

An Efficient Resource Allocation Algorithm with Load Balancing Mechanism to Enhance QoS Parameters in Fog Environment

Mohammad Aknan*

Dept. of Computer Science and Engineering, National Institute of Technology, Patna, India

E-mail: mohammada.phd19.cs@nitp.ac.in

ORCID iD: <https://orcid.org/0000-0003-1608-5574>

*Corresponding author

Maheshwari Prasad Singh

Dept. of Computer Science and Engineering, National Institute of Technology, Patna, India

E-mail: mps@nitp.ac.in

ORCID iD: <https://orcid.org/0000-0002-4121-4959>

Rajeev Arya

Dept. of Electronics and Communication Engineering, National Institute of Technology, Patna, India

E-mail: rajeev.arya@nitp.ac.in

ORCID iD: <https://orcid.org/0000-0002-0346-2150>

Received: March 11, 2026; Revised: May 22, 2026; Accepted: July 22, 2026; Published: October 8, 2026

Abstract: With the proliferation of Internet of Things (IoT) applications, a massive amount of data has been produced, requiring an efficient platform to store and process this data. Cloud computing has the ability to tackle such enormous data, but cannot provide real-time response to latency sensitive IoT applications. Fog computing delivers the cloud service at network edge with rapid response to IoT applications, but the computation offloading decision, unpredictable demands, unbalanced workload among fog servers and heterogeneity become challenging issues. Hence, this study applied an effective approach named Spider Monkey Optimization (SMO) to address the mentioned challenges and employs the proposed framework to ensure the Quality of Service (QoS) parameters performance. This framework uses an adaptive approach to allocate the workload based upon the computation ability of the resource, and continuously monitoring the workload among the fog nodes avoids the possibility of overloading and underloading the fog nodes. Exploration and exploitation ability of the SMO algorithm reduces the chances of being trapped in the local optimum and decides the optimal offloading destination. The performance of the proposed approach is assessed in a simulation environment, showing that the proposed algorithm reduces parameters such as latency, communication overhead, cost and energy consumption compared to baseline approaches Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Whale Optimization Algorithm (WOA) using the same experimental environments.

Index Terms: Fog Computing, IoT, Load Balancing, Latency, SMO, Resource Allocation.

1. Introduction

The role of IoT applications has increased in various areas such as industry, academia, defence, and many more to make the environment smart so that people can use the real time applications like autonomous vehicles, smart transportation system, smart agriculture, smart healthcare, smart home, video surveillance system, etc. [1]. The massive amount of data is produced by these IoT devices and is difficult to handle due to the resource-constrained nature of IoT devices [2]. To deal with such an amount of exponential data is not easy and requires the computing paradigm. Cloud computing provides a prominent solution by offering infinite resources for storage, networks as well as processing of data through virtualization technology on a pay-as-you-go basis [3]. On demand resource provisioning, elasticity, pool of resources, scalability and service monitoring are major characteristics of cloud computing due to which it has been adopted by industries and academia. Multi-tenancy is an additional feature in the cloud paradigm where multiple users

can get the services over a single physical infrastructure through heterogeneous platform. The cloud paradigm offers Everything as a Service (XaaS) to IoT applications anywhere and anytime. It resolves the issues of resource-constrained nature of IoT devices by offering the services in above mentioned way, but latency is a major issue with cloud services and they are not appropriate for latency based real time IoT applications.

The fog paradigm has been introduced to extend the cloud services with limited computational resources for IoT applications at edge of network in distributed manner [4]. It provides support to the latency based real-time IoT applications connected by billions of devices and addresses the problem of connectivity, delay, reliability and capacity. The fog computing architecture is connected via either an IoT gateway or wireless network with IoT devices and core wired network with cloud computing. It reduces the data exchange between IoT devices and the cloud network that leads to improved performance by optimizing the bandwidth as well as latency. However, it is not an easy task to offer the services to IoT network by limited fog nodes without compromising the performance. Hence, there is a need for the services of cloud and fog paradigms in a collaborative manner where latency based applications are computed by fog devices and high computation oriented applications are executed over cloud platform. Fluctuating user demand, offloading decision, balancing workload among the fog nodes, heterogeneity of resources are major issues in collaborative cloud fog environment. Load balancing along with offloading and service placement becomes a noticeable issue due to limited resources of fog computing and existing algorithms focus only on resource allocation and are unaware of the overloading and underloading of fog nodes [5-6]. Some algorithms care about the load balancing but use a reactive approach that is not an apposite solution for real-time applications.

Numerous approaches based upon heuristic, metaheuristic and artificial intelligence have been proposed by the researchers to find the solution of mentioned issues, but most of the existing techniques resolve the issues partially and none of the approaches provide a prominent solution. The task offloading problem comes under the class of NP-Complete problems and it is difficult to find the optimal solution in polynomial time. Initially, researchers have applied traditional heuristic approaches to find the solution, but heuristic approaches are problem-dependent and cannot provide optimum solution. AI-based approaches especially reinforcement learning provide better solutions compared with heuristic approaches, but need a large dataset for training, and the complexity of the model is high [7-8]. Meta-heuristic approaches are application-independent and can be applied in various domains to search the optimal solution. These algorithms can find the approximate solution in reasonable amount of time. Researchers have applied swarm intelligence-based approaches such as PSO, Genetic Algorithm (GA), BAT, Grey Wolf Optimization (GWO) and many more approaches for offloading decision and resource allocation to enhance the Quality of Service (QoS) parameters like time, energy, communication overhead, cost and throughput without violating the constraints. The proposed metaheuristic algorithm suffers from being trapped in local optima, slow convergence rate, inefficient exploration and exploitation, which makes the algorithm unable to balance the workload and degrade the performance parameters. An adaptive framework has been proposed by the authors that monitors the fog nodes continuously to balance the workload and an SMO-based approach is integrated with the framework that addresses the mentioned challenges and allocate the load to best possible resource by optimal offloading decision. The proposed approach uses a reactive strategy that allocates the workload based upon the available computing capacity of the resource and minimizes the possibility of overloading and underloading and enhances the QoS parameters performance. The proposed framework not only formulates the resource management problem as a multi-objective optimization model but also addresses core limitations in current technologies, including poor scalability, lack of real-time responsiveness, and inefficiencies in energy consumption. By monitoring fog nodes continuously and adjusting task allocations based on real-time conditions, our framework minimizes overloading and underloading scenarios while optimizing key QoS metrics such as latency, energy, communication overhead, and cost.

The major contribution of the article is as follows:

- To define the mathematical model for offloading decision and optimal resource allocation in collaborative fog cloud environment.
- To design and propose an adaptive framework integrated with SMO algorithm that decides whether tasks will execute locally or offloaded to a fog server considering bandwidth and deadline as constraints.
- The proposed approach selects the optimal server and continuously monitors the resources with the aim to avoid the condition of overloading or underloading and enhance the QoS parameter performance.
- The proposed framework performance is evaluated via conducting extensive simulation experiments, which demonstrate that the SMO algorithm significantly improves latency, communication overhead, energy consumption and cost compared with benchmark baseline approaches.

The rest of the paper is organized as follows: state-of-the-art approaches are explored with their advantages, objectives and limitations in section 2. The mathematical model is cast for QoS performance parameters in section 3. An adaptive framework is proposed and depicted in section 4 along with its components and proposed algorithm. The performance of the proposed work is assessed and validated in section 5. At last, the conclusion and future work are described.

2. Related Work

The main goal of the developed approaches is to decide the offloading destination and find the optimal resource for upcoming traffic to discard the chances of overloading and underloading conditions. The existing framework tried to utilize the resources of fog nodes in an effective manner, and transfer the workload to other fog nodes, if any one is over-utilized. There are some approaches such as fundamental, approximate, exact algorithm and hybrid approaches based upon greedy, dynamic, meta-heuristic, and AI techniques that have been proposed to search the optimal fog server and improve the QoS performance metric [6]. A greedy matching technique has been proposed by the authors for dynamic task offloading in autonomous vehicles to address the concerns of high mobility and complexity of traffic [9]. The proposed approach is applied in fog network to reduce the communication overhead and improve the response time of requests. Heuristic-based online task offloading is depicted by the authors for parallelism task distribution between mobile cloud computing and cloud platform; the approach maintains the load balancing among the mobile nodes [10]. The developed approach explores concurrent tasks concept to test and validate the results improvement in terms of significant QoS parameters. Heuristic and fundamental approaches failed to work in a dynamic environment due to various factors like fluctuating demand, unpredictable workload, heterogeneity, and limited resources that are unable to decide offloading at run time, leading to the problem of over-utilized and underutilized resources.

Artificial Intelligence based approaches have been introduced to counter the mentioned challenges and provide the optimal solution of the problem faced by end users. Machine learning, deep learning and meta-heuristic techniques are the most prominent approaches that come under Artificial Intelligence (AI) techniques. Deep reinforcement learning (DRL) technique has been utilized by the authors for offloading decision with the help of information gained from the environment and applies Salp Swarm Algorithm (SSA) for optimal resource allocation to optimize the latency, cost and energy consumption for IoT applications [11]. The authors have constructed a multi-device collaborative framework for offloading decision using graph neural network by establishing the relationship between mobile devices and mobile edge computing [12]. The proposed work utilizes the concept of reinforcement learning based upon the rewards and penalty to optimize the objective function in terms of latency, cost and energy consumption. The proposed algorithm provides the rewards if the task is offloaded at the optimal destination and executed within the deadline; otherwise, it imposes a penalty if deadline is missed. Workload forecasting is a major issue because it is difficult to predict the upcoming applications. Hence, authors have integrated Long Short Term Memory (LSTM) with DRL to predict the workload for better offloading decision and to train the DRL model for best possible resource selection [13]. The proposed work overcomes the issue of load balancing and service placement, but training the LSTM model as well as the DRL model over a large dataset is a major issue for IoT applications. Few mode variants of DRL have been proposed named Online Dependent Task Offloading, integration of graph attention network with DRL for optimal offloading, load balancing and service placement and resolve the issue up to a level, but major issues with DRL are that it requires a massive amount of training to train both the target network and the main network [14-15].

Meta-heuristic approaches can be applied in various domains of engineering and have the ability to provide an approximate solution to the problem by adaptive offloading decision, optimal resource allocation. The authors have utilized the concept of GA to determine how to offload and where to offload the tasks in vehicular edge computing and improve the energy consumption as well as the overall system performance [16]. The authors have employed Ant Colony Optimization (ACO) approach to solve the offloading issues in resource-constrained environment where ACO uses concentration of pheromone concept, if it is higher than maximum chances of food (optimal resource), otherwise search for the resource to execute the demand [17]. ACO is unable to deal with overloading and underloading issues in fog/edge computing environment. PSO based algorithm has been applied by the authors for offloading and allocation in the Internet of Drone (IoD) to enhance delay as well as processing cost [18]. The proposed strategy uses the concept of particles in swarm intelligence and initializes randomly at the start. After that, it moves towards the fitness function based upon the velocity and position of the particle that changes frequently based upon the upcoming demand and capacity of resources. The authors have applied a bat-inspired algorithm that utilizes food-searching behaviour to find the optimal solution of the problem [19]. Dynamic local search approach has been incorporated to regulate probabilities of selecting the three search methods that improve the desired QoS parameters and overall system performance.

IoT devices are limited battery-driven and resource-constrained in nature; hence, the authors have proposed GWO to minimize the power consumption and offload the computation-oriented tasks to suitable computing paradigm based upon the latency [20]. The proposed GWO approach employs a position-based mapping to get the high-quality solution that improves the influential parameters like energy consumption and delay. An AI-enabled sustainable framework has been developed by the authors for a multi-layered architecture that decides the optimal offloading based upon the upcoming workload [21]. Fuzzy-based approach is integrated to classify the tasks for better offloading decision and WOA is utilized to find a promising solution by allocating the optimum resource and enhancing significant parameters. The existing meta-heuristic achieved the objectives up to a level but several shortcomings in the mentioned approaches give the opportunity to explore a new algorithm that maintains the exploration and exploitation balance that helps to balance the workload and optimize the QoS parameters. Hence, authors have incorporated SMO-based approach in the proposed framework that decides not only the offloading decision but also discover the optimal resource in a collaborative environment for IoT applications.

3. Problem Formulation for IoT Applications

The objective of this work is to search the optimal offloading destination for unpredictable IoT requests generated by IoT devices and optimize the QoS parameters within defined constraints. We have considered a set of n IoT devices that generate the tasks $\{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \dots, \mathcal{L}_n\}$ and assume that each device generates a single task in a defined time period. Offloading decision is defined based upon offloading vector $V = \{v_1, v_2, v_3, \dots, v_n\}$, where v_n represents the offloading decision with three possibilities either executed locally means IoT devices ($v_n=0$) or allocated to fog nodes ($v_n=1$) for the processing and sent to the cloud ($=2$) for the processing. C_n^l, C^f and C^c depict the computation capacity of computing paradigm as in order IoT devices, Fog and cloud data centre.

3.1. Task Model

All the generated tasks $\{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \dots, \mathcal{L}_n\}$ are processed by IoT devices initially, depending upon the computation and storage capacity of devices. If IoT devices are unable to process, then the tasks are offloaded to a fog server or the cloud data centre depending upon the characteristics of the tasks and availability of resources. The authors have made the assumption that all the tasks are non-partition based and context switching is not allowed i.e., partial execution of the tasks is not considered in this work. $Q_i = \{\mathcal{L}_s, \mathcal{R}_n\}$, depicts the task model where task size is denoted by $\mathcal{L}_s$ and no. of CPU cycles required is denoted by $\mathcal{R}_n$. $\mathcal{L}_s$ and $\mathcal{R}_n$ relationship is defined in equation 1.

$$\mathcal{R}_n = \phi * \mathcal{L}_s \quad (1)$$

3.2. Computation Model

Local Processing

IoT devices collect the data that needs to be processed by IoT devices as per the given equation 2. Latency to execute the tasks locally is defined. $\mathcal{L}_n^l$

$$\mathcal{L}_n^l = \frac{\mathcal{R}_n}{C_n^l} \quad (2)$$

Energy consumption (E_{Total}^l) is computed by equation 3 over IoT devices.

$$E_{Total}^l = \mathcal{L}_n^l * \rho_l = \frac{\mathcal{R}_n}{C_n^l} * \rho_l \quad (3)$$

The power consumption over IoT is represented by ρ_l in unit amount of time and C_n^l represents the total cost.

$$C_n^l = \sum_{n=1}^N \alpha * E_n^l + \beta * \mathcal{L}_n^l \quad (4)$$

α and β are two parameters to adjust the weight factors that are between 0 and 1.

3.3. Fog Computing Model

IoT devices are resource-constrained and cannot process the applications that require high computational power and need to offload the upcoming tasks over fog servers with decision unit ($v_n=1$). Time required to compute the IoT tasks over fog server is computed by equation 5.

$$\mathcal{L}_n^f = \frac{\mathcal{R}_n}{C^f} \quad (5)$$

Fog server consumes more energy compared with IoT devices due to being highly resource-oriented; therefore, there is a need to allocate the fog servers in an optimal manner so that energy consumption can be reduced. Power is consumed by various components of fog server like CPU, storage, memory, and transceiver, etc., but maximum power is consumed by CPU [22]. The calculation of energy consumption over fog server is given in equations 6 to 7. Therefore, to calculate the power (energy) consumption of applications, we require data on processing time and CPU utilization. Resource utilization (U_t) is expressed in terms of time as follows:

$$U_t = \sum_{a=1}^b U_{t,a} \quad (6)$$

Here, b represents end-user applications that require $U_{t,a}$ resources for execution at time t [23]. The total energy consumption (E_{Total}) of fog server resources is defined as follows:

$$E_{Total} = (E_{max} - E_{min}) * U_t + E_{min} \quad (7)$$

$E_{\max}$ & $E_{\min}$ denote the maximum and minimum energy consumption required to process the computation demand using resources $\mathcal{R}_j$. Overall offloading cost of fog server to compute the application is given in equation 8.

$$C_n^f = \sum_{n=1}^N \alpha * E_{Total}^f + \beta * L_n^f \quad (8)$$

3.4. Offloading Over Cloud Data Centre

Few tasks require high computational resources that are offloaded over the cloud data centre ($v_n=2$). These types of tasks are not latency sensitive and do not follow the strict deadline, i.e., they can afford the latency during the computation. The authors have calculated the computation delay (L_n^c) and energy consumption (E_{Total}^c) of tasks over cloud data centre considering constant value of propagation delay (t_p). The transmission ($L_{n,t}^c$) and processing delay ($L_{n,p}^c$) are defined by equations 9 and 10.

$$L_{n,t}^c = \frac{T_s}{B_n^w} \quad (9)$$

$$L_{n,p}^c = \frac{R_n}{c^c} \quad (10)$$

The overall time taken by the cloud data centre to process the request is computed by equation 11.

$$L_n^c = L_{n,t}^c + L_{n,p}^c + t_p \quad (11)$$

The consumption of total energy over cloud data centre to execute the IoT tasks is given in equation 12.

$$E_{Total}^c = E_{n,t}^c + E_{n,p}^c \quad (12)$$

Energy consumption during the transmission and propagation time is given in equations 13 and 14.

$$E_{n,t}^c = L_{n,t}^c * \rho_t \quad (13)$$

$$E_{n,p}^c = L_{n,p}^c * \rho_p \quad (14)$$

Total cost to offload and execute the IoT applications over the cloud is given by the equation 15.

$$C_n^c = \sum_{n=1}^N \alpha * E_{Total}^c + \beta * L_n^c \quad (15)$$

Upcoming tasks can be executed locally, on fog nodes or on the cloud data centre depending upon the offloading decision that is taken based upon upcoming demand, size and remaining computation capacity of resources. Hence, overall cost is the summation of all these three and can be calculated by the given equation 16.

$$Q_{all} = \sum_{n=1}^N \frac{(1-v_n)(2-v_n)}{2} C_n^l + v_n(2-v_n) C_n^f + \frac{v_n(v_n-1)}{2} C_n^c \quad (16)$$

3.5. Problem Formulation

The main aim of this work is to minimize the value of that optimizes the performance Q_{all} parameters such as time, energy and overall service cost of tasks. Offloading decision plays a vital role to minimize the value of Q_{all} based upon several factors that leads to allocate the optimal resource to the upcoming tasks, and the problem comes under the category of optimization problems and can be formulated as follows:

$$\text{Min } (Q_{all}) = \min Q_{all} = \sum_{n=1}^N \frac{(1-v_n)(2-v_n)}{2} C_n^l + v_n(2-v_n) C_n^f + \frac{v_n(v_n-1)}{2} C_n^c$$

Subject to constraints: $C_1 \ v_n \in \{0, 1, 2\}, \forall n \in N$

4. SMO-based Proposed Framework for Dynamic Offloading Decision

The proposed framework is divided into three computing paradigm layers: IoT layer, Fog layer and Cloud layer. The multilayer framework ensures the processing of IoT tasks by allocating the resources out of the above-mentioned computing paradigm based upon their requirement. In the IoT layer, several sensors are available to collect data from the environment based on applications like healthcare, transportation, smart home, agriculture, etc. After collecting the data, IoT devices check whether they are capable of executing the tasks/data within the defined time period or not. If yes, then

the request is processed over the IoT device, otherwise offloaded to the fog server or the cloud server depending upon the demand of tasks. IoT devices are connected with the fog server via wireless or wired network, and all IoT devices send the data to the central controller fog server named resource broker, which is responsible for balancing the workload among fog nodes by allocating the resources in an effective manner. Resource broker utilizes the SMO algorithm to distribute the workload in a fair manner based upon the upcoming request and available remaining capacity of resources. Sometimes, it is difficult to find the best fit resource for upcoming tasks because it takes time to check the available capacity of all resources and leads to an increase in the delay that is not affordable for real-time IoT applications.

Hence, authors have categorized the fog servers in the form of three clusters, one cluster handles highly computation-oriented tasks because it has enough resources to handle the demand of tasks. The second cluster is for low computation-oriented tasks, requiring a smaller number of resources, and delay is not affordable. At last, the third cluster is for normal applications that come between high and low computation-oriented resources. It is the responsibility of the resource broker to check the demand of upcoming task and keep the record of every resource, how much capacity is available to process the applications. Resource information and Task scheduler are two modules that are directly linked with resource broker to provide the information continuously either event based or time based. The Resource Information module keeps all the updates about the resources like how many resources are underutilized or overutilized and takes the decision to balance the workload among the resources. This module sends the information after a time interval to resource broker. The task scheduler module utilizes the SMO algorithm to map the upcoming request with available resources in an efficient manner. Task scheduler sends task resource pair information to resource broker so that it can take appropriate decision to optimize the QoS parameters. The third layer is cloud based and offers the services to complex and diverse tasks that require high-end resources and minimum latency can be afforded [24-25].

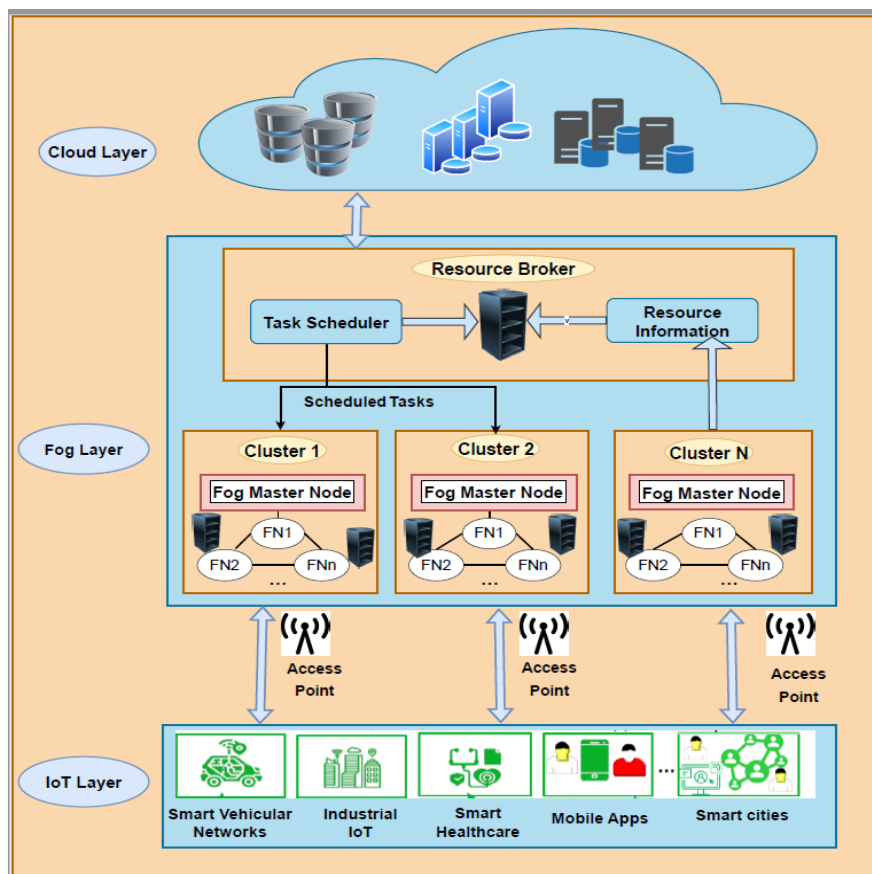


Fig. 1. Proposed SMO-based framework for offloading and resource allocation decision

Proposed SMO-based Scheduling Algorithm

This section illustrates the role of the SMO algorithm and why the authors have employed the SMO algorithm to solve the offloading issues along with discussing the mapping with upcoming tasks and available resources in collaborative fog cloud environment.

Spider Monkey Optimization (SMO): SMO is one of the promising metaheuristic approaches to find the solution of the NP-Complete (NPC) problem in approximation form within a short period of time. Numerous metaheuristic or AI-based approaches have been applied to place the IoT services over optimal fog nodes, but failed to search the optimal solution of problems due to heterogeneity, diverse task range, being trapped in local minima, unbalanced exploration and

exploitation ratio, etc. Therefore, SMO algorithm has been introduced to counter the mentioned issues utilizing the intelligent foraging behaviour. Generally, these monkeys live in a group and try to search for the best food source in different directions by sharing the information [26-29]. The aim of this work is to cut down the cost along with time and energy consumption using SMO where the distance between food source and monkey decides the quality of the solution. If it is near to the monkey, it means that we are near to the optimal solution; otherwise more iterations are required to get the optimal solution. There are 6 stages in SMO algorithm with 4 control parameters described as below:

Initialization: In initialization phase, $SM_1, SM_2, \dots, SM_n$ N monkeys are generated, which are uniformly distributed in the search space. Each monkey is initialized as

$$SM_{ij} = SM_{minj} + R_U(0,1) * (SM_{maxj} - SM_{minj}) \quad (17)$$

Where SM_{minj} and SM_{maxj} are lower and upper bounds of search space, $R_U \in (0,1)$.

- **Local Leader Phase (LLP):** Each monkey attempts to update the position as the experience gained from local group members and local leader. Further, the calculated fitness value of the new position is compared with the old one, if it is better than the previous one, it is updated with the new one as given in equation 18.

$$SM_{New,ij} = SM_{ij} + R_U(0,1) * (LL_{kj} - SM_{ij}) + R_U(-1,1) * (SM_{rj} - SM_{ij}) \quad (18)$$

- **Global Leader Phase (GLP):** Each monkey's position is updated by leveraging the experience of global leader and local group member so that it can explore more space to search the better solution.

$$SM_{New,ij} = SM_{ij} + R_U(0,1) * (GL_j - SM_{ij}) + R_U(-1,1) * (SM_{rj} - SM_{ij}) \quad (19)$$

Where GL_j points to the position of the global leader with j^{th} dimension.

$$P_i = 0.9 * \frac{f_i}{f_{max}} + 0.1 \quad (20)$$

The probability P_i plays a crucial role in updating the monkeys' positions and is determined based on the fitness value, as described in Equation 20. This equation indicates that a monkey with a higher fitness value has a greater chance of improving its position compared to others in the group.

- **Global Leader Learning Phase (GLL):** The algorithm employs a greedy selection approach to identify the optimal solution in this stage. If the position remains unchanged, the GLL count is incremented by 1; otherwise, it is reset to 0.
- **Local Leader Learning phase (LLL):** This algorithm employs a greedy approach to achieve the optimal solution within local group. If there is no improvement in the achieved solution, then LLL count for that group is increased by 1, otherwise its value is reset to 0.
- **Local Leader Decision phase (LLD):** If there is no progress in the LL position after a predefined number of iterations, it can be enhanced either by leveraging random initialization or with the assistance of LL, GL, and P_i as described in Equation 21.

$$SM_{New,ij} = SM_{ij} + R_U(0,1) * (GL_j - SM_{ij}) + R_U(0,1) * (SM_{ij} - LL_{kj}) \quad (21)$$

The above-mentioned equation 21 tries to update the position if $R_U(0,1) \geq P_i$ otherwise equation 22 performs the updating procedure of the monkey position

$$SM_{New,ij} = SM_{ij} + R_U(0,1) * (GL_j - SM_{ij}) + R_U(0,1) * (SM_{rj} - LL_{kj}) \quad (22)$$

- **Global Leader Decision phase (GLD):** The position of GL is continuously monitored. If it remains unchanged for a predefined number of iterations (GLL), GL divides the population into smaller groups to search the solution in different directions.

The concepts of fusion and fission play a crucial role in searching for the optimal solution in an integrated fog-cloud environment. Fission increases the exploration possibility and helps to find out the solution in the whole search space while fusion improves the quality of the solution through exploitation. The proposed algorithm aims to balance exploration and exploitation, enhancing result quality. All phases of the SMO approach work together to achieve an optimal solution in terms of latency, energy consumption, and cost. The authors have implemented an SMO-based approach, as presented in Algorithm 1, to compute results by optimizing the fitness function.

SMO-based algorithm for Offloading and Allocation

Input: $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \dots, \mathcal{L}_n$

Output: Min (Q_{all})

Begin

Create or initialize diverse range of tasks $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \dots, \mathcal{L}_n$

$R_1, R_2, R_3, \dots, R_m, R_{m+1}$ are m resources in fog servers and one data centre for cloud environment

Defined variables: iterations (k), deadline $\partial(\mathcal{L}_i)$, SM_i =ith monkey, dimension (D), Minimum_monkey=10 in a group

For i=1 to N //initialize the spider monkeys

For j=1 to D

Each SM_i with D dimension

$SM_{ij} = SM_{minj} + R_U(0,1) * (SM_{maxj} - SM_{minj})$

$SM_{New,ij} = SM_{ij} + R_U(0,1) * (LL_{kj} - SM_{ij}) + R_U(-1,1) * (SM_{rj} - SM_{ij})$ //Local leader phase

$SM_{New,ij} = SM_{ij} + R_U(0,1) * (GL_j - SM_{ij}) + R_U(-1,1) * (SM_{rj} - SM_{ij})$ //Global leader phase

Start to evaluate the time, cost and energy consumption using problem formulation equations

Min (Q_{all}) = min $Q_{all} = \sum_{n=1}^N \frac{(1-v_n)(2-v_n)}{2} C_n^l + v_n(2-v_n) C_n^f + \frac{v_n(v_n-1)}{2} C_n^c$

End for loop

End of loop

while($K_{max} \neq \emptyset$)

Do

For i=1 to N // for all the monkeys

For j=1 to D

$SM_{New,ij} = SM_{ij} + R_U(0,1) * (LL_{kj} - SM_{ij}) + R_U(-1,1) * (SM_{rj} - SM_{ij})$ //Local leader phase

If $f(\rho_j)_{new} < f(\rho_j)_{old}$

$f(\rho_j) = f(\rho_j)_{new}$

$SM_{New,ij} = SM_{ij} + R_U(0,1) * (GL_j - SM_{ij}) + R_U(-1,1) * (SM_{rj} - SM_{ij})$ //Global leader phase

$P_i = 0.9 * \frac{f(\rho_j)_i}{f(\rho_j)_{max}} + 0.1$

If $f(\rho_j)_{new} < f(\rho_j)_{old}$

$f(\rho_j) = f(\rho_j)_{new}$

To achieve the optimal value of parameters, a greedy selection approach is applied

If $f(\rho_j)$ is not enhanced

Then GLL++ and LLL++;

If $R_U(0,1) \geq P_i$

Then $SM_{New,ij} = SM_{ij} + R_U(0,1) * (GL_j - SM_{ij}) + R_U(0,1) * (SM_{ij} - LL_{kj})$

Else $SM_{New,ij} = SM_{ij} + R_U(0,1) * (GL_j - SM_{ij}) + R_U(0,1) * (SM_{rj} - LL_{kj})$

If $f(\rho_j)$ is not the optimal solution

Then groups of monkeys are further divided //fission

Else Merge all the groups into a single unified group //fusion

Repeat until get Min (Q_{all})

End

5. Performance Evaluation

This section depicts the experimental setup and results discussion to assess the efficacy of the proposed framework. The authors have tested the performance of the proposed work and compared it with other state-of-the-art approaches such as ACO [17], PSO [18] and WOA [21] to validate the results in the same simulation environment. The rationale behind choosing these three algorithms for benchmarking is: (a) it is a de facto standard to compare the performance of a meta-heuristic approach against other meta-heuristics, and (b) these three algorithms have already been used to solve the resource allocation problem in fog server domain. The overall goal of this work is to enhance the influential parameter so that users can get the optimal services at minimum cost by leveraging the SMO approach without affecting the defined constraints.

In fog computing, challenges such as optimal offloading decision-making, unpredictable demand from edge devices, and workload imbalance among heterogeneous fog nodes are critical and complex. These challenges are specifically formulated in our framework as a multi-objective optimization problem involving the minimization of latency, cost, and energy consumption while ensuring efficient utilization of fog resources. Unlike existing methods that often assume static workloads or ignore the temporal variability in task arrivals and fog node availability, our study considers a dynamic and resource-constrained fog environment hosting latency-sensitive IoT applications.

The uniqueness of our fog computing scenario lies in its high heterogeneity and real-time constraints, where

decisions must be decentralized and adaptive to environmental fluctuations. SMO is particularly well-suited to this context due to its dynamic grouping, local–global communication strategy, and exploitation–exploration balancing mechanism. This enables it to dynamically select optimal offloading destinations and prevent the system from converging to local minima, which is a common limitation in PSO, ACO, and WOA.

5.1. Simulation Setup

Conducting experiments in a real-time environment to evaluate the performance of the proposed work is challenging due to high costs and various obstacles that can cause delays in the experimentation process. Hence, the authors have set up the Python-based simulation environment including numbers of fog servers, IoT devices and cloud data centre to assess the proposed framework performance. Efficacy of the work is tested in heterogeneous environment by varying the number of tasks and fog servers to perform the execution of applications and validate the results. The experiments have been conducted on the system having an Intel Core i7 processor, 16GB RAM and a 64-bit processor and also including several python libraries to smoothly execute the tasks. The request service length is defined as [300, 600] Kbits, deadline is 1 to 10 seconds and computation speed of resources varies from 120 to 800 cycles/second [24].

5.2. Results and Discussion

The authors have considered three significant parameters named time, cost and energy consumption to evaluate and analyse the performance in collaborative fog cloud computing paradigm. Multiple test case scenarios have been considered by changing the simulation parameters to assess the efficiency of the proposed framework, which was compared with all baseline benchmark approaches.

A. Processing Time

Average Processing Time parameters play a very significant role in making a strong analysis for the proposed algorithm against the other three state-of-the-art algorithms. Initially, the authors have considered 10 fog nodes, 100 IoT devices and 1 data centre to conduct the experiment and applied the proposed SMO algorithm that computes the 100 tasks generated by IoT devices in approximately 130 seconds. After that, the authors have implemented baseline approaches over the same simulation environment and found that the proposed approach performance was superior compared to others. Further, the number of tasks is increased by step 100 up to 500 and applied the proposed approach as well as benchmark approaches. The achieved computational results proved that the proposed approach reduces the processing time of applications up to 20%, 15%, and 10% approximately compared with ACO, PSO and WOA as shown in Figure 2. In addition, the number of fog servers is increased by step size 10 up to 30 and Figures 2-4 show the average processing time parameter comparative analysis of the proposed algorithm with benchmark baseline algorithm with 10, 20, and 30 fog servers, respectively. The simulation result proves that our proposed algorithm performs significantly better than the other three algorithms in the same simulation environment at defined parameters. The simulation-based results also validate the efficacy of the algorithm because we have tested and verified the performance at several case scenarios. The results obtained through simulation further validate the effectiveness of the algorithm, as its performance has been thoroughly tested and verified across multiple case scenarios. By simulating diverse real-world conditions, we have ensured that the algorithm performs reliably under various circumstances, demonstrating its timeliness, efficiency and robustness in solving the given problem.

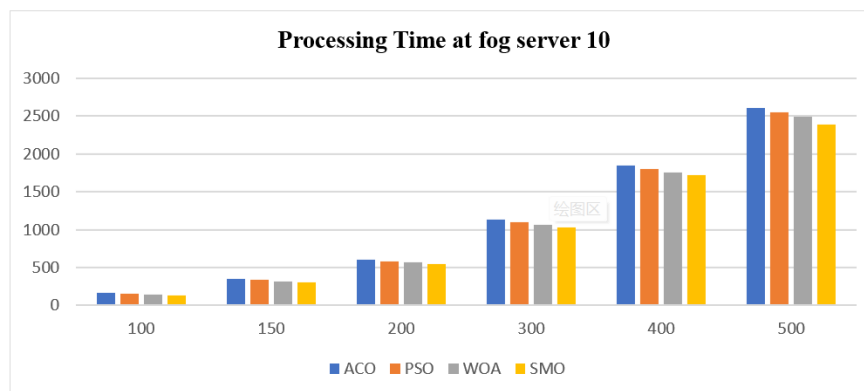


Fig. 2. Average processing time result at fog server 10

B. Energy Consumption

Energy consumption in fog computing is a critical concern due to the need for continuous processing, storage, and data transmission at the network edge. Efficient resource management and optimization techniques, such as workload balancing and computation offloading, help reduce energy consumption while maintaining performance. Implementing energy-aware scheduling algorithms can further enhance sustainability and extend the lifespan of fog nodes. In the initial stage of simulation, we planned to analyse the energy consumption parameters with all tasks on different resources based

on a series of tasks. However, with such experiments, we did not reach the upper bound of energy consumption parameters. Initially, we have started the experiment for 100 tasks over 10 fog servers utilizing the SMO algorithm. As shown in Figure 5, we observed that the proposed algorithm saves energy by an average of 16%, 12% and 10% approximately compared to ACO, PSO and WOA respectively. One more interesting observation we found with the comparison results is that with the increment of tasks, the energy saving amount also increases as shown in Figure 5 from tasks 100 to 500. To check the energy consumption with higher fog servers, we performed the simulation experiments on 20 and 30 fog servers. Figure 6 demonstrates the result of energy consumption with the same set of tasks against the 20 fog servers. The comparative result shows that the proposed algorithm cut down the energy consumption rate by an average of 19%, 15% and 12% approximately compared to benchmark approaches ACO, PSO and WOA respectively. In addition, a similar result was found in Figure 7 for 30 fog servers.

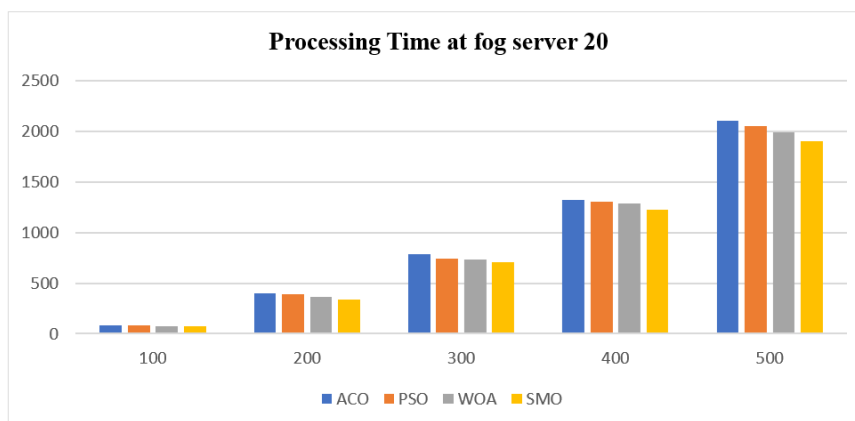


Fig. 3. Average processing time vs number of task comparison result for fog server 20

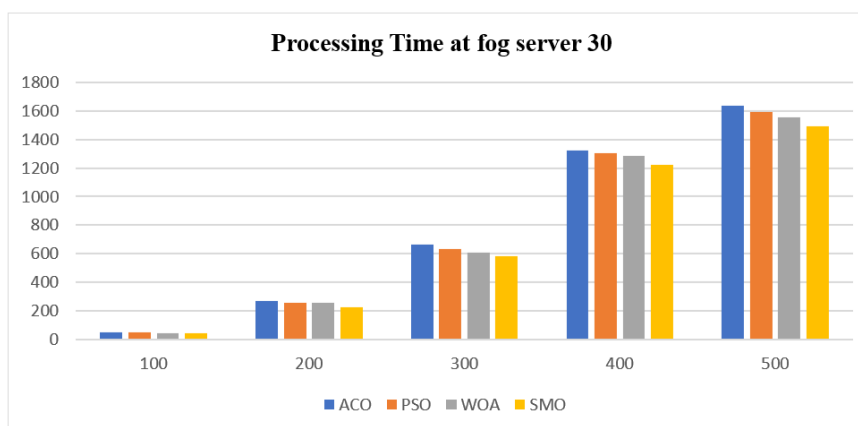


Fig. 4. Average processing time vs number of task comparison result for fog server 30

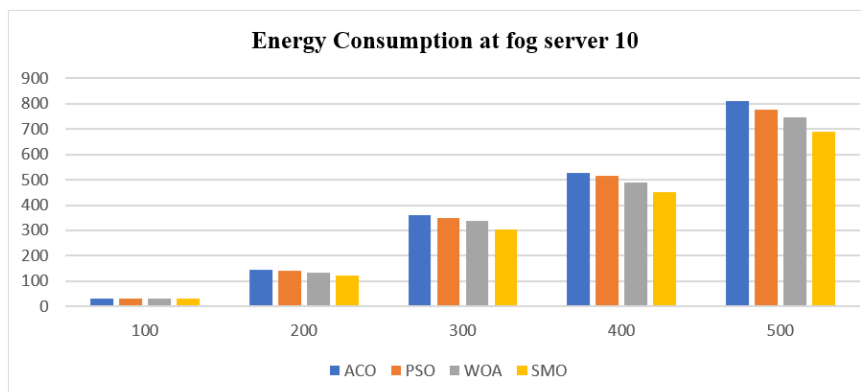


Fig. 5. Energy consumption vs number of task comparison result for fog server 10

C. Total Computation Cost

The computation cost in collaborative fog cloud environment refers to the expense associated with executing the

tasks at the network edge or IoT devices or cloud data centre depending upon offloading decision, including energy usage, resource allocation, and infrastructure maintenance. Several experiments have been conducted to evaluate the performance of the proposed algorithm. Cost is one of the most significant parameters that can be defined as the amount paid by end users for the services. The presented research work considers the Amazon Elastic Compute Cloud (EC2) model for calculating the cost parameter. The authors have considered three test cases to assess the performance where the number of fog servers varies from 10 to 30 against the task sets 100, 200, 300, 400, and 500.

Test Case 1: Initially, the number of fog servers is fixed at 10, and the upcoming IoT tasks are varied from 100 to 500 with a step size of 100 each time. The proposed SMO algorithm has been applied to get the finest resource for these tasks. The proposed SMO algorithm utilizes the properties of fusion and fission to improve exploration and exploitation ability that help to find the best possible resources in defined search space, where state-of-the-art meta-heuristic algorithm failed. We have compared the experimental results of proposed approach with state-of-the-art algorithm to validate the performance. The results based upon the simulation, shown in Figure 8, verify that the proposed approach has the least cost compared to the other benchmark approaches. The integrated SMO-based proposed framework minimizes the overall cost by an average of 8.98%, 11.67 %, and 15.41% compared to ACO, PSO and WOA, respectively.

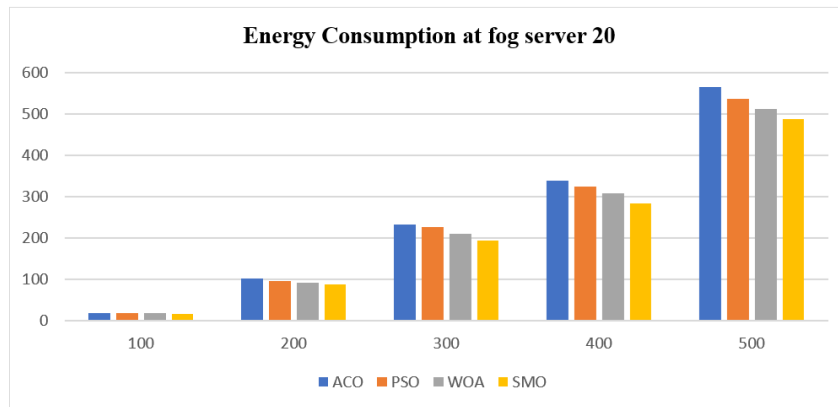


Fig. 6. Energy consumption vs number of task comparison result for fog server 20

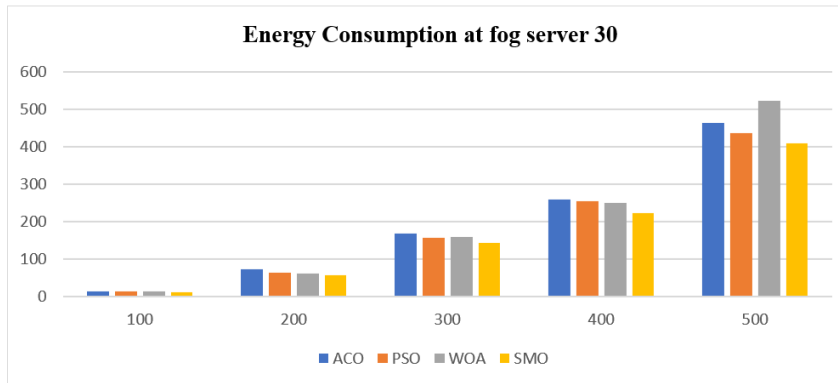


Fig. 7. Energy consumption vs number of task comparison result for fog server 30

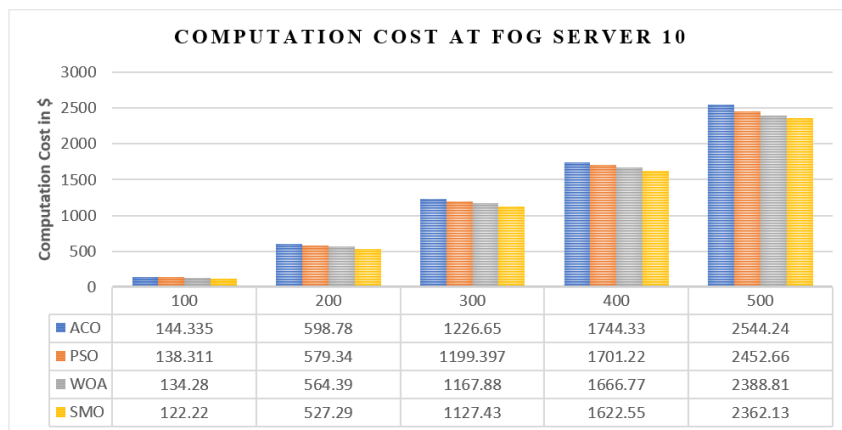


Fig. 8. Total computation cost vs number of task comparison for fog server 10

Test Case 2: In order to test the consistency of our proposed algorithm, we have increased the fog servers from 10 to 20 with the same task set and applied the proposed algorithm along with state-of-the-art approaches. Figure 9 demonstrates the comparative result analysis of total computation cost with the varying number of tasks. It is concluded from the result that the proposed algorithm reduces the total computation cost by an average of 8.6%, 11.2%, and 15.9% compared with the benchmark approaches ACO, PSO and WOA respectively in the same simulation scenarios.

Test Case 3: Similar to fog servers 10 and 20, the experiments were tested at 30 fog servers with the same number of tasks set to check the upper bound of our proposed algorithm. Figure 10 demonstrates the total computation cost comparative analysis of the proposed algorithm with baseline approaches at 30 fog servers. The results demonstrate the potential enhancement in the overall computation cost by applying the SMO algorithm compared with other state-of-the-art approaches. The outcome of all three simulation experiments confirms that the proposed algorithm computes the IoT-based tasks at a lower cost compared with the others.

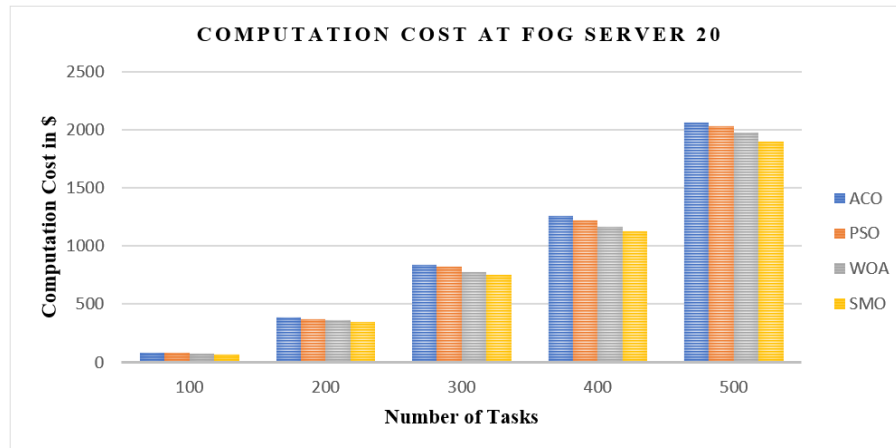


Fig. 9. Total computation cost vs number of task comparison for fog server 20

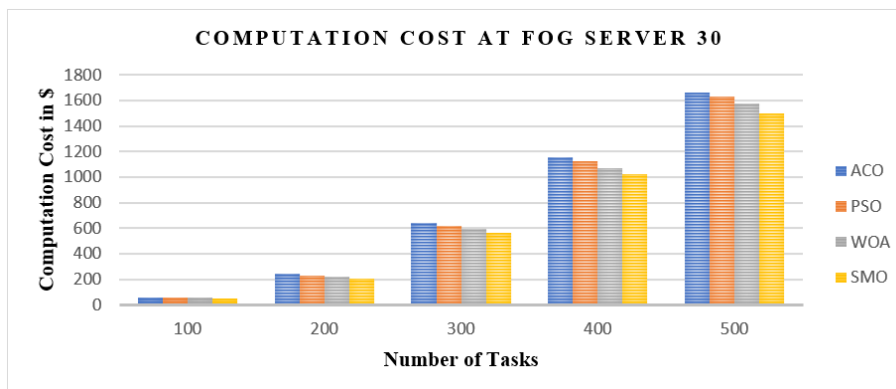


Fig. 10. Total computation cost vs number of task comparison for fog server 30

6. Conclusion and Future Work

With the rapid growth of IoT devices, an enormous amount of data is produced, necessitating an efficient computing paradigm for storage and processing. Cloud computing has the capability to manage such vast amounts of data and offer services on a pay-as-you-go basis, but it struggles to provide real-time responses for latency-sensitive IoT applications. Therefore, fog computing extends the services of cloud to the network edge, enabling faster response times for IoT applications. However, several challenges arise, including computation offloading decisions, unpredictable demand, workload imbalance among fog servers, and system heterogeneity. To counter these challenges, the authors have proposed an intelligent framework that offloads the upcoming requests based upon their demand and availability of resources. The proposed framework utilizes SMO algorithm to find an efficient solution and optimize the QoS parameters. The framework employs an adaptive strategy for workload allocation, dynamically distributing tasks based on the computational capacity of resources. Additionally, it continuously monitors workload distribution across fog nodes to prevent overloading or underutilization. The SMO algorithm's exploration and exploitation capabilities help avoid local optima, ensuring optimal offloading decisions. The performance of the SMO-integrated proposed framework is evaluated in a simulation environment, demonstrating that the algorithm significantly improves key QoS parameters, including latency, communication overhead, cost, and energy consumption, compared to baseline approaches under the same experimental conditions. The proposed framework can be extended using AI-based workload prediction approaches such

as LSTM and Gated Recurrent Unit (GRU). In addition, data can be secured using blockchain-based approach that increases the trust level of the users.

All the Declarations and Statements

Author Contributions Statement

Mohammad Aknan (corresponding author): Conceptualization, methodology, simulation, validation, formal analysis, investigation, resources, writing—original draft, visualization, result analysis

Rajeev Arya: Methodology, validation, formal analysis, investigation, writing—review & editing.

Maheshwari Prasad Singh: Writing – Review and Editing, and Project Management: Reviewed and edited the manuscript, ensured clarity and coherence, and helped coordinate project milestones and deadlines, supervision.

All authors have read and agreed to the published version of the manuscript.

Conflict of Interest Statement

The authors declare no conflict of interest.

Funding Declaration

This research received no external funding.

Data Availability Statement

The simulation was performed by varying the number of fog nodes and tasks. Results and code can be accessed via the corresponding author (mohammada.phd19.cs@nitp.ac.in).

Ethical Declarations

This study did not involve human or animal subjects requiring ethical approval.

Acknowledgments

The authors would like to express their sincere gratitude to the Department of Computer Science, National Institute of Technology, Patna for providing the necessary academic environment and resources that supported this research.

Declaration of Generative AI in Scholarly Writing

The authors confirm that no generative artificial intelligence (GAI) tools, including transformer-based models, were used in the creation of this work. All datasets, images, text, and other research content were developed, curated, or written by the authors without assistance from any generative AI system. The design, analysis, and interpretation of results are entirely the product of the authors' own research effort.

Abbreviations

The following abbreviations are used in this manuscript:

ACO - Ant Colony Optimization
AI - Artificial Intelligence
DRL - Deep Reinforcement Learning
GA - Genetic Algorithm
GLD - Global Leader Decision Phase
GLL - Global Leader Learning Phase
GLP - Global Leader Phase
GRU - Gated Recurrent Unit
GWO - Grey Wolf Optimization
IoD - Internet of Drone
IoT - Internet of Things
LLP - Local Leader Phase
LLD - Local Leader Decision Phase
LLL - Local Leader Learning Phase
LSTM - Long Short-Term Memory
PSO - Particle Swarm Optimization
QoS - Quality of Service
SMO - Spider Monkey Optimization
SSA - Salp Swarm Algorithm

WOA - Whale Optimization Algorithm
XaaS - Everything as a Service

Appendix

Not applicable (no additional materials beyond figures/tables in main text).

References

- [1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future direction," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645-1660, 2013.
- [2] S. S. Goel, A. Goel, M. Kumar, and G. Molto, "A review of Internet of Things: qualifying technologies and boundless horizon," *Journal of Reliable Intelligent Environments*, vol. 7, no. 1, pp. 23-33, 2021.
- [3] M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, "A comprehensive survey for scheduling techniques in cloud computing," *Journal of Network and Computer Applications*, vol. 143, no. 1, pp. 1-33, 2019.
- [4] P. H. Vilela, J. J. P. C. Rodrigues, P. Solic, K. Saleem, and V. Furtado, "Performance evaluation of a Fog-assisted IoT solution for e-Health applications," *Future Generation Computer Systems*, vol. 97, pp. 379-386, 2019.
- [5] H. Sulimani, W. Y. Alghamdi, T. Jan, G. Bharathy, and M. Prasad, "Sustainability of load balancing techniques in fog computing environment," *Procedia Computer Science*, vol. 191, pp. 93-101, 2021.
- [6] M. H. Kashani and E. Mahdipour, "Load balancing algorithms in fog computing," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1505-1521, 2022.
- [7] A. Robles-Enciso and A. F. Skarmeta, "A multi-layer guided reinforcement learning-based tasks offloading in edge computing," *Computer Networks*, vol. 220, no. 1, pp. 109476, 2023.
- [8] M. Tang and V. W. S. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 1985-1997, 2020.
- [9] S. Tian, X. Deng, P. Chen, T. Pei, S. Oh, and W. Xue, "A dynamic task offloading algorithm based on greedy matching in vehicle network," *Ad Hoc Networks*, vol. 123, pp. 102639, 2021.
- [10] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," *Proceedings of the 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 352-357, 2014.
- [11] Z. Aghapour, S. Sharifian, and H. Taheri, "Task offloading and resource allocation algorithm based on deep reinforcement learning for distributed AI execution tasks in IoT edge computing environments," *Computer Networks*, vol. 223, pp. 109577, 2023.
- [12] S. Wang, S. Zhao, H. Gui, X. He, Z. Lu, B. Chen, Z. Fan, and S. Pang, "Energy-efficient collaborative task offloading in multi-access edge computing based on deep reinforcement learning," *Ad Hoc Networks*, vol. 103743, 2024.
- [13] Y. Tu, H. Chen, L. Yan, and X. Zhou, "Task offloading based on LSTM prediction and deep reinforcement learning for efficient edge computing in IoT," *Future Internet*, vol. 14, no. 2, pp. 30, 2022.
- [14] X. Chen, J. Cao, Y. Sahni, S. Jiang, and Z. Liang, "Dynamic Task Offloading in Edge Computing based on Dependency-aware Reinforcement Learning," *IEEE Transactions on Cloud Computing*, vol. 12, no. 2, pp. 594-608, 2024.
- [15] Z. Cao, X. Deng, S. Yue, P. Jiang, J. Ren, and J. Gui, "Dependent Task Offloading in Edge Computing Using GNN and Deep Reinforcement Learning," *IEEE Internet of Things Journal*, pp. 21632-21646, 2024.
- [16] H. Li, W. Li, and X. Zhang, "A genetic algorithm for task offloading problem in vehicular edge computing," *Proceedings of the 2022 China Automation Congress (CAC)*, pp. 1-6, 2022.
- [17] X. Chen, Y. Mao, H. Wang, Y. Xu, X. Xiong, and D. Li, "ACO-based Task Offloading Method with Limited Resource in Edge Computing," *IEEE Sustainable Power and Energy Conference (iSPEC)*, pp. 1-6, 2021.
- [18] S. Zaidi, M. A. Attalah, K. Lazhar, and C. T. Calafate, "Task Offloading Optimization Using PSO in Fog Computing for the Internet of Drones," *Drones*, vol. 9, no. 1, pp. 23, 2024.
- [19] C. Lin, D. Deng, S. Suwatharachaitiwong, and Y. Li, "Dynamic weighted fog computing device placement using a bat-inspired algorithm with dynamic local search selection," *Mobile Networks and Applications*, vol. 25, pp. 1805-1815, 2020.
- [20] X. Wang, Z. Zhou, H. Chen, and Y. Zhang, "Task offloading and power assignment optimization for energy-constrained mobile edge computing," *Proceedings of the 2021 Ninth International Conference on Advanced Cloud and Big Data (CBD)*, pp. 1-6, 2021.
- [21] M. Kumar, G. K. Walia, H. Shingare, S. Singh, and S. S. Gill, "AI-Based sustainable and intelligent offloading framework for IIoT in collaborative cloud-fog environments," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 4, pp. 1414-1422, 2023.
- [22] Xinqian Zhang, Tingming Wu, Mingsong Chen, Tongquan Wei, Junlong Zhou, Shiyan Hu and Rajkumar Buyya, "Energy-aware virtual machine allocation for cloud with resource reservation," *Journal of Systems and Software*, vol. 147, pp. 147-161, 2019.
- [23] S. Singh and I. Chana, "QRSF: QoS-aware resource scheduling framework in cloud computing," *The Journal of Supercomputing*, vol. 71, no. 1, pp. 241-292, 2015.
- [24] G. K. Walia and M. Kumar, "Computational Offloading and resource allocation for IoT applications using decision tree-based reinforcement learning," *Ad Hoc Networks*, vol. 120, no. 1, pp. 103751, 2025.
- [25] H. K. Apat and B. Sahoo, "LESP: A fault aware internet of things service placement in fog computing," *Sustainable Computing: Informatics and Systems*, vol. 101097, 2025.
- [26] S. Kumar, R. Kumari, and V. K. Sharma, "Fitness Based Position Update in Spider Monkey Optimization Algorithm," *Procedia Computer Science*, vol. 62, pp. 442-449, 2015.
- [27] A. Sharma, A. Sharma, B. K. Panigrahi, D. Kiran, and R. Kumar, "Ageist spider monkey optimization algorithm," *Swarm and Evolutionary Computation*, vol. 28, pp. 58-77, 2016.
- [28] K. Gupta, K. Deep, and J. C. Bansal, "Spider monkey optimization algorithm for constrained optimization problems," *Soft Computing*, vol. 21, no. 23, pp. 6933-6962, 2017.

- [29] K. Gupta, K. Deep, and J. C. Bansal, "Improving the local search ability of spider monkey optimization algorithm using quadratic approximation for unconstrained optimization," *Computational Intelligence*, vol. 33, no. 2, pp. 210-240, 2017.

Authors' Profiles



Mohammad Aknan is currently pursuing his PhD in the Department of Computer Science and Engineering at NIT Patna. He has over 12 years of experience in industry and academia.



Maheshwari Prasad Singh is currently working as a Professor in the Department of Computer Science and Engineering. Prof. Singh has over two decades of experience in academia and has published more than 180 research papers in refereed journals and international conferences.



Rajeev Arya is working as an Assistant Professor in the Department of Electronics and Communication Engineering at SVNIT Surat. His research area includes Wireless Sensor Networks, Device-to-Device Communication, 5G/6G Network, UAV Communications, IoT, quantum computing and quantum dot cellular automata. Dr Arya has published more than 60 research papers in various journals.

How to cite this paper:

Mohammad Aknan, Maheshwari Prasad Singh, Rajeev Arya, "An Efficient Resource Allocation Algorithm with Load Balancing Mechanism to Enhance QoS Parameters in Fog Environment", *International Journal of Intelligent Systems and Applications (IJISA)*, Vol. 18, No. 5, pp. 22-36, 2026. DOI: 10.5815/ijisa.2026.05.02