Journal of Information Systems Engineering and Management
2025, 10(22s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Delay Aware Global Information Processing for Offloading Computation at Core Level Using Nature-Inspired Algorithm

Sarkarsinha Harsinha Rajput[1] Dr. Manoj Eknath Patil[2]
Ph.D. Research Scholar, Department of Computer Engineering,
SSBT's College of Engineering & Technology,
Jalgaon, Maharashtra 425001, India.
Email: bs.rajput26@gmail.com
Associate Professor, Department of Computer Engineering,
SSBT's College of Engineering & Technology,
Jalgaon, Maharashtra 425001, India.
Email: mepatil@gmail.com

ARTICLE INFO

Received: 21 Dec 2024

Revised: 04 Feb 2025

Accepted: 18 Feb 2025

ABSTRACT

Introduction: Nowadays, fog computing has emerged as a promising solution for handling the prompt processing of tasks in Internet of Things (IoT)-based applications. One of the key advantages of fog computing is that it reduces service completion time by offloading tasks from IoT devices to the fog server. Therefore, scheduling of tasks becomes vital, where emergency and non-emergency tasks can be prioritized to offload data to the nearby fog servers, which improves the Quality of Service (QoS). Due to the dynamic nature of the IoT environment, traffic load varies over time, making it difficult to select the optimal fog server for task offloading.

Objectives: This research introduces a novel task offloading for the Fog Cloud scenario using the improved Coati Optimization Algorithm based on Genetic Operators (ICOA-GO) algorithm.

Methods: Initially, the Delay aware Four Queue model with Fuzzy logic (DFQM-Fuzz) is designed to queue incoming tasks into four different priorities. In the DFQM-Fuzz, Highly urgent (HU) and Urgent (U) tasks are considered as first and second priorities and those tasks are offloaded through the Fog Server. The Non Urgent (NU) and No Deadline (ND) tasks are offloaded through the Cloud Server. Furthermore, when there is no resource to offload through the Fog Server, the task is offloaded through the Cloud Server.

Results: The proposed improved Coati Optimization Algorithm based on Genetic Operators (ICOA-GO) algorithm optimizes the selection of Cloud and Fog Servers. The analysis based on Energy Utilization, Task Rejection Ratio, and Makespan yielded values of 93.589, 1, and 1.96073.

Conclusions: By integrating the DFQM-Fuzz and ICOA-GO improves task prioritizing and offloading efficiency while reducing execution time, energy consumption, and cost.

Keywords: Fog Server, Optimization Algorithm, Task Offloading, Prioritization, Internet Of Things (Iot), Task Rejection.

INTRODUCTION

The recent advancements of information technology have led to the widespread use of mobile devices in daily life. In recent years, mobile phones, wearable technology, smart devices, industrial gadgets, smart devices, and other items have been connected to the Internet (Deb et al, 2021). In addition to having limited energy and resources (CPU, storage, and memory), these devices handle a significant amount of data. The Open Fog Consortium and the European Telecommunications Standards Institute (ETSI) have given definitions and guidelines for computing overhead (Jazayeri et al, 2021). The deployment of smart systems such as smart factories, smart grids, smart cities, smart supply chains and logistics, and smart factories has become impossible without the Internet of Things (IoT) (Tran-Dang & Kim, 2023). Cloud computing requires sufficient resources to perform activities effectively; it is still a challenging task. However, due to limited spectrum resources, sporadic network connectivity, and the large physical distance between IoT devices and faraway cloud servers, cloud computing-based solutions fail to provide the desired QoS for delay-sensitive applications (Gasmi et al, 2022; Mishra et al, 2023).

Fog computing is used to expand cloud computing resources closer to data generation sources. It enables services and applications to meet their QoS levels by allowing fog computing devices to handle and offload most tasks on behalf of cloud servers in a distributed manner (Hussain & Beg, 2021). Fog Computing Systems (FCSs) consist of interconnected fog computing devices that are required for IoT-based systems to provide low response latency and uninterrupted services and applications across the things-to-cloud spectrum (Salehnia et al, 2024). For real-time tasks, it is essential to offload such execution methods to external platforms like Fog for fast processing. However, these methods only concentrate on selecting a single Fog Node (FN) and completing all tasks there (Mazumdar et al, 2021; Tran-Dang & Kim, 2021). IoT Sensor Nodes (SNs) use Directed Acyclic Task Graphs (DATGs) to express interdependent subtasks. Certain operations at the same level in DATG could be executed concurrently to significantly reduce processing delay (Deng et al, 2021).

Effective resource allocation strategies are necessary for FCSs to perform task offloading and reap the benefits of fog computing (Yu-Jie et al, 2022). Some researchers have enhanced their allocation technique by adopting a centralized approach to resource distribution. However, self-centered IoT users struggle to maximize their personal Quality Of Experience (QoE). They may fail to execute the procedures required to optimize system performance (Bai et al, 2021; Meena et al, 2021). To reduce processing time at fog nodes, various application forms' virtual parallel queues are taken into consideration. However, the system's performance suffers due to the lack of a load-balancing mechanism. These capable systems' processing and storage capacities determine their QoE (Abdulazeez & Askar, 2023). As a result, a queueing system offers a comprehensive solution for processing a large number of requests in accordance with an appropriate scheduling pattern. These queueing models have enough predictive capacity to forecast behavior and performance in both low- and high-traffic conditions (Razaq et al, 2021). Thus, this study introduces a novel delay-aware scheduling and server selection method. The objectives of the research are to introduce Delay-aware computation offloading on Fog system using global information processing and improved nature-inspired computational intelligence algorithm. It also implements a delay-aware Four Queue Model With Fuzzy Logic (DFQM-Fuzz) model for scheduling the offloading tasks in Fog level. This study aimed to present an improved Coati Optimization Algorithm based on Genetic Operators (ICOA-GO), it is employed to offload the computations in fog and cloud computing environments, and to compare the performance of the proposed approach with the recently developed approaches for Delay aware computation offloading at the core level.

The rapid expansion of the IoT has led to an unprecedented increase in data generation, necessitating efficient processing and management to derive actionable insights (LEE et al, 2020). Traditional cloud computing paradigms face significant challenges in handling the latency-sensitive nature of IoT applications due to the physical distance between data sources and centralized data centers (Angel et al, 2021). Fog computing emerges as a viable solution by bringing computation closer to the edge, thereby reducing latency and improving real-time data processing capabilities (Das & Inuwa, 2023). However, efficient offloading of computational tasks in a distributed environment remains a complex task. To increase performance and guarantee task completion on time, this research is motivated by the need to optimize computation offloading in IoT-based fog computing systems.

The integration of IoT with fog computing systems poses significant challenges in terms of delay-aware computation offloading (Sabireen & Neelanarayanan, 2021). Current methods often struggle with inappropriate task distribution, resulting in increased latency, inefficient resource consumption, and potential system bottlenecks. These issues are exacerbated by the heterogeneous and dynamic nature of fog computing environments, in which devices with various computational capabilities must collaborate to process data effectively. Traditional optimization techniques fail to adequately address the complexities involved, resulting in poor performance and user dissatisfaction (Laroui et al, 2021).

To address these issues, this study proposes developing an enhanced nature-inspired computational intelligence algorithm and a strong framework that can efficiently handle the dynamic and diverse nature of fog computing environments by utilizing enhanced nature-inspired computational intelligence algorithms.The proposed algorithm aims to optimize the offloading decisions by considering factors such as computational delay, energy consumption, and network latency, and can significantly improve the performance and reliability of core-level computing systems, ensuring that latency-sensitive applications meet their stringent requirements.

LITERATURE REVIEW

Sadoon Azizi et al. 2022 developed two semi-greedy-based algorithms named priority-aware semi-greedy (PSG) and PSG-with multi-start procedure (PSG-M) to map the IoT tasks in FNs efficiently. In order to achieve QoS for IoT tasks, the task scheduling problem was first developed to reduce the problem of FN's energy consumption (Azizi et al, 2022). This technique enhances the percentage of IoT tasks that achieve the deadline requirement, energy consumption, makespan, and deadline violation time. The percentages of tasks meet their deadline requirements, such as 95.2% for PSG and 96.5 for PSG-M when FNs is 60.

Naveen Chauhan et al. 2021 developed a multi-class open queueing model that is utilized to maintain the traffic on various Delay-aware applications offloading (DAAO). This technique enhanced the performance of a 14.30% service rate and reduced the loss rate to 2.0%. The multi-class Brownian model was used to design FN's architecture, which can serve several customers (Chauhan et al, 2021). This technique was designed as a Weighted-Fair Queueing (WFQ), non-WFQ and load-balancing algorithm.

Moreover, Maryam Keshavarznejad et al. 2021 developed a task offloading in the form of a multi-objective optimization issue to reduce total power consumption and delay in executing tasks. This technique utilizes two meta-heuristic methods named the Non-Dominated Sorting Genetic Algorithm (NSGA-II) and the Bees algorithm (Keshavarznejad et al, 2021). This technique reduces the time consumption and delay; the maximum time consumption is 5000 joules for offloading.

Similarly, Parmeet Kaur and Shikha Mehta created a QoS-aware task offloading technique based on a novel nature-inspired optimization algorithm named the Smart Flower Optimization Algorithm (SFOA). This technique is used to offload delay-sensitive operations in the IoT, lowering execution costs and deadlines (Kaur & Mehta, 2022). This technique yields a minimum execution time of 75.1 seconds for 300 tasks and 55.6 seconds for 1,000 tasks.

Sanjaya Kumar Panda et al. 2023 developed a multi-objective task offloading technique named EDP-TO for load balancing. The multi-objective function is used to select FNs for offloading. This technique divides the tasks into many sub-tasks and allocates them to the appropriate FNs (Panda et al, 2023). This concept reduces the overall delay. Here, 60% of FNs were active nodes in all three scenarios, such as 3, 69, and 18. The summary of related works is presented in Table 1.

Table 1. Summary of related works

Author name and reference	Technique used	Performance	Limitation
Sadoon Azizi et al.	PSG, PSG-M	95.2% for PSG and 96.5 for PSG-M when FNs are 60	It can execute only one task at a time in each FN
Naveen Chauhan et al.	A multi-class open queueing model	loss rate as 2.0%	It is only utilized for a single centralized fog server
Maryam Keshavarznejad et al.	NSGA-II and the Bees algorithm	Maximum time consumption is 5000 joule for offloading	This technique does not perform on clustering fog nodes
Parmeet Kaur and Shikha Mehta	SFOA	55.6 least execution time for 1000 tasks	The worst solution towards the best local/global solution occurred by this technique.
Sanjaya Kumar panda et al.	EDP-TO for load balancing	Mean energy consumption and mean delay	Complex to perform

OBJECTIVES

This research introduces a novel delay-aware scheduling and server selection method. The objectives of the research are:

- To introduce Delay-aware computation offloading on Fog system using global information processing and improved nature-inspired computational intelligence algorithm.
- To implement Delay aware Four Queue model with Fuzzy logic (DFQM-Fuzz) model for scheduling the offloading tasks in Fog level.

- To present an improved Coati Optimization Algorithm based on Genetic Operators (ICOA-GO), it is employed to offload the computations in fog and cloud computing environments.
- To compare the performance of the proposed approach with the recently developed approaches for Delay aware computation offloading at the core level.

METHODOLOGY

Study Design

The complexity of computation offloading increases with the number of offloading tasks, making it a nontrivial and NP-hard problem. The existing studies mostly focused on minimizing the overall communication delay, processing cost, and time. However, a priority during task scheduling in optimal fog server selection for computation offloading based on their source requirements and deadline constraints is challenging. In this study, a Global Information Processing model is designed to schedule tasks generated by IoT devices and process them in the appropriate computing resource to achieve QoS. Global Information Processing (GIP) at the fog-cloud level for computation offloading entails strategically distributing computational tasks over a layered architecture of fog and cloud computing. Initially, the Delay aware Four Queue model with Fuzzy logic (DFQM-Fuzz) is developed, in which highly urgent and urgent tasks are en-queued in the first and second queues, respectively, for offloading to a nearby fog server while taking the delay factor into consideration. Non-urgent and no-deadline tasks will be queued in the third and fourth queues, respectively, for direct cloud offloading due to their low delay factor. Here, factors such as task size, arrival time, and delay are considered while scheduling the task into four priorities. Fog computing and cloud computing are combined in the GIP to maximize data processing and management, especially from IoT devices. This GIP model aims to balance the computational load between local fog nodes and remote cloud servers at the core level, improving efficiency, reducing latency, and enhancing overall system performance.

Furthermore, to solve the issue of selecting the best fog server for computation offloading, an improved Coati Optimization Algorithm based on Genetic Operators (ICOA-GO) is used to offload calculations in fog computing environments. In rare cases where fog nodes lack adequate resources, offloading computation will take place in the cloud. An enhanced version of the proposed meta-heuristic method achieves three objectives: 1) minimizing task execution time, 2) minimizing energy usage for connected devices, and 3) minimizing execution costs for using server resources. A customized mutation operation is applied to the Coati Optimization Algorithm (COA), extending the functionality of the Standard COA to improve global search abilities. The system model is presented in Figure 1.

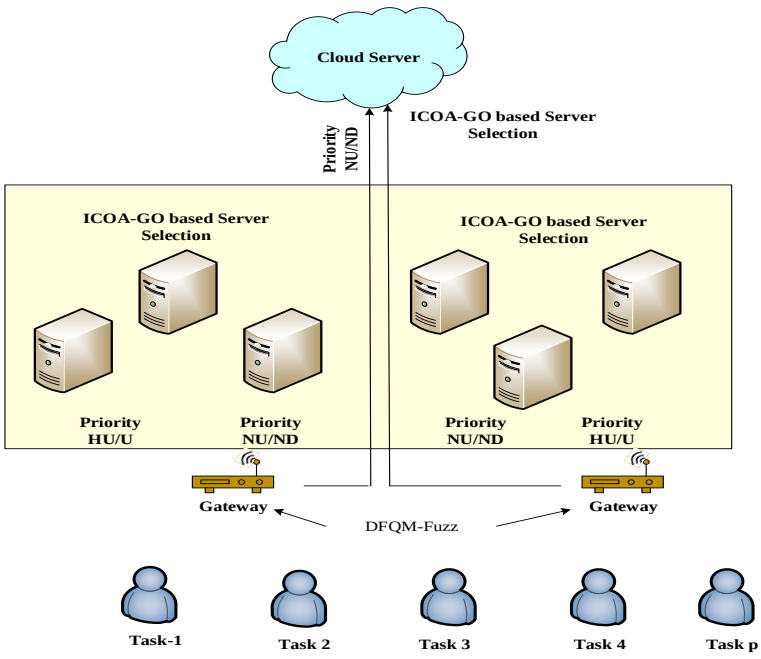

Figure 1. System Model

Delay aware task scheduling using DFQM-Fuzz

The Delay aware Four Queue model with Fuzzy logic (DFQM-Fuzz) is used for delay-aware task scheduling. The incoming tasks are offloaded into four various queues: Highly urgent (HU), Urgent (U), Non-Urgent (NU), and No Deadline (ND). Here, the DFQM-Fuzz model is used to offload the task into four different queues. Task size, arrival time, and delay limits are considered while categorizing the task. Let the incoming tasks be denoted as: $H = \{H_1, H_2, ...H_p\}$, wherein $H_p(n)$ symbolizes the p^{th} task at the time n. The IoT devices are indicated as $Q = \{Q_1, Q_2, ...Q_p\}$, and the fog nodes distributed in the fog layer are signified as $B = \{B_1, B_2, ...B_q\}$. In order to schedule the task into four various queues, fuzzy logic is employed by considering task size, arrival time, and delay as factors.

Task Size: Task size is the computational workload required to complete a task, which is measured in bits. Larger tasks need more processing time and resources and are considered lower priority unless they are urgent.

Arrival Time: Arrival time is the time when the task is generated or received in the system. Earlier arrival times of the task are assigned with higher priority.

Delay: Delay is the time taken for a task to experience transmission, queuing, and processing lags. Tasks with higher delays are assigned with higher priority to minimize QoS violations.

By considering the three factors, fuzzy-based decision-making is employed for prioritizing the task and is defined in Table 2.

Table 2. Sample Fuzzy rule set

Task Size	Arrival Time	Delay	Priority
Small	Early	Low	Highly Urgent (HU)
Large	Late	High	Urgent (U)
Medium	Normal	Moderate	Non-Urgent (NU)
Small	Late	High	No deadline (ND)

After categorizing the task into four various queues, the HU and U are offloaded to the Fog Server and the NU and ND are offloaded through the Cloud Server. The final decision in prioritizing the queues $\gamma_r(n)$ is defined as:

$$\gamma_r(n) = \begin{cases} HU, U & task\ is\ offloaded\ through\ Fog \\ NU, ND & task\ is\ offloaded\ through\ Cloud \end{cases} \quad (1)$$

Furthermore, when there is no resource to offload through the Fog Server, the task is offloaded via the Cloud Server. The ICOA-GO algorithm is used here to process both cloud and fog-based offloading.

3.2 Server Selection using ICOA-GO Algorithm

The proposed ICOA-GO algorithm is used to choose the server that will process the task. In order to improve the convergence rate, the genetic operator (Sivanandam et al, 2008) and Coati solution (Dehghani et al, 2023) update are integrated into the design of the proposed ICOA-GO algorithm. The algorithm considers energy consumption, task execution delay, and execution cost as its fitness function when selecting the server.

3.2.1 Multi-objective Fitness Function

The multi-objective fitness MF for the server selection based on energy consumption EC, task execution delay TED and execution cost ExC is defined as:

$$MF = \alpha \cdot TED' + \beta \cdot EC' + \delta \cdot ExC' \tag{2}$$

where the weights concerning the normalized value of task execution delay, energy consumption and execution cost are symbolized as α , β and δ respectively.

Task execution delay: The delay is determined by the size of the task and the rate at which data is transmitted. A smaller task execution delay results in higher fitness values and is considered as follows:

$$TED = \sum_{k=1}^{K} \frac{S_k}{DT_k} \tag{3}$$

Where, K signifies the total number of tasks, S_k indicates the size of the k^{th} task in bits, and DT_k notates the data transmission rate for the k^{th} task in bits per second. The normalized value of the TED symbolized as TED' .

Energy consumption: The energy consumed by each task is proportional to the power required and the time taken to execute the task. Reducing energy usage is critical for battery-powered devices and is formulated as:

$$EC = \sum_{k=1}^{K} Q_k \cdot TED_k \tag{4}$$

where, Q_k symbolizes the energy consumption for executing the k^{th} task (in joules). The normalized value of energy consumption is signified as EC' .

Execution cost: Execution cost is associated with the expense of utilizing computational resources in cloud or edge servers. Lower costs make the system more economically efficient.

$$ExC = \sum_{k=1}^{K} ExC_k \tag{5}$$

where, ExC_k symbolizes the cost incurred for using server resources to execute the k^{th} task. The normalized value is symbolized as ExC' .

3.2.2 Design of ICOA-GO Model

The proposed ICOA-GO algorithm is designed by integrating conventional coati optimization with genetic operators such as selection, crossover, and mutation. The ICOA-GO algorithm is initialized as follows:

$$A_f : a_{fl} = E_l + R.\left(J_l - E_l\right),\ f = 1,2,.....b, l = 1,2.....g \tag{6}$$

Here, the search bounds of the algorithm are notated as $J_l,$ and E_l respectively, which have the dimension l and R symbolize the random parameter. The solution estimated by the f^{th} Procyonidae search agent is signified as A_f and s symbolizes the decision variable. The total population of the Procyonidae search agent is

denoted as b, and the solution derived by the f^{th} Procyonidae search agent is $a_{f,l}$, wherein its dimension is signified as l. The population of the Procyonidae search agent is interpreted in matrix form A as:

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_f \\ \vdots \\ A_b \end{bmatrix}_{b\times g} = \begin{bmatrix} a_{1,1} \cdots a_{1,l} \cdots q_{1,s} \\ \vdots \quad \ddots \quad \vdots \quad \vdots \\ a_{r,1} \cdots a_{f,l} \cdots a_{f,g} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ a_{b,1} \cdots a_{b,s} \cdots a_{p,s} \end{bmatrix}_{b\times g} \tag{7}$$

Fitness: The feasibility of the solution is evaluated based on the multi-objective fitness function formulated in equation (2).

Diversification: In the diversification phase, the Procyonidae search agents climb the tree, allowing the algorithm to explore distinct areas of the search space and avoid premature convergence on local optimums. In this stage, the Procyonidae search agent uses target-searching using high-dimensional portions of the problem's search space to identify potential areas. The solution obtained by the Procyonidae search agent during the diversification stage is modeled as follows:

$$A_f^{i_1} : a_{f\,l}^{i_1} = a_{f,l} + R.\left(N_l - K.A_f\right) for \;\; f = 1,2.....\left\lfloor \frac{b}{2} \right\rfloor \text{ and } l = 1, 2 \ldots g \tag{8}$$

Here, the target identified by the Procyonidae search agent is notated as N, and the arbitrary parameter that has the bounds $\{1,2\}$ is symbolized as K. Here, in order to enhance the convergence rate of the algorithm, genetic operators like selection, crossover, and mutation are incorporated in the diversification phase of the ICOA-GO algorithm.

Selection: In the selection phase, the best Procyonidae search agent is chosen based on the fitness to perform the reproduction.

Crossover: The offspring are introduced during the crossover phase by combining the parents. The proposed ICOA-GO algorithm combines the two diverse solutions to create a new offspring solution for more effectively exploring the search area.

Mutation: The mutation process of the genetic algorithm increases the algorithm's randomness, which helps to prevent local optimal solution trapping.

Thus, the solution of the Procyonidae search agents is updated in the diversification phase after performing the genetic operations is signified as $\left(A_f^{i_1} : a_{f\,l}^{i_1}\right)'$. In the diversification period, another group of Procyonidae search agents wait below the trees, awaiting prey to fall. This stage represents a group of agents focusing on areas where high-potential solutions may emerge, prepared to refine them further. The location of the prey in the diversification stage is formulated as follows:

$$N^Y : N_l^Y = E_l + R.\left(J_l - E_l, \qquad l = 1,2,....,g\right), \tag{9}$$

$$A_f^{i_1} : a_{f,l}^{i_1} = \begin{cases} A_{f,l} + R.\left(N_c^Y - I\, a_{f,l}\right). & , \quad Fit_{N^Y} < Fit_f, \\ A_{f,l} + R.\left(a_{f,l} - N_l^Y\right). & , \quad else \end{cases} \tag{10}$$

$$\text{For f} = \left\lfloor \frac{b}{2} \right\rfloor + 1, \left\lfloor \frac{b}{2} \right\rfloor + 2, \ldots . b \quad \text{and l=1, 2... g} \tag{11}$$

When a Procyonidae search agent calculates a new position, the algorithm determines whether it improves the objective function. It prevents coatis from moving to worse positions, which could lead to local optimal trapping. The solution updation is interpreted as:

$$A_f = \begin{cases} A_f^{i_1}, & Fit_f^{i_1} < Fit_f, \\ A_f, & else, \end{cases} \tag{12}$$

where the solution derived by the f^{th} procyonidae search agent is symbolized as $A_f^{i_1}$ and the fitness is defined as $Fit_f^{i_1}$.

Intensification: Attacking the prey represents the fine-tuning of a solution and is utilized to refine the solution to find the best possible outcome in a smaller focused region. The intensification step represents exploiting a known solution to improve it further by intensively searching its identified position in the diversification phase. The solution evaluated by the Procyonidae search agent is defined as:

$$E_l^W = \frac{E_C}{V},\ J_l^W = \frac{J_l}{V} \ where\ V = 1,2,\ldots . v_{\max} \tag{13}$$

$$A_f^{i_2} : a_{f\,l}^{i_2} = a_{f\,l} + (1 - 2R).\left(E_l^W + R.\left(J_l^W - E_l^W\right)\right) \tag{14}$$

where, f =1, 2... b, l=1, 2... g

The Procyonidae search agent's solution update during the intensification phase is evaluated to determine if this position enhances the objective function. It is evaluated based on:

$$A_f = \begin{cases} A_f^{i2}, & Fit_f^{i_2} < Fit_f, \\ A_f, & else, \end{cases} \tag{15}$$

where the solution derived by the f^{th} Procyonidae search agent is symbolized as $A_f^{i_2}$ and the fitness is defined as $Fit_f^{i_2}$.

Termination: The acquisition of a global best solution or the attainment of maximal iteration terminates the algorithm processing. The pseudo-code for the ICOA-GO algorithm is presented in Algorithm 1.

Table 3. Algorithm 1: Pseudo-code for ICOA-GO algorithm

Pseudo-code for ICOA-GO algorithm	
1	Initialize the search agent, dimension, and population.
2	The Procyonidae search agents are located arbitrarily.
3	Estimate the feasibility
4	while v<V
5	{
6	The solution estimation based on diversification is evaluated through $A_f^{i_1} : a_{f\,l}^{i_1} = a_{f,l} + R.\left(N_l - K.A_f\right) for \;\; f = 1,2.....\left\lfloor \frac{b}{2} \right\rfloor$
7	$A_f = \begin{cases} A_f^{i_1}, & Fit_f^{i_1} < Fit_f, \\ A_f, & else, \end{cases}$ The fitness is evaluated through
8	The solution estimation based on intensification is evaluated through $E_l^W = \frac{E_C}{V},\; J_l^W = \frac{J_l}{V} where \; V = 1,2,....v_{\max}$
9	$A_f = \begin{cases} A_f^{i2}, & Fit_f^{i_2} < Fit_f, \\ A_f, & else, \end{cases}$ The fitness is estimated as
10	}
11	The best outcome is returned
12	v=v++
13	Stop

Here is the solution obtained using the ICOA-GO algorithm; the server selection is used to perform the task offload.

RESULT AND DISCUSSION

The proposed approach is implemented in iFogSim (java), and its performance is measured by analyzing numerous metrics and demonstrated by comparing the results to other current methods that are implemented. The proposed model implements and compares existing methods such as the Whale optimization algorithm (WOA), Bat optimization algorithm (BAT), Round robin (RR), and random optimization (ROP).

The makespan signifies the time taken to perform the given work, as depicted in Figure 2. The analysis uses a variety of VMs to demonstrate the scalability of the proposed work offloading architecture. Here, reducing high-priority task delays and optimizing virtual machine utilization is made possible by the DFQM-Fuzz model-based task prioritization based on urgency. Additionally, the GIP model prevents bottlenecks and underutilization of particular VMs by offloading tasks to the most appropriate VMs in fog or cloud layers. Additionally, the ICOA-GO-based server selection lowers the makespan and assigns tasks to virtual machines (VMs) with adequate resources. The detailed analysis based on makespan is presented in Table 4.

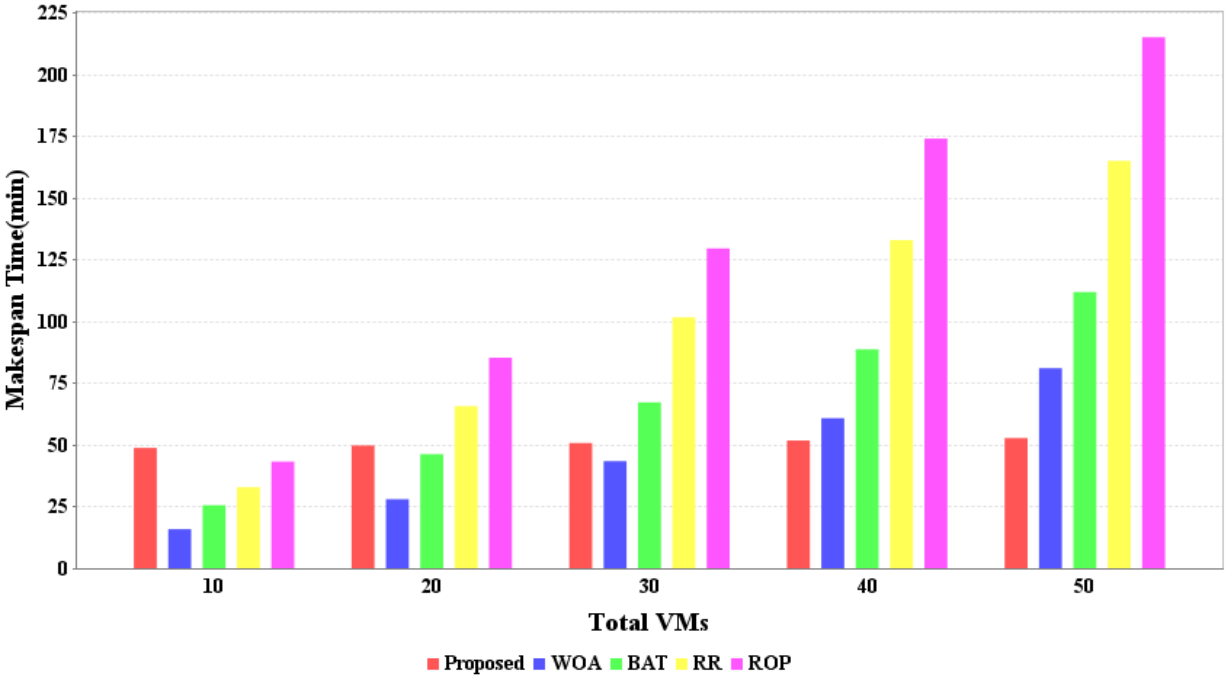

Figure 2. Makespan based on VM

Table 4. Makespan based on VM

Total VMs/Methods	10	20	30	40	50
Proposed	48.832	49.832	50.8327	51.8327	52.8327
WOA	15.8964	28.0399	43.4596	60.9015	81.1608
BAT	25.592	46.2988	67.2951	88.7953	112.048
RR	32.8925	65.7818	101.79	133.102	165.248
ROP	43.2971	85.3937	129.719	174.257	215.348

The makespan analysis based on various tasks is portrayed in Figure 3, and its detailed analysis is presented in Table 5. In this case, the DFQM-Fuzz model-based task queuing model helps minimize queuing delays for high-priority tasks by prioritizing tasks based on urgency. Then, the urgent tasks with HU and U are processed in nearby fog nodes to reduce transmission time and low-priority tasks are offloaded to the cloud for efficient handling. Thus, the minimal makespan is evaluated by the proposed model compared to the existing techniques for various tasks.

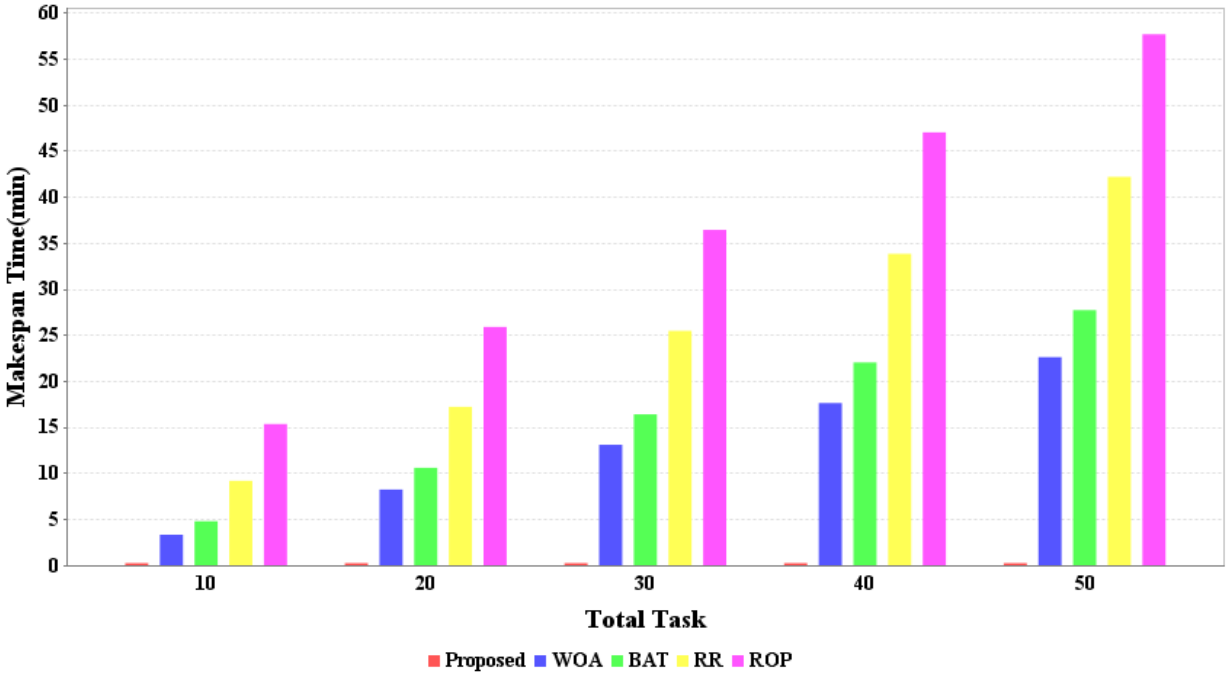

Figure 3. Makespan based on Task

Table 5. Makespan based on Task

Total tasks/methods	100	300	500	700	900
Proposed	1.96073	1.96073	1.96073	1.96073	1.96073
WOA	3.36915	8.41556	13.0753	17.7084	22.3496
BAT	4.86923	10.7064	16.5526	22.3535	28.2007
RR	9.35894	18.0012	26.7764	35.3507	44.2151
ROP	15.3327	26.0155	36.6461	47.4334	58.1581

The task rejection ratio based on VM indicates the percentage of tasks that cannot be processed due to insufficient resources on VMs and is presented in Figure 4. In this, the ICOA-GO algorithm selects the optimal server to schedule the task with sufficient computational capacity. As a result, the suggested model minimizes the task rejection ratio in comparison to traditional approaches. Furthermore, by adding a genetic operator to the traditional coati algorithm, the local optimal trapping problems are resolved, and the best solution for the global is obtained. The optimal server selection for offloading assists in minimizing the task rejection ratio, and the detailed analysis is portrayed in Table 6.

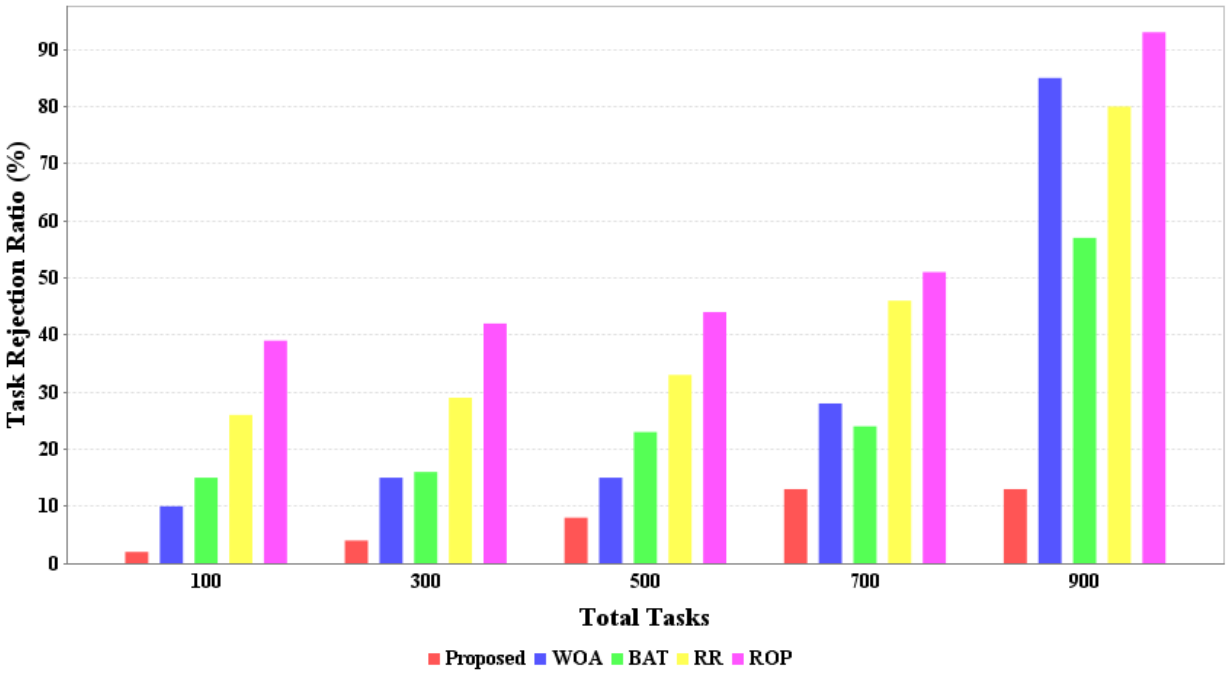

Figure 4. Task Rejection Ratio based on Task

Table 6. Task Rejection Ratio based on Task

Total Tasks/Methods	100	300	500	700	900
Proposed	2	4	8	13	13
WOA	10	15	15	28	85
BAT	15	16	23	24	57
RR	26	29	33	46	80
ROP	39	42	44	51	93

The task rejection ratio by varying the number of VMs is portrayed in Figure 5, and a detailed analysis of it is presented in Table 7. In this case, the DFQM-Fuzz model helps to minimize rejections and delays by en-queuing the task according to priority. In addition, by combining fog and cloud resources, the GIP model ensures that large resource-intensive tasks are accommodated. Also, the inclusion of cloud resources as a fall-back minimizes the rejection of non-urgent and no-deadline tasks. Thus, superior performance is acquired by the proposed model.

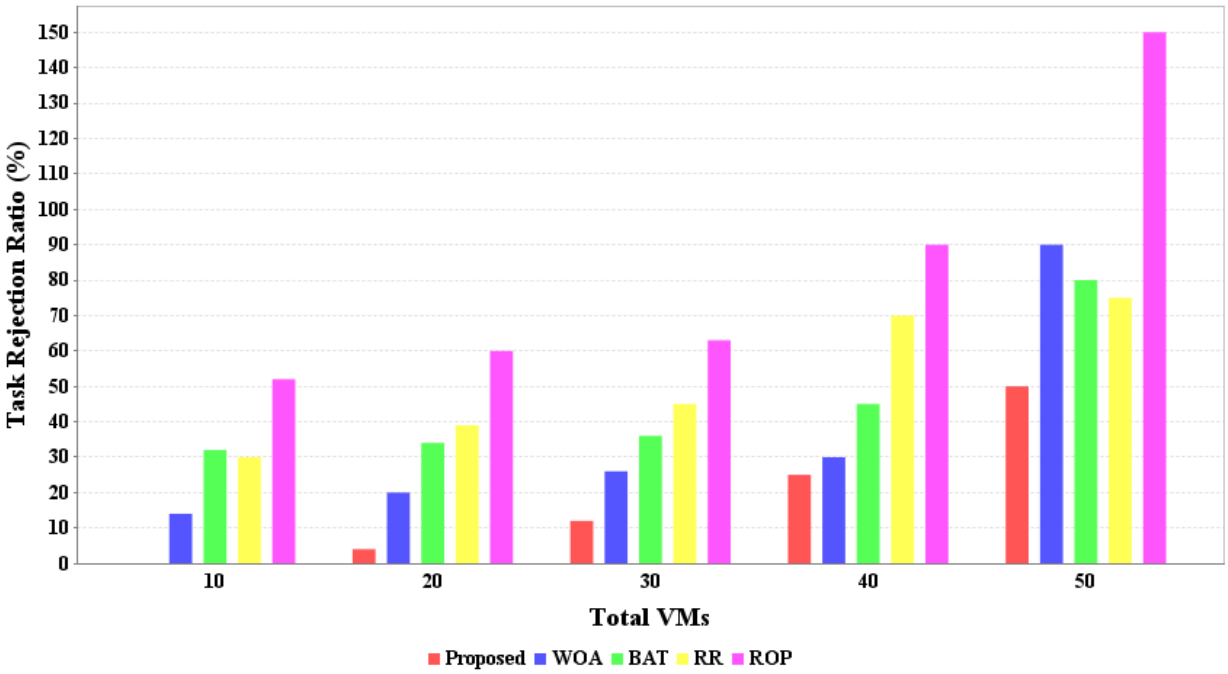

Figure 5. Task Rejection Ratio based on VM

Table 7. Task Rejection Ratio based on VM

Total VMs/Methods	10	20	30	40	50
Proposed	1	5	6	17	34
WOA	11	12	13	15	68
BAT	15	17	22	35	87
RR	24	26	31	44	99
ROP	39	41	45	45	72

The energy used to process individual tasks is measured by the energy utilization analysis shown in Figure 6. As a result, the GIP model helps to reduce transmission and processing energy by offloading tasks to nearby fog nodes whenever feasible. Furthermore, the ICOA-GO-based server offloading considers energy usage to be one of the fitness factors that aid in the efficient use of energy. The detailed energy utilization analysis based on various tasks is presented in Table 8.

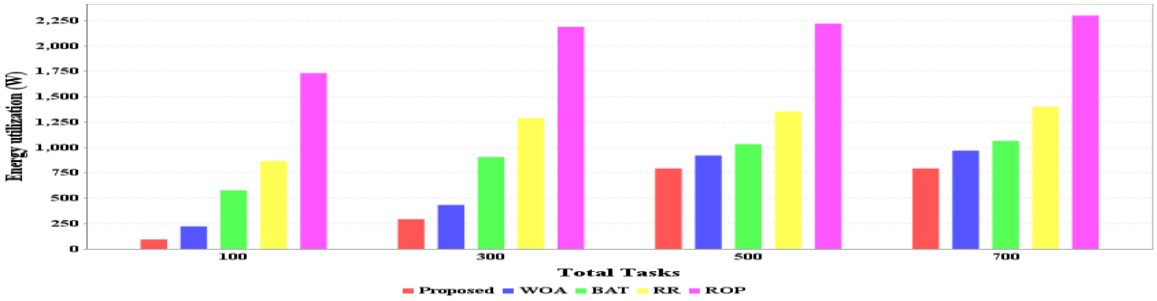

Figure 6. Energy Utilization based on Task

Table 8. Energy Utilization based on Task

Total tasks/Methods	100	300	500	700
Proposed	93.589	293.589	793.589	793.589
WOA	222.523	433.477	921.477	969.477
BAT	577.97	1001.97	1065.97	1113.97
RR	866.955	1290.95	1354.95	1402.95
ROP	1733.91	2189.91	2221.91	2269.91

Figure 7 demonstrates the latency analysis of the proposed model based on offloading and non-offloading mechanisms. The analysis shows that the suggested offloading model minimizes latency because of the optimal server selection and delay-aware queuing design.

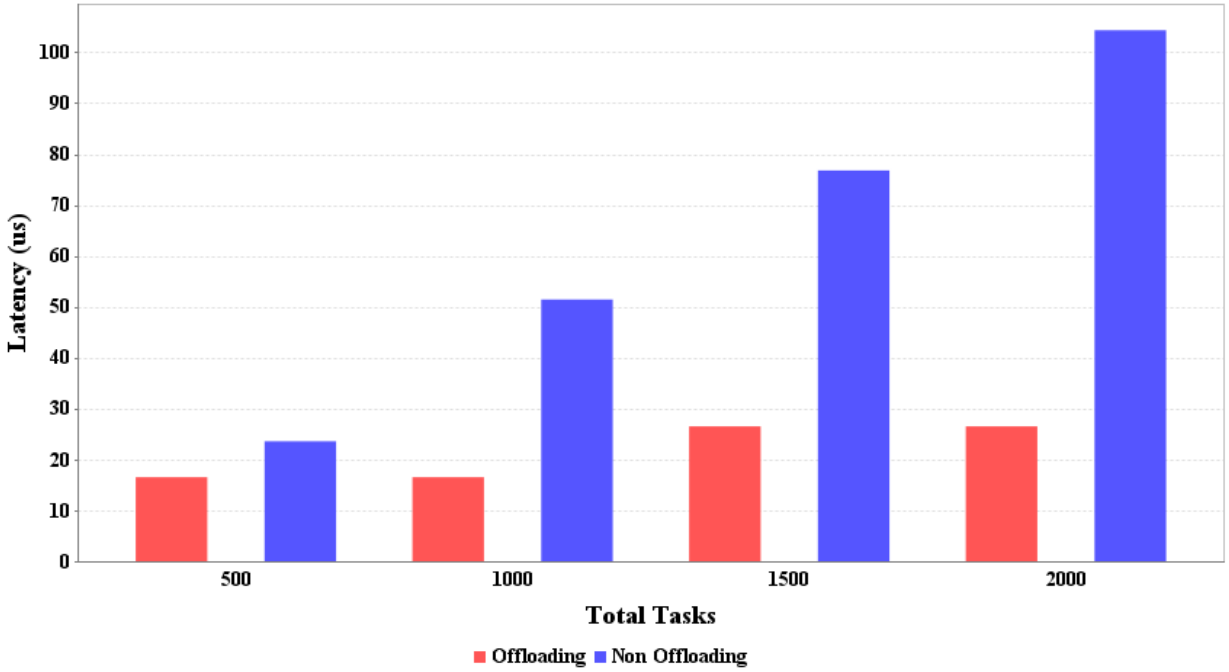

Figure 7. Analysis based on Latency

The ablation study of the proposed method is presented in Figure 8. The analysis demonstrates the superiority of the proposed model with a minimal task rejection ratio. Let the analysis with 900 tasks; the proposed DFQM- Fuzz + ICOA - GO method acquired the task rejection ratio of 69. Still, the proposed method with criteria 1 accomplished the task rejection ratio of 144. Criteria 1 indicates that the proposed model offloads HU and U tasks only via fog and NU and ND tasks only via the cloud. The DFQM – Fuzz + COA yielded a task rejection ratio of 154, and the genetic operators are not included in the proposed task offloading model. Thus, the incorporation of genetic operators into the COA aids the proposed technique in minimizing task rejection rates by selecting the optimal server without local optimal trapping. Furthermore, ICOA-GO (proposed model without fuzzy-based task prioritization) yielded a task rejection ratio of 167. Thus, the ablation analysis demonstrates the superiority of the proposed model with fuzzy-based priority scheduling and optimal server selection.

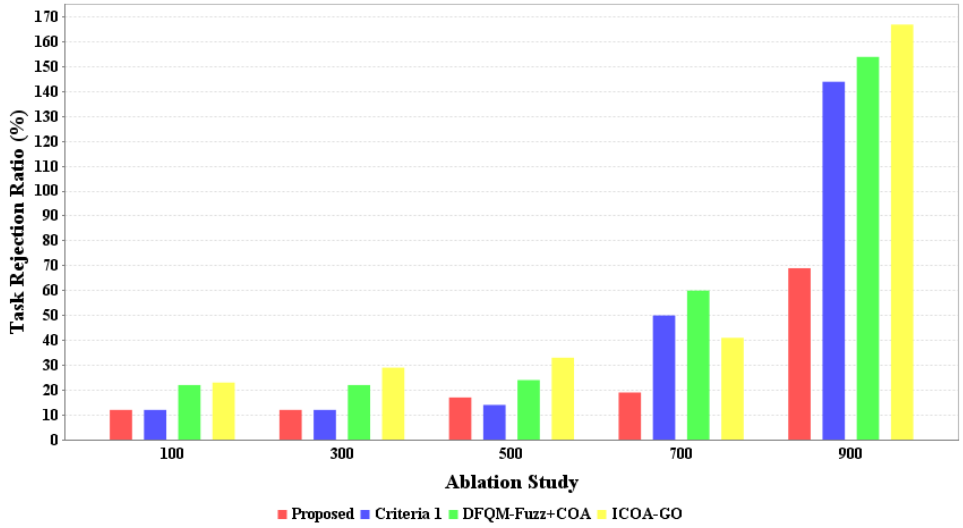

Figure 8. Ablation Study

Comparative Discussion

The comparative discussion based on the best case is portrayed in Table 9. Here, the minimal energy utilized by the proposed method is 93.589, which is 57.94%, 83.81%, 89.20%, and 94.60% compared to conventional WOA, BAT, RR, and ROP methods. Here, the minimal Task Rejection Ratio by the proposed method is 1, which is 90.91%, 93.33%, 95.83%, and 97.44% compared to conventional WOA, BAT, RR, and ROP methods. Here, the minimal makespan by the proposed method is 1.96073, which is 41.80%, 59.73%, 79.05%, and 87.21% compared to conventional WOA, BAT, RR, and ROP methods.

Table 9. Comparative Discussion

Metrics/ Methods	WOA	BAT	RR	ROP	Proposed
Energy Utilization	222.523	577.97	866.955	1733.91	93.589
Task Rejection Ratio	11	15	24	39	1
Makespan	3.36915	4.86923	9.35894	15.3327	1.96073

Here, the analysis indicates the superiority of the proposed method compared to the existing methods.

CONCLUSION

This paper introduced a novel task offloading model for the GIP system using delay-aware prioritization and optimal server selection. In this case, the DFQM-Fuzz model is used to prioritize the incoming task offloading. The parameters like task size, arrival time, and delay are considered by the fuzzy model to prioritize the incoming task. Here, prioritization assists the model in eliminating task rejection by crossing the deadline. For efficient task offloading, the optimal server selection is employed using the ICOA-GO algorithm. The proposed ICOA-GO algorithm incorporates the genetic operators to acquire the best global solution. The optimal best server selection is employed using the ICOA-GO by considering energy consumption, task execution cost, and task execution delay as its fitness for efficient task offloading. The analysis based on Energy Utilization, Task Rejection Ratio, and Makespan acquired the values of 93.589, 1, and 1.96073, respectively.

LIMITATIONS AND FUTURE RESEARCH

Integrating the DFQM-Fuzz and ICOA-GO improves task prioritizing and offloading efficiency while reducing execution time, energy consumption, and cost. The reliance on fog servers with limited resources poses challenges when high task density is considered, which leads to computational bottlenecks. Future enhancements should be

made to the GIP model by integrating advanced AI techniques for dynamic task scheduling, supporting real-time adaptability and scalability.

AUTHORS BIOGRAPHIES

1Mr. Sarkarsinha H. Rajput, completed his B.E. from D. N. Patel College of Engineering, Shahada Dist: Nandurbar (M.S.) and M.E. from SSBT's College of Engineering and Technology, Bambhori, Jalgaon (M.S.). He has been working as Assistant Professor in SSBT's College of Engineering and Technology since 2009. He is pursuing a Ph.D. in Computer Science & Engineering from Kavayitri Bahinabai Chaudhari North Maharashtra University, Jalgaon (M.S.). His areas of interest are Fog Computing, Cloud Computing, and Machine Learning.

2Dr. Manoj Eknath Patil has completed PhD degree in Computer Science & Engineering from Jodhpur National University, Jodhpur, Rajasthan. Currently, he is working as an Associate Professor & Head, at the Department of Computer Engineering, SSBT's College of Engineering & Technology, Jalgaon (M.S.) and recognized PhD Guide in Kavayitri Bahinabai Chaudhari North Maharashtra University, Jalgaon (M.S.). He has 39 research papers in reputed peer-reviewed journals in addition to 20 papers in International Conferences to his credit. He is a Life Member of ISTE. His research interests are Wireless Sensor Networks, Web Security, Cloud Computing, Fog Computing.

REFRENCES

[1] Abdulazeez, D. H. & Askar, S. K. (2023) Offloading mechanisms based on reinforcement learning and deep learning algorithms in the fog computing environment. Ieee Access, 11, 12555-12586.

[2] Angel, N. A., Ravindran, D., Vincent, P. D. R., Srinivasan, K. & Hu, Y.-C. (2021) Recent advances in evolving computing paradigms: Cloud, edge, and fog technologies. Sensors, 22(1), 196.

[3] Azizi, S., Shojafar, M., Abawajy, J. & Buyya, R. (2022) Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach. Journal of network and computer applications, 201, 103333.

[4] Bai, W., Ma, Z., Han, Y., Wu, M., Zhao, Z., Li, M. & Wang, C. (2021) Joint optimization of computation offloading, data compression, energy harvesting, and application scenarios in fog computing. IEEE Access, 9, 45462-45473.

[5] Chauhan, N., Banka, H. & Agrawal, R. (2021) Delay-aware application offloading in fog environment using multi-class Brownian model. Wireless Networks, 27(7), 4479-4495.

[6] Das, R. & Inuwa, M. M. (2023) A review on fog computing: issues, characteristics, challenges, and potential applications. Telematics and Informatics Reports, 10, 100049.

[7] Deb, P. K., Misra, S. & Mukherjee, A. (2021) Latency-aware horizontal computation offloading for parallel processing in fog-enabled IoT. IEEE Systems Journal, 16(2), 2537-2544.

[8] Dehghani, M., Montazeri, Z., Trojovská, E. & Trojovský, P. (2023) Coati Optimization Algorithm: A new bio-inspired metaheuristic algorithm for solving optimization problems. Knowledge-Based Systems, 259, 110011.

[9] Deng, X., Yin, J., Guan, P., Xiong, N. N., Zhang, L. & Mumtaz, S. (2021) Intelligent delay-aware partial computing task offloading for multiuser industrial Internet of Things through edge computing. IEEE Internet of Things Journal, 10(4), 2954-2966.
[10] Gasmi, K., Dilek, S., Tosun, S. & Ozdemir, S. (2022) A survey on computation offloading and service placement in fog computing-based IoT. the Journal of Supercomputing, 78(2), 1983-2014.
[11] Hussain, M. M. & Beg, M. S. (2021) CODE-V: Multi-hop computation offloading in Vehicular Fog Computing. Future Generation Computer Systems, 116, 86-102.
[12] Jazayeri, F., Shahidinejad, A. & Ghobaei-Arani, M. (2021) A latency-aware and energy-efficient computation offloading in mobile fog computing: a hidden Markov model-based approach. The Journal of Supercomputing, 77, 4887-4916.
[13] Kaur, P. & Mehta, S. (2022) Improvement of Task Offloading for Latency Sensitive Tasks in Fog Environment. Energy Conservation Solutions for Fog-Edge Computing Paradigms, 49-63.
[14] Keshavarznejad, M., Rezvani, M. H. & Adabi, S. (2021) Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms. Cluster Computing, 1-29.
[15] Laroui, M., Nour, B., Moungla, H., Cherif, M. A., Afifi, H. & Guizani, M. (2021) Edge and fog computing for IoT: A survey on current research activities & future directions. Computer Communications, 180, 210-231.
[16] LEE, H. S., SIA, B. K., CHONG, S. C. & LOW, C. W. (2020) Investment in Research & Development or Size Expansion? The Case of Internet of Things Companies, 2020 IEEE 8th Conference on Systems, Process and Control (ICSPC). IEEE.
[17] Mazumdar, N., Nag, A. & Singh, J. P. (2021) Trust-based load-offloading protocol to reduce service delays in fog-computing-empowered IoT. Computers & Electrical Engineering, 93, 107223.
[18] Meena, V., Gorripatti, M. & Suriya Praba, T. (2021) Trust enforced computational offloading for health care applications in fog computing. Wireless Personal Communications, 119(2), 1369-1386.
[19] Mishra, K., Rajareddy, G. N., Ghugar, U., Chhabra, G. S. & Gandomi, A. H. (2023) A collaborative computation and offloading for compute-intensive and latency-sensitive dependency-aware tasks in dew-enabled vehicular fog computing: A federated deep Q-learning approach. IEEE Transactions on Network and Service Management, 20(4), 4600-4614.
[20] Panda, S. K., Pounjula, T., Ravirala, B. & Taniar, D. (2023) An Energy, Delay and Priority-Aware Task Offloading Algorithm for Fog Computing.
[21] Razaq, M. M., Tak, B., Peng, L. & Guizani, M. (2021) Privacy-aware collaborative task offloading in fog computing. IEEE Transactions on Computational Social Systems, 9(1), 88-96.
[22] Sabireen, H. & Neelanarayanan, V. (2021) A review on fog computing: Architecture, fog with IoT, algorithms and research challenges. Ict Express, 7(2), 162-176.
[23] Salehnia, T., Seyfollahi, A., Raziani, S., Noori, A., Ghaffari, A., Alsoud, A. R. & Abualigah, L. (2024) An optimal task scheduling method in IoT-Fog-Cloud network using multi-objective moth-flame algorithm. Multimedia Tools and Applications, 83(12), 34351-34372.
[24] Sivanandam, S., Deepa, S., Sivanandam, S. & Deepa, S. (2008) Genetic algorithm optimization problems. Introduction to genetic algorithms, 165-209.
[25] Tran-Dang, H. & Kim, D.-S. (2021) FRATO: Fog resource based adaptive task offloading for delay-minimizing IoT service provisioning. IEEE Transactions on Parallel and Distributed Systems, 32(10), 2491-2508.
[26] Tran-Dang, H. & Kim, D.-S. (2023) Dynamic collaborative task offloading for delay minimization in the heterogeneous fog computing systems. Journal of Communications and Networks, 25(2), 244-252.
[27] Yu-Jie, S., Hui, W. & Cheng-Xiang, Z. (2022) Balanced computing offloading for selfish IoT devices in fog computing. IEEE Access, 10, 30890-30898.