IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 September 2024, accepted 27 October 2024, date of publication 30 October 2024, date of current version 8 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3488032

RESEARCH ARTICLE

Multi-Objectives Firefly Algorithm for Task Offloading in the Edge-Fog-Cloud Computing

FATEN A. SAIF[1], ROHAYA LATIP[2], (Member, IEEE), ZURINA MOHD HANAPI[2], (Member, IEEE), SHAFINAH KAMARUDIN[2], A. V. SENTHIL KUMAR[3], AND AWADH SALEM BAJAHER[2]

[1]Department of Information Technology, Gulf Colleges, Hafar Al Batin 39952, Saudi Arabia
[2]Department of Communication Technology and Network, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia
[3]Department of MCA, Hindusthan College of Arts & Science, Coimbatore 641028, India

Corresponding authors: Faten A. Saif (f-saif500@hotmail.com), Rohaya Latip (rohayalt@upm.edu.my), and Shafinah Kamarudin (shafinah@upm.edu.my)

This work was supported in part by University Putra Malaysia, and in part by the Ministry of Education (MOE) Malaysia.

ABSTRACT This study proposes a two-stage strategy to ensure the successful execution of critical tasks. In the first stage, an Enhanced Task Offloading (ETO) algorithm is introduced to determine the appropriate computing layer—edge, fog, or cloud—for offloading incomplete tasks. The algorithm makes this decision by assessing the availability of idle computing resources relative to the task's computational requirements. Additionally, it verifies the status of the server (on/off) before offloading; if the server is unavailable, the algorithm proceeds to check the next layer. In the second stage, the strategy employs a Multi-objective Firefly (MFA) algorithm to assign the optimal computational device within the selected layer. Experimental simulations compare the proposed strategy with a benchmark task offloading algorithm. The results demonstrate the superiority of the proposed strategy, achieving reductions in energy consumption and delay and maximizing resource utilization compared to the baseline algorithms.

INDEX TERMS Task offloading, optimization, firefly algorithm, metaheuristic, edge-fog-cloud computing, Internet of Things, energy consumption, transmission delay, Pareto.

I. INTRODUCTION

The Information Technology (IT) sector has experienced significant advancements, and it is projected that over 50 million Internet of Things (IoT) devices will be connected to the internet in the coming years [1]. Moreover, the widespread availability of steady, high-speed internet has facilitated the production of a wide variety of Internet of Things (IoT) devices, which have dramatically increased over the past decade. These devices include wearable technology, wireless sensors, and innovative IoT applications. On one hand, applications such as facial recognition, natural language processing, and augmented reality generate vast amounts of data [2].

This shift has led to the development of edge computing, a distributed computing paradigm that brings data processing and computation closer to the source where data is generated or utilized. By minimizing the need to transmit all data to centralized cloud servers, edge computing reduces latency, enables real-time data analysis, and conserves bandwidth. Notable applications of edge computing include smart homes, autonomous vehicles, retail environments, telecommunications, environmental monitoring, and industrial IoT. This technology enhances efficiency, responsiveness, and reliability across various industries and use cases [3].

However, despite the advanced capabilities of smart devices, they are still unable to efficiently handle the computational demands of emerging IoT applications, such as smart healthcare, virtual reality, and smart transportation. These applications require substantial processing power, which exceeds the capabilities of many devices. For example,

The associate editor coordinating the review of this manuscript and approving it for publication was Zhengmao Li.

running IoT applications on computers or mobile phones often encounters limitations due to restricted battery life, processing power, and storage capacity [4]. These limitations make it challenging to process IoT applications on mobile devices, necessitating the use of alternative resources to complete the operations. To overcome these restrictions and meet the requirements of computation-intensive, delay-sensitive tasks, offloading tasks to cloud computing is essential. Cloud computing, as a centralized platform with abundant computational resources, enables the remote processing and storage of complex tasks [5].

Despite the significant benefits of cloud computing, several limitations persist. Cloud data centres may struggle to handle the continuous requests from IoT devices due to the considerable distance between the cloud and end-user devices. Transmitting large volumes of data over the internet to remote cloud data centres consumes significant bandwidth and results in network delays [6]. Fog computing addresses the shortcomings of cloud computing but does not replace it. Instead, it serves as a bridge between IoT devices and cloud computing. Fog computing consists of multiple heterogeneous devices, such as base stations, routers, switches, and surveillance cameras, which are ubiquitously connected and can be deployed in locations like power poles, vehicles, and commercial centers. The primary role of fog computing is to process simple, delay-sensitive tasks that require immediate responses, leveraging the shorter distance between IoT devices and fog nodes. These decentralized devices, positioned at the edge of the network, provide real-time processing of raw data from sensors. Therefore, task offloading plays a crucial role in ensuring efficient task processing [7].

ask offloading is a decision-making process that determines where the tasks of IoT applications should be offloaded—either to a specific fog node or cloud server—based on the application's requirements and the availability of resources on the nodes. It involves transferring certain tasks remotely to fog nodes or cloud servers to address the challenge of running resource-intensive applications on devices with limited resources. Once the tasks are executed remotely, the results are sent back to the terminal devices to enable the IoT applications to function [8].

However, the dynamic nature of the IoT environment presents various challenges that must be addressed. One significant issue is the offloading of tasks when a resource node leaves the system due to technical failure or power outage. For instance, in an intensive care unit, patients' vital signs must be monitored in real-time, with notifications sent to doctors to provide immediate medical attention in emergencies. If a sensor device shuts down, it could jeopardize patients' lives. Additionally, offloading tasks to the cloud can reduce energy consumption on users' devices but may introduce transmission delays due to the long distance between IoT devices and cloud data centers. Conversely, offloading tasks to fog nodes minimizes transmission delays due to their proximity but increases the energy consumption of users' devices. Therefore, optimizing both delay and energy consumption is essential to improving task offloading efficiency.

As a result, researchers and developers have focused on enhancing various strategies for task offloading in cloud-fog environments to enable the remote execution of IoT applications across distributed resources, such as fog nodes and cloud servers, rather than relying on terminal devices. Task offloading has thus emerged as an optimization problem, typically addressed using two common approaches: optimal and heuristic. Optimal approaches provide the best possible solutions but often require longer execution times due to their high complexity, making them suitable primarily for small-scale problems. In contrast, heuristic approaches are less complex, ensure minimal execution time, reduce energy consumption, and are better suited for distributed computing environments. Additionally, heuristic approaches offer near-optimal solutions [9].

Various techniques have been proposed for offloading tasks to appropriate computational devices based on conflicting Quality of Service (QoS) parameters, such as energy consumption, delay, cost, and makespan. Optimizing multiple conflicting objectives falls under the category of Multi-objective Optimization Problems (MOPs), which are addressed by identifying a set of trade-off solutions, known as the Pareto front. There are two common strategies for solving MOPs: the first involves finding a set of Pareto-optimal solutions in a single run, while the second utilizes a non-Pareto approach based on decomposition methods [10] Both approaches aim to find optimal solutions for MOPs. Recently, many studies have focused on employing meta-heuristic algorithms to solve MOPs due to their lower complexity, which results in reduced execution time [11]. The Firefly algorithm, while a well-known optimization method, has yet to be fully explored for task offloading in solving MOPs within cloud-fog computing environments.

The Firefly algorithm is a meta-heuristic optimization technique within the field of swarm intelligence, developed by Yang. It mimics the swarming and flashing behaviours of tropical fireflies [9]. The Firefly Algorithm (FA) is a high-level, problem-independent method that incorporates a set of strategies and rules for finding solutions. Additionally, FA excels at handling multimodal functions more efficiently and intuitively [12]. The strategy of the Firefly Algorithm (FA) is based on fireflies being attracted to others with higher flash intensity values. Fireflies can divide and regroup into subgroups due to the stronger attraction between nearby fireflies compared to those at longer distances. This allows each subgroup to converge around a local optimum. Since light intensity decreases with distance, the attraction between fireflies can be either local or global, depending on the absorption coefficient.

The main advantage of the FA algorithm is its speed compared to other meta-heuristic algorithms, ensuring minimal execution time and energy consumption. For instance, FA outperforms the PSO algorithm in terms of nearby

attraction and automated regrouping, making FA popular due to its accuracy and fast convergence in finding solutions [13]. The Firefly Algorithm (FA) enhances convergence speed due to its global search capabilities. Moreover, as iterations progress, FA reduces problem randomness, leading to more precise results. The goal of convergence in optimization algorithms is to achieve global or near-global solutions, often providing better outcomes than traditional techniques [14]. optimum, rather than making large jumps as seen in genetic algorithm techniques. This approach ensures a higher likelihood of obtaining optimal solutions [12].

The Firefly Algorithm (FA) is traditionally designed for solving single-objective problems, but its inherent features make it well-suited for addressing Multi-objective Optimization Problems (MOPs) as well [15].

This study extends the existing research by addressing the challenge of checking resource availability before task transfer and selecting the optimal computing device to maximize resource utilization. It proposes a Multi-objective Firefly Algorithm (MFA) as a solution for task offloading in cloud-fog computing. The MFA algorithm ensures that unfinished tasks are transferred to a suitable layer, allowing them to be completed on another node when the resource node leaves the system. The primary objectives are to reduce delay and energy consumption.

The main contributions of this study are as follows:

- A mathematical framework with queue theory which is a renowned and extensively utilized method to manage the flow of data within a network and in this study was developed to reduce power consumption and transmission delay via efficient task offloading.
- Enhanced Task offloading to select the suitable among computing layer among edge-fog-cloud for offloading unfinished tasks based on task computing unit.
- A Multi-objectives Firefly (MFA) algorithm is proposed for offloading unfinished tasks to another suitable computational resource in the selected layer by finding the optimal computation device instead of randomly selecting. The main objectives of the MFA algorithm are to reduce the transmission delay and energy consumption during tasks offloading in cloud-fog computing.
- The simulation results demonstrate that the proposed algorithm outperforms the Task offloading and PSO algorithms in reducing delay and energy consumption by 25% and 23%, respectively and maximizing the resource utilization by 86%.

The remainder of this paper is organized as follows: The ''Cloud-Fog Cooperation System'' section presents a mathematical framework of a fog-cloud cooperation system using queueing theory. The ''Description of Delay and Energy Consumption'' section outlines the functions of delay and energy consumption across the three layers (Edge-Fog-Cloud). The ''Task Offloading Algorithm'' section explains the strategy for selecting the appropriate layer for task offloading. The ''Multi-objective Firefly (MFA) Algorithm'' section details how to select suitable computational devices to maintain QoS by reducing delay and energy consumption. The ''Simulation Settings and Results'' section provides a brief description of the simulation process and outcomes. Finally, the ''Conclusion'' section summarizes the study's main findings, limitations, and suggestions for future work.

II. RELATED WORK

Several studies have been conducted to address the task offloading problem in cloud-fog computing systems. This section reviews the strengths and weaknesses of existing offloading strategies in a fog-cloud environment, focusing on various QoS parameters [16]. This study addresses the problem of task offloading when a node leaves the system by transferring unfinished tasks to another node based on the idle rate. The main goal is to reduce delay and energy consumption in edge-fog-cloud computing by proposing a task offloading algorithm. However, the study only focuses on offloading tasks according to the computation unit.

Where in [17] A joint energy-efficient task assignment (JEETA) has been proposed, utilizing the dynamic application-partitioning algorithm (DAPTS) to determine whether tasks should be offloaded in heterogeneous cloud-fog computing environments while aiming to reduce energy consumption. However, the proposed algorithm resulted in increased response time due to the distribution of computation across different locations. Regard to [18] a new Microservice Container Fog System (MSCFS) framework has been proposed to address the challenges of running mobility and delay-sensitive applications with minimal cost. Additionally, the Cost-Aware Computational Offloading and Task Scheduling (CACOTS) framework was introduced to solve the task scheduling problem through multiple phases, including task sequencing, resource matching, and scheduling. However, the proposed algorithm could be further improved by considering transient failures.

In contrast, [3] The problem of computation offloading and task scheduling for DNN-based applications in cloud-fog environments has been formulated using greedy and genetic algorithm-based approaches, with the aim of reducing delay. However, this strategy suffers from high complexity, which increases execution time and, consequently, energy consumption.

Correspondingly, [19] The problem of computation offloading in cloud-fog computing has been addressed by proposing a method that jointly optimizes offloading decisions using Successive Convex Approximation (SDR) and random extraction for decision-making, alongside the allocation of computational resources. This approach aims to minimize the maximum weighted cost of delay and energy consumption (EC) across all user equipment (UEs), a mixed-integer non-linear programming problem. However, this strategy, classified as an exact approach, consumes high energy and does not fully meet the requirements on the fog node side.

Additionally, [20] The resource scheduling and task offloading problem in hybrid cloud-fog computing has been addressed by proposing two efficient algorithms based on a deep reinforcement learning framework. These algorithms aim to reduce the weighted sum of energy, time, and rent cost (ETRC). However, the adoption of artificial intelligence increases the complexity and execution time compared to meta-heuristic approaches.

In line with, [5] proposes a three-tier Cloud of Things (CoT) architecture and a task offloading mathematical model to manage incoming tasks from IoT devices. The model determines whether to offload tasks to the fog, cloud, or collaboratively to both based on execution time, delay, and energy consumption. However, the presence of a gateway increases energy consumption and system overload.

Similarly, [21] The computation offloading process in cloud-fog computing has been investigated by proposing an optimization strategy that models the interactions of the offloading process based on microeconomic theory, aiming to minimize energy consumption. However, this approach results in long execution times for offloading decisions.

Indeed, [22] a new intelligent computation offloading strategy has been proposed using the MEC (Mobile Edge Computing) architecture and adopting artificial intelligence (AI) technology in cloud-fog computing. The prediction strategy is based on the LSTM (Long Short-Term Memory) algorithm to offload tasks and reduce task delay. However, it does not implement fine-grained caching, offloading, and scheduling of computation tasks, which could further improve the efficiency of computation offloading.

Regard to [2] a new task offloading approach has been proposed to optimally decide when and where to offload tasks—either to a fog node or a cloud server—to reduce delay, maximize task execution, and balance the load, using a Q-learning-based algorithm. However, the approach becomes less efficient as the number of fog nodes exceeds 15, leading to increased system complexity.

As mentioned in [14] a sustainable infrastructure for fog-cloud computing has been proposed to process resource-intensive and delay-sensitive applications. The strategy involves offloading tasks using the Firefly Algorithm to identify the optimal computational device based on computational time and energy consumption, while also managing delay. However, the approach does not account for task prioritization during offloading.

As others have highlighted [23] A game-theoretic approach has been proposed for the computation offloading decision-making problem across multiple mobile terminal devices in mobile-edge cloud computing, based on Nash equilibrium. However, this study does not account for QoS parameters or address the issue of mobile users leaving during the computation process.

As reported in [24] the problem of reducing makespan while maintaining the heterogeneity of cloud-fog computing has been addressed by formulating it as a Mixed Integer Linear Programming (MILP) model and proposing an optimal approach using the Logic-Based Benders Decomposition (LBBD) principle in cooperation with MILP. However, LBBD may not be the most efficient option for finding the optimal solution, as increasing communication overhead can reduce overall efficiency.

Referring to [25] an Optimal Joint Data Center (DC), Offloading, and Resource Allocation (JCORA) strategy has been proposed for joint DC and computation offloading in hierarchical fog-cloud computing to minimize the maximum weighted energy and service delay cost (WEDC). However, this strategy has high complexity, leading to increased energy consumption. Even more, computational load incurred. According to a study in [26] the problem of hybrid computation offloading has been investigated, focusing on maintaining the various computational and communication capabilities of two offloading destinations to reduce both communication and computation energy consumption while ensuring the completion of tasks within delay constraints. However, more QoS parameters should be considered to effectively evaluate the overall performance.

As mention in [27] Computation offloading in mobile-cloud-edge environments, involving multiple Wireless Devices (WDs) equipped with energy harvesting capabilities to collect renewable energy from the environment, has been addressed. However, the strategy incurs longer execution times during task scheduling, and the overall system utility converges once the energy arrival rate reaches a certain threshold.

In [28] a relative value algorithm was designed to determine the optimal task offloading scheme and improve the total long-term reward of the Vehicular Fog-Cloud Computing (VFCC) system. The study formulated the task offloading problem in the VFCC system as a semi-Markov decision process (SMDP) to enhance the system's long-term performance. However, the study could be improved by addressing the periodic arrival of vehicles. Even more, [29] the study explored computation offloading in cloud-fog computing with non-orthogonal multiple access (NOMA) and addressed resource allocation by proposing an integrated fog-cloud approach. This approach offloads tasks to nearby fog nodes or cloud centers cooperatively to reduce energy costs and delay. However, the study did not optimize resource utilization, leading to increased energy consumption as usage increased.

As mention in [30] the problem of task offloading in cloud-fog computing has been discussed, with a proposed strategy based on fuzzy logic algorithms. This approach considers task requirements (e.g., CPU demand, network demand, and delay sensitivity), resource utilization, and resource heterogeneity. The strategy also addresses task scheduling in cloud-fog computing, focusing on factors such as average processing delay, network delay, service time, VM utilization, and task failure. However, assigning priority to large computational resources leads to increased energy consumption.

In [31] the study focused on computation offloading in cloud-fog computing, taking into account the diverse com-

putational, communication, and security capabilities. The proposed algorithms, MO and GO, aim to increase the cloud-fog provider's revenue without compromising performance or security requirements, while also reducing costs. However, there is resource wastage when extra CPU is allocated to offload tasks to servers with higher security levels.

In contrast, [32] the study discussed computation offloading in marine vehicular cloud-fog computing for Unmanned Surface Vehicle (USV) clusters. The proposed mechanism is an optimized learning-based computation task offloading approach, namely the Adaptive Upper Confidence Bound (AUCB) algorithm, based on Multi-Armed Bandit (MAB) theory, aimed at reducing the average computation task offloading delay. However, it requires further optimization to decrease the cost of exploration and achieve balanced computing performance.

In line with [33] the study formulated resource allocation and computation offloading as time cost and energy minimization problems, addressed by the proposed ETCORA algorithm. However, it did not fully leverage resource utilization.

Regard to [34] the study investigated the placement of Virtual Machines (VMs) in cloud-fog computing to reduce energy consumption. However, it should also consider resource utilization to further improve system performance.

Where in [35] the study formulated the energy-efficient computation offloading and dynamic resource scheduling (eoDS) problem, addressing it through the proposed eoDS algorithm to reduce completion time and energy consumption. However, the approach did not take resource utilization into account.

Referring to [36] the study discussed the problem of dynamic task offloading, focusing on deciding where tasks should be offloaded based on application requirements, computation needs, and data to select the appropriate resources. A machine learning-regression algorithm was proposed to address this. However, the algorithm has high complexity, leading to longer execution times compared to other algorithms.

Reference [37] an optimal joint offloading scheme based on resource occupancy prediction has been proposed to address the problem of computation offloading with limited edge resources. The scheme aims to minimize the average task delay and reduce the task offloading failure rate. However, when processing tasks in real-time, the CPU frequently performs a series of data fetch operations (reading and processing packets) initiated by parallel threads. This can result in the CPU becoming a bottleneck in overall computing performance.

Most recent studies on task offloading have primarily focused on either fog-cloud or edge-cloud computing. However, the few studies that address edge-fog-cloud computing often overlook the varying nature of each platform when considering optimization objectives. Hence, offloading tasks to fog computing will reduce the delay due to the short distance to the IoT devices while increasing the consumption of users' energy. In contrast, offloading tasks to the cloud will reduce the consumption of users' energy but increase the transmission delay due to the extended distance from the cloud data center to the IoT devices. As a result, offloading tasks across different platforms simultaneously presents significant challenges in optimizing conflicting objectives.

Even more, most studies solved the problem of task offloading through an optimal approach. Whereas the optimal approach is highly complex, increasing the execution and energy consumption. Therefore, the heuristic approach is consistent approach with the aim of this study, which is concerned with reducing delay and energy consumption in cloud-fog computing.

Based on the points discussed, this study has selected the Firefly Algorithm (FA) over other heuristic algorithms due to its ability to quickly obtain optimal solutions during task offloading. Additionally, FA is capable of efficiently addressing complex optimization problems that involve multiple conflicting objectives [12]. Therefore, this research proposes an improved Multi-Objective Firefly Algorithm (MFA) to optimize the conflicting objectives in offloading tasks across edge-fog-cloud computing using a multi-objective optimization approach, see Table 1.

III. SYSTEM MODEL DESCRIPTION

The cloud-fog system architecture in Figure 1 was adopted [16] because of its simplicity and feasibility of implementation. The system comprises M end devices that communicate with each other via wireless links, F fog nodes, and C cloud servers. Terminal devices communicate with each other via a wireless channel. Fog nodes communicate directly with the terminal devices. The role of fog nodes is to provide services to end-user devices. Cloud servers are responsible for fog nodes. Whereas, the traffic model captures a variant of the power and capacity.

The modeling of the entire cloud-fog computing system using Poisson processes that are utilized in scenarios where when the continuous occurrences of specific events which appear to occur at the specific rate, but randomly. Also, the wireless channels between terminal devices and fog required an exponentially distributed [38].

The traffic model follows an M/M/1 queue at the end devices, M/M/C queue at the fog node, and M/M/∞ queue at the cloud server. The task is characterized by the number of tasks, length of the task input I_t, task deadline d_t, flag of task execution u_t, and required computing unit by task ψ_t. See Table 2 for the primer notations.

IV. DELAY DESCRIPTION AND ENERGY DESCRIPTION

This section to describe the main equations of energy consumption and delay on (Edge-Fog-Cloud) that implemented in the study [14].

A. EDGE COMPUTING

It refers to End user devices and the study assume the service rate as μ of the end-user device i follows an exponential distri-

TABLE 1. Summary of notations.

Symbol	Definition
P^i_{ed}, P^j_{fog}, P^K_{cloud}	Power consumption of end device *i*, fog node j, cloud server k (a unit of power)
D^i_{ed}, D^j_{fog}, D^k_{cloud}	Delay of end device *i*, fog node j, cloud server k (a unit of time)
P_{total}, D_{total}	Total power and delay of the system
λ	Arrival rate of task (Request/ unit of time)
μ	Service rate of task (Request/ unit of time)
X^i_{ed}, Y^j_{fog}, Z^k_{cloud}	The assigned workload to end device *i*, fog node j, cloud server k. (request)
X^{min}_{ed}, Y^{min}_{fog}, Z^{kmin}_{cloud}	Minimum workload of end device layer, fog node layer, cloud server layer.

bution, with an M/M/1 task queue. In addition, the generation of tasks from the end device is based on at Poisson process with an average arrival rate λ. P_{ed} is the power of end-device *i* and T_{ed} is its processing time. The power consumption of X^i_{ed} for the task's execution at the end device calculated by [16].

$$P^i_{ed} \triangleq T_{ed} \times P_{ed} = \frac{X^i_{ed}}{\mu - \lambda} \times P_{ed} \tag{1}$$

We consider computing latency because tasks performed on mobile terminal devices have little communication delay. As deduced from queue theory, the delay is described as

$$D^i_{ed} \triangleq \frac{\lambda}{\mu\,(\mu - \lambda)}. \tag{2}$$

B. FOG COMPUTING

The task queue in fog node j is modelled as M/M/C. The power consumption reflects the amount of computation, which is a monotonically increasing and strictly convex function. Quadratic and piecewise linear functions are two alternatives to this function [39] Fog nodes can flexibly adapt to any function of energy consumption as long as they meet these two attributes:1) there is a direct relationship; that is, increasing energy consumption increases the computation amount. 2) The power consumption margin increases for each fog device. The power energy expression P^j_{fog} of the fog node is related to the workload Y^j_{fog} as follows:

$$P^j_{fog} \triangleq a{Y^j_{fog}}^2 + bY^j_{fog} + c \tag{3}$$

where a > 0 and b and, c ≥ 0 are pre-determined parameters.

The fog node *j* consists of both communication and computing delays. The computing delay D^{com}_{fog} is related to waiting time. Using queue theory, we can express the computing delay as follows:

$$D^{com}_{fog} \triangleq \frac{Q_L}{\lambda} \times Y^j_{fog} \tag{4}$$

where Y^j_{fog} is the workload allocated to fog node j and. Q_L is the average queue length. As a result of task execution at the fog node, communication is related to the input length of the tasks. The communication delay D^{comm}_j is expressed as follow

$$F_{comm}(I_t) \triangleq \begin{cases} yI_t & u_t \epsilon cloud \\ \varepsilon I_t & u_t \in fog \end{cases} \tag{5}$$

where I_g is the input length of the task t ($\gamma \gg \varepsilon$). Therefore, the communication delay of the fog node is $D^{comm}_{fog} = \varepsilon I_g$. The fog node delay is composed of computing and communication delays, which can be expressed as follows:

$$D^j_{fog} \triangleq = D^{com}_{fog} + D^{comm}_{fog} \tag{6}$$

C. CLOUD COMPUTING

For cloud server k, the task queue is modelled as an M/M/∞ queue. Assuming that every cloud server has several homogeneous computing machines and that the CPU frequency of all machines is equal, this implies that the energy consumption for all servers is the same. The approximate power consumed by every machine on cloud server k can be obtained by utilizing the frequency of the CPU machine function. f_k:$A_k Z^k_{cloud} + B_k$, where A_k and B_k are positive constants [40] Assigning more workload to the cloud server implies more power-on. Whenever the assigned workload decreases, some cloud servers are turned off to save energy. The power consumption of the cloud server P^k_{cloud} is related to the on/off state of the machine.

$$P^k_{cloud} \triangleq \sigma_k n_k (a_k Z^k_{cloud} + b_k), \tag{7}$$

where a_k and b_k are the positive constants. σ_k indicates the on/off state of cloud server k, where 1 denotes the cloud server on and 0 indicates its off state. n_k denotes the number of on-state machines on the cloud server. Owing to the heavy computational resources of cloud servers, the computing delay can be assumed to be negligible; thus, the delay is the communication delay that defines as:

$$D^k_{cloud} \triangleq \gamma I_i \tag{8}$$

V. ENHANCES TASK OFFLOADING ALGORITHM

According to the dynamic nature of the IoT network, one computing may leave the system due to the power-off. Thus, offloading tasks to another computational device is mandatory to guarantee the execution of the unfinished tasks.

The main challenge is completing the uncompleted tasks in the queue. This research has solved this problem in to two stages. First, selecting suitable computing layer using Enhanced Task Offloading algorithm which extend from [16]. Second, after selecting the layer, the MFA algorithm will choose the optimal computational device, as shown in

Figure 2 The Enhance Task Offloading algorithm is summarized in Algorithm 1 that offload unfinished tasks to the suitable layer.

$$C_{idle}^{j} = (1 - \delta_i)C_i \tag{9}$$

where δ_i is the current CPU utilization rate, C_i is the total computing cell of the devices, u_t indicates the task execution flag where 1,2, and 3 refer to Md, FN, CS respectively.

Assuming there are n_1 Edge computing and n_2 Fog computing. In case the tasks will offload to the computing in the belonging layer. First, it must satisfy the conditions; 1) check the status of computing where $\sigma_k = (1, 0)$ to indicates the on/off power. 2) The idle computing resource of the Edge computing is more than computing unit of task ψ_t. Then, if the Edge computing have enough idle computing resources to complete task t. Then, the task flag will change to $u_t = 1$, which means tasks will offload to the layer of the Edge computing using the MFFA Algorithm to find the optimal computational device. Otherwise, the task flag will change to $u_t = 2$, which means the tasks will offload to the Fog layer to complete unfinished tasks in the queue using the MFFA Algorithm to find the optimal computational device in case the tasks satisfy the layer condition. Otherwise, offload to the Cloud Computing.

Where σ_k indicates the status of server [1 = on,0 = off], C_{idle} is the idle computing resource in the layer. which is as follows $u_t = 2$, which means the tasks will offload to the Fog layer to complete unfinished tasks in the queue using the MFA Algorithm to find the optimal computational device in case the tasks satisfy the layer condition. Otherwise, offload to the Cloud Computing.

Algorithm 1 Enhance Task offloading Algorithm

BEGIN

1. ***IF*** $(u_t \epsilon \{mt\})\{$
2. $n_1 = mtnode\_set()$
3. ***FOR*** $(i = 1; i \leq n_1; i++)$
4. $\{$***IF*** $C_{idle}^{i} > \psi_t \& \sigma_k = 1$
5. $(u_t = 1)$ *// edge computing*
6. *MFA Algorithm to find the optimal edge computing}*
7. ***Else***

a $u_t = 2$ *// fog computing}*
$F\ (u_t \epsilon \{Fog\})\{$

8. $n_2 = Fognode\_set()$
9. ***FOR***$(j = 1; j \leq n_2; j++)$
10. $\{IF\ C_{idle}^{j} > \psi_t \& \sigma_k = 1$
11. $(u_t = 2)$ *// Fog Layer*
12. *Use MFA to find the optimal fog node}*
13. ***Else***
14. $u_t = 3$ *// cloud layer}*
15. *Return* u_t
16. ***End***

VI. MULTI-OBJECTIVES OPTIMIZATION PROBLEM (MOP)

The main issue in this study is that offloading tasks to fog computing will reduce the delay but increase the energy consumption of the users' devices. Concurrently, offloading tasks to cloud computing will decrease the energy consumption of users' devices but increase the delay due to the long distance between the cloud and the end users, which means vice versa.

This study is concerned with optimizing two objectives, delay and energy consumption, and it is challenging to reduce both of them simultaneously, which is called conflicting objectives.

Thus, main purpose of MOP is to optimize conflicting multi-objectives simultaneously. With m decision variables and n objectives, it can be defined as:

$Min(y = f\ (x) = [f_1\ (x)\ , \ldots\ldots\ldots, f_n\ (x)])$, where $x = (x_1, \ldots\ldots, x_m) \epsilon X$ is an m -dimensional decision vector, X is the search space, $y = (y_1, \ldots\ldots, y_m) \epsilon Y$ is the objective vector; and Y is the objective space.

Generally, there is no single optimal solution with respect to other objectives. In this type of problem, the desired solution is regarded as at set of possible solutions that are optimal for a single objective or more. These solutions are considered Pareto optimal sets. The main Pareto concepts used in the MOP are as follows:

(i) Pareto dominance. For two decision vectors x_1 and x_2, dominance (indicated by $\prec$): is known as

$$x_1 \prec x_2 \Leftarrow\Rightarrow \forall_i f_i\ (x_i) \leq f_i\ (x_i) \wedge \exists_i\ (x_i) < f_i(x_i).$$

The decision vector x_1 dominates x_2, in this case, x_1 outperforms x_2 for at least single objective.

(ii) Pareto optimal set. Pareto optimal set P_s is the set of all Pareto optimal decision vectors.

$$P_s = \{x_1 \epsilon X, |\exists x_2\ \epsilon X, x_2 \prec x_1,$$

where decision vector x_1 is said to be Pareto optimal when it is not dominated by any other decision vector, x_2, in the set.

(iii) Pareto optimal front. The Pareto optimal front P_F is an image of the Pareto optimal set in the objective space.

$$P_F = \{f\ (x) = (f_1\ (x)\ , \ldots\ldots, f_n\ (x))\ |x \epsilon P_s\}$$

VII. MULTI-OBJECTIVE FIREFLY (MFA) ALGORITHM

It is one of the optimization Algorithms that are appropriate to solve the MOP effectively, [43]. The study has chosen the FA algorithm to improve instead of other metaheuristic algorithms due to its features. First, addressing edge-fog-cloud environments involves balancing conflicting objectives like reducing energy consumption, delay, and optimizing resource utilization, and thus the FA algorithm can be adept at multi-objective optimization, making it suitable for simultaneously optimizing these conflicting objectives, as the movement of fireflies is affected by both attraction and randomness, allowing exploration of a various solution space [12].

TABLE 2. Summary of related studies in task offloading.

Ref	Challenge	Technique	Objectives	Strength	Limitation
[16]	Offloading tasks to another node to complete unfinished tasks	Task Offloading algorithm	Energy consumption and Delay	Reduce energy	Network overhead
[17]	Deciding offloading tasks	joint energy-efficient task assignment (JEETA) utilizing the dynamic application-partitioning algorithm (DAPTS)	Energy Consumption	Completing workflow tasks within a given deadline	more response time
[18]	mobility and delay-sensitive applications	Microservice container fog system (MSCFS)	Cost	Enhance server utilization	considering transient failure
[3]	computation offloading and task scheduling for DNN-based applications	Greedy and genetic algorithms	Delay	Handling online tasks	Consuming more time
[19]	the computation offloading problem in a mixed fog/cloud system	Computation offloading and resource allocation algorithm (CORA)	Reduce the maximal weighted cost of delay and energy consumption	Balance the EC among all the UEs	Consuming a high energy
[20]	task offloading and resource scheduling problem	Efficient task offloading and resource scheduling algorithms based deep learning	Minimize the weighted sum of energy, time, and rent cost (ETRC)	Near-optimal performance and significantly decreased the ETRC compared	increase the complexity and the execution time
[5]	Manipulating the incoming tasks and decide whether to offload tasks	proposing a three-tier CoT architecture, and a task offloading mathematical model	Execution time, delay and energy consumption	Saves time and power	Overloading
[21]	the computation offloading	Rich theory of microeconomics	Energy consumption	Reducing Energy	long execution time on offloading decision
[22]	Improving offloading of MEC architecture	A new intelligent computation offloading based MEC architecture	Delay	Reduce task delay with the increasing data and subtasks	Considering content caching problem
[2]	Selecting the best fog node to offload each task	new task offloading approach	Delay	delay, executing more tasks, and balance the load	optimizing each new task may take a long time
[14]	finding an optimal computing device for offloading tasks	Firefly algorithm	Energy consumption and computational time	computational time, energy consumption, CO 2 emission, and Temperature emission	Did not consider the priority of tasks for offloading
[23]	computation offloading decision-making problem	Game theory	computation overhead	computation offloading performance and scales well as the user size increases	Considering when the mobile user left during the computation
[24]	reducing the makespan while maintaining the heterogeneity of cloud-fog computing	Logic-Based Benders Decomposition (LBBD) principle with Mixed Integer Linear Programming (MILP) models	makespan	Reducing overall solution time comparing	communication overhead
[25]	joint DC and computation offloading	Optimal Joint DC, Offloading, and Resource Allocation (JCORA)	minimize the maximum weighted energy and service delay cost (WEDC) of all users	Reduce service delay cost	high complexity

TABLE 2. *(Continued.)* **Summary of related studies in task offloading.**

[26]	the hybrid computation offloading	hybrid computation offloading solution	Reduce the energy consumption	achieves lower energy consumption than with meeting the deadline	consider more QoS parameters
[27]	computation offloading at mobile-cloud-edge	CGMS and DGMS based Multi-user Multi-task offloading scheme	study the convergence speed	higher overall system utility	spends more execution time during scheduling tasks
[28]	find the optimal task offloading	find the optimal task offloading	improve the total long-term	achieve higher system reward than	improve the periodic arrival of vehicles
[29]	computation offloading in cloud-fog computing with non-orthogonal multiple access (NOMA)	integrated fog and cloud computing approach	minimizing the total system cost of energy	Reducing cost	did not utilize the resources optimally and more energy consumption
[30]	task offloading	novel task offloading approach for	minimize the overall service time for latency-sensitive applications	service time and resource utilization. It can also reduce the overall task failures	Assigning the priority will increase the energy consumption
[31]	computation offloading	MO and GO algorithms	increase the cloud-fog provider's revenue	achieves higher revenue	wasting of resources when allocating extra CPU to offload to the server
[32]	computation offloading in marine vehicular cloud-fog computing for USV clusters	Adaptive Upper Confidence Boundary (AUCB) algorithm based on MAB theory	Delay	fast converge to the optimal computation task offloading and light input data loads	the cost of exploration and accomplish balanced computing performance
[33]	resource allocation and computation offloading	ETCORA algorithm	time cost and energy minimization	reducing energy consumption and completion time of requests	Did not take advantage of full resource utilization
[34]	placement of VMs in cloud-fog computing	Mixed Integer Linear Programming (MILP) model	Reduce energy consumption	Recue energy effectively	Long execution time
[35]	energy efficient computation offloading and dynamic resource scheduling (eoDS) problem	proposing an eoDS algorithm	Energy consumption and completion time	Reduce delay and energy	Did not utilize the resource optimally
[36]	dynamic task offloading	machine learning based on a logistic regression algorithm	Execution time	High accuracy	Consuming energy
[37]	computing offloading with limited edge resources	optimal joint offloading scheme based on resource occupancy prediction	Delay and failure rate	Delay and failure rate	Limited bandwidth
This study	Completing unfinished tasks by offloading to another computing layer and selecting an optimal device	Multi-objective Firefly (MFA) algorithm	Reducing Energy, delay	Reducing energy, delay and maximizing the resource utilization	Real-world scenario

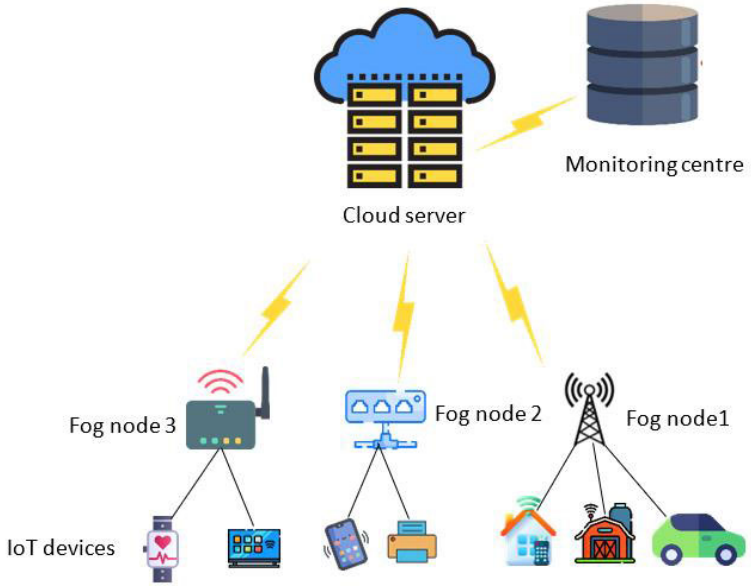

FIGURE 1. The cloud-fog computing architecture.

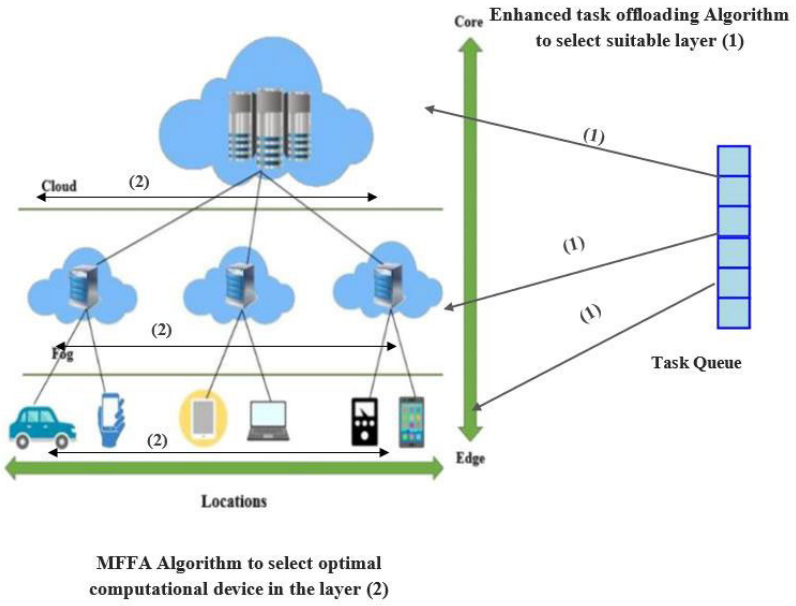

FIGURE 2. Enhance task offloading and MFA algorithms for offloading tasks in the edge-fog-cloud computing.

Even more, the Firefly algorithm utilizes both global and local search mechanisms. In task offloading, there's a require to search for global optimal solutions while refining those solutions locally to suit the dynamic nature of the edge, fog, and cloud platforms. Firefly's technique of brighter fireflies attracting others guarantee both exploration (global search) and exploitation (local refinement). Furthermore, according to the highly dynamic nature of Edge-fog-cloud computing, with fluctuating network conditions, workload variations, and resource availability [44].

The Firefly Algorithm's adaptability to various intensities (brightness) allows it to dynamically adjust its search based on current conditions, making it responsive to changes in the environment and resources. Also, it is relatively simple to implement compared to other optimization algorithms. Its flexible design allows it to be customized for different optimization problems, which is crucial for a hybrid computing environment like edge-fog-cloud, where multiple parameters need to be optimized based on real-time data. In addition, task offloading in edge-fog-cloud systems often

needs fast decision-making to ensure that performance is not compromised.

The FA algorithm converges relatively quickly to good solutions, which is essential for real-time offloading tasks where delays could degrade the system's overall performance. Moreover, task offloading in edge-fog-cloud systems often requires quick decision-making to ensure that performance is not compromised [44].

The FA Algorithm converges relatively quickly to good solutions, which is essential for real-time offloading tasks where delays could degrade the system's overall performance. Besides, Edge-fog-cloud architectures can be scaled to support a large number of devices and services. The FA algorithm is highly scalable and can handle the complexity of large networks with many interconnected devices and platforms, ensuring that the solution remains efficient even as the number of tasks and nodes grows. All these characteristics of the FA algorithm make it an ideal option for offloading tasks in edge-fog-cloud computing [12].

However, this research aims to solve the problem of unfinished tasks when the computing device leaves the system by offloading tasks to another computing device in the belonged or upper layer by implementing Algorithm 1 and then proposing the MFA, which starts after selecting the suitable layer then finding the optimal computational device horizontally in the same layer and deploy the unfinished tasks on that computation device. The main objectives are reducing the energy consumption and delay the resource utilization in edge-fog-cloud computing. The detail of the algorithm is discussed below in algorithm 2, which illustrates how to find the optimal computational device in the selected layer. The MFA algorithm process initiates sorting tasks in ascending order according to the deadline to guarantee fast processing of sensitive tasks and reduce transmission delay.

Then, generating a MOP function in each iteration that indicates a fitness function based on the two QoS parameters, energy consumption and transmission delay.

The fitness function based on combining the objectives into one objective using a weighted-sum approach:

Then, generating a MOP function in each iteration that indicates a Fitness Function based on the two QoS parameters, energy consumption and delay that are calculated according to selected platform that are clarified in IV Section [16]. The Fitness Function based on combining the objectives into one objective using a weighted-sum approach:

$$F_n = Min(\alpha_1.Total\ Energy) + ((1-\alpha_2).Total\ Delay) \quad (10)$$

where F_n is the fitness function value for measuring the optimal degree of executing devices to assign tasks to the suitable computing devices. Where (α_1&(α_2) are refer to the priority of the objectives. Hence, α is the Energy-delay balance factor where α ($\alpha \in [0, 1]$) is the balance coefficient between total Energy and total delay. $\alpha = 0.5$ means that total Energy and total Delay have same priority in optimizing. When $\alpha > 0.5$, our mechanism focuses on minimizing the energy with higher priority than total delay, which is the case task will be late to obtain minimum energy consumption. Inversely, when $\alpha < 0.5$, the Delay is more prioritized than Energy, i.e., the user has an instant better task delay.

In the MFA Algorithm finding the optimal computational devices depending on the light intensity variation of the Fireflies and attractiveness function. Thus, for minimum optimization problem [13], the computing server light intensity represented by (Firefly) at specific location x can be selected as $IS(x) \propto F(x)$. The light intensity of the computing server position varies with the distance r and is expressed as follows [14].

$$IS_i(r) = IS_{0i}.e^{-\gamma r^2} \quad (11)$$

where IS_0 indicates the source computing device light intensity, S_i referred to the index of computing resource, γ represents a fixed light absorption coefficient. Generally, the light intensity is associated with the minimization problem unlike the Fitness Function.

The attractiveness AS depends on the distance r and is expressed as follows [14].

$$AS_i = AS_{0i}.e^{-\gamma r^2} \quad (12)$$

The above equations attractiveness function is appropriate for single objective. So, the attractiveness function requires improving it to MOP, which will enhance the problem's randomness and convergence speed by the normalization which can be expressed as:

$$FN_r(d_i) = \frac{f_r(d_i) - f_r^{min}}{f_r^{max} - f_r^{min}} \quad (13)$$

The normalization of each objective of the particle while finding the optimal executing devices is based on the maximum and minimum value of the associated objective function. The essential role of normalizing the objective function is eliminating the effect of different amplitudes on multiple objectives. Where r indicates the total number of objectives of computational device d_i, f_r^{max}, f_r^{min} are the maximum and minimum fitness function value of rth objective that are gotten from non-dominated solution among the available devices. In line 16, in each iteration the condition conducting to compare the new FN_j and previous FN_i result. In case the new result of FN_j is better, then calculating the distance between two computational devices. In case the $FN_j < FN_l$ not satisfy the IF condition, it means the new result of FN_j is not better than previous one FN_i, then find the optimal device according to [14]. Otherwise, calculate the modified attractiveness function in equation (6.5). The MFA algorithm pointed to the ideal computing device in line with the minimum ED value, defined as:

$$gbest(d_i) = \min \sqrt{\sum_{x=1}^{2} ED(f_r(d_i), f_r(d_j))} = \sqrt{\sum_{x=1}^{2} ED(f_r(d_i) - f_r(d_j))^2} \quad (14)$$

where ED value $ED\ (\ d_i, d_j)$ between two particles d_i and d_i in 2-D space is defined as:

$$ED\left(f_r\left(d_i\right), f_r\left(d_j\right)\right) = \begin{cases} \left(f_r\left(d_i\right) - f_r\left(d_j\right)\right) & IF\ \left(\left(f_r\left(d_i\right) > f_r\left(d_j\right)\right)\right) \\ 0 & Otherwise \end{cases} \quad (15)$$

After that, comparing Pareto dominance relation between the computing and put the non-dominance one. Then select the one device with minimum utilization rate from the set as the attractiveness one to reduce the delay that expressed as follow [45].

$$utilization\ rate\ UR_n = \frac{VMs\ requested\ MIPS}{VMs\ available\ MIPS} * 100 \quad (16)$$

Consequently, update the position of the computational device as:

$$d_{i+1} = d_i + \beta_0 e^{-\gamma v_{ij}^2}\left(d_j + d_i\right) + \alpha\varepsilon_i \quad (17)$$

where $\beta 0$ is the initial attractiveness at r = 0, α being the randomization parameter in the interval [0, 1], ε is a vector of random numbers drawn from the uniform distribution. Finally, the parameter γ represents the variation of the attractiveness, and its value helps to determine the speed of the convergence of the algorithm. In most cases, the values of, γ vary [0.01, 100]. However, in line 31 if the condition of $(FNj \leq \Delta)$ does not satisfied, then the algorithm randomly generated new $FFmin$ solutions are considered as fireflies to find optimal as in [14]:

$$FF^{min} = FF^{min} + \alpha.Rand(1/2) \quad (18)$$

Thus, it supposed to improve the speed of convergence during selecting the execution component of MFA algorithm and finding the appropriate devices for every unfinished task with maintaining the QoS objectives.

VIII. SIMULATION SETTINGS

This section to verify the effectiveness of the proposed MFA algorithm for Task offloading to complete unfinished tasks during the processing. we investigate the performance of the proposed MFA algorithm for task offloading by evaluating various QoS parameters such as energy consumption and delay recording to the calculation on the section IV. In addition, The evaluation conducting between the proposed algorithm (MFA) with the existing algorithm that proposed in [14] and PSO algorithm in reducing the two objectives energy consumption and delay. Even more, the study conducting the comparison based on maximizing the resource utilization as performance metric. It determines the suitable utilizing of resources that are giving to the user to schedule the workload by exploiting the idle time gaps. Resource utilization can be calculating as [46]

$$AVG\ resource\ utilization = \frac{\sum_{i=1}^{N} T_{VM_i}}{Makespan \times N} \quad (19)$$

where T_{VM_i} is the time taken by the VM_i to finish all tasks, and N is the number of resources.

Algorithm 2 Multi-objectives Firefly (MFA) Algorithm

Input: *Set the objectives of computational devices;*
Define the constant values of n, I_0, a, γ
Output:*Finding the optimal device*
BEGIN
Initialize
Temp=0;
$Q[] = \emptyset$
Min=arr [0]

1. $Q \leftarrow t$; *// receiving tasks then insert to the Queue*
2. ***For*** $(i = 1; i \leq t; i++)\{$ *// sorting tasks in ascending according to the deadline*
3. ***For*** $(int\ j = i + 1; j < length; j++)\{$
4. ***IF***$(arr\ [i] > arr\ [j])\ \{$
5. $temp = arr\ [i]$;
6. $arr\ [i] = arr\ [j]$;
7. $arr[j] = temp;\}\}$
8. *Retrun Q*$\}$ *// in ascendingorder*
9. ***For*** *each* j : 1 *to* N *do* *//r= Maximum number of iterations*
10. $FN_j = E + (1 - RU)$
11. ***END For***
12. ***While*** $\left(FN_j \leq \Delta\right)$ *do //*Δ *= Threshold value*
13. $min = arg\ min\ FN_j$
14. ***For each*** i : 1 *to* n *do*
15. $FN_r\ (d_i) = \frac{f_r(d_i - f_r^{min})}{f_r^{max} - f_r^{min}}$ *// calculate the modified attractiveness function*
16. ***IF*** $\left(FN_j < FN_i\right)$ *// new* FN_j *and previous* FN_i *result*
17. $gbest\ (d_i)\ \|\ d_i - d_j\ \| = \sqrt{\sum_{x=1}^{D}(f_r(d_i) + f_r(d_j))^2}$ *// Calculate the Euclidean between the computational devices*
18. *Compare the Pareto dominance relationship between each pair of the computation device and put the non-dominance ones.*
19. *Calculate* UR_n = *(device requested MIPS/ device available MIPS)* $*$ 100 *// calculate the resource utilization for computation devices*
20. ***For*** $(m = 1; m \leq length; m++)$
21. ***IF*** $(arr\ [m] < Min)$
22. $Min = arr\ [m]$ *; //sorting server ascending according to the resource utilization*
23. ***End IF***
24. ***END For***
25. *Select the computation device with minimum resource utilization* UR_n *from the non-dominant sets as attractive ones}*
26. *Update the position of the computational server as in Eq. (17)*
27. ***End if***
28. ***End for***
29. ***End for***
30. $FN^{min} = FN^{min} + \alpha.Rand(1/2)$ *//*FN^{min} *indicates to the minimum value of fitness function obtained from algorithm*
31. *Find the optimal computational device // by new value of* FN^{min} *that obtained from Eq. (18)*
32. ***End for***
33. ***End***

The experiments were conducted on a scenario with seven mobile terminal devices, three fog nodes, and one cloud server in the fog-cloud computing system. It can extend to more mobile terminal devices, fog nodes, and cloud servers with similar results. Examining the performance of our method was conducting under different workloads,

TABLE 3. Simulation key parameters.

Parameters	Value
μ	4.6
λ	3.2
a	1
b	2.4
e	3.5
workload	[30 50 90 150 200]
population size	50
α	0.1
β	0.6
γ	0.6
Δ	0.0001
Number of VMs	5
MIIPs	9726
VM memory (RAM)	0.5 GB

we selected five groups of tasks. Their total workloads are 30, 50, 90, 150, and 200. The lengths of the tasks are generated randomly because we cannot predict the task length in reality. More or fewer tasks can achieve similarly. The experimental study was conducted through simulations using MATLAB R2018b on a computer equipped with a Core i7 and 8 GB RAM machine running the Windows operating system 11 to validate the performance of our proposed method. Validating the proposed algorithm by performing 30 independent runs referring to [47] that the best Pareto front achieved by 30 runs is for providing a qualitative comparison. The simulation parameters of the MFA algorithm are summarized in Table 3, referring to [8], [11], and [46].

IX. RESULTS AND DISCUSSIONS

The purpose of the simulation experiments was to reduce energy consumption and delay for IoT applications. Figure 3 demonstrates the Energy consumption for the MFA,Task offloading algorithm [16], and PSO algorithms versus the number of workloads [30,50,90,100,150].

The task offloading algorithm consistently shows higher energy consumption across all workload groups compared to the other two algorithms. While the PSO algorithm and the MFA algorithm have much closer results in terms of energy consumption, the MFA algorithm outperforms the PSO slightly in most cases. MFA appears to be the best algorithm across all workload groups due to its consistently lower energy consumption compared to both task offloading and PSO. Thus, the MFA algorithm likely incorporates more efficient resource management and task scheduling, which helps it adapt better to the varying number of workloads. It seems to balance energy consumption effectively, even when the workload increases (as shown in the 150 and 200 workload groups). The MFA algorithm outperforms the Task Offloading algorithm by 66.67%, depending on the workload size. This shows that the MFA algorithm is significantly more energy-efficient, particularly in larger workloads like 90, 150, and 200.

Hence, compared to PSO, MFA also shows a consistent advantage and outperforms by 11.11%. However, the gap between MFA and PSO is smaller than with the Task Offloading algorithm, indicating that PSO is somewhat competitive but still less efficient overall.

MFA is the best algorithm because it consistently demonstrates superior energy efficiency, particularly in larger workloads. The percentage reductions in energy consumption show that it scales well with increasing workloads, making it a strong solution for edge-fog-cloud computing environments where energy optimization is critical.

While in TABLE 4, it is obvious that across all workload sizes, MFA outperforms Task Offloading by a significant margin, with percentage differences ranging from 50.4% to 89.15%.

The difference becomes more pronounced as the workload size increases, showing that MFA scales much better and is more energy-efficient at larger workloads.

For example, at Workload 90, the difference is 89.15%, meaning Task Offloading consumes almost twice as much energy as MFA. Similarly, at Workload 200, MFA is 79.4% more efficient. Whereas, the differences between PSO and MFA are much smaller, ranging from 6.31% to 7.12%. This shows that PSO is more competitive with MFA but still consumes slightly more energy across all workloads.

The difference remains consistent across all workload sizes, meaning that while PSO performs close to MFA, MFA still consistently leads in energy efficiency.

In Figure 4, it shows that across all workload groups, the MFA algorithm consistently demonstrates the lowest delay compared to both Task Offloading and PSO algorithms. As the workload increases (from 30 to 200), the difference in delay becomes more pronounced, with the MFA algorithm providing significant delay reduction, especially for larger workloads.

This suggests that the MFA algorithm is more efficient in minimizing delay, particularly in high-workload scenarios.

In TABLE 5 that presents the delay results for three algorithms (Task Offloading, MFA, and PSO) across five different workload sizes. In workload size 30 The MFA algorithm shows the smallest delay, with the Task Offloading algorithm having the highest delay.

MFA reduces delay by approximately 21.7% compared to Task Offloading, and PSO reduces delay by around 16.6% compared to Task Offloading. While in workload 50, MFA has the lowest delay, followed by PSO, both of which outperform Task Offloading. MFA reduces delay by 38.2% compared to Task Offloading, while PSO reduces delay by about 34.1%.

Whereas in 90 workloads, MFA still demonstrates the lowest delay, with PSO slightly higher than MFA but still better than Task Offloading.

MFA reduces delay by 37.5%, and PSO reduces delay by 33.6% compared to Task Offloading.

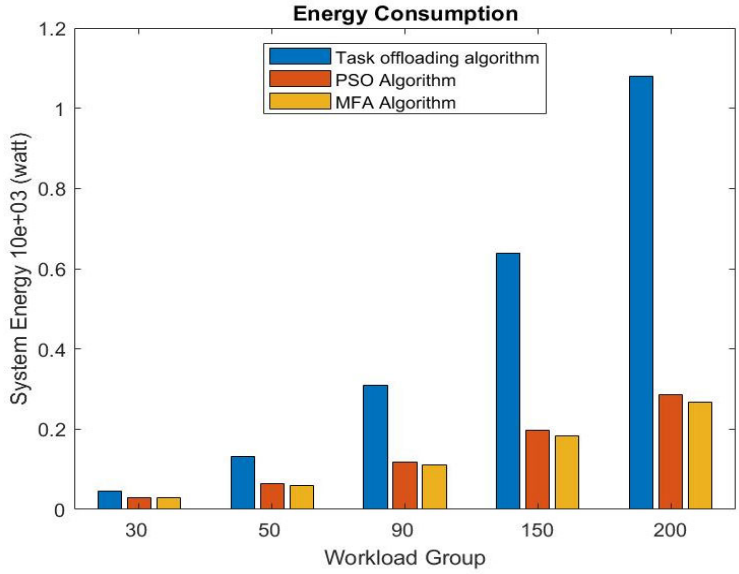

FIGURE 3. Energy consumption comparison among the MFA with comparison algorithms via various workload size.

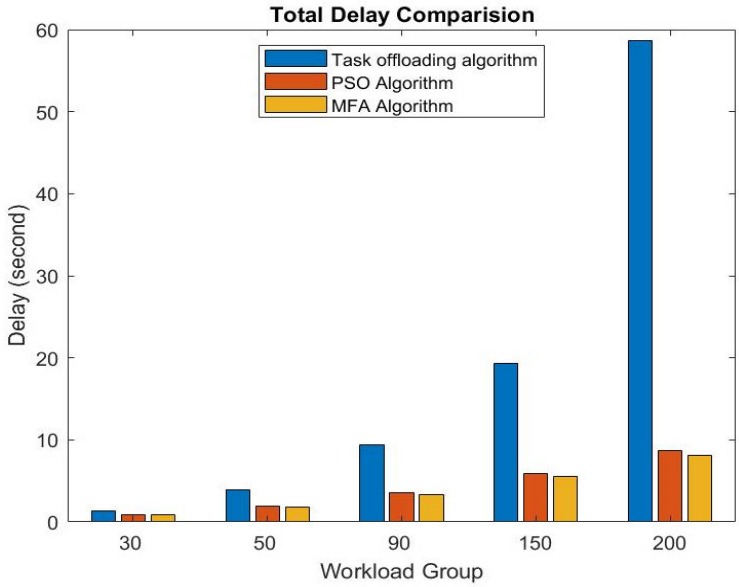

FIGURE 4. Delay comparison among the MFA with comparison algorithms via various workload size.

TABLE 4. Result of energy consumption based on various workload.

Workload size	Task offloading	MFA	PSO
30	1.093	0.8564	0.9114
50	2.965	1.832	1.954
90	5.365	3.352	3.562
150	9.34	5.572	5.932
200	15.61	8.076	8.651

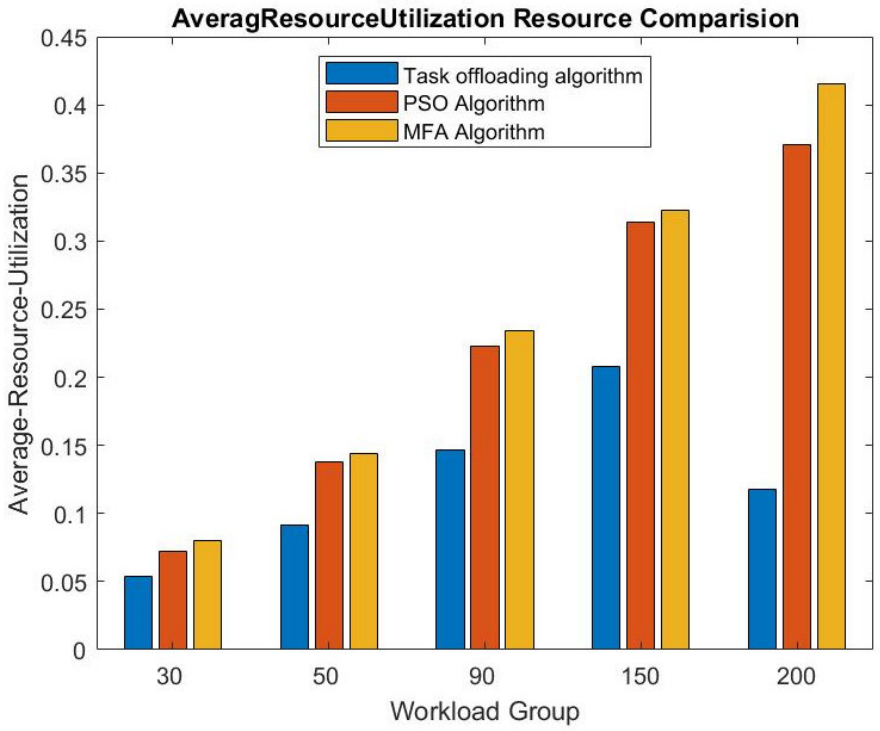

FIGURE 5. Resource utilization among the MFA with comparison algorithms via various workload size.

TABLE 5. Result of delay based on various workload between MFA algorithm and the comparison algorithms.

Workload size	Task offloading	MFA	PSO
30	0.04605	0.02831	0.03013
50	0.0911	0.06058	0.06461
90	0.2096	0.1108	0.1178
150	0.3396	0.1842	0.1961
200	0.479	0.267	0.286

TABLE 6. Result of resource utilization based on various workload between MFA algorithm and the comparison algorithms.

Workload size	Task offloading	MFA	PSO
30	0.05744	0.07376	0.5399
50	0.0976	0.1337	0.09171
90	0.1563	0.222	0.1469
150	0.221	0.313	0.2077
200	0.1253	0.4064	0.1178

Besides, workload size is 150 presents that MFA continues to show the smallest delay, followed by PSO. MFA reduces delay by 40.4%, while PSO reduces delay by 36.5% compared to Task Offloading.

While in 200 workload size, MFA remains the most effective in reducing delay.

MFA reduces delay by 48.3%, and PSO reduces delay by 44.6% compared to Task Offloading.

Figure 5 which shows the average resource utilization comparison between the Task Offloading Algorithm, PSO Algorithm, and MFA Algorithm across different workload groups. MFA Algorithm consistently achieved the highest resource utilization across all workload sizes, from small (30) to large (200). This indicates that MFA is the most efficient algorithm for managing and maximizing the use of available computational resources.

While PSO Algorithm performed well but consistently fell slightly behind MFA in resource utilization. It still showed significantly better results than the Task Offloading algorithm, making it a viable option but not as optimal as MFA.

Whereas Task Offloading Algorithm had the lowest resource utilization across all workload groups. Its inefficiency in utilizing resources becomes more pronounced as the workload increases, suggesting it is not as effective in managing resources compared to MFA and PSO.

In TABLE 6, MFA Algorithm consistently achieved the highest resource utilization across all workload sizes, from small (30) to large (200). This indicates that MFA is the most efficient algorithm for managing and maximizing the use of available computational resources.

PSO Algorithm performed well but consistently fell slightly behind MFA in resource utilization. It still showed significantly better results than the Task Offloading algorithm, making it a viable option but not as optimal as MFA.

Task Offloading Algorithm had the lowest resource utilization across all workload groups. Its inefficiency in utilizing resources becomes more pronounced as the workload increases, suggesting it is not as effective in managing resources compared to MFA and PSO.

Overall, MFA is the most efficient algorithm for managing energy, delay, and resource utilization, while Task Offloading algorithm shows the weakest performance, particularly under larger workload conditions. PSO serves as a middle ground, performing better than Task Offloading but not as efficiently as MFA.

X. CONCLUSION

This study proposed a strategy to complete unfinished tasks when the computation node leaves the system by offloading tasks to another node considering reducing the energy consumption and delay. Solve the overhead load by selecting the optimal computation device with minimum resource utilization.

The strategy is based on two stages: First proposed an Enhance Task offloading Algorithm to select the appropriate layer (Edge-Fog-Cloud) according to resource capacity availability compared to the task's computation. Then, implementing the proposed Multi-objectives Firefly called the MFA algorithm for task offloading to complete unfinished tasks by selecting the optimal computational device with assigning tasks to the minimum resources utilization rate to reduce the delay and solve the load overhead. The proposed algorithm aims to minimize power consumption and delay objectives.

The simulation results reveal that MFA outperforms Task Offloading algorithm the comparison approach in reducing energy consumption and delay by about 23% and 25%, respectively. The proposed algorithm maintains its stability and increases linearly with increasing workloads. It proves it can handle the enormous increase in generating requests from IoT devices. In addition, the MFA algorithm outperforms the benchmark by maximizing the resource utilization by 86%.

When comparing the results between the MFA and PSO algorithms, there is a slight difference in both energy consumption and delay, with MFA outperforming PSO by 6%. This advantage is attributed to the nature of metaheuristic algorithms, which have lower complexity, resulting in faster processing and reduced energy consumption. However, the difference in resource utilization is more pronounced, with MFA achieving 96% utilization compared to PSO. This significant improvement is due to MFA's strategy of offloading tasks to the most suitable platforms and selecting optimal devices, effectively preventing idle resources.

The main limitations of this study include the consideration of resource heterogeneity and the lack of discussion on the success rate. For future work, a comparison with other baseline algorithms will be necessary to further validate the effectiveness of the proposed algorithm.

Additionally, this study was conducted using simulations and holds potential for extension to real-world scenarios. Future research could focus on optimizing additional objectives such as transmission costs, computing resources, and load balancing. Moreover, applying more advanced algorithms to solve offloading problems and adopting AI-based approaches for predicting and analyzing incoming tasks could enhance the overall system performance.

ACKNOWLEDGMENT

This study is supported by the Universiti Putra Malaysia and the Ministry of Higher Education Malaysia under Grant Number (FRGS/1/2023/ICT11/UPM/02/3). We are sincerely grateful for the facilities and funding provided, which were essential for the completion and publication of this study.

REFERENCES

[1] B. Rababah, T. Alam, and R. Eskicioglu, "The next generation Internet of Things architecture towards distributed intelligence: Reviews, applications, and research challenges," *SSRN Electron. J.*, vol. 12, no. 2, 2020, doi: 10.2139/ssrn.3640136.

[2] S. Aljanabi and A. Chalechale, "Improving IoT services using a hybrid fog-cloud offloading," *IEEE Access*, vol. 9, pp. 13775–13788, 2021, doi: 10.1109/ACCESS.2021.3052458.

[3] Z. Chen, J. Hu, X. Chen, J. Hu, X. Zheng, and G. Min, "Computation offloading and task scheduling for DNN-based applications in cloud-edge computing," *IEEE Access*, vol. 8, pp. 115537–115547, 2020, doi: 10.1109/ACCESS.2020.3004509.

[4] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of Internet of Things," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 46–59, Jan. 2018, doi: 10.1109/TCC.2015.2485206.

[5] M. Aazam, S. u. Islam, S. T. Lone, and A. Abbas, "Cloud of things (CoT): Cloud-fog-IoT task offloading for sustainable Internet of Things," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 1, pp. 87–98, Jan. 2022, doi: 10.1109/TSUSC.2020.3028615.

[6] H. Lin, S. Zeadally, Z. Chen, H. Labiod, and L. Wang, "A survey on computation offloading modeling for edge computing," *J. Netw. Comput. Appl.*, vol. 169, Nov. 2020, Art. no. 102781, doi: 10.1016/j.jnca.2020.102781.

[7] S. O. Ogundoyin and I. A. Kamil, "Optimization techniques and applications in fog computing: An exhaustive survey," *Swarm Evol. Comput.*, vol. 66, Oct. 2021, Art. no. 100937, doi: 10.1016/j.swevo.2021.100937.

[8] H. Tan, Z. Han, X.-Y. Li, and F. C. M. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9, doi: 10.1109/INFOCOM.2017.8057116.

[9] F. A. Saif, R. Latip, Z. M. Hanapi, M. A. Alrshah, and S. Kamarudin, "Workload allocation toward energy consumption-delay trade-off in cloud-fog computing using multi-objective NPSO algorithm," *IEEE Access*, vol. 11, pp. 45393–45404, 2023, doi: 10.1109/access.2023.3266822.

[10] M. H. Horng and T. W. Jiang, "The codebook design of image vector quantization based on the firefly algorithm," in *Computational Collective Intelligence. Technologies and Applications* (Lecture Notes in Computer Science), vol. 6423. Berlin, Germany, 2010, pp. 438–447, doi: 10.1007/978-3-642-16696-9_47.

[11] X. S. Yang and M. Karamanoglu, "Swarm intelligence and bio-inspired computation: An overview," in *Swarm Intelligence and Bio-Inspired Computation*. Amsterdam, The Netherlands: Elsevier, 2013, pp. 3–23, doi: 10.1016/B978-0-12-405163-8.00001-6.

[12] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications* (Lecture Notes in Computer Science), vol. 5792. Berlin, Germany: Springer, 2009, pp. 169–178, doi: 10.1007/978-3-642-04944-6_14.

[13] M. Adhikari, T. Amgoth, and S. N. Srirama, "Multi-objective scheduling strategy for scientific workflows in cloud environment: A firefly-based approach," *Appl. Soft Comput.*, vol. 93, Aug. 2020, Art. no. 106411, doi: 10.1016/j.asoc.2020.106411.

[14] M. Adhikari and H. Gianey, "Energy efficient offloading strategy in fog-cloud environment for IoT applications," *Internet Things*, vol. 6, Jun. 2019, Art. no. 100053, doi: 10.1016/j.iot.2019.100053.

[15] X.-S. Yang, "Multiobjective firefly algorithm for continuous optimization," *Eng. Comput.*, vol. 29, no. 2, pp. 175–184, Apr. 2013, doi: 10.1007/s00366-012-0254-1.

[16] G. Li, J. Yan, L. Chen, J. Wu, Q. Lin, and Y. Zhang, "Energy consumption optimization with a delay threshold in cloud-fog cooperation computing," *IEEE Access*, vol. 7, pp. 159688–159697, 2019, doi: 10.1109/ACCESS.2019.2950443.

[17] A. Abro, Z. Deng, K. A. Memon, A. A. Laghari, K. H. Mohammadani, and N. Ul Ain, "A dynamic application-partitioning algorithm with improved offloading mechanism for fog cloud networks," *Future Internet*, vol. 11, no. 7, p. 141, Jun. 2019, doi: 10.3390/fi11070141.

[18] X. Zhao and C. Huang, "Microservice based computational offloading framework and cost efficient task scheduling algorithm in heterogeneous fog cloud network," *IEEE Access*, vol. 8, pp. 56680–56694, 2020, doi: 10.1109/ACCESS.2020.2981860.

[19] J. Xu, Z. Hao, R. Zhang, and X. Sun, "A method based on the combination of laxity and ant colony system for cloud-fog task scheduling," *IEEE Access*, vol. 7, pp. 116218–116226, 2019, doi: 10.1109/ACCESS.2019.2936116.

[20] Q. Zhang, L. Gui, S. Zhu, and X. Lang, "Task offloading and resource scheduling in hybrid edge-cloud networks," *IEEE Access*, vol. 9, pp. 85350–85366, 2021, doi: 10.1109/ACCESS.2021.3088124.

[21] R. Besharati and M. H. Rezvani, "A prototype auction-based mechanism for computation offloading in fog-cloud environments," in *Proc. 5th Conf. Knowl. Based Eng. Innov. (KBEI)*, Feb. 2019, pp. 542–547, doi: 10.1109/KBEI.2019.8734918.

[22] Y. Miao, G. Wu, M. Li, A. Ghoneim, M. Al-Rakhami, and M. S. Hossain, "Intelligent task prediction and computation offloading based on mobile-edge cloud computing," *Future Gener. Comput. Syst.*, vol. 102, pp. 925–931, Jan. 2020, doi: 10.1016/j.future.2019.09.035.

[23] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016, doi: 10.1109/TNET.2015.2487344.

[24] S. Li, W. Chen, Y. Chen, C. Chen, and Z. Zheng, "Makespan-minimized computation offloading for smart toys in edge-cloud computing," *Electron. Commerce Res. Appl.*, vol. 37, Sep. 2019, Art. no. 100884, doi: 10.1016/j.elerap.2019.100884.

[25] T. T. Nguyen, V. N. Ha, L. B. Le, and R. Schober, "Joint data compression and computation offloading in hierarchical fog-cloud systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 293–309, Jan. 2020, doi: 10.1109/TWC.2019.2944165.

[26] X. Meng, W. Wang, and Z. Zhang, "Delay-constrained hybrid computation offloading with cloud and fog computing," *IEEE Access*, vol. 5, pp. 21355–21367, 2017, doi: 10.1109/ACCESS.2017.2748140.

[27] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 726–738, Sep. 2019, doi: 10.1109/TSC.2018.2826544.

[28] Q. Wu, H. Ge, H. Liu, Q. Fan, Z. Li, and Z. Wang, "A task offloading scheme in vehicular fog and cloud computing system," *IEEE Access*, vol. 8, pp. 1173–1184, 2020, doi: 10.1109/ACCESS.2019.2961802.

[29] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. M. Leung, "Distributed resource allocation and computation offloading in fog and cloud networks with non-orthogonal multiple access," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12137–12151, Dec. 2018, doi: 10.1109/TVT.2018.2872912.

[30] J. Almutairi and M. Aldossary, "A novel approach for IoT tasks offloading in edge-cloud environments," *J. Cloud Comput.*, vol. 10, no. 1, p. 28, Apr. 2021, doi: 10.1186/s13677-021-00243-9.

[31] Y. Yang, X. Chang, Z. Han, and L. Li, "Delay-aware secure computation offloading mechanism in a fog-cloud framework," in *Proc. IEEE Int. Conf Parallel Distrib. Process. Appl., Ubiquitous Comput. Commun., Big Data Cloud Comput., Social Comput. Netw., Sustain. Comput. Commun. (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, Dec. 2018, pp. 346–353, doi: 10.1109/BDCLOUD.2018.00061.

[32] K. Cui, B. Lin, W. Sun, and W. Sun, "Learning-based task offloading for marine fog-cloud computing networks of USV cluster," *Electronics*, vol. 8, no. 11, p. 1287, Nov. 2019, doi: 10.3390/electronics8111287.

[33] H. Sun, H. Yu, G. Fan, and L. Chen, "Energy and time efficient task offloading and resource allocation on the generic IoT-fog-cloud architecture," *Peer-Peer Netw. Appl.*, vol. 13, no. 2, pp. 548–563, Mar. 2020, doi: 10.1007/s12083-019-00783-7.

[34] H. A. Alharbi, T. E. H. Elgorashi, and J. M. H. Elmirghani, "Energy efficient virtual machines placement over cloud-fog network architecture," *IEEE Access*, vol. 8, pp. 94697–94718, 2020, doi: 10.1109/ACCESS.2020.2995393.

[35] H. Sun, H. Yu, and G. Fan, "Towards energy and time efficient resource allocation in IoT-fog-cloud environment," in *Service-Oriented Computing—ICSOC* (Lecture Notes in Computer Science), vol. 11434. Berlin, Germany: Springer, 2019, pp. 387–393, doi: 10.1007/978-3-030-17642-6_32.

[36] M. M. Bukhari, T. M. Ghazal, S. Abbas, M. A. Khan, U. Farooq, H. Wahbah, M. Ahmad, and K. M. Adnan, "An intelligent proposed model for task offloading in fog-cloud collaboration using logistics regression," *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–25, Jan. 2022, doi: 10.1155/2022/3606068.

[37] Z. Sun, H. Yang, C. Li, Q. Yao, D. Wang, J. Zhang, and A. V. Vasilakos, "Cloud-edge collaboration in industrial Internet of Things: A joint offloading scheme based on resource prediction," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17014–17025, Sep. 2022, doi: 10.1109/JIOT.2021.3137861.

[38] S. Atapattu, C. Weeraddana, M. Ding, H. Inaltekin, and J. Evans, "Latency minimization with optimum workload distribution and power control for fog computing," in *Proc. IEEE Wireless Commun. Netw. Conf.*, May 2020, no. I, pp. 1–6, doi: 10.1109/WCNC45663.2020.9120694.

[39] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016, doi: 10.1109/JIOT.2016.2565516.

[40] L. Rao, X. Liu, M. D. Ilic, and J. Liu, "Distributed coordination of internet data centers under multiregional electricity markets," *Proc. IEEE*, vol. 100, no. 1, pp. 269–282, Jan. 2012, doi: 10.1109/JPROC.2011.2161236.

[41] L. Chen, K. Guo, G. Fan, C. Wang, and S. Song, "Resource constrained profit optimization method for task scheduling in edge cloud," *IEEE Access*, vol. 8, pp. 118638–118652, 2020, doi: 10.1109/ACCESS.2020.3000985.

[42] J. Xu, Z. Hao, R. Zhang, and X. Sun, "A method based on the combination of laxity and ant colony system for cloud-fog task scheduling," *IEEE Access*, vol. 7, pp. 116218–116226, 2019.

[43] T. Apostolopoulos and A. Vlachos, "Application of the firefly algorithm for solving the economic emissions load dispatch problem," *Int. J. Combinatorics*, vol. 2011, pp. 1–23, Dec. 2011, doi: 10.1155/2011/523806.

[44] X.-S. Yang. (2010). *Nature-Inspired Metaheuristic Algorithms*. Luniver Press. [Online]. Available: https://www.researchgate.net/publication/235979455_Nature-Inspired_Metaheuristic_Algorithms

[45] M. Farid, R. Latip, M. Hussin, and N. A. W. A. Hamid, "Weighted-adaptive inertia strategy for multi-objective scheduling in multi-clouds," *Comput., Mater. Continua*, vol. 72, no. 1, pp. 1529–1560, 2022, doi: 10.32604/cmc.2022.021410.

[46] H. Ben Alla, S. Ben Alla, A. Touhafi, and A. Ezzati, "A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment," *Cluster Comput.*, vol. 21, no. 4, pp. 1797–1820, Dec. 2018, doi: 10.1007/s10586-018-2811-x.

[47] B. M. H. Zade, N. Mansouri, and M. M. Javidi, "Multi-objective scheduling technique based on hybrid hitchcock bird algorithm and fuzzy signature in cloud computing," *Eng. Appl. Artif. Intell.*, vol. 104, Sep. 2021, Art. no. 104372, doi: 10.1016/j.engappai.2021.104372.

FATEN A. SAIF received the Bachelor of Computer Science degree from Sana'a University, in 2008, the master's degree in computer science from University Putra Malaysia, in 2019, and the Ph.D. degree in computer science specializing in parallel and distributed system from the Faculty of Computer Science and Information Technology, University Putra Malaysia, in 2024. She is currently an Assistant Professor with the Gulf Colleges, Saudi Arabia. Her research interests include computer networks, workflow and task scheduling, cloud computing, the Internet of Things, optimization, workload allocation, and task offloading.

ROHAYA LATIP (Member, IEEE) received the Bachelor of Computer Science degree from Universiti Teknologi Malaysia, in 1999, and the M.Sc. degree in distributed systems and the Ph.D. degree in distributed database from Universit Putra Malaysia. She was the Head of the HPC Section, Universiti Putra Malaysia, from 2011 to 2012, and consulted the Campus Grid Project and also the Wireless for Hostel in Campus UPM Project. She is currently an Associate Professor with the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia. She is also the Head of the Department of Communication Technology and Networks. She is also a Co-Researcher with the Institute for Mathematic Research (INSPEM). Her research interests include big data, cloud and grid computing, network management, and distributed database.

ZURINA MOHD HANAPI (Member, IEEE) received the B.Sc. degree in computer and electronic system from the University of Strathclyde, Glasgow, U.K., in 1999, the M.Sc. degree in computer and communication system engineering from Universiti Putra Malaysia (UPM), Malaysia, in 2004, and the Ph.D. degree from Universiti Kebangsaan Malaysia, in 2011. She is currently an Associate Professor with the Department of Communication Technology and Network, Faculty of Computer Science and Information Technology, UPM, where she has been a Lecturer, since 2004. She has authored over 70 articles in the cited journals and conferences in the area of security and wireless sensor networks. Her current research interests include security, routing, wireless sensor networks, wireless networks, distributed computing, and cyber-physical-systems. She is a member of Malaysian Security Committee Research.

SHAFINAH KAMARUDIN received the Diploma degree in computer science and the Bachelor of Computer Science and Master of Science degrees from Universiti Putra Malaysia, in 2000, 2003, and 2009, respectively, and the Ph.D. degree from Universiti Kebangsaan Malaysia, in 2016. She is currently a Senior Lecturer with Universiti Putra Malaysia. She has actively written book chapters and published numerous papers in journals and conferences. Her current research include computer networks, management information systems, ICT for agriculture, and scholarship for teaching and learning (SoTL).

A. V. SENTHIL KUMAR has industrial experience for five years and teaching experience of 27 years. He has to his credit 20 book chapters, 214 papers in international and national journals 58 papers in international conferences in international and national conferences, and edited 11 books (IGI Global, USA). He is the Editor-in-Chief of many journals and a Key Member of India, Machine Intelligence Research Laboratories (MIR Laboratories). He is an editorial board member and a reviewer for various international journals. He is also a committee member for various international conferences. He is a Life Member of International Association of Engineers (IAENG) and Systems Society of India (SSI); a member of the Indian Science Congress Association, Internet Society (ISOC), International Association of Computer Science and Information Technology (IACSIT), and Indian Association for Research in Computing Science (IARCS); and a committee member of various international conferences.

AWADH SALEM BAJAHER received the bachelor's degree in computer science (systems and networking) from Universiti Tenaga Nasional, in 2016, and the master's degree in computer science from Universiti Putra Malaysia, in 2018, where he is currently pursuing the Ph.D. degree with the Faculty of Computer Science and Information Technology. His research interests include the Internet of Things, cloud computing, computer networks, workflow and task scheduling, and distributed computing.