduling	Mechanism	Environment	Primary Objectives				
				Latency	Makespan	Cost	QoS	Utilization
MTFP [18]	Traditional /Deterministic	Matrix-based	Fog	√	×	√	√	×
RAPTS [19]	Traditional /Deterministic	Priority queuing	Fog	×	×	×	√	√
MOFA [20]	Learning-Based	Clustering	Fog-Cloud	√	×	√	√	√
RADISH [21]	Learning-Based	Bayesian optimization	Cloud	√	√	×	√	×
Fuzzy-GEC [22]	Hybrid (Learning& Meta- Heuristic)	Fuzzy logic, Genetic	Cloud	×	×	×	√	√
OSCAR [23]	Learning-Based	Machine learning	Cloud	√	×	√	√	√
FuzzyPSG [24]	Heuristic	Fuzzy logic, Priority-aware	Fog	√	×	√	×	×
TRAP [25]	Heuristic	Batch processing, Priority ranking	Fog	√	×	√	×	×
DLJSF [26]	Heuristic	Data locality-aware	Fog-Cloud	×	√	√	×	×
MCT-SMO [27]	Meta- Heuristic	Spider monkey optimization	Fog	√	×	√	×	×
LAMOM [28]	Meta- Heuristic	Multi-objective ranking	Fog	√	√	×	√	×
MPSO [29]	Meta- Heuristic	Modified PSO	Fog	√	×	×	√	√
CODA [30]	Meta- Heuristic	Chaotic opposition-based	Fog	×	√	√	√	×
EH-SMO (Proposed)	Meta- Heuristic	SMO & PSO	Fog-Cloud	√	√	√	√	√

3.3 Energy Consumption

The energy requirements of the system are broken down into three key components. Energy spent during task upload is derived from (6), while the processing energy needed at the fog node is calculated using (7). Finally, (8) determines energy for downloading results.

$$E_{up} = T_{up} \times P_{trans} \tag{6}$$

$$E_{exe} = T_{exe} \times P_{cpu} \tag{7}$$

$$E_{down} = T_{down} \times P_{recive} \tag{8}$$

Where P_{trans} is transmission power consumption. P_{cpu} is the power consumed by CPU during execution, P_{recive} is power consumed during result delivery.

3.4 Total Cost

The total cost at fog nodes comprises execution and transmission costs, calculated by (9) and (10) respectively.

$$C_{exe} = T_{exe} \times C_f \tag{9}$$

$$C_{trans} = T_{up} \times C_t \tag{10}$$

Where C_f represents cost usage and C_t represents the transmission cost per unit time for the fog node.

3.5 Problem Formulation

Let $T = \{\tau_1, \tau_2, \dots, \tau_m\}$ represent the set of tasks to be processed by fog nodes $F = \{f_1, f_2, \dots, f_n\}$. The objective is to find optimal task-to-node mapping that minimizes: makespan, energy consumption and cost constraints.

$$E_{total} = E_{up} + E_{exe} + E_{down} \tag{11}$$

$$C_{total} = C_{exe} + C_{trans} \tag{12}$$

The multi-objective optimization problem is formulated as:

$$\min \text{Ш} = \text{Ŧ} + E_{total} + C_{total} \tag{13}$$

$$F(n) = \frac{1}{\text{Ш}} \tag{14}$$

Table 2. Notations used in EH-SMO algorithm

Symbol	Definition
F	Set of fog nodes
P_o	Processing power of fog node
T	Set of tasks
S	Task size (MB)
T_{length}	Task processing Length (MI)
$\mathcal{T}$	Makespan (Maximum completion time)
T_{total}	Total completion time
C_f	Cost per unit time for fog node
C_t	Transmission cost per unit time
SM_i	Spider monkey i position vector
V_i	Velocity vector
c_1, c_2	Local acceleration coefficients
c_3,c_4	Global acceleration coefficients
P_{best_i}	Personal best position
g_{best_i}	Global best position
$Fit_{best_{i,j}}$	Selection fitness for task i and fog node j
$d_{i,j}$	Normalized distance between task i and fog node j
$V_{i,j}$	Normalized velocity component
α, β, γ	Weight factors for selection fitness

4 EH-SMO ALGORITHM

Resource management in IoT is a challenging generalized problem that requires meta-heuristic algorithms. This paper proposes an Enhanced Hybrid Spider Monkey Optimization (EH-SMO) algorithm for task scheduling in the fog computing environment. The algorithm enhances traditional SMO by integrating the PSO local search mechanism and modifying the standard position update equations. The algorithm employs a hierarchical structure where SMO represents potential task-to-node mappings. Each SM's position is evaluated based on the weighted cost function Ш which considers time, energy and cost metrics. In addition, the algorithm introduces a PSO velocity component to enhance local search capability during position updates. The EH-SMO's flow chart is shown in Figure 3.

For population structure, as illustrated in Algorithm 1, each SM_i is defined as a position vector $\{x_{i1}, x_{i2}, \dots, x_{in}\}$ where x_{ij} represents the fog node assigned to task τ_j, and a velocity vector $V_i = \{v_{i1}, v_{i2}, \dots, v_{in}\}$ for PSO-based movement. The population is organized into k groups $G = \{g_1, g_2, \dots, g_k\}$, where each group maintains a Local Leader (LL) representing the best solution within the group and tracks improvement through a Local Leader Count (LLC) bounded by Local Leader Limit (LLL). At the global level, the best solution among all groups is maintained as the Global Leader (GL), with updates monitored through Global Leader Count (GLC) and limited by Global Leader Limit (GLL). This hierarchical organization facilitates both local exploitation through group-based search and global exploration through leader-guided movement, while maintaining adaptation through leader update mechanisms.

4.1 Position Update Rules

The position updates in EH-SMO combine SMO's swarm intelligence with PSO's velocity mechanism through two main phases. In the Local Leader Phase, each SM updates its velocity and position considering both local group information and personal experience as following:

$$V_{i(t+1)} = w \times V_{i(t)} + c_1 r_1 \left(LLk - X_{i(t)}\right) + c_2 r_2 \left(p_{best_i} - X_{i(t)}\right) \quad (15)$$

$$X_{i(t+1)} = X_{i(t)} + V_{i(t+1)} \quad (16)$$

Where w is inertia weight, c_1, c_2 are acceleration coefficients, r_1, r_2 are random numbers in [0,1], LLk represents the local leader position of group k, and p_{besti} is the personal best position.

In the Global Leader Phase, the position update incorporates global best information as in (17).

This dual-phase update mechanism ensures both local exploitation through group-based movement and global exploration through leader-guided search.

$$V_{i(t+1)} = w \times V_{i(t)} + c_3 r_3 \left(GL - X_{i(t)}\right) + c_4 r_4 \left(g_{best_i} - X_{i(t)}\right) \quad (17)$$

$$X_{i(t+1)} = X_{i(t)} + V_{i(t+1)} \quad (18)$$

Where GL represents global leader position, g_{best_i} is global best position, c_3, c_4 are acceleration coefficients, and r_3, r_4 are random numbers in [0,1].

4.2 Selection Operator

The selection operator in EH-SMO identifies optimal task-fog node pairs by incorporating both PSO velocity and distance metrics. For a task τ_i and fog node f_j, the selection fitness is calculated based on a new hybrid measure that combines velocity-based search efficiency with distance-based resource matching:

$$fit_{best_{i,j}} = \frac{\alpha}{d_{(i,j)}} + \beta \times F(n) + \gamma \times V_{(i,j)} \quad (19)$$

Where $d_{i,j}$ is the normalized distance between task ti and fog node f_j, F_n is the fitness value from equation (14), $V_{i,j}$ is the normalized velocity component from PSO, α, β, and γ are weight factors ($\alpha + \beta + \gamma = 1$).

4.3 Learning Phases

The EH-SMO algorithm implements a dual learning mechanism through Local Leader Learning Phase and Global Leader Learning Phase. In Local Leader Learning, if $F_{LL_{new}} > F_{LL_{current}}$, the Local Leader position is updated; otherwise, LLC is incremented and exceeding LLL triggers group reorganization.

Similarly, in Global Leader Learning, if $F_{GL_{new}} > F_{GL_{current}}$, the Global Leader position is updated; otherwise, GLC is incremented and exceeding GLL initiates group redistribution based on the Maximum number of Groups (MG), thus maintaining balanced exploration and exploitation.

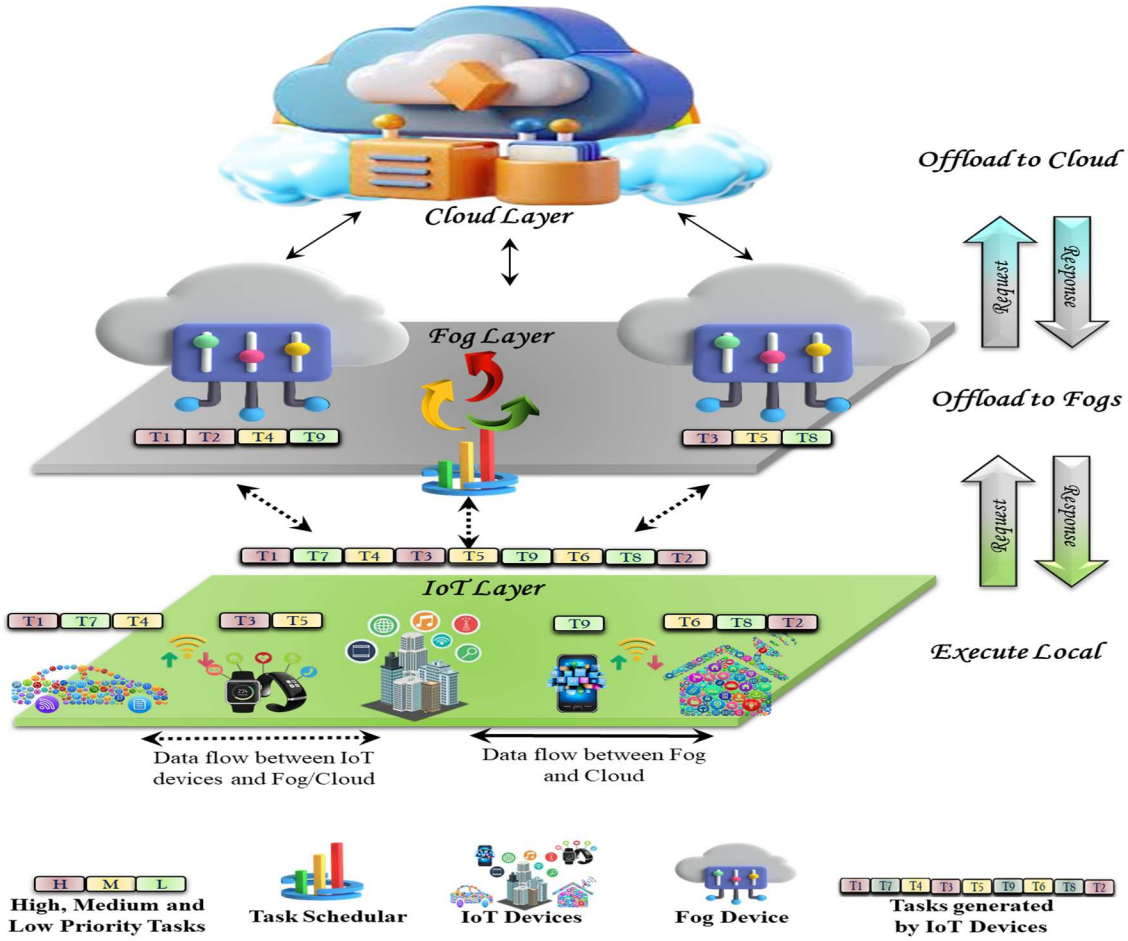


Figure 2: System model of the proposed EH-SMO algorithm

5 SIMULATION RESULTS

5.1 Simulation Setup and Parameters

The proposed EH-SMO algorithm is implemented using iFogSim simulator environment [31]. In default settings, task loads vary from 250 to 2000 in a fog computing environment. The simulation experiments are performed on a Windows 11 computer with an Intel i7 CPU 3.2 GHz and 16GB RAM. Table 3 provides a detailed list of the simulation parameters. The experiments analyze performance in terms of makespan, energy consumption, resource utilization, and convergence metrics. The EH-SMO compared with two baseline algorithms:

i. **MCT-SMO** [27]: A heuristic initialization-based SMO algorithm that enhances traditional SMO through minimum completion time heuristic initialization, specifically targeting cost and service time optimization in fog computing environments.

ii. **MPSO** [29]: Modified PSO algorithm employing a dual-phase optimization for module placement and task allocation, optimizing latency and energy consumption through sophisticated particle update mechanisms.

Table 3. Simulation parameters

Parameter	Value
Number of fog nodes	40-120
Task data size	0.1-5 MB
Task length	100-1000 Megacycles
Communication bandwidth	100 Mbps
Simulation area	5km x 5km urban
Simulation time	1000 sec

5.2 Results and Analysis

5.2.1 Average Makespan

The makespan results, calculated by (5), demonstrate EH-SMO's effectiveness in minimizing task completion times through its hybrid optimization approach. Figure 4 illustrates the average makespan for various computational tasks across the three algorithms. In Figure 4(a), with 40 fog nodes, EH-SMO achieves 17% lower average makespan than MPSO and 32% over MCT-SMO. The superior performance of EH-SMO can be attributed to its hybrid architecture that combines global exploration through spider monkey behavior with PSO's efficient local search capabilities.

In Figure 4(b), with 80 fog nodes, EH-SMO's intelligent position update mechanism enables it to

efficiently navigate the larger solution space, resulting in an average makespan of 22% and 37% lower than MPSO and MCT-SMO, respectively. In Figure 4(c), with 120 fog nodes, the gap widens especially agains the MCT-SMO by about 50% due to it's initialization overhead. This demonstrates EH-SMO's ability to maintain efficiency through superior load distribution and adaptive parameter control even at high node counts. The consistent performance of EH-SMO highlights how its hybrid optimization approach effectively manages complex search spaces while baseline algorithms show significant degradation at scale.

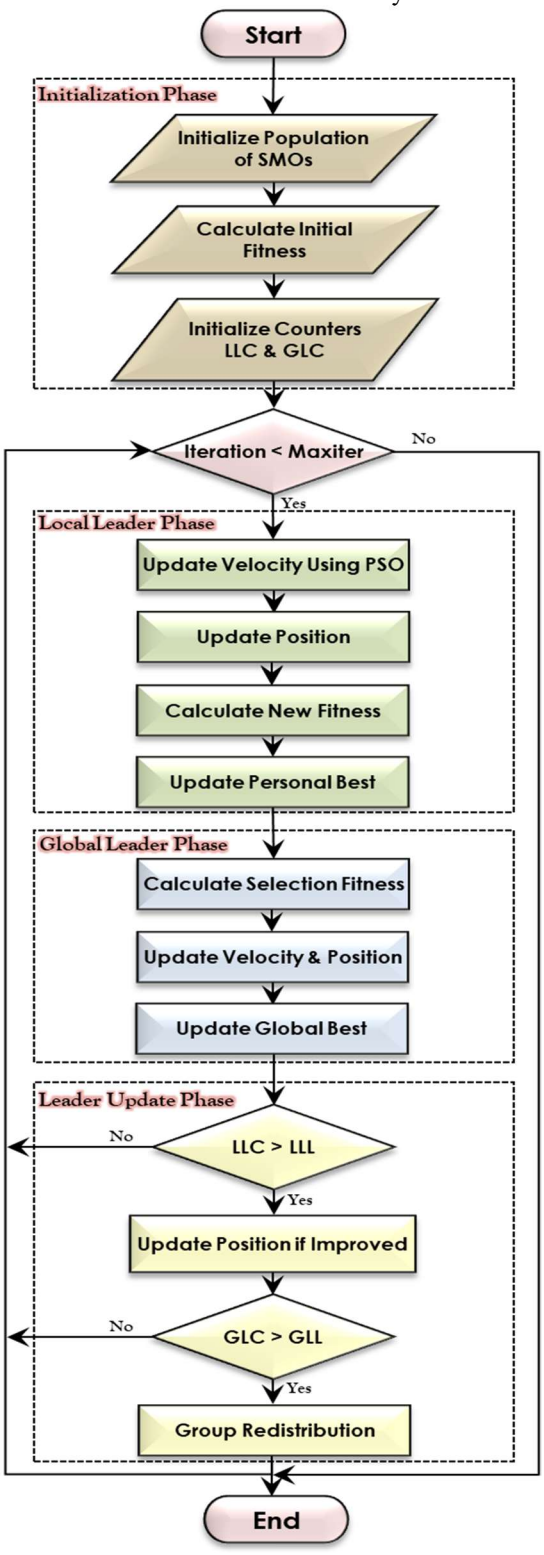


Figure 3: EH-SMO's flow chart

Algorithm 1: EH-SMO algorithm

Input: Tasks $T=\{t_1, t_2, ..., t_n\}$, Fog nodes $F=\{f_1, f_2, ..., f_m\}$
Output: Optimal task scheduling solution
Initialize population of SMs with random pos. & vel.
Calculate initial fitness using (14)
Select initial LL and GL
Initialize LLC, GLC, LLL, GLL
while (iteration < MaxIter) **do**
  for each group k **do**
    for each SMi in group k **do**
      Calculate new velocity using (16)
      Update position using (17)
      Calculate new fitness F(n)
      Update personal best if improved
    end for
    for each SMi in population **do**
      Calculate selection fitness using (15)
      Calculate new velocity using (18)
      Update position using (19)
      Update personal, global best if improved
    end for
    if F(LLnew) > F(LLcurrent) **then**
      Update Local Leader
    else
      LLC++
    end if
    if F(GLnew) > F(GLcurrent) then
      Update Global Leader
    else
      GLC++
    end if
    if LLC > LLL **then**
      for each SMi in group k **do**
        $P = r_1(GL - SMi) + r_2(SMi - LLk)$
        Update position if improved
      end for
      LLC = 0
    end if
    if GLC > GLL **then**
      if number_of_groups < MG then
        Divide largest group into two
        Select new LL for each group
      else
        Combine two closest groups
        Select best LL
      end if
      GLC = 0
      LLC = 0
    end if
  end for
end while
return best solution found

5.2.2 Energy Consumption

Energy consumption evaluation, calculated by (11), reveals distinct patterns in resource allocation efficiency. In Figure 5(a), with 40 fog nodes, algorithms demonstrate linear growth due to sufficient resource availability. EH-SMO achieves through efficient resource management, an average energy consumption of 32% and 46% lower than MCT-SMO and MPSO, respectively. In Figure 5(b), the 80 fog nodes scenario reveals emerging system stress patterns. EH-SMO achieves exceptional efficiency through its adaptive parameter control, maintains controlled energy consumption. MCT-SMO shows moderate performance degradation averaging 37% higher as its initialization-based approach struggles to optimize larger task sets. While MPSO demonstrates significant efficiency loss averaging 53% higher as its local search limitations compound with increased problem complexity. In Figure 5(c), with 120 fog nodes, dramatic exponential patterns emerge, highlights the algorithms' scalability characteristics under higher system stress. EH-SMO maintains remarkable efficiency through its hybrid optimization approach, with relatively controlled growth. MCT-SMO exhibits severe exponential increase, revealing fundamental limitations in its initialization strategy at scale, while MPSO exhibits the worst scalability, demonstrating how its local search limitations become more catastrophic in complex scenarios. The stark differences in curve patterns highlight EH-SMO superior ability to maintain controlled energy consumption even as system complexity increases significantly.

5.2.3 Resource Utilization

Resource Utilization (RU), represets the ratio of used resources to total available resources during the makespan. It is calculated by (20).

$$RU = \left(\frac{\sum_{i \in T} \left(T_{length_i} / P_{o_j} \right)}{\sum_{f \in F} \left(P_{o_f} \cdot \mathrm{T} \right)} \right) \times 100 \qquad (20)$$

Figure 6 reveals critical differences in resource management capabilities across algorithms.

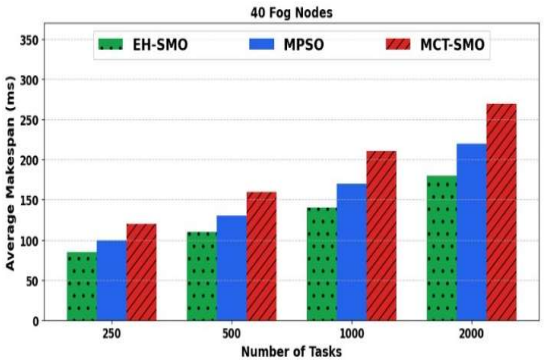
In Figure 6(a), with 40 fog nodes, EH-SMO maintains peak utilization through its effective resource allocation strategy and dynamic workload balancing. Its hybrid architecture enables 17.0% better utilization than MPSO and 37.9% over MCT-SMO.With 80 fog nodes in Figure 6(b), exponential patterns emerge. EH-SMO maintains robust scaling via adaptive parameter control and efficient search space exploration. MPSO suffers from premature convergence in the enlarged solution space, leading to resource underutilization about 24.8% lower. MCT-SMO's performance deteriorates due to increased initialization overhead and lack of dynamic load balancing. Figure 6(c), showing 120 fog nodes, clearly demonstrates stepwise behavior. EH-SMO scales effectively through its sophisticated task distribution mechanism, significantly outperforming both MPSO and MCT-SMO by about 28.8% and 45.7%, respectively. These issues highlight inefficiencies in MPSO's reliance on localized search strategies and MCT-SMO's static initialization, which result in suboptimal resource utilization. In contrast, EH-SMO leverages a hybrid approach that dynamically adjusts task allocation based on real-time resource availability and workload patterns, enabling it to maintain robust performance under demanding conditions. These patterns quantitatively demonstrate how baseline limitations become more pronounced at scale, while EH-SMO's hybrid approach enables sustained performance.

5.2.4 Adaptive Resilience

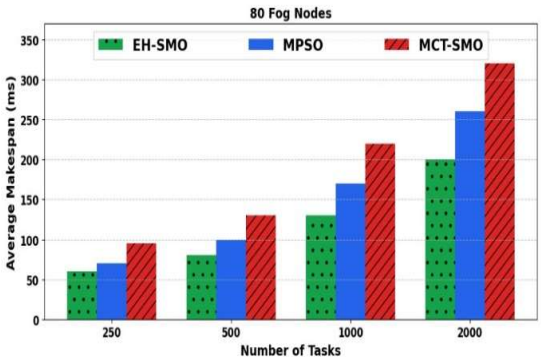
A comparative analysis of three algorithms is evaluated under light, medium and high workload conditions to highlight the adaptability of EH-SMO in handling dynamic fog computing environments. As shown in Table 2, EH-SMO can dynamically adapts to fog computing environments by continuously monitoring network parameters and workload variations. The results demonstrate that EH-SMO significantly outperforms MCT-SMO and MPSO in terms of makespan, energy consumption, and resource utilization across all workload scenarios.

Table 4. Consolidated Performance Metrics by Load Type

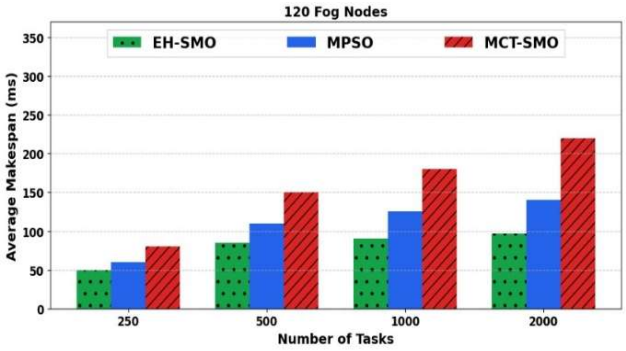
Metric &Load type		EH-SMO	MCT-SMO	MPSO	Over MCT-SMO	Over MPSO
Makespan (*ms*)	**Light**	78.3	122.5	95.0	36.1%	17.6%
	Medium	120.0	203.3	155.0	41.0%	22.6%
	Heavy	149.2	253.3	193.7	41.1%	23.0%
Energy (*KJ*)	**Light**	178.3	198.2	211.7	10.0%	15.8%
	Medium	228.0	277.0	347.3	17.7%	34.4%
	Heavy	303.7	416.2	553.2	27.0%	45.1%
Resource Utilization (%)	**Light**	97.9	58.4	76.1	40.3%	22.3%
	Medium	92.2	52.7	70.8	42.8%	23.2%
	Heavy	86.5	46.7	65.7	46.0%	24.0%



(a) Under 40 fog nodes

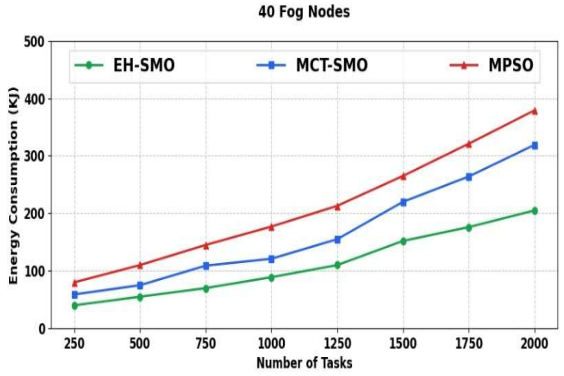


(b) Under 80 fog nodes

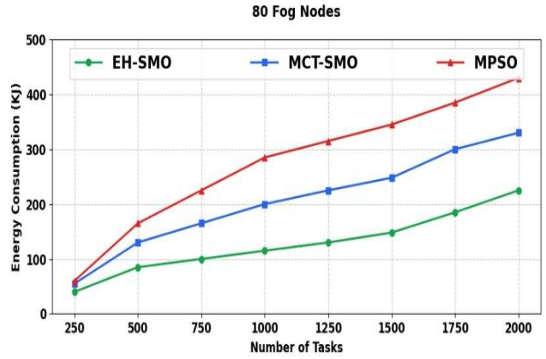


(c) Under 120 fog nodes

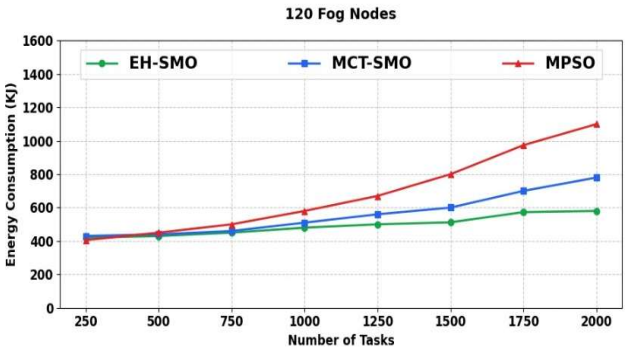
Figure 4: Average task makespan under a varied number of tasks



(a) Under 40 fog nodes

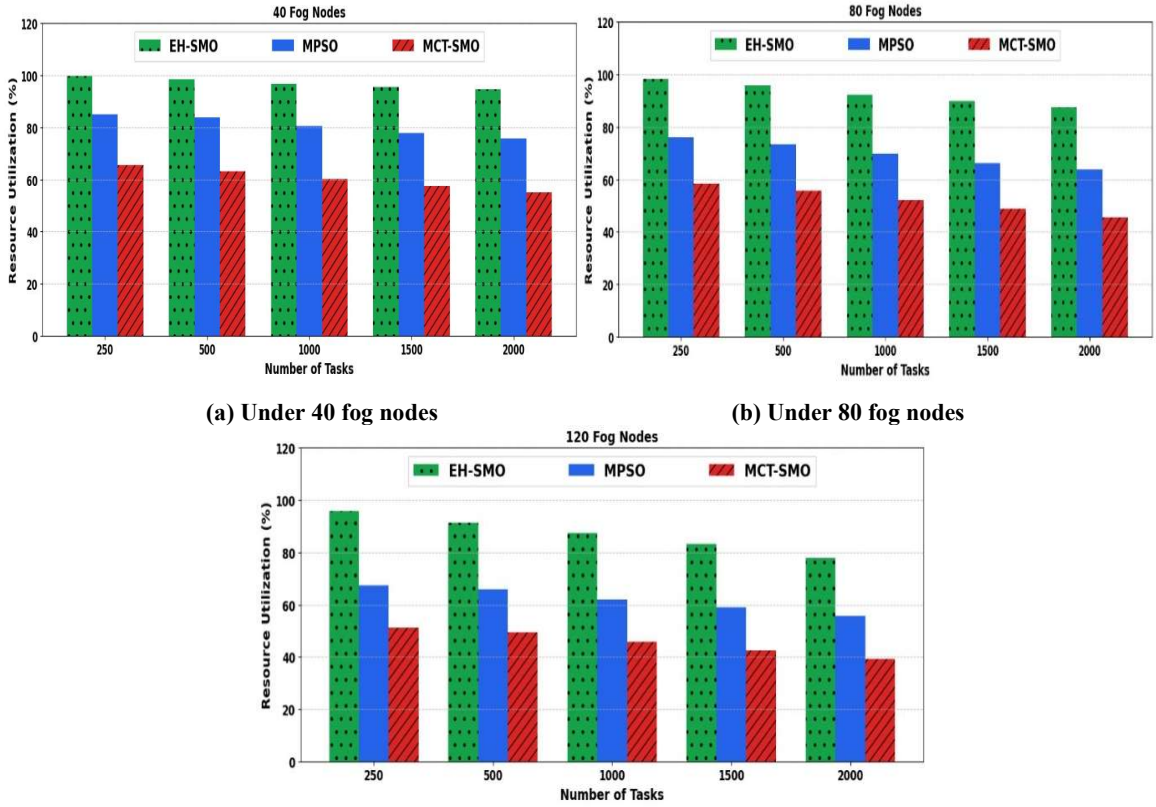


(b) Under 80 fog nodes



(c) Under 120 fog nodes

Figure 5: Energy consumption under a varied number of tasks



(a) Under 40 fog nodes

(b) Under 80 fog nodes

(c) Under 120 fog nodes

Figure 6: Resource utilization under a varied number of tasks

6 CONCLUSION

This paper has presented EH-SMO, a novel hybrid optimization algorithm for task scheduling in fog computing environments that integrates Spider Monkey Optimization with Particle Swarm Optimization through an innovative dual-phase learning architecture. Simulation results confirm EH-SMO's superior performance, reducing makespan by up to 17% and 32% and improving resource utilization by about 18% and 37%, maintaining the best efficiency even under heavy computational loads compared to MPSO and MCT-SMO, respectively. Also, EH-SMO demonstrates remarkable 45% and 33% lower energy efficiency compared to baseline algorithms, keeping consumption controlled even as system complexity increases significantly. These substantial improvements, particularly in high-density scenarios and complex task distributions, validate the effectiveness of EH-SMO hybrid approach in addressing the fundamental challenges of fog computing resource management. Future research directions could explore real-time parameter adaptation mechanisms, multi-objective optimization under uncertainty, and integration with emerging IoT architectures to further advance fog computing capabilities.

7 REFERENCES

[1] R. O. Aburukba, M. AliKarrar, T. Landolsi, K. El-Fakih, "Scheduling Internet of Things requests to minimize latency in hybrid fog-cloud computing," Future Generation Computer Systems, vol. 111, pp. 539-551, 2020.

[2] R. Rani, N. Kumar, M. Khurana, A. Kumar, A. Barnawi, "Storage as a service in fog computing: a systematic review," Journal of Systems Architecture, vol. 116, pp. 102033, 2021.

[3] P. Datta, B. Sharma, "A survey on IoT architectures, protocols, security and smart city based applications," Proc. of the 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1-5, 2017.

[4] Y. L. Jiang, Y. S. Chen, S. W. Yang, C. H. Wu, "Energy-efficient task offloading for time-sensitive applications in fog computing," IEEE Systems Journal, vol. 13, no. 3, pp. 2930-2941, 2019.

[5] M. De Donno, K. Tange, N. Dragoni, "Foundations and evolution of modern computing paradigms: Cloud, IoT, edge, and

fog," IEEE Access, vol. 7, pp. 150936-150948, 2019.
[6] M. Kaur, R. Aron, "An energy-efficient load balancing approach for scientific workflows in fog computing," Wireless Personal Communications, vol. x, pp. 1-25, 2022.
[7] K. Ostrowski, K. Małecki, P. Dziurzański, and A.K. Singh, "Mobility-aware fog computing in dynamic networks with mobile nodes: A survey," Journal of Network and Computer Applications, p.103724, 2023.
[8] R. O. Aburukba, M. AliKarrar, T. Landolsi, K. El-Fakih, "Scheduling Internet of Things requests to minimize latency in hybrid fog-cloud computing," Future Generation Computer Systems, vol. 111, pp. 539-551, 2020.
[9] H. Wadhwa, R. Aron, "TRAM: technique for resource allocation and management in fog computing environment," Journal of Supercomputing, vol. 78, no. 1, pp. 667-690, 2022.
[10] H. Nashaat, W. Hashem, R. Rizk, R. Attia, "DRL-Based Distributed Task Offloading Framework in Edge-Cloud Environment," IEEE Access, vol. 12, pp. 33580 - 33594, Mar. 2024.
[11] M. M. S. Maswood, M. R. Rahman, A. G. Alharbi, D. Medhi, "A novel strategy to achieve bandwidth cost reduction and load balancing in a cooperative three-layer fog-cloud computing environment," IEEE Access, vol. 8, pp. 113737-113750, 2020.
[12] E. Hosseini, M. Nickray, S. Ghanbari, "Optimized task scheduling for cost-latency trade-off in mobile fog computing using fuzzy analytical hierarchy process," Computer Networks, vol. 206, pp. 108752, 2022.
[13] J. Baek, G. Kaddoum, "Online partial offloading and task scheduling in SDN-fog networks with deep recurrent reinforcement learning," IEEE Internet of Things Journal, vol. 9, pp. 11578-11589, 2022.
[14] H. Wadhwa, R. Aron, "Optimized task scheduling and preemption for distributed resource management in fog assisted IoT environment," Journal of Supercomputing, 2022.
[15] M. Yang, H. Ma, S. Wei, Y. Zeng, Y. Chen, Y. Hu, "A multi-objective task scheduling method for fog computing in cyber-physical-social services," IEEE Access, vol. 8, pp. 65085-65095, 2020.
[16] Z. Movahedi, B. Defude, and A. M. Hosseininia, "An efficient population based multi-objective task scheduling approach in fog computing systems," J. Cloud Comput., vol. 10, no. 1, pp. 1–31, Dec. 2021.
[17] P. Hosseinioun, M. Kheirabadi, S. R. K. Tabbakh, R. Ghaemi, "Task scheduling approaches in fog computing: A survey," Transactions on Emerging Telecommunications Technologies, vol. 33, no. 3, 2022.
[18] N. Kaur, A. Mittal, "MTFP: matrix-based task-fog pairing method for task scheduling in fog computing," International Journal of Information Technology, pp. 1-14, 2024.
[19] M. Hussain, S. Nabi, M. Hussain, "RAPTS: resource aware prioritized task scheduling technique in heterogeneous fog computing environment," Cluster Computing, vol. x, pp. 1-25, 2024.
[20] M. Mokni, S. Yassa, J. E. Hajlaoui, M. N. Omri, R. Chelouah, "Multi-objective fuzzy approach to scheduling and offloading workflow tasks in Fog-Cloud computing," Simulation Modelling Practice and Theory, vol. 123, pp. 102687, 2023.
[21] A. B. Kanbar, K. Faraj, "Region aware dynamic task scheduling and resource virtualization for load balancing in IoT-fog multi-cloud environment," Future Generation Computer Systems, vol. 137, pp. 70-86, 2022.
[22] K. Lalitha Devi, K. D. Thilak, C. Shanmuganathan, K. Kalaiselvi, "Fuzzy-GEC an energy-aware hybrid task scheduling on the cloud," Proc. of Advances in Data-Driven Computing and Intelligent Systems, pp. 443-458, 2024.
[23] H. Wadhwa, R. Aron, "Optimized task scheduling and preemption for distributed resource management in fog-assisted IoT environment," Journal of Supercomputing, vol. 79, no. 2, pp. 2212-2250, 2023.
[24] P. Shukla, S. Pandey, P. Hatwar, A. Pant, "FAT-ETO: Fuzzy-AHP-TOPSIS-based efficient task offloading algorithm for scientific workflows in heterogeneous fog-cloud environment," Sadhana, vol. 48, no. 1, pp. 80, 2023.
[25] N. Kaur, A. Kumar, R. Kumar, "TRAP: task-resource adaptive pairing for efficient scheduling in fog computing," Cluster Computing, vol. 25, no. 6, pp. 4257-4273, 2022.
[26] E. Khezri, R. O. Yahya, H. Hassanzadeh, M. Mohaidat, S. Ahmadi, M. Trik, "DLJSF: Data-locality aware job scheduling IoT tasks in fog-cloud computing environments," Results in Engineering, vol. 21, pp. 101780, 2024.
[27] S. S. Hajam and S. A. Sofi, "Spider monkey optimization-based resource allocation and scheduling in fog computing environment", High-Confidence Computing, vol. 3, no. 3, pp. 100149, 2023.
[28] L. Altin, H. R. Topcuoglu, F. S. Gürgen, "Latency-aware multi-objective fog scheduling: addressing real-time constraints in distributed

environments," IEEE Access, vol. 12, pp. 58459-58471, 2024.
[29] T. Hameed, B. Jamil, H. Ijaz, "Efficient resource scheduling in fog: a multi-objective optimization approach," Proc. of the Pakistan Academy of Sciences: A. Physical and Computational Sciences, vol. 61, no. 1, pp. 19-31, 2024.
[30] R. Ghafari, N. Mansouri, "CODA: chaotic opposition-based differential evolution algorithm for task scheduling in fog computing," Journal of Computational Science, vol. 74, pp. 102152, 2023.
[31] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," Software: Practice and Experience, vol. 47, no. 9, pp. 1275-1296, 2017.