IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 18 March 2023, accepted 11 April 2023, date of publication 13 April 2023, date of current version 12 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3266822

RESEARCH ARTICLE

Workload Allocation Toward Energy Consumption-Delay Trade-Off in Cloud-Fog Computing Using Multi-Objective NPSO Algorithm

FATEN A. SAIF, ROHAYA LATIP, (Member, IEEE), ZURINA MOHD HANAPI, (Member, IEEE), MOHAMED A. ALRSHAH, (Senior Member, IEEE), AND SHAFINAH KAMARUDIN

Department of Communication Technology and Network, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia

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

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

ABSTRACT The Internet of Things (IoT) generates massive data from smart devices that demand responses from cloud servers. However, sending tasks to the cloud reduces the power consumed by the users' devices, but increases the transmission delay of the tasks. In contrast, sending tasks to the fog server reduces the transmission delay due to the shorter distance between the user and the server. However, this occurs at the user end's expense of higher energy consumption. Thus, this study proposes a mathematical framework for workload allocation to model the power consumption and delay functions for both fog and clouds. After that, a Modified Least Laxity First (MLLF) algorithm was proposed to reduce the maximum delay threshold. Furthermore, a new multi-objective approach, namely the Non-dominated Particle Swarm Optimization (NPSO), is proposed to reduce energy consumption and delay compared to the state-of-the-art algorithms. The simulation results show that NPSO outperforms the state-of-the-art algorithm in reducing energy consumption, while NGSA-II proves its effectiveness in reducing transmission delay compared to the other algorithms in the experimental simulation. In addition, the MLLF algorithm reduces the maximum delay threshold by approximately 11% compared with other related algorithms. Moreover, the results prove that metaheuristics are more appropriate for distributed computing.

INDEX TERMS Energy consumption, delay, multi-objectives optimization, NPSO, optimization, pareto solutions, particle swarm optimization, workload allocation.

I. INTRODUCTION

Internet of Things (IoT) applications have contributed to modern society in various ways. Examples include smart manufacturing, smart homes, smart vehicles, intelligent transportation, and agriculture. The increasing growth of billions of IoT devices with sensors and actuators can be felt in the environment. Smart devices and humans can communicate instantly to create a more innovative world comprising of smart cities, transportation systems, health services, industries, homes, farming, and security. The proliferation of these connected IoT devices and sensors lead to the generation of a massive amount of data at every moment [1]. This generated data creates big data, which requires adequate storage, processing, and analysis to obtain valuable outputs that satisfy user requirements. It also requires a vast processing capability beyond the capacity of terminal devices, owing to their limited battery life, power computation, and storage capacity. The obtained information is then transferred to the data centres in the cloud for processing; this requires enormous computational resources and should be sufficiently distributed geographically to enable users

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

to send their requests everywhere and simple tasks should be executed on terminal devices [2]. Despite the enormous capabilities of cloud computing, owing to its high resource computation and scalability to process the enormous amount of data generated from IoT devices, delivering such a large amount of data via the Internet to the remote cloud data center consumes a large amount of bandwidth and leads to network delay [3]. Consequently, the Quality of Service (QoS) is degraded, particularly for time-sensitive tasks, owing to the significant distance between the cloud and IoT devices, which is a challenge. From this viewpoint, fog computing addresses the shortcomings of cloud computing by acting as a bridge between IoT devices and the cloud [4]. Fog computing is middleware containing multiple heterogeneous devices that are ubiquitously connected, such as base stations, routers, switches, surveillance cameras, and others that can be deployed in places such as power poles, vehicles, and commercial centres. These devices are decentralized to provide instant processing of raw data from the sensors. Furthermore, fog nodes can communicate within the same layer regarding processing problems [5]. One of the main advantages of fog computing is its proximity to end users. Thus, facilitating instant data processing and storage reduces the transmission delay for time-critical tasks, such as those in emergency health-monitoring applications [6]. Furthermore, it supports the mobility of terminal devices. In addition to the benefits of fog computing in processing time-sensitive tasks, it still needs to be improved [7]. Limitations in computation capacity and battery life are challenges experienced during task execution in fog nodes. The node cannot process complex or large tasks and characterize by high energy consumption. All these factors prevent fog computing from satisfying user requirements. Despite the limitations of cloud and fog nodes, exploiting their benefits by incorporating them into a paradigm called cloud-fog computing will be meritorious [3].

Cloud-fog computing is a promising platform that can serve billions of IoT devices by processing delay-sensitive data in applications that require real-time response [8]. It contains multiple local fog nodes and remote cloud centres to reduce delays and network traffic while increasing energy efficiency. Furthermore, it enables the processing of tasks from IoT applications via a suitable layer between fog and the cloud. Despite the significant advantages of cloud-fog computing, it still faces a critical challenge during allocation owing to the variety of task characteristics such as task input length, task computing unit, and degree of delay sensitivity. These factors are barriers to allocating workload in cloud-fog computing and need to be improved to guarantee QoS.

It is an essential technique that significantly reduces the financial cost incurred in data processing, especially when end users choose the appropriate resource to send their workload [7]. In addition, they play a leading role in determining the most suitable layer for processing tasks to meet user requirements according to their characteristics. Nevertheless, the variety of IoT-generated tasks makes workload allocation challenging because transmitting complex tasks to the cloud centres for processing increases network delay, which makes it even less energy-consuming. In contrast, executing time-sensitive tasks in the fog node reduces the delay but increases the energy consumption [9]. On the one hand, the dynamic nature of massive IoT-generated data leads to network overload, which makes it fail to meet the requirements of workload allocation. Thus, workload allocation in cloud-fog computing is critical and must be well-planned to overcome these problems, mainly because they are associated with IoT applications with various generated data requiring an instant response [10].

Workload allocation involves assigning tasks to the appropriate layer between fog nodes and the cloud server to guarantee efficient task execution and promote smart manufacturing in various respects. For instance, in intensive care, patients are connected to wearable devices, such as thermometers, blood pressure monitors, and electrocardiography, to detect vital signs. Hence, the CCTV was placed in a room to record the patient's motions. CCTV creates a film, is considered a complex task, and should be transferred to the cloud for analysis, processing, and storage. In a critical case, the sensor detects the sensed information and sends a direct notification to the doctor for an instant response or action. Such notifications require rapid processing and analysis from a shorter distance to save a patient's life. In this case, these critical signs are sent to the fog node by leveraging its proximity. The significant role of workload allocation when deciding where tasks must be sent for execution is shown in Figure 1.

Therefore, classifying the problem of workload allocation is the first step for improving and finding an appropriate approach to obtain the solution. Thus, workload allocation is an optimization problem. Typically, there are two standard techniques for solving optimization problems: optimal and heuristic approaches. Optimal approaches, such as simplex techniques and branch-and-bound guarantees, obtain optimal solutions compared to heuristic approaches. Nevertheless, they require a longer execution time, making them suitable only for small problems. Similarly, they must converge when used to solve complex problems. On the other hand, heuristic approaches provide near-optimal solutions with less complexity, which implies less execution time. Hence, they are ideal for solving complex problems [11]. Many heuristic approaches have been used extensively to solve complex tasks involving non-linear, high-dimensional, multi-model, and real-time systems [12]. Hence, there are various common approaches, such as the Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Simulated Annealing (SA), and Tabu Search (TS) [5]. This study adopts the PSO algorithm instead other metaheuristics approaches because it is much simpler than other metaheuristic approaches. In addition, it has a more rapid convergence owing to the use of fewer operators than the other approaches [13]. Having fewer operators means simple complexity which leads to a shorter execution time and a reduction in energy consumption. Moreover, the PSO is widely used in optimization functions such as workload

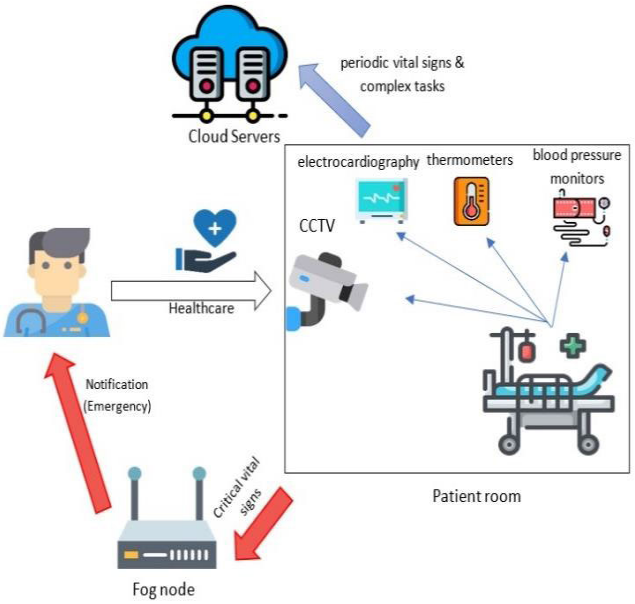

FIGURE 1. Allocating the workload from the patient to the cloud-fog layers.

and resource allocation, task scheduling, and fuzzy system control, and that aligns with the aim of this study. These advantages make the PSO the best option in the workload allocation with a variety of generating tasks from IoT applications.

However, the PSO algorithm has some shortcomings. It gets stuck in the local optimum solution owing to its decreasing population diversity, and particle reduction as the iterations proceed, affecting its ability to obtain the best solution [14]. Therefore, it is necessary to address this problem by finding a solution to increase the population diversity and overcome the limitation of falling in the local optimum solution, which is the main challenge of heuristic approaches, and leads to accomplishing the desired scheduling objectives (i.e., energy consumption, delay, cost, makespan, etc.).

At this point, Multiple-objective problems (MOP), rather than having a single objective, attract considerable critical attention. MOP can provide trade-off solutions that satisfy conflicting objectives. Thus, there are two main techniques to solve the MOP: the decomposition technique for generating non-Pareto solutions and finding Pareto optimal solutions through an individual run in the problem space [15]. A Pareto-optimal or non-dominated solution guarantees the best performance. If there is no best solution, it provides at least one best value, and the remaining solutions are of the same quality [16]. The essential problem of this study is the increasing demand for cloud and fog servers with the astronomical volume of data generated from IoT applications and the rapid increase in users' requests. Thus, transferring tasks from end users' devices to fog nodes will increase the energy consumed by fog nodes on the user side and reduce transmission delay owing to the short transmit distance between users and fog nodes. In contrast, sending tasks to the cloud servers will reduce the energy consumption of the end users but increase the transmission delay owing to the long transmission distance between end users and cloud servers. Therefore, this critical problem requires an effective trade-off strategy. Thus, the Pareto-optimal solution can significantly solve conflicting objectives while allocating workloads in cloud-fog computing, and this solution solves the main challenge in this study.

It is mandatory to propose a practical algorithm for reducing transmission delay and energy consumption during workload allocation and provide reasonable performance in reducing delay and energy consumption to enhance the processing of tasks uploaded by IoT devices and speed up the execution time in various applications such as smart manufacturing, smart homes, and smart transportation. Therefore, this study contributes significantly to the research on workload allocation in cloud-fog computing by deploying MOP. It proposes a trade-off technique between the energy consumed and the delay threshold. To the best of our knowledge, this is the first study that addresses using an NPSO algorithm for workload allocation in cloud-fog computing and proposes an MLLF algorithm to reduce delay threshold. The main contributions of this study are as follows:

- A mathematical framework with queue theory was developed to reduce power consumption and delay via efficient workload allocation.
- An optimal online algorithm called MLLF was proposed to reduce the maximum upper bound of the delay threshold.
- An NPSO algorithm was proposed to perform workload allocation for each layer using MOP to reduce energy consumption and delay.
- The experimental results show the superiority of the NPSO algorithm in reducing energy consumption and the NSGA in reducing transmission delay, Also, the MLLF algorithm in reducing the delay threshold by 11% compared with the First Comes First Serve (FCFS), scheduling to minimize lateness (STML), and (LLF) algorithms.

This paper is structured as follows: Section II reviews related works, discusses relevant algorithms and workload allocation techniques, and illustrates comparative studies on workload allocation in cloud-fog computing. Section III presents the mathematical framework of the cloud-fog integration system in detail. Section IV illustrates the proposed algorithm, and Section V presents a performance evaluation and describes the simulation environment. The simulation and numerical results are presented in Section VI. finally, the concluding remarks are presented in Section VII. along with potential future works.

II. RELATED WORK

Workload allocation in distributed computing has attracted significant attention over the years because it affects QoS. Thus, scholars have proposed various techniques to determine where and how to allocate the workload while maintaining the QoS of the target applications. This section briefly

reviews prior studies on workload allocation in cloud-fog computing. As mentioned in [17], a new method called BCM-XCS utilizes learning classifier systems (LCS called XCS) for workload allocation and resource management in cloud-fog computing. It uses a GA algorithm and considers four dimensions: workload, delay, power consumption, and battery status. The goal was to strike a balance between the power consumption at the fog layer and the delay at the cloud layer. The simulation results show that the proposed algorithm reduces the processing delay by about 42%. Also, it can recharge the renewable batteries utilized at the edge layer around 18 % more than comparing algorithms. However, adopting AI requires massive computation and memory resources that are more appropriate when executing at the task on a cloud platform and not on a cloud-fog computing platform. In addition, GA has more parameters and, thus, a higher complexity and computation time than other heuristic approaches. Similarity, [18] proposed a modified energy-efficient workload allocation model within the IoT-fog-cloud framework to decrease power consumption and delay using the NSGA II algorithm. Solar or wind energy can be used as renewable energies for complementary power supply to reduce the energy consumption of fog devices. The experiment results present the proposed algorithm's ability to reduce the energy consumption of fog nodes and reduce task delay of IoT devices. The NSGA II algorithm also reduces energy consumption and delay for various workloads. However, offloading of several edge nodes must be considered. In addition, the central controller requires information on all fog nodes. As others have highlighted, [3] proposed a delay-aware online workload allocation and scheduling algorithm (DAOWA) for cloud-fog computing to decrease task service delay while satisfying QoS requirements. The simulation results reveal that the DAOWA algorithm can reduce the task service delay and optimize the policy of allocating the workload respecting the system status. However, this study excluded energy consumption, which is a significant consideration in cloud-fog computing. The optimal workload allocation in cloud-fog computing was introduced in [4] to trade-off delay and energy consumption by dividing the primary problem into three subproblems using an approximation method. The numerical result demonstrates that the fog node complements the cloud and does not substitute it. Also, the fog node can improve the performance of cloud computing by sacrificing humble computation resources to maintain the communication bandwidth and minimize transmission delay. However, this mechanism is characterized by an overhead. As reported in [19] workload allocation in cloud-fog computing with an energy-delay trade-off was studied. This problem is formulated as a MOP that is solved using the NSGA II algorithm. The simulation result illustrates the proposed algorithm's ability to reduce energy consumption and delay. Also, allocating workload to fog devices is about 25%. However, GA consumes more energy during computation than other meta-heuristic approaches because it adopts more parameters and does not consider other state-of-the-art comparative algorithms for evaluation. Referring to [20] proposed the idea of distributing workloads in an IoT edge-cloud computing system while guaranteeing energy efficiency and minimal delay via a delay-based workload allocation algorithm (DBWA). The algorithm uses the Lyapunov drift-plus-penalty theory to reduce the system's energy consumption with a granular delay for each job. The result shows that the proposed algorithm can better balance workload distribution than other algorithms. At the same time, it reduces the delay by about 40%. It reduces the energy consumption of the vehicular network, this strategy attempt to reduce battery consumption by exploiting renewable energies. However, the non-availability of green energy results in sending all requests to the cloud, and edge servers lose their ability to reduce delays. Similarly, the authors of [21] proposed a queue-theory-based workload allocation in cloud-fog computing to achieve a trade-off between energy consumption and delay. It aims to determine an optimal workload when new tasks are generated following the dynamic nature of IoT applications. The problem was solved using non-linear programming to reduce energy consumption while utilizing (STML) algorithm to reduce the delay threshold. They proposed a cloud-fog cooperation algorithm that considers computational energy. Even more, they presented a task offloading algorithm for nodes that run off to another node in the same or upper layer. The numerical analysis presents that energy consumption was reduced around 22%, while the delay reduces by 12.5 %, less than the compared approach FCFS algorithm. However, optimal optimization requires higher computation and energy consumption. According to a study in [22] proposed an energy-efficient model for workload allocation and computation for multiple VM configurations. Discussing the trade-off between energy cost and SRP, they theoretically demonstrated an ideal SRP that achieves the lowest energy cost. They determined the closed-form conditions to reach this minimal energy point and investigated the joint task allocation and computation frequency configuration challenge. The simulation was conducted on synthetic and real-world workload trace data. The results reveal that the proposed algorithm can perform minimum energy consumption compared to conventional fixed-SRP configuration schemes. However, the scalability of VMS was challenging in this study. Adding more VMs increase the energy cost, and the maintenance and orchestration of VMs make it challenging to reduce the delay and overhead problems.

It is evident from previous studies that standard techniques implemented in investigating workload allocation in cloud-fog computing are solved by an optimal technique or artificial techniques that both have robust performance. However, these methods have high complexity that leads to higher resource computation and energy consumption, which could be unfavourable in a cloud-fog computing environment that requires minimum delay and energy consumption. However, some studies have utilized GA, a popular meta-heuristic approach. Generally, the meta-heuristic approach provides a reasonable high-quality solution with lower complexity than

the optimal solution, which guarantees a reduction in energy consumption. Nevertheless, the GA consumes more energy than other meta-heuristic approaches because it requires more parameters to obtain the results. This factor leads to more computation time and energy consumption and does not meet the service requirements required for processing IoT applications in a cloud-fog computing environment.

From this perspective, finding a suitable approach to meet the platform's demands is essential. Thus, the PSO algorithm has a minimal number of parameters while providing reasonable solutions compared to other metaheuristics, making it the ideal option for workload allocation in cloud-fog. Therefore, this study employs a MOP approach (NPSO algorithm) because it requires fewer parameters, which leads to reduced execution time and decreased energy consumption, which are the main objectives of this study. In addition, there is the ability of optimal solutions to reduce the delay. Thus, this study proposes an optimal online MLLF algorithm to reduce the maximum delay threshold and guarantee that the delay requirements are met. Thus, reducing the response time and processing more tasks from the IoT devices is necessary. Based on the above points, this study meets the requirements for reducing delay and energy consumption.

III. CLOUD-FOG SYSTEM AND PROBLEM FORMULATION

The cloud-fog system proposed in Figure 2 was adopted because of its simplicity and feasibility of implementation [22]. The system comprises M end devices that communicate with each other via wireless links, F fog nodes, and C cloud servers. The role of fog nodes is to provide services to end-user devices. Cloud servers are responsible for fog nodes. While the monitoring center moderates task scheduling in the cloud, thereby reducing communication costs. Whereas, the traffic model captures a variant of the power and capacity. The modelling of the entire cloud-fog computing system using queue theory. 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 workload allocation procedure is described as follows: if the arrival rate of the generated tasks is less than the service rate of the end devices, then process tasks are at the end devices. However, they are sent to the fog node if the arrival rate of the tasks exceeds the service rate of the devices. Otherwise, the tasks are sent to the cloud servers. The average task arrival rate followed a Poisson process that are utilized in scenarios where when the continuous occurrences of specific events which appear to occur at the specific rate, but randomly [23]. Also, the wireless channels between terminal devices and fog required are exponentially distributed. The task is characterized by the number of tasks, length of the task input, task deadline, flag of task execution, and required computing unit by task. Table 1 lists the primary notations used in this study.

A. DELAY THRESHOLD DEFINITION

In this study, the delay threshold was modelled as $D_{total} \leqq D_{max}$. Hence, to reduce the delay, we reduce because it affects

Algorithm 1 MLLF Algorithm

BEGIN

For scheduling instant x_i
For each $r \in R\,(x_i)$ do
  Calculate Laxity $(x_i) = D_i - (T_i + C_i)$
End for
$n \leftarrow$ adding $(R\,(x_i)\,,$ laxity $(x_i))$
for $(i = 0; i < n; ++i)\{$
  for $(j = i + 1; j < n; ++j)\{$
IF $(\text{num}\,[i] > \text{num}\,[j]$
  $a = \text{num}\,[i]\,;$
  $\text{num}\,[i] = \text{num}\,[j]\,;$
  $\text{num}\,[j] = a;\ \}\ \}$
  $\text{rlist} = \text{num}\,[j]$
Else IF $(\text{num}\,[i] = \text{num}\,[j])\{$
  Calculate $\text{WT}(x_i) = x_{begin} - x_{arrive}$
  $\text{WT}(x_j) = x_{begin} - x_{arrive}$
rlist = assign priority to the max $(\text{WT}\,(x_i)\,, \text{WT}\,(x_j))\,;\}$
  End For
End For
  rlist $\leftarrow$ sort $(R\,(x_i)\,,$ laxity $(x_i))$
  Return rlist

END

FIGURE 2. The architecture of cloud-fog computing.

the task completion time and workload allocation. Thus, it is essential to determine an algorithm that can guarantee at reduction in D_{max}. Generally, in a real-time system, there are two categories of scheduling: online and offline [24]. In offline scheduling, the decision (resource allocation, execution time, deadline, and priority allocation) is made before the system start-up time because it has complete knowledge of the tasks for execution. Meanwhile, in online scheduling, the decision is made during the system run time by assigning priority to tasks, which is more appropriate for an IoT environment with a dynamic nature and unpredictable tasks. Hence, there are two online algorithms: the Earliest Deadline

TABLE 1. Summary of notations [21].

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
D_{max}	Upper bound of the delay threshold (a unit time)
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.
X^*_{ed}, Y^*_{fog}, Z^*_{cloud}	Optimal workload of end device layer, fog node layer, cloud server layer.

First (EDF) algorithm and (LLF) algorithm. This study chose the LLF algorithm because it is more dynamic than the EDF. Fixed deadlines characterize the EDF, whereas the deadlines of the LLF change over time and are more flexible to IoT tasks. This study proposes a new algorithm, namely MLLF, an online-based algorithm derived from LLF.

The LLF algorithm is an optimal algorithm based on calculating the laxity value for each task and then assigning the priority according to the minimum values to guarantee reduce of the delay, the maximum tolerable delay that tasks can experience. However, The LLF performs poorly when more than one task has equal laxity values. Thus, an enhancement is required to tackle this challenge while finding a way to reduce delay in the system, which leads to improved response time, as a higher priority is assigned to the maximum waiting time in the queue to release tasks from the system.

Algorithm 1 presents the proposed MLLF, initially calculate the laxity of each task, where D_i is the task deadline, T_i is the current system time, C_i represents the worst-case execution time (WCET). After that, sorting the tasks in ascending order based on the laxity values in the store called (n). Then, sorting the laxity value from store n in ascending order. In case, the equal values of laxity the algorithm improve it by calculating the waiting time of tasks with equal laxity value. After that, assign the priority to the task with maximum waiting time and add it to the *r* list that indicates the task laxity list, where the waiting time (WT) is the duration between the arrival of the requested task in the queue of the server and when it starts processing. x_{begin} represents when the task starts execution, and x_{arrive} is when the task arrives in the queue. This technique can fasten the responding time.

The validation of the MLLF performance is provided in the Results section.

B. END USER DEVICE

We assume the service rate μ of the end-user device *i* follows an exponential distribution, 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 energy 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

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

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

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

C. FOG NODE

The task queue in fog node j is modelled as M/M/C. The energy 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 [4]. 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 energy 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:

$$P^j_{fog} \triangleq aY^{j^2}_{fog} + bY^j_{fog} + c \tag{3}$$

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 follows:

$$F_{comm}\left(I_g\right) \triangleq \{\gamma I_g \varepsilon I_g, \begin{array}{l} u_t \in cloud, \\ u_t \in fog \end{array} \tag{5}$$

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:

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

D. 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 energy 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 [25]. 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

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

IV. MULTI-OBJECTIVES OPTIMIZATION PROBLEM (MOP)

The 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 $x_1 \prec x_2 \Longleftrightarrow \forall_i f_i(x_i) \leq f_i(x_i) \wedge \exists_i (x_i) \prec f_i(x_i)$. 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.

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

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.

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

V. THE PROPOSED ALGORITHM NPSO

PSO was developed by Kennedy and Eberhart and is built on the behaviour of the animals' social system [26]. It is a population-based evolutionary algorithm known as swarm, where; each swarm has many particles indicated as a solution. Every particle has previous local information regarding the best solution and the global best solution found by the entire swarm. The search space of PSO has D dimensions, and the technique utilized is the velocity vector for every particle without selection operators, which increases the computation time. Furthermore, the PSO particles moved Towards the search space. PSO relies on the direction and velocity to select at particle's motion. However, the main shortcoming of PSO is that it traps the local optimal solution [14]. Thus, the motivation is to improve the limitation of PSO by proposing NPSO algorithm by adapting a mutation operator to increase the diversity of the particle population to avoid falling into the local optimum search and improve its exploration in the search space to facilitate finding the solution to the MOP. Each particle had its own position and velocity. The velocity of the *i th* particle at the *t th* iteration is denoted as $v^t_i = v^t_1, v^t_2, \ldots\ldots\ldots v^t_{nd}$ while its position is $s^t_i = s^t_1, s^t_2, \ldots\ldots\ldots s^t_{nd}$. D denotes the dimension of the search space. $Pbest^t_i$ refers to the best previous position based on the best fitness value determined by the *i th* particle at the *t th* iteration. This is indicated by $Pbest^t_i = Pbest^t_1, Pbest^t_2, \ldots\ldots\ldots Pbest^t_{nd}$. $Gbest^t_i$ is the global best particle position found in the entire generation based on the fitness function. It depends on the optimization problem and it chosen from the top of the external archive. The global best particle is denoted as $Gbest^t_i = Gbest^t_1, Gbest^t_2, \ldots\ldots\ldots Gbest^t_{nd}$. The main equation of the NPSO is described as follows. The NPSO strategy is shown in Algorithm 2.

A. THE VELOCITY OF EACH PARTICLE IS UPDATED AS FOLLOWS

$$v^{t+1}_i = w \cdot v^t_i + C_1 \cdot r_1 \left(Pbest^t_{id} - s^t_{nd}\right) + C_2 \cdot r_2 (Gbest^t_{nd} - s^t_{nd}), \tag{9}$$

r_1 and r_2 are random numbers in $[1, 0]$. C_1 C_1 is the cognitive coefficient with respect to the prior exploration of particles and C_2 is a social coefficient that depends on the experience of swarms. Here, w refers to the inertia weight, which is formulated as a non-linear decreasing function to balance the exploration of the global and local search space. A larger value of inertia weight indicates a greater global search ability (*i*.e., searching for a new area), whereas a smaller value of inertia weight indicates a greater local search area (*i*.e., current search area) [27]. This study adopted a new technique [28] to improve the inertial weight of the algorithm as follows

$$w(t) = \omega_{max} * \frac{1}{1 + (t/H_1)^{H_2}} + \omega_{min} \tag{10}$$

where $\omega(t)$ varies with the number of iterations. H_1 is the max. number of iterations, H is 10, and t denotes the number of current iterations. Where ω_{max} is 0.5, ω_{min} is 0.4.

B. THE POSITION OF EACH PARTICLE IS UPDATED AS FOLLOWS

$$s^{t+1}_{nd} = s^t_{nd} + v^{t+1}_i \tag{11}$$

s_{nd}^{t} is the current position of particle i at time t and s_{nd}^{t+1} is the new position of the particle.

C. FITNESS FUNCTION

The fitness function evaluates the movement of a particle using optimization problems to obtain the Pareto solution. The fitness value indicates the quality of the solution and the essential metrics for guaranteeing the diversity of the solution [29]. Thus, the fitness function is used to select a solution based on the minimum delay and energy consumption. The fitness function computes using the weighted sum approach that based on gathering various objectives into single objective function, that can be calculated by

$$\mathrm{F(n)} = \alpha_1 * \mathrm{Energy} + (1 - \alpha_2) * \mathrm{delay} \tag{12}$$

Hence, Energy and delay are scaled and weighted based on their importance and incorporate to provide the objective function. Where α is the energy-delay balance factor and it value varies between [0, 1]. It is a significant factor to assign the priority to the objective. In the scenario of fog-cloud computing, we focused on allocating workload depending on the delay and energy consumption simultaneously. Thus, the total energy and total delay balance coefficient. $\alpha_1 = \alpha_2 = 0.5$ means that the total energy and total delay are assigned the same priority in the optimization procedure. In this article, a general problem is considered that gives equal preference to both the objectives.

D. ADAPTIVE MUTATION

Incorporating a mutation operator with PSO plays an essential role in maintaining the diversity of the particle population, which overcomes the challenge of having PSO trapped in local optimal solutions. The mutation operator randomly changes the position of the particle, thereby enhancing the local search capabilities. Moreover, it improves the accuracy of the obtained solutions, as shown in Figure 3.

E. EXTERNAL ARCHIVE

The external archive adopted in the proposed algorithm for storing Pareto solutions was obtained from each iteration until the termination condition was satisfied, which significantly affected the search performance [13]. It maintains the diversity of the solutions and a uniform distribution based on the Pareto front. Typically, the external archive is pruned if the number of non-dominated solutions exceeds its maximum size based on the crowding distance (CD). The external archive strategy is described in Algorithm (3).

F. CROWDING DISTANCE (CD)

The purpose of CD is to maintain the spread of solutions uniformly along the Pareto front. When the external archive is full, the crowding distance evaluates the solutions and determines which solutions should be deleted and which must remain in the external archive. Because we cannot determine the priority among solutions in the archive, we adopt the CD

Algorithm 2 NPSO Algorithm

BEGIN

1. set (number of Particle NP, size of particle swarm N)
2. set EA = Ø; // initialize an empty archive, to restore non dominated solutio
3. initialize randomly $\{v_i^t, s_i^t, Pbest_{id}^t, Gbest_{id}^t\}$
4. For i = 1 to N (N the size of particle swarm)
5. Update $v_i^{t+1} = w.v_i^t + C_1.r_1\left(Pbest_{id}^t - s_{nd}^t\right) + C_2.r_2(Gbest_{nd}^t - s_{nd}^t)$
6. Update $s_{nd}^{t+1} = s_{nd}^t + v_i^{t+1}$
7. Compute F(n) = α * Energy + (1 − α) * delay
8. Add the non-dominated solutions found in S into EA // algorithm 2
9. Update $Pbest_{id}^t$
10. If $(f\left(Pbest_{id}^t\right) < F(n)$
    Then $f\left(Pbest_{id}^t\right)$
11. Elseif $(f\left(Pbest_{id}^t\right) > F(n)$
    Then F(n)
12. Else (random select
13. End for
14. Update $Gbest_{id}^t$ from EA// from external archive
15. For i = 1 to N (N the size of particle swarm)
16. Update $v_i^{t+1} = w.v_i^t + C_1.r_1\left(Pbest_{id}^t - s_{nd}^t\right) + C_2.r_2(Gbest_{nd}^t - s_{nd}^t)$
17. Update $s_{nd}^{t+1} = s_{nd}^t + v_i^{t+1}$
18. Mutate particle
19. Compute F(n) = α * Energy + (1 − α) * delay
20. Update the External Archive E
21. Update the $Pbest_{id}^t$
22. End For
23. Retain the best pareto solution in external archive EA

END

to choose solutions with a higher crowding distance. The primary purpose of CD is to limit the archive size and reduce the complexity of the algorithm by arranging the particles in descending order. In this case, it is necessary to choose the minimum value, $Gbest_{nd}^t$. Crowding distance can be defined as

$$CD = \frac{f_{i+1}^t - f_{i-1}^t}{f_{max}^t - f_{min}^t} \tag{13}$$

where f_{max} and f_{min} indicate the maximum and minimum values of particles in each iteration, respectively, and t. f_i^t indicates the particle weights of the i th and i+th particles in the iteration. where f_{i+1}^t represents the next objective function value of the particle, f_{i-1}^t represents the previous objective function value of this particle.

VI. SIMULATION SET-UP AND RESULT

This section presents the verification of the effectiveness of the proposed NSPO algorithm for workload allocation. The objective function are the delay and energy consumption. MATLAB R2018b was used for the simulation to compare the NPSO algorithm with a non-linear programming

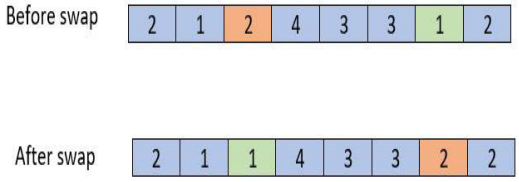

FIGURE 3. Mutation operation.

Algorithm 3 External Archive E

BEGIN

1. For $i = 1$ to N (population size)
2. IF EA=0 $|EA < N$
3. EA ← F(n)// adding the pareto solutions
4. Else
5. Calculate the CD
6. Delete minimum CD
7. End if
8. End For
9. Sorting the Non-dominated solutions in descending order

END

TABLE 2. Simulation key parameters [21].

Parameters	Value
μ	4.6
λ	3.2
P_{ed}	23
a	1
b	2.4
e	3.5
a_k	[3.206 4.485 2.370]
b_k	[68 53 70] × 10^{-20}
workload	[30 50 90 150 200]
Population size	50
Number of iterations	30
r_1, r_2	0.5
C_1, C_2	1.5

approach [21], MOPSO-CD [30], and NSGA-II algorithms. The cloud-fog scenario considers seven end-users, three fog nodes, and one cloud server. The extensive simulation was conducted with five groups of tasks, and their total workloads are 30, 50, 90, 150, and 200, on a computer equipped with Windows 11, CPU Core i7, and 8 GB of RAM. The length of the tasks is randomly generated because they cannot be predicted in reality. Validating the proposed algorithm by performing 30 independent runs, for each run, the best obtained delay and energy consumption were recorded to get average result [31]. The main parameters for validating the performance of the algorithms are summarized in Table 2 referring to [20], and NPSO parameters referring to [32] that indicate and the represents the suitable settings to provide the best convergence rate.

First, it is obvious from Figure 4 that the quantitative comparison in terms of delay illustrates a set of tasks scheduled by the FCFS, STML, LLF, and MLLF algorithms. The X-axis indicates 100 tasks divided into 10 groups, and the y-axis represents delay. The simulation results show that the maximum delay incurred by the FCFS is approximately 125.10, STML

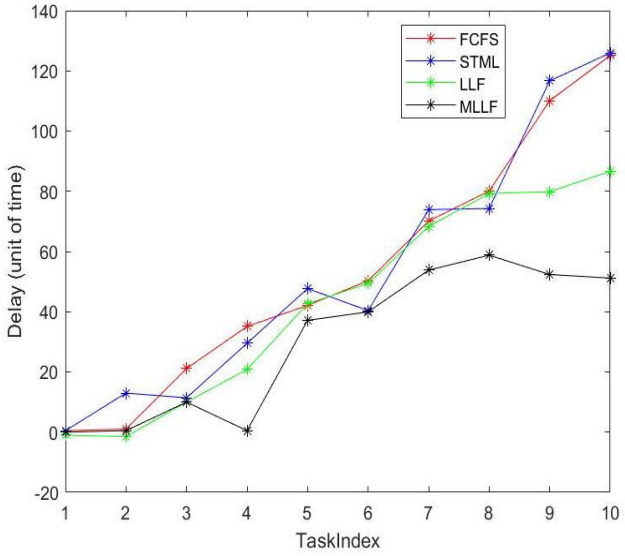

FIGURE 4. Delay of the FCFS-STML-LLF-MLLF.

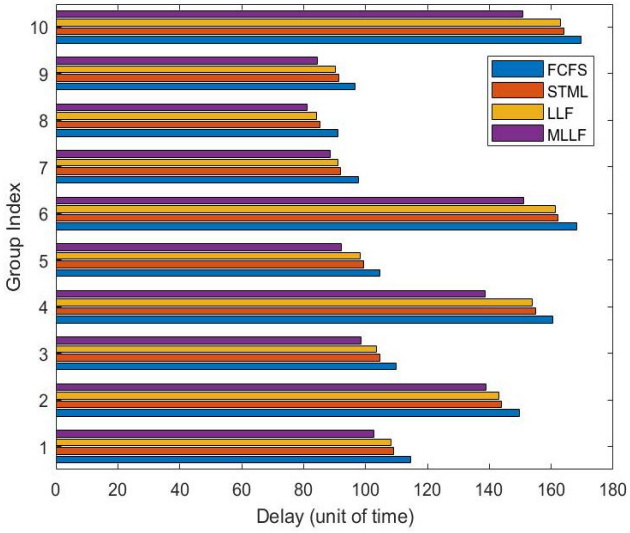

FIGURE 5. Max delay of FCFS- STML-LLF-MLLF.

is approximately 126.00, LLF is approximately 86.67, and MLLF is approximately 51.71. From the simulation experiments, it is evident that the MLLF outperforms the compared algorithms with respect to reducing the delay to the minimum value, which can guarantee a reduction in transmission delay.

The reduction in the maximum delay of all approaches related to 10 groups of tasks is shown in Figure 5, where the X-axis indicates the FCFS, STML, LLF, and MLLF algorithms, and the Y-axis Depicts the results of 100 tasks divided into ten groups. A comparison among various algorithms was made to determine the maximum delay. We observed a stable reduction in delay when the number of tasks executed was increased, and MLLF performed best in reducing the maximum delay by approximately 11% compared to the other algorithms. This affects the task completion time and effectively allocates the workload.

TABLE 3. Result of energy consumption based on various workload.

	Workload Size				
Algorithms	**30**	**50**	**90**	**150**	**200**
MOPSO-CD	**0.1354**	**0.4189**	**1.339**	**3.445**	**6.3**
NPSO	**0.1354**	**0.4147**	**1.304**	**3.365**	**5.96**
NSGA-II	**0.2084**	**0.6497**	**1.942**	**3.48**	**8.377**
Nonlinear Programming	**0.1385**	**0.4472**	**1.485**	**3.654**	**7.437**

TABLE 4. Result of delay based on various workload.

	Workload Size				
Algorithms	**30**	**50**	**90**	**150**	**200**
MOPSO-CD	**386.4**	**1009**	**1775**	**2374**	**3701**
NPSO	**386.4**	**880**	**1214**	**1445**	**2340**
Nonlinear Programming	**456.7**	**1238**	**2144**	**3137**	**4641**
NSGA-II	**32.63**	**101.7**	**304**	**742.6**	**1312**

Figure 6 compares the proposed algorithm, NPSO, non-linear optimization [20], MOPSO-DC, and NSGA-II regarding the energy consumption when D_max = 100. Recall the average delay threshold of the MLLF in Figure 5; that is, the normal delay threshold is $(-\infty, 100)$. The workload was divided into five groups (30, 50, 90, 150, and 200). The results show the superiority of the NPSO algorithm over comparison techniques in reducing energy consumption, followed by MOPSO-CD with a slight difference during a small workload size. Then, the obvious difference appears with the increasing workload size, which proves the effectiveness of the NPSO in reducing energy consumption with an increase in the number of workload groups. The worst performance in reducing energy consumption goes to NSGA-II because it has selection and mutation operations that increase the algorithm's complexity and consume more energy. See Table 3.

Figure 7 shows the performance comparison among the NPSO algorithm with a non-linear optimization approach, MOPSO-CD, and NSGA-II algorithms in terms of delay when D_max = 100 is the average delay threshold of the MLLF in Figure 3. The normal delay threshold is $(-\infty, 100)$. Moreover, the delay threshold at D_max=100 guarantees the completion of all tasks in time. In the simulation. The workload was divided into five groups (30, 50, 90, 150, and 200). It can be observed that the lowest task transmission delay is for the NSGA-II due to its ability to process the workload in the shortest time based on two operators. In contrast, followed by the NPSO algorithm over non-linear optimization and MOPSO-CD in reducing delay due to applying the MLLF algorithm in reducing the delay threshold when allocating the workload utilizing the NPSO, which results in a lower delay when allocating the workload compared to MOPSO-CD. See Table 4.

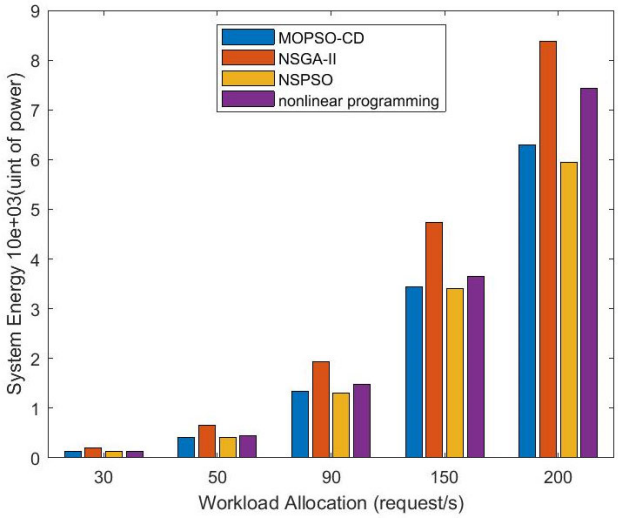

FIGURE 6. Energy consumption comparison among NPSO with state of art techniques in $D_{max} = 100$.

VII. DISCUSSION AND CONCLUSION

This paper proposes a new MOP approach, namely the NPSO algorithm for workload allocation in cloud-fog computing. The study adopted the mathematical framework for describing the energy consumption and delay functions with queue theory. We solve the delay optimization problem by proposing the MLLF algorithm to reduce the delay threshold and an external archive to store the non-dominated solution from the Pareto optimal solutions. The simulation results prove that

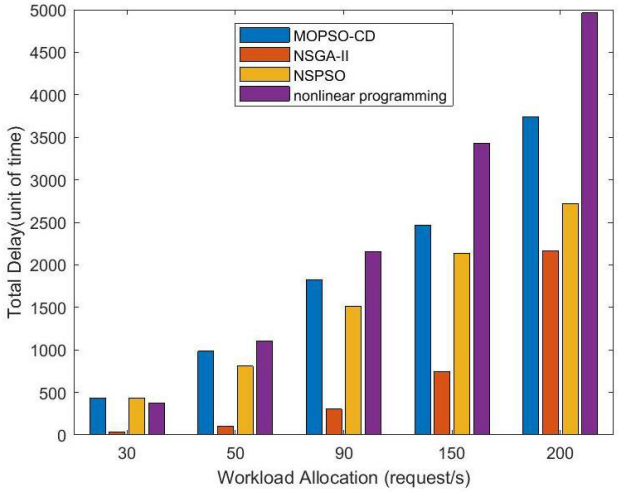

FIGURE 7. Delay comparison among NPSO with state of art techniques in $D_{max} = 100$.

the proposed MLLF effectively reduces the delay compared to the other approaches (FCFS, STML, and LLF) by approximately 11%. Moreover, Pareto optimal solution is adopted to find the optimal workload by reducing energy consumption and delay.

Furthermore, the proposed NPSO algorithm can reduce energy consumption compared to Nonlinear programming, NSGA-II, and MOPSO-CD methods. While reducing the transmission delay, the best performance goes to NGSA-II due to the operations that accelerate the processing of workload and lead to reducing the transmission delay. Then, the NPSO algorithm achieves better performance than MOPSO-CD owing to utilizing the Proposed MLLF algorithm to reduce the delay threshold, which contributes to reducing the transmission delay. The worst performance was for Nonlinear programming in reducing transmission delay and energy consumption. Overall, the NPSO algorithm can perform a reasonable performance in balancing the delay and energy consumption during the allocating of the workload between fog and cloud computing. Furthermore, the results prove that the metaheuristics approaches are more appropriate for distributed computing than the optimal approach.

The main limitations of this study are considering the heterogeneity of fog nodes, conducting experiments in a real-world system instead of simulation, and considering more objectives to solve many-objective optimization. In future work, we will overcome the limitations of this study for enhancement. In addition, adopting artificial intelligence to predict the allocation workload.

ACKNOWLEDGMENT

This study supported technically and financially by Universiti Putra Malaysia.

REFERENCES

[1] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "IoT Middleware: A survey on issues and enabling technologies," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 1–20, Feb. 2017, doi: 10.1109/JIOT.2016.2615180.

[2] R. Tyagi and S. K. Gupta, "A survey on scheduling algorithms for parallel and distributed systems," in *Advances in Intelligent Systems and Computing*, vol. 718. Singapore: Springer, 2018, pp. 51–64, doi: 10.1007/978-981-10-7656-5_7.

[3] Li, Guo, Ma, Mao, and Guan, "Online workload allocation via fog-fog-cloud cooperation to reduce IoT task service delay," *Sensors*, vol. 19, no. 18, p. 3830, Sep. 2019, doi: 10.3390/s19183830.

[4] 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.

[5] R. O. Aburukba, M. AliKarrar, T. Landolsi, and K. El-Fakih, "Scheduling Internet of Things requests to minimize latency in hybrid fog–cloud-computing," *Future Gener. Comput. Syst.*, vol. 111, pp. 539–551, Oct. 2020, doi: 10.1016/j.future.2019.09.039.

[6] Z. He, Q. Zhao, H. Mei, and L. Peng, "Optimal scheduling of IoT tasks in cloud-fog computing networks," in *Proc. Int. Symp. Artif. Intell. Robot.*, 2021, pp. 103–112, doi: 10.1007/978-3-030-56178-9_8.

[7] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2017, doi: 10.1109/ACCESS.2017.2778504.

[8] 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.

[9] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, Feb. 2018, doi: 10.1109/JIOT.2017.2780236.

[10] M. Han, T. Zhang, Y. Lin, and Q. Deng, "Federated scheduling for typed DAG tasks scheduling analysis on heterogeneous multi-cores," *J. Syst. Archit.*, vol. 112, Jan. 2021, Art. no. 101870, doi: 10.1016/j.sysarc.2020.101870.

[11] R. O. Aburukba, T. Landolsi, and D. Omer, "A heuristic scheduling approach for fog-cloud computing environment with stationary IoT devices," *J. Netw. Comput. Appl.*, vol. 180, Apr. 2021, Art. no. 102994, doi: 10.1016/j.jnca.2021.102994.

[12] F. A. Saif, R. Latip, M. N. Derahman, and A. A. Alwan, "Hybrid metaheuristic genetic algorithm: Differential evolution algorithms for scientific workflow scheduling in heterogeneous cloud environment," in *Proc. Future Technol. Conf.*, vol. 3, 2023, pp. 16–43, doi: 10.1007/978-3-031-18344-7_2.

[13] Y. Xie, Y. Zhu, Y. Wang, Y. Cheng, R. Xu, A. S. Sani, D. Yuan, and Y. Yang, "A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud–edge environment," *Future Gener. Comput. Syst.*, vol. 97, pp. 361–378, Aug. 2019, doi: 10.1016/j.future.2019.03.005.

[14] A. M. Manasrah and H. B. Ali, "Workflow scheduling using hybrid GA-PSO algorithm in cloud computing," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–16, Jan. 2018, doi: 10.1155/2018/1934784.

[15] 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.

[16] J. Sun, H. Li, Y. Zhang, Y. Xu, Y. Zhu, Q. Zang, Z. Wu, and Z. Wei, "Multiobjective task scheduling for energy-efficient cloud implementation of hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 587–600, 2021, doi: 10.1109/JSTARS.2020.3036896.

[17] M. Abbasi, M. Yaghoobikia, M. Rafiee, A. Jolfaei, and M. R. Khosravi, "Efficient resource management and workload allocation in fog–cloud computing paradigm in IoT using learning classifier systems," *Comput. Commun.*, vol. 153, pp. 217–228, Mar. 2020, doi: 10.1016/j.comcom.2020.02.017.

[18] M. Abbasi, E. Mohammadi-Pasand, and M. R. Khosravi, "Intelligent workload allocation in IoT–fog–cloud architecture towards mobile edge computing," *Comput. Commun.*, vol. 169, pp. 71–80, Mar. 2021, doi: 10.1016/j.comcom.2021.01.022.

[19] M. Abbasi, E. M. Pasand, and M. R. Khosravi, "Workload allocation in IoT-fog-cloud architecture using a multi-objective genetic algorithm," *J. Grid Comput.*, vol. 18, no. 1, pp. 43–56, Mar. 2020, doi: 10.1007/s10723-020-09507-1.

[20] M. Abbasi, M. Yaghoobikia, M. Rafiee, M. R. Khosravi, and V. G. Menon, "Optimal distribution of workloads in cloud-fog architecture in intelligent vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4706–4715, Jul. 2021, doi: 10.1109/TITS.2021.3071328.
[21] 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.
[22] W. Zhang, Z. Zhang, S. Zeadally, H.-C. Chao, and V. C. M. Leung, "Energy-efficient workload allocation and computation resource configuration in distributed cloud/edge computing systems with stochastic workloads," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1118–1132, Jun. 2020, doi: 10.1109/JSAC.2020.2986614.
[23] 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. (WCNC)*, May 2020, pp. 1–6, doi: 10.1109/WCNC45663.2020.9120694.
[24] N. Ahmad, N. Javaid, M. Mehmood, M. Hayat, A. Ullah, and H. A. Khan, *Fog-Cloud Based Platform for Utilization of Resources Using Load Balancing Technique*, vol. 22. Cham, Switzerland: Springer, 2019, doi: 10.1007/978-3-319-98530-5_48.
[25] 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.
[26] Q. S. Khalid, S. Azim, M. Abas, A. R. Babar, and I. Ahmad, "Modified particle swarm algorithm for scheduling agricultural products," *Eng. Sci. Technol., Int. J.*, vol. 24, no. 3, pp. 818–828, Jun. 2021, doi: 10.1016/j.jestch.2020.12.019.
[27] W. Lin, Z. Lian, X. Gu, and B. Jiao, "A local and global search combined particle swarm optimization algorithm and its convergence analysis," *Math. Problems Eng.*, vol. 2014, pp. 1–11, Oct. 2014, doi: 10.1155/2014/905712.
[28] H. Yu, Y. Gao, and J. Wang, "A multiobjective particle swarm optimization algorithm based on competition mechanism and Gaussian variation," *Complexity*, vol. 2020, pp. 1–23, Nov. 2020, doi: 10.1155/2020/5980504.
[29] Z. Xu, Q. Geng, H. Cao, C. Wang, and X. Liu, "Uncertainty-aware workflow migration among edge nodes based on blockchain," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, pp. 1–12, Dec. 2019, doi: 10.1186/s13638-019-1583-1.
[30] C. R. Raquel and P. C. Naval, "An effective use of crowding distance in multiobjective particle swarm optimization," in *Proc. 7th Annu. Conf. Genetic Evol. Comput.*, Jun. 2005, pp. 257–264, doi: 10.1145/1068009.1068047.
[31] 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.
[32] R. Hassan, B. Cohanim, O. de Weck, and G. Venter, "A comparison of particle swarm optimization and the genetic algorithm," in *Proc. 46th AIAA/ASME/ASCE/AHS/ASC Struct., Struct. Dyn. Mater. Conf.*, Apr. 2005, pp. 1138–1150, doi: 10.2514/6.2005-1897.

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

ZURINA MOHD HANAPI (Member, IEEE) received the B.Sc. degree in computer and electronic systems 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 Networks, Faculty of Computer Science and Information Technology, UPM, where she has been a Lecturer, since 2004. She has authored more than 70 papers in 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 the Malaysian Security Committee Research.

MOHAMED A. ALRSHAH (Senior Member, IEEE) received the B.Sc. degree in computer science from Naser University, Libya, in 2000, and the M.Sc. and Ph.D. degrees in communication technology and networks from Universiti Putra Malaysia (UPM), in May 2009 and February 2017, respectively. He is currently a Senior Lecturer with the Department of Communication Technology and Networks, Faculty of Computer Science and Information Technology, UPM. He has published a number of articles in high-impact-factor scientific journals. His research interests include high-speed TCP protocols, high-speed wired and wireless networks, parallel and distributed algorithms, WSN, the IoT, and cloud computing.

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

FATEN A. SAIF received the bachelor's degree in computer science degree from Sana'a University, in 2008, and the master's degree in computer science from Universiti Putra Malaysia (UPM), in 2019, where she is currently pursuing the Ph.D. degree with the Faculty of Computer Science and Information Technology. Her research interests include computer networks, workflow and task scheduling, cloud computing, the Internet of Things, optimization, workload allocation, and task offloading.