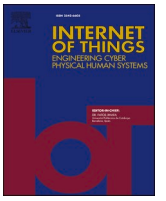
Internet of Things 34 (2025) 101775

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.sciencedirect.com/journal/internet-of-things



Joint optimisation of time and energy consumption for data aggregation in fog-enabled IoT networks

Sira Yongchareon

Department of Data Science and AI, School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, New Zealand

ARTICLE INFO

Keywords:
Fog computing
IoT
Optimization
Energy consumption
Time consumption

ABSTRACT

Fog computing extends cloud capabilities to the network edge, enabling Internet-of-Things (IoT) devices to offload computation to nearby fog nodes rather than a remote cloud. Offloading aggregation tasks reduces data redundancy and accelerates analytics while easing device energy use and backhaul load. Yet end-to-end completion time—comprising execution, transmission, and queueing—can still be substantial, creating a challenging time-energy trade-off. We formulate data-aggregation offloading as a multi-objective optimization problem that jointly minimizes latency (makespan) and energy under compute and bandwidth constraints. To solve it, we develop an NSGA-III-based method that searches for Pareto-optimal offloading and scheduling decisions across sensor and fog nodes. Comprehensive simulations and systematic experiments demonstrate that our approach consistently outperforms state-of-the-art baselines, delivering lower latency and energy consumption with better scalability.

1. Introduction

The Internet of Things (IoT) is a network of interconnected smart objects—such as wearable and mobile devices [1,2] —that communicate over wireless networks and embed sensing and computing capabilities [3], enabling real-time collection and processing of data about users and their surroundings [4]. Moreover, IoT devices can perform on-device data aggregation and analysis, coordinating these operations and governing data utilisation throughout [5].

Given IoT devices' limited battery and compute resources, cloud computing offers scalable aggregation, analysis, and storage capabilities [6]. Data produced by IoT devices can be transmitted to the cloud for processing—an approach known as task offloading [7]. However, cloud offloading consumes network bandwidth and introduces latency. To mitigate these drawbacks, fog computing places processing and storage closer to devices [8]; offloading to nearby fog nodes reduces latency and alleviates network congestion. Additionally, performing data aggregation in fog computing reduces redundancy and accelerates analysis while lowering storage demands [9]. Offloading tasks—such as aggregation and storage—can be distributed across fog nodes [10–12], but naïve distribution can incur substantial time and energy costs; indiscriminate offloading rarely shortens the makespan on fog nodes. Efficient task placement and scheduling are therefore essential to minimize latency and energy use. However, this joint optimization is NP-hard, and the search space grows exponentially with the numbers of offloaded tasks, sensor nodes, and fog nodes.

Recent work has explored offloading and scheduling for fog/edge-assisted IoT. Moving computation from a remote cloud to the network edge reduces service delay by bringing processing closer to devices [15]. Early fog-only frameworks propose neighbour or threshold-based offloading to minimise delay, but they overlook the transmission overheads and energy costs introduced by inter-fog

E-mail address: sira.yongchareon@aut.ac.nz.

https://doi.org/10.1016/j.iot.2025.101775

Available online 24 September 2025

communication and therefore do not jointly optimise delay and energy [16–18]. Fog-cloud hybrids balance delay and power by pushing heavy tasks to the cloud [19]; however, they reintroduce WAN latency and backhaul congestion and often assume a single cloud ingress, creating a potential single point of failure [19].

Multi-objective offloading has been considered, but common scalarisation approaches collapse objectives into a single weighted sum, which can miss Pareto-optimal solutions at the vertices of the feasible region [20]. Most approaches optimise a single metric (delay or energy) or use scalarisation, yielding limited coverage of the true time-energy Pareto front; (ii) many depend on cloud offloading, which adds WAN latency, congestion, and single-point-of-failure risk [19]; (iii) aggregation-specific modelling at the fog (processing + storage) is underdeveloped, particularly the interplay of execution, transmission, and queueing; and (iv) several methods induce imbalance or hotspots by not explicitly controlling workload distribution across fog nodes [16–24].

In this paper, we propose a novel Multi-objective Optimisation Method (MUOM) that can address the aforementioned gaps by (a) formulating aggregation offloading with explicit execution, transmission, and queueing time components and corresponding energy terms; (b) directly searching the Pareto front via NSGA-III (rather than collapsing objectives) [20]; (c) operating within a two-layer fog to avoid WAN penalties while improving locality; and (d) producing balanced schedules that reduce makespan and energy simultaneously. As shown in our evaluations, this yields lower latency and energy than state-of-the-art baselines with better scalability.

The remainder of the paper is organised as follows. We discuss related work in Section 2. Section 3 outlines our proposed system model and problem formulation. Section 4 presents our multi-objective optimisation method for fog computing, followed by the evaluation of the proposed method and result discussions in Section 5. Finally, Section 6 concludes the paper.

2. Related work

Computation tasks in IoT devices can be offloaded to remote cloud/servers for processing due to the devices' low computational and power capacity [13,14]. However, by doing so, there are challenges in managing network overhead, increased communication, computational energy, and cost. Compared with the cloud, fog computing can reduce network overhead and congestion by alleviating the workload from a remote cloud to the edge of a network close to IoT devices [15]. Complementing architecture-level studies, Abolhassani Khajeh et al. survey real-time scheduling for IoT, cataloguing task models, evaluation parameters, and solution families (heuristics, meta-heuristics, learning-based) [30]. The review underscores the prevalence of latency-centric objectives and heterogeneous setups, while revealing comparatively limited treatment of joint latency-energy optimisation under realistic networking effects. Complementary to offloading and scheduling, the fog data-reduction literature—covering compression, sampling, and in-network aggregation—shows that pushing reduction near data sources can alleviate congestion and device energy use; however, these techniques are typically studied in isolation from offloading/scheduling policies across fog nodes [32]. Our research treats aggregation as a first-class workload and jointly optimises where and when it executes to balance latency and energy at the fog layer.

Several works have proposed different approaches to offloading tasks from IoT devices to fog nodes [16–24]. In [16], Yousefpour et al. proposed a framework for task offloading to reduce the service delay of IoT-cloud applications in fog computing. The framework provides a minimising policy for the service delay, and the policy considers the service delay based on the load of each fog node. The offloading task is transmitted to the best neighbouring fog node if the service delay exceeds the threshold value. Otherwise, the task is accepted at the same fog node for processing. In their framework, fog nodes are interconnected in a distributive manner, which leads to data transmission overhead. Therefore, the minimising policy cannot optimise the transmission overhead that may be incurred. Also, the proposed framework does not consider optimising energy consumption for service delay. Similarly, Yousefpour et al. in [17,18] do not consider fog nodes' transmission and energy overhead for service delay optimisation in fog computing.

Another study partially focused on computing IoT tasks at fog computing [19]. For further computation of tasks, IoT data can be forwarded from fog nodes to the cloud, and offloading tasks to the cloud can reduce the workload and power consumption of the fog nodes that are needed to compute heavyweight tasks. However, transmitting tasks to the cloud increases network overhead and computational and communication costs. Further, the study assumed that the cloud is connected to fog nodes using a single communication point. In this context, the communication network between fog nodes and the cloud is vulnerable to a single point of failure threat.

In [21], Jiang et al. proposed a meta-heuristic method that investigates the placement of tasks offloaded to the fog nodes. Based on the meta-heuristic method's cost function, the study considered communication cost, computation cost, and power consumption of fog nodes. In this method, a technique for priority mapping is used to schedule the placement of tasks to help reduce the power consumption, communication, and computational cost of fog nodes during the placement. However, the optimal solutions for optimising fog nodes' energy and power consumption for the tasks, including data processing and data storage, are not provided.

Liu et al. [20] proposed a method for optimising energy consumption, execution time delay, and payment cost in fog computing. Their method transforms a multi-objective problem into a single-objective problem using scalarisation and interior point techniques. These techniques are intuitively not satisfying as they do not visit a problem's vertices but only cover its interior region. The techniques can find an optimal solution from an interior region without considering a problem's vertices.

Naqvi et al. [22] proposed a meta-heuristic method based on the ant colony optimisation (ACO) method to optimise response times of smart grid applications in fog computing. This study did not consider the optimisation of transmission time and transmission energy. Further, the ACO method depends on profiling offloaded tasks, which incurs high transmission overhead. Hussein et al. [24] further enhanced the ACO method to optimise the transmission time of offloaded tasks at fog nodes. Still, the new ACO method is based on a single-objective optimisation of the transmission time. Also, the proposed method offloads the aggregation tasks to the cloud for further processing and storage, which results in more network overhead.

In [23], Binh et al. proposed a method based on a genetic algorithm (GA) for offloading tasks and scheduling tasks at fog nodes. The

main objective of the method is to achieve a trade-off between offloading tasks, scheduling tasks, and monetary cost to efficiently complete tasks in fog and cloud systems. The proposed GA achieved high cost and performance efficiency for offloading tasks to fog nodes.

Beyond classic cloud-offloading, hierarchical designs increasingly push computation toward the edge to curb WAN latency and backhaul load. Javadzadeh et al. model real-time scheduling over a cloud-fog-dew stack via a mixed-integer nonlinear program, targeting reductions in power consumption and Internet traffic, and scale their approach with NSGA-II [31]. While their results highlight the benefits of deeper edge tiers, the formulation is not aggregation-specific and does not explicitly capture queueing/waiting at fog nodes—factors that dominate end-to-end makespan in data-aggregation workloads.

However, none of these methods [16–24] considered multi-objective problems concerning time and energy consumption for computing tasks such as data aggregation in fog computing. In conclusion, there is still a key challenge in finding optimal solutions concerning time consumption (including transmission, execution, and waiting) and energy consumption for efficient task offloading in fog computing. To address this challenge in this paper, we propose a new multi-objective optimisation method (MUOM) in fog computing.

3. System model and problem formulation

In this section, we discuss our proposed system model and problem formulation for the optimisation of time consumption and energy consumption for data aggregation in fog computing.

Data aggregation formulation regarding time and energy consumption is based on the divide-and-conquer scheme for data aggregation proposed in [12]. Our notations used in this paper, along with their descriptions, are listed in Table 1.

3.1. System model

Our system model is illustrated in Fig. 1, based on the model presented in [12]. Fig. 1 comprises fog nodes, data-owner, sensor nodes, and end-user devices. This model divides a fog computing layer into two sub-layers: Fog layer 1 and Fog layer 2. Fog layer 1 comprises fog nodes for computation and analysis of IoT tasks. In Fog layer 2, fog nodes are organised in a cluster and responsible for aggregation and storage tasks.

IoT devices, i.e. sensor nodes, can be connected to Fog layer 1 through a local area network (LAN) connection and both layers (Fog layer 1 and Fog layer 2) are interconnected through a LAN connection. End-user devices and the data owner are connected to Fog layer 2 and Fog layer 1, respectively, through a wide area network (WAN) connection.

Let $SN_k = \{sensor_1, \ldots, sensor_k\}\ (1 \le k \le \max(sensor))$ represents a set of sensor nodes for generating IoT data, and a data owner can be denoted as $DT = \{t_1, \ldots, t_i\}(1 \le i \le \max(dataowner))$. A data owner defines the data utilisation and privacy policies for their generated data at sensor nodes.

We also use FN to represent a set of fog nodes in both layers, and it can be denoted as $FN = \{fn_1, fn_2, \ldots, fn_l\}(1 \le l \le N)$, where $FN_l = \{fn_{l,j} \mid 1 \le j \le FN_l\}$ represents a set of computing tasks in the l^{th} fog node. Let $w_{l,j}$ be a workload of the l^{th} fog node. Also, $pre\ (fn_{l,j})$ indicates the precursory computing task, which is waiting in a queue for $fn_{l,j}$.

Based on the network design and setup presented in [12], we consider a scenario where SN_k sends data to FN in Fog layer 1. FN has to perform FN_l which is a set of computation tasks including data authentication, data analysis, data encryption, scheduling transmission of channel and $fn_{l,j}$ allocation to FN in Fog layer 2. According to the t_i policies, FN_l is processed, such as encrypted, analysed,

Table 1
Key notations and description.

Notation	Description
$\mathbf{fn_{l,j}}$	A computing task j on the l^{th} fog node
$\mathbf{T(fn_{l,j})}$	Total time consumption of task j in the l^{th} fog node
$\mathbf{SN_k}$	k^{th} sensor node
FN	A set of fog nodes
$\boldsymbol{t_i}$	i^{th} data-owner
$\mathbf{T_{exe}}$	Execution time
$\mathbf{T_{trans}}$	Transmission time
$\mathbf{T_{wait}}$	Waiting time
$\mathbf{w_{l,j}}$	Workload of task j on the l^{th} fog node
$\mathbf{T\left(fn_{l,j}^{1}\right)}$	Total time consumption of task j on the l^{th} miner node in fog layer 1
$\mathbf{T\left(fn_{l,j}^{2}\right)}$	Total time consumption of task j on the l^{th} fog node in fog layer 2
$\mathbf{E(fn_{l,j})}$	Total energy consumption of task j on the l^{th} fog node
$\mathbf{E_{trans}}$	Energy consumption of transmission
$\mathbf{E_{exe}}$	Energy consumption of execution
$\mathbf{E_{wait}}$	Energy consumption of waiting for execution of the precursor executing task
$\mathbf{E\left(fn_{l,j}^{1}\right)}$	Total energy consumption of task j on the l^{th} miner node in fog layer 1
$\mathbf{E\left(fn_{l,j}^{2}\right)}$	Total energy consumption of task j on the l^{th} fog node in fog layer 2

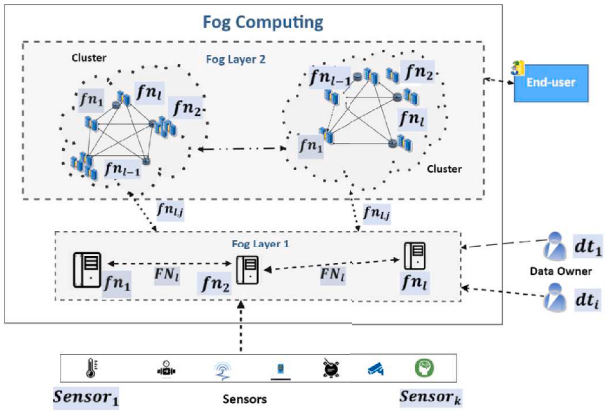


Fig. 1. System Model.

and authenticated at Fog layer 1. Fog layer 2 receives processed $fn_{l,j}$ from Fog layer 1. Then *FN* in Fog layer 2, stores and aggregates processed $fn_{l,j}$. On a request of the end-user device, *FN* in Fog layer 2 sends aggregated data in $fn_{l,j}$ to the end-user device.

Note that we adopt a static-snapshot model with stable links and stationary arrivals to cleanly study the latency-energy trade-off for aggregation at the fog layer. This abstraction is common in edge/fog scheduling and avoids confounding factors such as mobility, transient outages, or burst-induced backlogs.

3.2. Time consumption model

In our proposed model, time consumption $T(fn_{l,j})$ is an amount of time consumed by fog nodes to perform computing tasks $fn_{l,j}$ for data aggregation. $T(fn_{l,j})$ consists of the transmission time T_{trans}, the execution time T_{exe} and the waiting time of the precursor executing task T_{wait}. The transmission time T_{trans} is the amount of time that each fog node in both fog layers takes to communicate with the nodes in fog layers, sensors, data-owner, and end-user devices. T_{trans} involves the communication time for $fn_{l,j}$ according to the tasks defined in [12], including data request, data transmission, cluster formation, data authentication, public and private keys, and table distribution.

The execution time T_{exe} is the time taken by *FN* to execute the $fn_{l,j}$. T_{exe} involves $fn_{l,j}$ for executing data encryption, data authentication, creation of the public and private key, creation of the policy table, aggregation, and cluster formation. The precursor executing task T_{wait} is the waiting time that each computing task $fn_{l,j}$ must wait in a queue to be executed by *FN*. The total $T(fn_{l,j})$ based on [3,25] becomes:

$$T(fn_{l,j}) = \sum_{FN_l \in FN} (T_{trans} + T_{exe} + T_{wait}) \tag{1}$$

The time consumption of both fog layers depends on the computing tasks $fn_{l,j}$, which are performed uniquely by each fog layer. Therefore, first, we compute the T_{trans}, T_{exe} *and* T_{wait} for both layers separately and then combine them to get the total time consumption $T(fn_{l,j})$, respectively.

For Fog layer 1, the transmission time T_{trans} can be calculated by

$$T^1_{trans} = \left(\sum_{fn_j \in fn_l} (fn_j) \right), Bandwidth_l,\ Num\ (FN) \tag{2}$$

In Fog layer 1, the transmission time T^1_{trans} of our proposed model depends on the transmission tasks fn_jThe total number of fog nodes and network bandwidth of fog computing. The transmission tasks fn_j are the tasks for fog nodes' authentication, data policy requests, hash keys and data requests, task allocation, and channel scheduling. Each of these fn_j is discussed as follows.

In an authentication task $Auth_i$, the fog nodes send an authentication token to the other fog nodes in both layers with whom they wish to communicate. The authentication token checks the fog node's authenticity within a network. For policy fn_j, the fog nodes request the data-owner t_i for data policies P_{LoP}.

Also, fog nodes processing data tasks such as data encryption and division in Fog layer 1 send a hash of private keys P_k to the end-user device for decryption of processed data. In the requested fn_j, fog node requests data M from sensor nodes SN_k in close proximity and fog node receive data M from a sensor node SN_k. The fn_j for allocation, allocates the computing tasks to chosen fog nodes in Fog layer 1. Another fn_j is the scheduling of transmission channels for a fog computing network.

The transmission time T^1_{trans} also depends on the network bandwidth $Bandwidth_l$. The bandwidth between fog nodes in both layers, sensor nodes, and data-owner can be computed by

$$Bandwidth_i = \begin{cases} \infty, & FN = 0 \\ Bandwidth_{i,L}, & FN_i = 1, 2, \ldots, N \\ Bandwidth_L, & SN_k = 1, 2, \ldots \max(sensor) \\ Bandwidth_{i,W}, & DT = 1,\ 2, \ldots \max\ (data - owner) \end{cases}$$

Let $Bandwidth_{l,\ L}$ be the bandwidth of a LAN network for the l^{th} fog node to the other fog nodes in both fog layers. $Bandwidth_L$ represents the bandwidth of a LAN network for the l^{th} fog nodes and sensor nodes SN_k, and $Bandwidth_{l,W}$ is WAN for the l^{th} fog node and the data owner DT.

In the execution of a computing task in Fog layer 1, the execution time T^1_{exe} of the l^{th} fog node is determined by the workload of the fog node and the computational capacity of the l^{th} fog node. Based on the formula [3], T^1_{exe} can be computed by

$$T^1_{exe} = \frac{\sum_{w_j \in fn_l}(w_j)}{C_{cap_l}} \tag{3}$$

The workload w_j at the l^{th} fog node consists of the following workloads for the computing task fn_j.

The w_j for encrypting data M at the fog node for processing data. Then the w_j for a division of encrypted M at the fog node. Also, the w_j is for generating and checking token-based authentication $auth_i$.

Further, the workload w_j for creating a hash of a private key p_k. The execution time T^1_{exe} also depends on the workload for creating a table with data-owner-defined policies.

For fog layer 1, the precursor T^1_{wait} is a waiting time in a queue for the execution of fn_j at the l^{th} fog node. The l^{th} fog node is represented as a tuple $(total\ (w_j),\ Num(FN_l))$, where $total\ (w_j)$ represents the total workload at the l^{th} fog node and $Num(FN_l)$ represents the number of computing tasks that are scheduled for the l^{th} fog node.

$$T^1_{wait} = \sum_{j=1}^{total\ (w_j)} pre\left(T^1_{exe,j}\right) \tag{4}$$

By combining (2), (3), and (4), (1) becomes:

$$T\left(fn^1_{l,j}\right) = \left(\begin{array}{c} \left(\sum_{fn_j \in fn_l}(fn_j)\right) + \left(\frac{\sum_{w_j \in fn_l}(w_j)}{C_{cap_l}}\right) \\ + \left(\sum_{j=1}^{total\ (w_j)} pre\left(T^1_{exe,j}\right)\right) \end{array} \right), Bandwidth_l,\ Num\ (FN) \tag{5}$$

Now for Fog layer 2, the transmission time T^2_{trans} taken by the l^{th} fog node is defined as:

$$T^2_{trans} = \left(\sum_{fn_j \in fn_l}(fn_j)\right),\ Bandwidth_l\ Num\ (FN),\ Num\ (C(FN)) \tag{6}$$

Similar to T^1_{trans}, the T^2_{trans} also depends on the transmission tasks fn_j, the network bandwidth $Bandwidth_l$, the total number of fog nodes $Num\ (FN)$ in Fog layer 2, and the number of fog nodes in a cluster $Num\ (C(FN))$.

The fn_j for T^2_{trans} includes authentication, sending aggregated data, receiving data blocks, cluster formation, and channel scheduling. Each of the fn_j is discussed in detail below.

For the token-based authentication fn_j, fog nodes send a token $Auth_i$ to other fog nodes in both fog layers and end-user devices with whom they wish to have data communication. Afterwards, fog nodes communicate with neighbouring fog nodes for cluster formation and scheduling of the transmission channel. Then fog nodes request and receive data blocks $block_i$ from Fog layer 1. Fog nodes also request the table policy. Another fn_j is for the communication with the end-user device for sending aggregated data E. Further, the fog node requests the fog nodes in the same layer to send blocks $block_i$ for aggregation. Besides fn_j, the transmission time T^2_{trans} also depends on $Bandwidth_l$. The bandwidth between fog nodes in both layers and end-user devices. The $Bandwidth_l$ is measured as

$$Bandwidth_i = \begin{cases} \infty, & FN = 0 \\ Bandwidth_{i,L}, & FN_i = 1, 2, \ldots, N \\ Bandwidth_{i,W}, & user_i = 1, 2, \ldots \max(end - userdevice) \end{cases}$$

where $Bandwidth_{l,\ L}$ represents the bandwidth of a LAN for the l^{th} fog node in both fog layers. $Bandwidth_{l,W}$ represents the bandwidth of WAN for the l^{th} fog node and the end-user device.

Similar to Fog layer 1, the execution time T^2_{exe} of the l^{th} fog node in Fog layer 2 is determined by the workload of the l^{th} fog node and the computational capacity of the l^{th} fog node.

$$T^2_{exe} = \frac{\sum_{w_j \in fn_l}(w_j)}{C_{cap_l}} \tag{7}$$

The workload w_j at the l^{th} fog node consists of the following workloads for computing tasks fn_j in Fog layer 2:

One of the w_j is for generating and checking token-based authentication, like Fog layer 1. Also, w_j is for processing the request of the data $block_i$. Further w_j involves the aggregation of encrypted M.

Similar to T^1_{wait}, the precursor waiting time T^2_{wait} is computed for Fog layer 2. Thus, the total time consumption $T\left(fn^2_{l,j}\right)$ for Fog layer 2 becomes:

$$T\left(fn^2_{l,j}\right) = \left(\begin{array}{c} \left(\sum_{fn_j \in fn_l} (fn_j) \right) + \left(\frac{\sum_{w_j \in fn_l}(w_j)}{C_{cap_l}} \right) \\ + \left(\sum_{j=1}^{total\ (w_j)} pre\left(T^2_{exe,j}\right) \right) \end{array} \right), Bandwidth_l Num\ (C(FN)),\ Num\ (FN) \tag{8}$$

From (5) and (8), the total time consumption $T(fn_{l,j})$ becomes:

$$T(fn_{l,j}) = \sum_{FN_l \in FN} \left(T\left(fn^1_{l,j}\right) + T\left(fn^2_{l,j}\right) \right) \tag{9}$$

3.3. Energy consumption model

Energy consumption $E(fn_{l,j})$ of FN is the consumption of energy in the transmission E_{trans}, the execution E_{exe} and the waiting for execution E_{wait} of the precursor computing task $fn_{l,j}$ based on [3,25].

Let E_{trans} be an energy consumption for transmission, which is the power consumed by each fog node in both fog layers to transmit data to the nodes in fog layers, sensors, data-owner, and end-user devices.

The energy consumption for execution E_{exe} represents a power consumed by each fog node to execute the computing tasks $fn_{l,j}$. The energy consumption for precursor execution E_{wait} is the energy consumption that each computing task $fn_{l,j}$ requires to wait in a queue to be executed by FN. Based on the time consumption in subsection B, energy for Fog layer 1 and Fog layer 2 can be computed as

$$E\left(fn^1_{l,j}\right) = \left(\begin{array}{c} \left(T^1_{trans} * p_{trans}\right) + \left(T^1_{exe} * (p_a + p_i)\right) \\ + \left(pre\left(T^1_{exe} * (p_a + p_i)\right)\right) \end{array} \right) \tag{10}$$

$$E\left(fn^2_{l,j}\right) = \left(\begin{array}{c} \left(T^2_{trans} * p_{trans}\right) + \left(T^2_{exe} * (p_a + p_i)\right) \\ + \left(pre\left(T^2_{exe} * (p_a + p_i)\right)\right) \end{array} \right) \tag{11}$$

where p_{trans} represents the power consumption during $fn_{l,j}$ transmission in both fog layers, and p_a *and* p_i represents the active and idle power consumption of a fn_l as represented in [26]. By combining (10) and (11), the total energy consumption $E(fn_{l,j})$ for both fog layers becomes

$$E(fn_{l,j}) = \sum_{FN_l \in FN} \left(E\left(fn^1_{l,j}\right) + E\left(fn^2_{l,j}\right) \right) \tag{12}$$

3.4. Problem formulation and constraints

In this paper, we focus on the two-objective (2 M) fitness function to optimise the time consumption $T(fn_{l,j})$ and energy consumption $E(fn_{l,j})$ of fog nodes FN in both fog layers. The problem can be formally defined in (13), and its constraint is shown in (14). The constraint guarantees that the gene values greater than the number of fog nodes FN will not be created by a chromosome.

$$\min T(fn_{l,j}),\ E(fn_{l,j}),\ \forall\ l \in \{1,\ 2,\ \ldots,\ N\}\ and\ j\ \in \{1,\ 2,\ ..,FN_l\} \tag{13}$$

$$s.t.\ \sum_{l=1}^{n} fn_l \leq N\ and\ \sum_{(lj)=1}^{n} fn_{l,j} \leq FN_l \tag{14}$$

where N represents the maximum number of fog nodes, FN_l represents the maximum set of computing task $fn_{l,j}$ and fn_l represents the fog node participating in computing task $fn_{l,j}$.

4. Multi-objective optimisation method (MUOM) in fog computing

In this section, we propose MUOM based on NSGA-III, an accurate and efficient method for solving optimisation problems with multiple objectives. In our proposed MUOM, the NSGA-III is used to optimise the time consumption $T(fn_{l,j})$ and energy consumption $E(fn_{l,j})$ for data aggregation in fog computing as represented in (9) and (12), respectively. NSGA-III has brilliant performance in search of an optimal solution and faster solution convergence. The NSGA-III introduces a selection method based on reference points rather than the traditional NSGA-II method [27]. In the selection generation for NSGA-III, the reference point guarantees the distribution's diversity for efficiently searching optimal solutions. We balance latency-energy trade-offs via Pareto optimisation (NSGA-III); when a single operating point is needed, we select the knee—the Pareto solution closest to the utopia point in min-max normalised (*T, E*) space. First, we encode a strategy for optimal solutions, and then we initialise the fitness functions, constraints, and the first generation of the population. Afterwards, we utilise the crossover and mutation operations for the new generation of solutions. The selection operations based on reference-point SAW [28] and MCDM [26] are chosen to select an optimal solution.

4.1. Encoding

We encode the optimal strategy for the time consumption $T(fn_{l,j})$ and energy consumption $E(fn_{l,j})$ problem. In a genetic algorithm (GA), chromosomes comprise several genes, representing an optimal strategy for *FN*. Fig. 2 illustrates an example of an optimal strategy. In this example, a chromosome is an instance of the optimal strategy. The chromosome is encoded in an array of (0, 2, 3) integers.

4.2. Fitness functions and constraints

In GA, fitness functions predict whether a possible strategy is optimal. The fitness functions include two categories: the time consumption $T(fn_{l,j})$ and energy consumption $E(fn_{l,j})$ for fog nodes *FN*, as represented in (9) and (12). The proposed MUOM aims to find an optimal strategy for minimising the fitness function's two categories, as shown in (13). The constraint associated with the fitness function is given in (14). We use NSGA-III to optimise the time consumption and energy consumption in fog computing. Also, NSGA-III addresses the multi-objective optimisation problem (2 M) with associated constraints.

Time consumption $T(fn_{l,j})$ is one of the fitness functions. Algorithm 1 presents the evaluation of time consumption $T(fn_{l,j})$. In this Algorithm, we input an optimal strategy denoted as $O(fn_{l,j})$. First, we calculate the transmission time T_{trans}, execution time T_{exe}, and waiting time T_{wait} for data aggregation (lines 3 to 8). Then, we compute the time consumption $T(fn_{l,j})$ by fog nodes (lines 9 and 10). The time consumption of both fog layers is the total time consumption in fog computing (line 12). Finally, the total time consumption is output in each task schedule.

Another fitness function is energy consumption $E(fn_{l,j})$. The evaluation of the energy consumption is elaborated in Algorithm 2. We first calculate transmission energy E_{trans}, execution energy E_{exe} and waiting energy E_{wait} for both fog layers (lines 3 to 10). Then the total energy consumption $E(fn_{l,j})$ is an output.

4.3. Initialisation

During the initial stages of GA, the GA parameters must be determined and initialised. The parameters include the possibility of crossover POP_c, mutation POP_m, population size N_{pop} and maximum iterations *Gen*. In GA, the optimal strategy of the computing task $fn_{i,j}$ is indicated by each chromosome,

$Crm_{l,i} = \left\{ g_{s,1},\ g_{s,2},\ \ldots,\ g_{s,G} \right\} (i = 1,\ 2,\ \ldots,\ N_{pop},\ G = Gen)$, which is denoted as an array of integers. The chromosome $Crm_{l,i}$ consists of gene $g_{s,l}$ and the gene $g_{s,l}$ be an optimal strategy of $fn_{l,j}$ in the s^{th} schedule.

4.4. Crossover and mutation

In the crossover operation, two new chromosomes are generated from combining two parent chromosomes. This operation is performed to acquire better chromosomes while exchanging part of the gene's fragments from parent chromosomes. Fig. 3 shows an example of a crossover operation. In this example, the crossover points for two chromosomes in a first schedule are determined. Then, genes are swapped around the crossover point to generate two new chromosomes.

After the crossover operation, the mutation operation is performed to generate better chromosomes. Some chromosome genes are modified with higher fitness values in mutation operation, as illustrated in Fig. 4.

$fn_{1,1}$	$fn_{1,2}$	$fn_{1,4}$	.....	$fn_{l,j}$
3	2	0	.....	3

Fig. 2. Example of Encoding Chromosomes.

Algorithm 2
Energy consumption evaluation.

Input: Optimal strategy $\boldsymbol{O}(\boldsymbol{fn}_{lj})$
Output: Energy consumption $\mathrm{E}(\boldsymbol{fn}_{lj})$
1. for $\boldsymbol{l} = 1$ ***to*** $\boldsymbol{N}$ do
2. for $\boldsymbol{j} = 1$ ***to*** $|\boldsymbol{FN}_l|$ do
3. Calculate $\boldsymbol{E}^1_{\boldsymbol{trans}}$ by (10)
4. Calculate $\boldsymbol{E}^2_{\boldsymbol{trans}}$ by (11)
5. Calculate $\boldsymbol{E}^1_{\boldsymbol{exe}}$ by (10)
6. Calculate $\boldsymbol{E}^2_{\boldsymbol{exe}}$ by (11)
7. Calculate $\boldsymbol{E}^1_{\boldsymbol{wait}}$ by (10)
8. Calculate $\boldsymbol{E}^2_{\boldsymbol{wait}}$ by (11)
9. $\boldsymbol{E}\left(\boldsymbol{fn}^1_{lj}\right) = \boldsymbol{E}^1_{\boldsymbol{trans}} + \boldsymbol{E}^1_{\boldsymbol{exe}} + \boldsymbol{E}^1_{\boldsymbol{wait}}$
10. $\boldsymbol{E}\left(\boldsymbol{fn}^2_{lj}\right) = \boldsymbol{E}^2_{\boldsymbol{trans}} + \boldsymbol{E}^2_{\boldsymbol{exe}} + \boldsymbol{E}^2_{\boldsymbol{wait}}$
11. end for
12. $\mathrm{E}(\boldsymbol{fn}_{lj}) = E\left(\boldsymbol{fn}^1_{lj}\right) + \boldsymbol{E}\left(\boldsymbol{fn}^2_{lj}\right)$
13. end for
14. return $\boldsymbol{E}(\boldsymbol{fn}_{lj})$

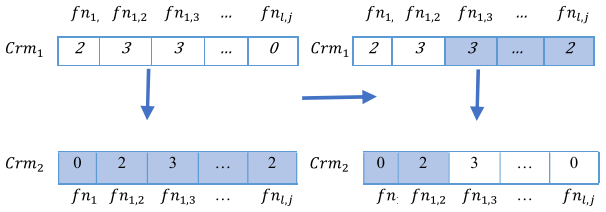


Fig. 3. Example of crossover operation.

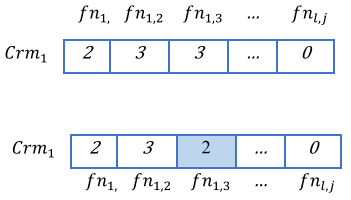


Fig. 4. Example of mutation operation.

4.5. Selection for the next generation

We aim to select chromosomes to generate individuals with higher fitness values for the next population. As discussed above, each chromosome represents an optimal strategy for 2 M objective. After crossover and mutation operations on the chromosomes, the population size increases to $2N_{pop}$ Algorithms 1 and 2 evaluate the 2 M fitness function values. For the next-generation population, optimal solutions evaluating 2 M fitness functions are sorted using a fast non-dominated method. This procedure is carried out to generate non-dominated fronts $\{NF_1 + NF_2 + \ldots + NF_l\}$ with higher fitness values.

Then, the generated fronts are randomly chosen to generate the next-generation population until the size of the selected solutions are N_{pop}. By adding up the generated fronts, if the size becomes N_{pop}, then the procedure for selecting optimal solutions is finished, and the next generation is generated. Otherwise, optimal solutions need to be selected from the last l^{th} non-dominated front NF_l until the population size becomes N_{pop}.

After selecting optimal solutions, we adopt a normalisation operation to normalise the 2 M fitness function for all chromosomes in the population. In $2N_{pop}$ population, we search for the minimum time consumption and energy consumption, denoted as $T^{min}(fn_{l,j})$ and $E^{min}(fn_{l,j})$. The 2 M objective values are computed as

$$T'(fn_{l,j}) = T(fn_{l,j}) - T^{min}(fn_{l,j}) \tag{15}$$

$$E'(fn_{l,j}) = E(fn_{l,j}) - E^{min}(fn_{l,j}) \tag{16}$$

Let φ^T, φ^E be a maximum value of time consumption and energy consumption in each dimension, which can be calculated by

$$\varphi^T = \max\left(T'\left(fn_{l,j}\right) / wt^t\right) \tag{17}$$

$$\varphi^E = \max\left(E'\left(fn_{l,j}\right) / wt^e\right) \tag{18}$$

where wt^t and wt^e represent a weight vector of the 2 M fitness function.

Also, optimal solutions are sorted and selected in a non-dominated the l^{th} front NF_l. This process is repeated until all the solutions are selected. The selection steps are elaborated in Algorithm 3. In this Algorithm, the u^{th} generation (parent) represented as Gen_u is the input, and the output is $(u+1)^{th}$ generation (child) denoted as Gen_{u+1}. Before sorting and selecting a solution for the next generation, we compute each fog node's time and energy consumption using Algorithms 1 and 2 (lines 2 and 3).

Then, the non-dominated sorting for individual chromosomes with the size population of Gen_u is carried out (line 5). This sorting results in non-dominated fronts. Besides this, the population is selected primarily. The selected population S_u is constituted from fronts $\{NF_1 + NF_2 + \ldots + NF_l\}$ until the size of S_u becomes or exceeds N_{pop} (lines 6 and 7). Otherwise, a further selection is carried out (line 9). After selection, normalisation is carried out, and the remaining optimal solutions are determined (lines 10–12). Finally, the next generation (child) population (Gen_{u+1}) is generated entirely.

4.6. Optimal selection using SAW and MCDM

In each population, chromosome represents an optimal solution to minimise time consumption $\mathrm{T}\left(fn_{l,j}\right)$ and energy consumption $\mathrm{E}\left(fn_{l,j}\right)$. To select the optimal chromosome from the population size N_{pop}, the reference-point-based SAW [28] and MCDM [26] are employed.

The $\mathrm{T}\left(fn_{l,j}\right)$ and $\mathrm{E}\left(fn_{l,j}\right)$ are negative criteria, as the higher the values are, the worse the solution becomes. Therefore, we normalise $\mathrm{T}\left(fn_{l,j}\right)$ and $\mathrm{E}\left(fn_{l,j}\right)$ in the i^{th} optimal strategy, as represented in (19) and (20).

$$N\left(T\left(fn_{l,j}\right)\right) = \begin{cases} \left(T_{max}\left(fn_{l,j}\right) - T\left(fn_{l,j}\right)\right) / T_{max}\left(fn_{l,j}\right) - T_{min}\left(fn_{l,j}\right), \\ \qquad T_{max}\left(fn_{l,j}\right) - T_{min}\left(fn_{l,j}\right) \neq 0 \\ I, \quad T_{max}\left(fn_{l,j}\right) - T_{min}\left(fn_{l,j}\right) = 0 \end{cases} \tag{19}$$

$$N\left(E\left(fn_{l,j}\right)\right) = \begin{cases} \left(E_{max}\left(fn_{l,j}\right) - E\left(fn_{l,j}\right)\right) / E_{max}\left(fn_{l,j}\right) - E_{min}\left(fn_{l,j}\right), \\ \qquad E_{max}\left(fn_{l,j}\right) - E_{min}\left(fn_{l,j}\right) \neq 0 \\ I, \quad E_{max}\left(fn_{l,j}\right) - E_{min}\left(fn_{l,j}\right) = 0 \end{cases} \tag{20}$$

where $T_{max}\left(fn_{l,j}\right), \quad T_{min}\left(fn_{l,j}\right)$ *and* $E_{max}\left(fn_{l,j}\right), \quad E_{min}\left(fn_{l,j}\right)$ represents the maximum and minimum time consumption and energy consumption. To calculate the maximum values, $N\left(T\left(fn_{l,j}\right)\right)$ and $N\left(E\left(fn_{l,j}\right)\right)$ need to be combined with the associated weights $\frac{1}{2N}$ as shown in (21).

$$N\left(Crm_{l,i}\right) = \sum_{l=1}^{N} \frac{1}{2N} . N\left(T\left(fn_{l,j}\right)\right) + \sum_{l=1}^{N} \frac{1}{2N} . N\left(E\left(fn_{l,j}\right)\right) \left(1 \leq i \leq N_{pop}\right) \tag{21}$$

where $N\left(Crm_{s,i}\right)$ represents the value of the i^{th} chromosome. The optimal solution represented by chromosome $N\left(Crm_{l,i}\right)$ can be computed as

Algorithm 3
Selection for the Next Generation.

Input: Parent Generation $\boldsymbol{Gen_u}$
Output: Child Generation $\boldsymbol{Gen_{u+1}}$
1. for $\boldsymbol{l} = 1$ ***to N*** **do**
    2. Calculate $\boldsymbol{T}\left(\boldsymbol{fn_{l,j}}\right)$ by Algorithm 1
    3. Calculate $\boldsymbol{E}\left(\boldsymbol{fn_{l,j}}\right)$ by Algorithm 2
4. end for
    5. Non-dominant sorting ($\boldsymbol{Gen_u}$) the POP solutions
    6. Constitute $\boldsymbol{S_u}$ from fronts $\{\boldsymbol{NF}_1 + \boldsymbol{NF}_2 + \ldots + \boldsymbol{NF}_l\}$
    7. Conduct Primary selection
    8. if size $(\boldsymbol{S_u}) < \boldsymbol{N_{pop}}$ then
    9. Conduct further selection
    10. Normalise solutions by (15- 18)
    11. Select remaining $\boldsymbol{z}$ solutions
    12. $\boldsymbol{Gen_{u+1}} = \boldsymbol{S_u} \cup \boldsymbol{NF_l}$
    13. else
    14. $\boldsymbol{Gen_{u+1}} = \boldsymbol{S_u}$
    15. end if
    16. return $\boldsymbol{Gen_{u+1}}$

$$N(C_l) = \max_{i=1}^{N_{pop}} N(Crm_{l,i})(1 \le l \le N) \tag{22}$$

4.7. Proposed MUOM overview

We aim at minimising $T(fn_{l,j})$ and $E(fn_{l,j})$ for data aggregation in fog computing based on NSGA-III to obtain an optimal strategy for reducing $T(fn_{l,j})$ and $E(fn_{l,j})$. Algorithm 4 elaborates on the overview of MUOM. In this Algorithm, we input the initialised population **N** and a maximum number of iterations *Gen*. The Algorithm outputs the optimal strategy for $T(fn_{l,j})$ and $E(fn_{l,j})$ in each schedule $(1 \le l \le N)$.

Firstly, the first-generation population is initialised. Then, the child population is generated using crossover and mutation operations (lines 2–5). The child population size becomes $2N_{pop}$ This Algorithm also calculates the 2 M fitness functions of $2N_{pop}$ solutions (lines 6 and 7), then the Algorithm selects the optimal individuals for the next generation. Next, the Algorithm evaluates the fitness function to select an optimal strategy using SAW and MCDM methods (lines 12 and 13). Finally, the optimal strategies are outputs (line 15).

By jointly choosing where to aggregate and how to assign tasks, MUOM (i) places aggregators closer to data sources to reduce hop count and payload, (ii) discourages inter-fog transfers that add extra rounds, and (iii) balances link usage to limit queueing—thereby lowering both transmission time and energy.

4.8. Complexity analysis

Let P be the population size, G the number of generations, S the number of tasks (e.g., packets/sensor items), F the number of fog nodes, E the number of active links, and $L \approx S$ the chromosome length. Per generation, NSGA-III performs non-dominated sorting in $O(P^2)$ (two objectives) and variation in $O(P\,x\,L)$. Our latency/energy evaluation aggregates execution, transmission, and queueing terms in $O(S + E)$. Thus the total time is $O(G(P^2{+}P(L + S + E)))$ and memory is $O(P\ x\ L)$. Since evaluation dominates and is independent across individuals, it can be parallelised across cores/nodes.

5. Experiments, evaluation, and discussions

This section presents our comprehensive simulation and experiments conducted to evaluate the performance of the proposed MUOM. A simulation-based evaluation is adapted because it controls environmental parameters and considers different experiment constraints and scenarios. First, we introduce the test case scenarios for simulation, followed by a simulation setup including simulation parameters. Then, we discuss the performance evaluation of the proposed MUOM and comparative analysis with state-of-the-art methods.

5.1. Fog computing test-case architecture

We designed a test-case scenario to evaluate the effectiveness of the proposed MUOM, which consists of three layers. In the first layer, we have 5–1000 sensor nodes to sense the heterogeneous data and generate a range of 1–1000 Kbps of data. Sensor nodes are randomly distributed within 50–400 m of fog nodes. The second layer is divided into Fog layer 1 and Fog layer 2. The number of fog nodes in each layer ranges from 10 to 1000. Fog layer 1 performs computing tasks $fn_{l,j}$ to process and analyse the data generated by

Algorithm 4
Proposed MUOM in Fog Computing.

Input: The population size N, Max Iteration ***Gen***
Output: The optimal method $\boldsymbol{O}(\boldsymbol{fn_{l,j}})$
The optimal time consumption $\boldsymbol{T}(\boldsymbol{fn_{l,j}})$
The optimal energy consumption $\boldsymbol{E}(\boldsymbol{fn_{l,j}})$
1. for $\boldsymbol{l} = 1$ ***to N*** do
2. $\boldsymbol{i} = 1$
3. while $\boldsymbol{i} \le$ ***Gen*** do
4. for individuals: current population do
5. Crossover and Mutation operation
6. Calculate time consumption by Algo (1)
7. Calculate energy consumption by Algo (2)
8. end for
9. Selection for the next generation by Algo (3)
10. $\boldsymbol{i}++$
11. end while
12. Evaluate objective function by (19–21)
13. Select an optimal strategy by (22)
14. end for
15. return $\boldsymbol{O}(\boldsymbol{fn_{l,j}})$, $\boldsymbol{T}(\boldsymbol{fn_{l,j}})$, $\boldsymbol{E}(\boldsymbol{fn_{l,j}})$

sensor nodes. Whereas Fog layer 2 performs computing tasks $fn_{l,j}$ for data storage and aggregation. Lastly, the third layer comprises data-owner and end-user devices connected to the second layer via the internet.

5.2. Simulation setup

A simulation is carried out via a Network Simulator based on a Linux system with Intel (R) Core (i7), RAM 16.0 GB, and CPU 3.40 GHz. In the simulation, our preset parameter values for the experiments conducted on the test-case scenario are presented in Table 2. We define a range of values of the corresponding parameters based on [27,29].

The time consumption $T(fn_{l,j})$ and energy consumption $E(fn_{l,j})$ of the fog nodes in fog layers are used to evaluate the performance of MUOM. The comparative methods are elaborated on below for the comparative analysis of the proposed MUOM.

5.2.1. Non-optimization method (N—OPT)

In this method, computing tasks $fn_{l,j}$ for data aggregation, including data encryption, data distribution, data storage, and additive aggregation, are carried out on fog layers without utilising any optimisation method to minimise $T(fn_{l,j})$ and $E(fn_{l,j})$ [12].

5.2.2. Fully cloud method (FCM)

All the computing tasks $fn_{l,j}$ are fully offloaded from fog nodes to the cloud to process, store, and aggregate data. NSGA-III method is considered for optimisation of $T(fn_{l,j})$ and $E(fn_{l,j})$.

5.2.3. Partial cloud method (PFCM)

This method is a partial offloading of $fn_{l,j}$ from fog nodes to the cloud using the Ant-Colony Optimization (ACO) method [24]. The method based on ACO aims to find an optimal solution for $T(fn_{l,j})$ for data aggregation at fog nodes, and cloud for partial processing and aggregation of data.

As discussed above, these comparative methods are implemented under the same simulation setup of fog layers and sensor nodes.

We select baselines that (i) optimise latency/energy under the same fog-layer aggregation model, (ii) require the same input information, and (iii) are reproducible. Recent energy-aware clustering, delay-tolerant aggregation, and reinforcement-learning approaches operate under different assumptions (e.g., cluster formation vs. fog scheduling, relaxed latency constraints, or online training with additional state features), which would hinder an apples-to-apples comparison here. We therefore treat these as complementary rather than directly comparable baselines.

5.3. Evaluation criteria

We evaluate the performance of the proposed MUOM in terms of evaluation metrics, including the number of fog nodes, the execution and transmission power, the computing capacity, the data size, the degree of workload imbalance, and the standard deviation of the workload imbalance.

The degree of workload imbalance shows the imbalance of workload among fog nodes. The imbalance can be calculated by considering the formula from [24], as shown in (23).

$$WL = \frac{(Max(R_l) - Min(R_l))}{R_{average}} \ , \quad l = 1,\ 2, \ldots N \tag{23}$$

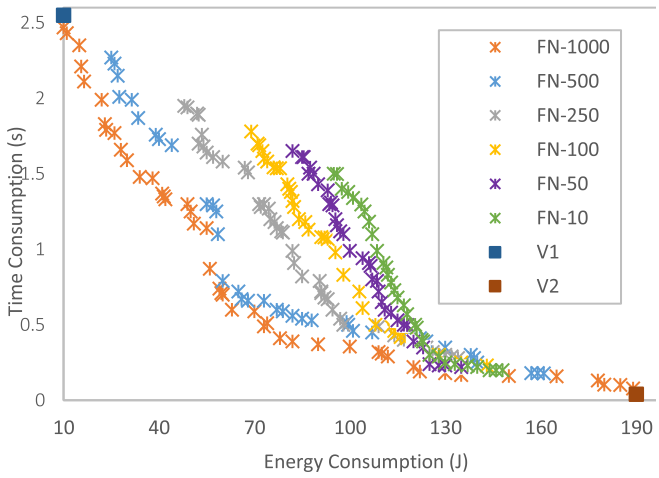
where $R_l = T(fn_{l,j})$. The workload imbalance WL is the difference of $T(fn_{l,j})$ of fog nodes to the average $T(fn_{l,j})$ of fog nodes.

The standard deviation evaluates the workload distribution among the fog nodes. The smaller the value of deviation, the higher the workload balanced between the fog nodes. The standard deviation can be calculated from [24] as

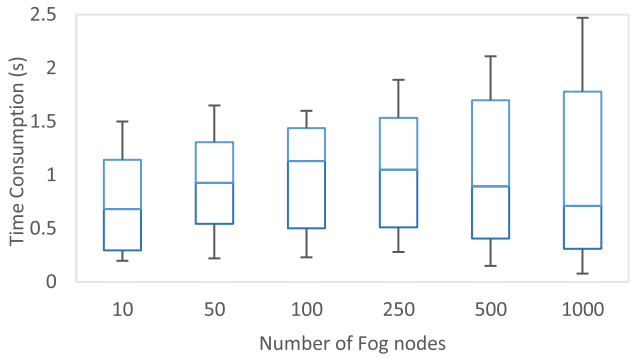
$$S.D = \sqrt{\frac{\sum_l (R_l - R_{average})^2}{N}} \tag{24}$$

Table 2
Parameters Settings.

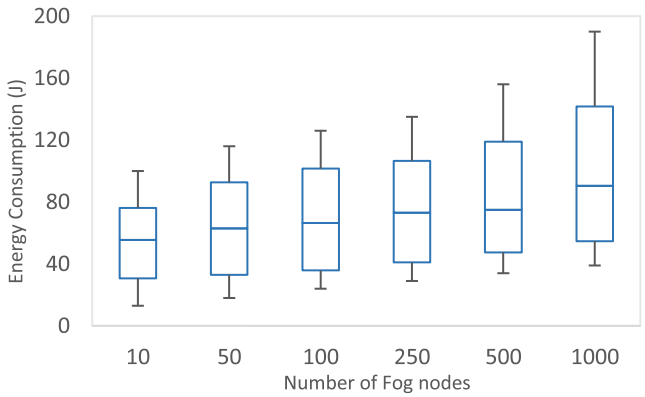
Parameter	Value
The Bandwidth of LAN	250 MB/s
The Bandwidth of WAN	20 MB/s
The Latency of LAN	(0.2–20) ms
The Latency of WAN	20 ms
Number of Fog nodes	10–1000
Number of Sensor nodes	5–1000
The Idle Power of Fog nodes	50 mW
The Active Power of Fog nodes	(100–500) mW
The Transmission Power	(100–500) mW
The computing capacity of Fog nodes	(1–50) GHz



(a)

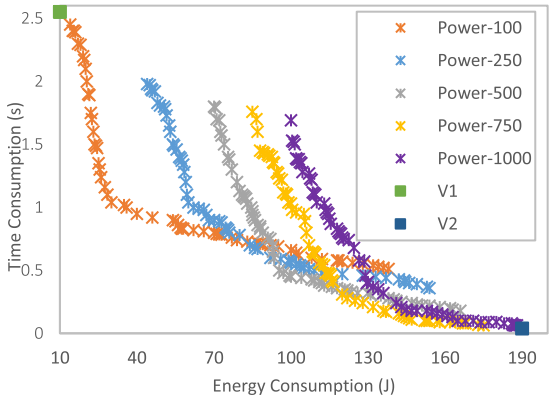


(b)

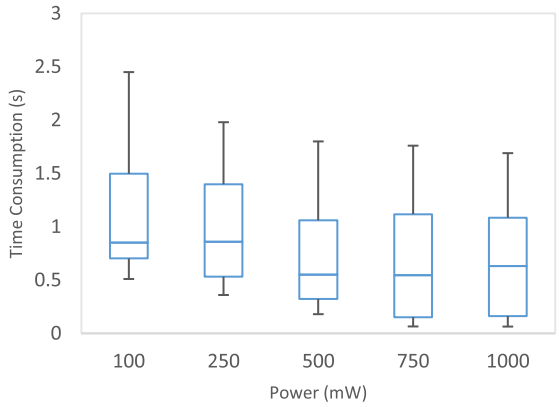


(c)

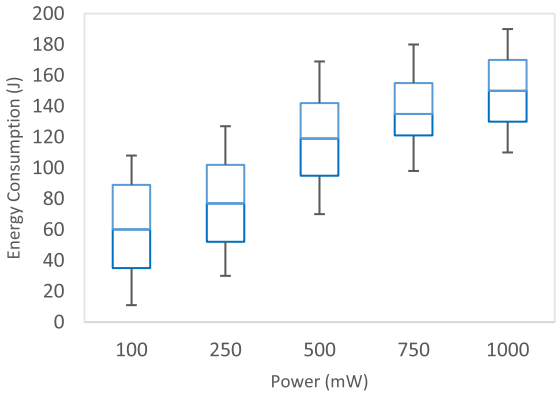
Fig. 5. Impact of the number of fog nodes. (a) Pareto front for optimal solutions. (b) Box plots of the time consumption in a varying number of fog nodes. (c) Box plots of the energy consumption in a varying number of fog nodes.



(a)



(b)



(c)

Fig. 6. Impact of execution and transmission power. (a) Pareto front for optimal solutions. (b) Box plots of the time consumption in varying power values. (c) Box plots of the energy consumption in varying power values.

5.4. Performance evaluation of proposed MUOM

This section evaluates MUOM to analyse the impact of the number of fog nodes, the execution and transmission power, and the computing capacity of fog nodes.

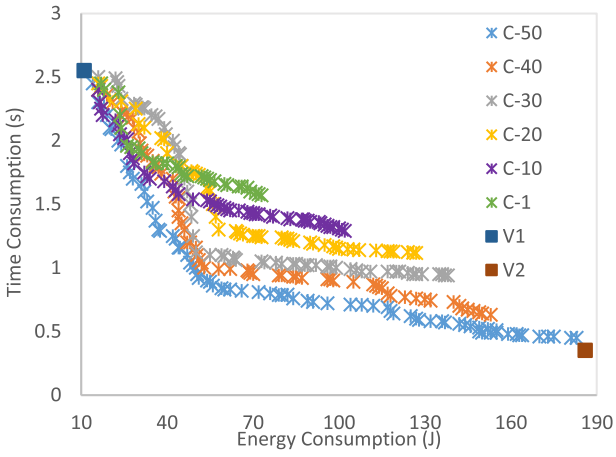


Fig. 7. Impact of the computing capacity of fog nodes. (a) Pareto front for optimal solutions. (b) Box plots of the time consumption in varying computing capacity values. (c) Box plots of the energy consumption in varying computing capacity values.

5.4.1. Impact of the number of fog nodes

We consider the impact of the number of fog nodes on $T(fn_{l,j})$ and $E(fn_{l,j})$ of our MUOM method. In the simulated network, fog nodes are responsible for performing the computing task $fn_{l,j}$ including data encryption, data division, distribution, storage, and additive aggregation. The number of fog nodes that can perform computing tasks ranges between 10 and 1000, as listed in Table 2. The results are illustrated in Figs. 5(a), (b), and (c).

According to the Pareto front chart in Fig. 5(a), we can observe that the MUOM method based on NSGA-III can get the optimal solutions $O(fn_{l,j})$ balance between extreme values of V1 and V2 for time consumption $T(fn_{l,j})$ and energy consumption $E(fn_{l,j})$. With the number of fog nodes increasing, the range of Pareto also increases, corresponding to the higher number of optimal solutions. For 1000 fog nodes (FN-1000), the optimal solutions are notably higher than the 500 fog nodes or fewer fog nodes. The higher number of optimal solutions is because of the higher computation tasks $fn_{l,j}$ with the increase in fog network size. We can also notice that the optimal solutions have a high degree of overlap for the number of fog nodes <250 at reaching particular time and energy levels.

From our analysis, it can be concluded that the dimension of decision-making for optimal solutions becomes more extensive with a higher number of fog nodes than with fewer fog nodes. Figs. 5(b) and (c) depict the impact of the number of fog nodes on the time consumption $T(fn_{l,j})$ and energy consumption $E(fn_{l,j})$, respectively. The average time consumption $T(fn_{l,j})$ and energy consumption $E(fn_{l,j})$ show a positive correlation with the increase in the number of fog nodes.

Although the increase in the number of fog nodes minimises the execution time T_{exe} by a division of computing tasks $fn_{l,j}$ among fog nodes. Still, the transmission time T_{trans} for requesting, transmitting, and authenticating $fn_{l,j}$ requires a larger amount of time than T_{exe}. Therefore, the overall time consumption $T(fn_{l,j})$ increases with an increase in the number of fog nodes, as shown in Fig. 5(b).

The relationship between energy consumption $E(fn_{l,j})$ and the number of fog nodes is shown in Fig. 5(c). Similar to T_{trans}, the transmission energy E_{trans} consumed by fog nodes for $fn_{l,j}$ including data transmission, data authentication, and data request becomes higher than the execution energy E_{exe}. This increase impacts the overall increase in energy consumption $E(fn_{l,j})$ for a higher number of fog nodes.

5.4.2. Impact of the execution and transmission power

We focus on the influence of execution and transmission power on the time consumption $T(fn_{l,j})$ and energy consumption $E(fn_{l,j})$ in this section. The range of power, including transmission p_{trans} and execution power p_a, p_i (idle and active power), varies from 100 to 1000 mW, as listed in Table 2. Fig. 6(a) depicts multiple optimal solutions in a Pareto chart with varying power values. From the Figure, we can observe that the proposed MUOM method based on NSGA-III can always find multiple optimal solutions between extreme V1 and V2 values. In addition, the optimal solutions for transmission and execution power greater than 500 have a high degree of overlapping. It can be concluded from the overlapping that the dimension of decision-making for optimal solutions becomes very small when the transmission and execution power reaches 500 mW.

The relationship between time consumption $T(fn_{l,j})$ and power is negatively correlated, as shown in Fig. 6(b). The processing and transmission speed of computing tasks $fn_{l,j}$ becomes faster with the more considerable power, which results in smaller T_{trans} and T_{exe} at fog nodes. In contrast, energy consumption $E(fn_{l,j})$ shows a positive correlation with power, as illustrated in Fig. 6(c). With an increase in p_{trans}, p_a and p_i, the higher energy consumption $E(fn_{l,j})$ is required to process $fn_{l,j}$ within a network.

Overall, we have noticed a slowdown in the trend for $E(fn_{l,j})$ increases and $T(fn_{l,j})$ decreases. Although the power is increasing evenly, still the degree of $T(fn_{l,j})$ and $E(fn_{l,j})$ is shrinking gradually. Therefore, the effect of transmission and execution power on the network is not higher than the impact of the other variables, including the number and computing capacity of fog nodes.

5.4.3. Impact of computing capacity of fog nodes

This section discusses the relationship between the computing capacity C_{cap} of fog nodes and the network performance. In our experiment, the range of computing capacity C_{cap} varies from 1 to 50 GHz, as listed in Table 2, and the results are depicted in Fig. 7.

The optimal solutions in the Pareto chart with a variation of fog nodes' computing capacity are shown in Fig. 7(a). We can observe that the multiple optimal solutions fall between extreme V1 and V2 values. The extreme value at the top left of Fig. 7(a) has the maximum time consumption $T(fn_{l,j})$ of 2.55 s and the minimum energy consumption $E(fn_{l,j})$ of 10 J. For the maximum time consumption $T(fn_{l,j})$ and the minimum energy consumption $E(fn_{l,j})$, each optimal set of Pareto has almost the same value as the extreme value at the top left. In addition to the apparent relationship, we also notice that the lower computing capacity C_{cap} values, i.e. C-1 and C-20, make the scope of the optimal solutions denser and smaller than values greater than 20 GHz.

Figs. 7(b) and (c) show the box plot relationship of time consumption $T(fn_{l,j})$ and energy consumption $E(fn_{l,j})$ with computing capacity C_{cap}. The time consumption $T(fn_{l,j})$ shows a negative correlation with computing capacity C_{cap}. As the computing capacity C_{cap} of fog nodes increase the execution time T_{exe}, and the transmission time T_{trans} for $T(fn_{l,j})$ becomes smaller. In contrast, the energy consumption $E(fn_{l,j})$ is positively correlated with computing capacity C_{cap}. The higher the computing capacity C_{cap}, the more energy is consumed by fog nodes for processing data.

From Figs. 7(b) and (c), it can be concluded that the time consumption $T(fn_{l,j})$ decreases moderately with the computing capacity C_{cap} increasing evenly. Similarly, the energy consumption $E(fn_{l,j})$ increases moderately with the computing capacity increase.

5.5. Comparative analysis

In this section, we evaluate the performance of the MUOM method with the N-opt, FCM, and PFCM methods. Time, energy

consumption, data size, and power consumption are the metrics used to assess MUOM and the performance of the other comparative methods. We also evaluate the standard deviation and compare the degree of imbalance of MUOM with that of PFCM.

5.5.1. Comparison of data sizes for time consumption and energy consumption

Figs. 8 and 9 present the time and energy consumption in data size from 1 to 1000 Kbps in processing the MUOM, N-opt, FCM, and PCFM methods. As discussed in Section B, all the computing tasks $fn_{i,j}$ in the N-opt methods are executed and transmitted without applying the fog layers' optimisation algorithm. Due to the lack of time and energy consumption optimisation, the transmission time and energy incur high overhead for data transmission, including encryption, authentication, and key distribution. With a larger data size, the transmission time and energy increase abruptly, resulting in an overall increase in time and energy consumption.

The reasons for the variation in FCM and PCFM results with data sizes for time and energy consumption are summarised in detail below.

On the one hand, the time consumption of FCM is higher than that of PFCM, as shown in Fig. 8. The fog nodes in FCM are connected to the cloud via WAN, which has lower bandwidth and higher latency compared to those interconnected in PFCM through LAN. In FCM, computing tasks and data storage are executed in the cloud. In contrast, PFCM executes computing tasks on fog layers and partially performs further execution on a cloud. Hence, less time is consumed when the computing task is executed on fog layers in PFCM than when executed entirely on the cloud in FCM.

Further, resources available for higher data size execution are limited and finite in fog layers. In the case of all fog nodes instantiated for computing tasks, the execution requests for the remaining tasks in the queue have to wait until the fog node's resources become available. In PFCM, only partial computing tasks are required to wait for resource availability in fog layers. Moreover, partial computing tasks are offloaded to the cloud. Also, the ACO algorithm optimises the time consumption for task offloading in fog layers.

On the other hand, the energy consumption in PFCM for computing tasks without optimising energy in fog layers is higher than FCM, as shown in Fig. 9. In PFCM, most computing tasks are executed at fog nodes, which requires higher energy consumption than all the computing tasks offloaded to the cloud in FCM. Further, ACO optimises only a single objective, i.e., time consumption in PCFM. Thus, energy consumption in PFCM is not optimised, and higher energy is consumed when the computing tasks are executed on fog layers than on the cloud, as in FCM.

Compared to MUOM, partial tasks offloading to the cloud increases the time and energy consumption of PCFM, as shown in Figs. 8 and 9. In MUOM, computing tasks are performed at fog layers, and no task is offloaded to the cloud. Furthermore, the optimal solutions provided in PCFM are based on an ACO algorithm with single-objective optimisation, i.e., time consumption, which incurs higher computational time and energy than MUOM. MUOM is based on NSGA-III, providing a hybrid strategy for multi-objective optimisation, i.e., time and energy consumption.

5.5.2. Comparison of power consumption

Figs. 10 and 11 illustrate the time and energy consumption in terms of power consumed by each of the four methods. Fig. 10 illustrates that the increase in power consumption reduces the overall time required for each method to execute and transmit computing tasks. The proposed MUOM method incurs significantly less time than the N-opt, FCM, and PFCM methods. Due to the lack of an optimisation algorithm, the time consumed by fog nodes with varying power consumption in the N-opt method is remarkably higher than the other three methods. In FCM, offloading tasks to the cloud requires a little higher power and time consumption than PFCM. Power consumption significantly impacts the energy consumed by executing and transmitting computing tasks, as shown in Fig. 11. We consider power to be one of the parameters used to measure the energy consumption of fog nodes.

We can conclude from Fig. 11 that the increase in power consumption results in higher energy consumption for executing and transmitting tasks. Besides, the average power consumption of MUOM is lower than that of FCM and PFCM. At the same time, the power consumption of MUOM is remarkably lower than that of the N-opt method. The lower power consumption of MUOM is the optimisation of energy consumption and the performance of computing tasks within a fog layer.

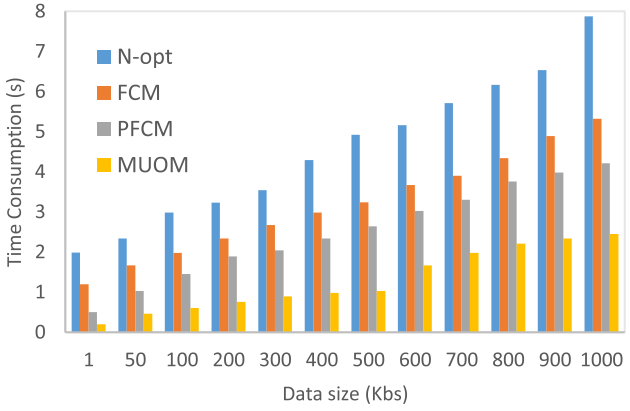


Fig. 8. Data size comparison for time consumption.

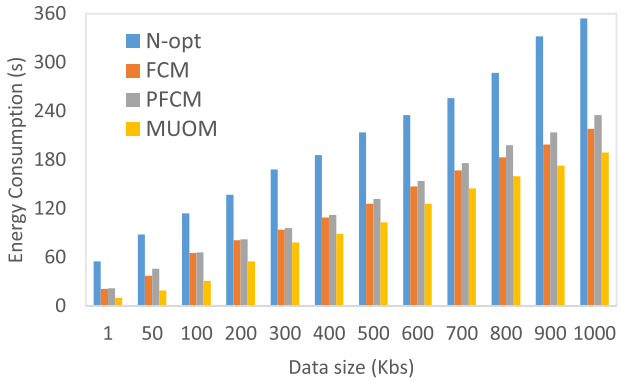


Fig. 9. Data size comparison for energy consumption.

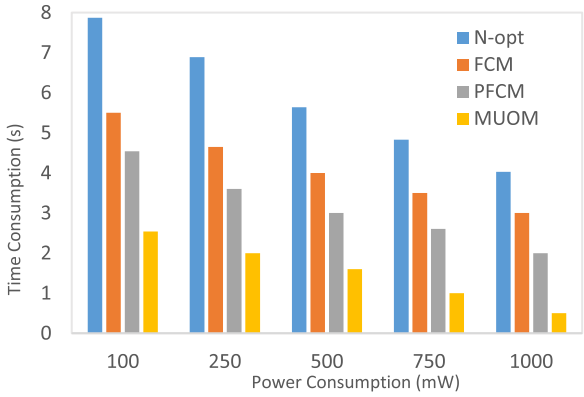


Fig. 10. Power comparison for time consumption.

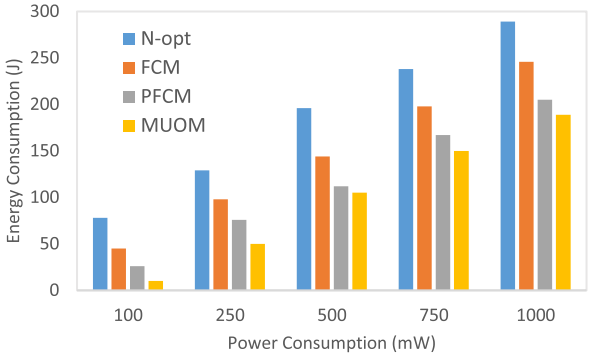


Fig. 11. Power comparison for energy consumption.

5.5.3. Comparison of workload imbalance

Fig. 12 depicts the degree of workload imbalance at the fog layers for the MUOM and PFCM methods with increasing fog nodes. The workload imbalance is evaluated according to (23), listed in subsection C. Fig. 12 shows that the degree of imbalance for MUOM is significantly less than PCFM, which means that MUOM effectively balances computing workload at fog nodes. The optimal solutions balance the workload among fog nodes to execute and transmit computing tasks.

In contrast, PFCM utilises the ACO algorithm only to optimise time, not to consider energy optimisation. Due to high execution and transmission energy, the workload imbalance in PFCM is remarkably higher than in MUOM.

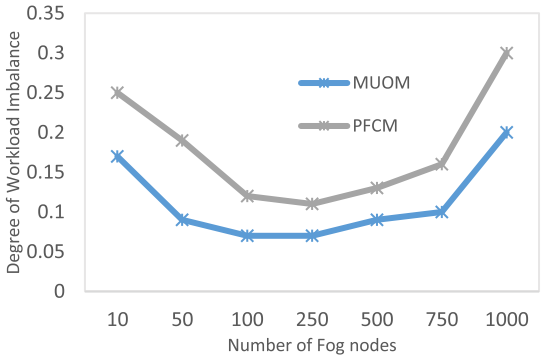


Fig. 12. The degree of workload imbalance.

Similarly, Fig. 13 shows the standard deviation of the workload distribution among fog nodes with an increasing number of fog nodes. We can see that the MUOM enhances the standard deviation compared to PFCM, indicating that smaller standard deviation values correspond to a higher workload balance and distribution among fog nodes.

Note that our experiments compute deterministic end-to-end metrics for fixed workloads and network parameters; as such, classical confidence intervals over repeated runs are not informative for this setup. When stochastic variability is introduced (e.g., random arrivals/links), we recommend reporting Pareto fronts across scenario samples along with bootstrap confidence intervals or percentile bands—a practice compatible with our framework but outside the scope of the present deterministic study.

6. Practical relevance & limitations

The results quantify the core time-energy trade-off under a clean baseline. In practice, adopting conservative parameters and periodically updating inputs with measurements preserves the usefulness of the optimisation without changing the underlying model or method. However, real fog-enabled IoT deployments may face (i) intermittent connectivity, (ii) time-varying/bursty data rates, and (iii) heterogeneous device capabilities. While these dynamics are outside our formal model, their qualitative impacts map directly onto our latency and energy components:

- *Intermittent connectivity* inflates effective transmission time and may create short-term queue build-ups. In deployment, this can be mitigated by conservative link budgets (e.g., using measured percentiles), lightweight retries/back-off, and periodic re-optimisation as conditions change.
- *Dynamic data generation* increases variability in queueing delay. Operators can smooth arrivals with short aggregation windows or admission control and evaluate plans against expected/percentile loads to avoid over-commitment.
- *Heterogeneity* alters per-node processing time and energy coefficients. Our optimisation logic is agnostic to how these coefficients are obtained (profiling/measurement) and thus remains applicable when devices differ in compute or power characteristics.

7. Conclusion

In this paper, we investigated the problem of time and energy consumption for data aggregation in fog computing. We provide two models for time and energy consumption in fog computing, and to optimise both models, we proposed MUOM based on the NSGA-III

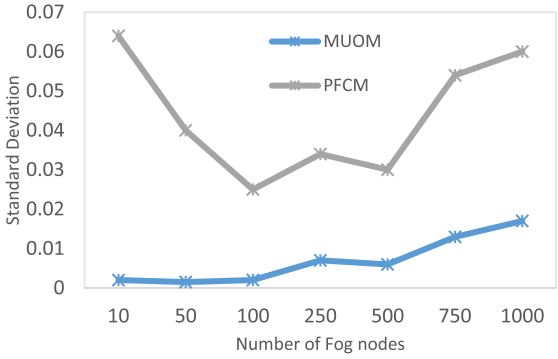


Fig. 13. Standard deviation of workload distribution.

algorithm. Furthermore, comprehensive experiments and evaluations are carried out to analyse and compare the performance of the proposed MUOM with the other methods. Our experiment results showed that our MUOM can always obtain the Pareto optimal solutions within the extreme values, and it outperforms the state-of-the-art methods in solving the optimisation problem. For future work, we will extend the proposed strategies to deal with more complex real-world scenarios of IoT. In addition, we will apply the proposed strategies for optimisation of data replication in Fog-enabled IoT.

CRediT authorship contribution statement

Sira Yongchareon: Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors thank Dr. Kinza Sarwar for her enormous help in designing and analysing the algorithms and the simulated IoT network presented in this paper.

Data availability

Data will be made available on request.

References

[1] Z. Guan, et al., APPA: an anonymous and privacy preserving data aggregation scheme for fog-enhanced IoT, J. Netw. Comput. Applic. 125 (2019) 82–92.
[2] R.H. Jhaveri, et al., Sensitivity analysis of an attack-pattern discovery based trusted routing scheme for mobile ad-hoc networks in industrial IoT, IEEe Access. 6 (2018) 20085–20103.
[3] X. Xu, et al., A computation offloading method over big data for IoT-enabled cloud-edge computing, Future Generat. Comput. Syst. 95 (2019) 522–533.
[4] X. Wang, et al., A distributed HOSVD method with its incremental computation for big data in cyber-physical-social systems, IEEe Trans. Comput. Soc. Syst. 5 (2) (2018) 481–492.
[5] Z. Chang, et al., Dynamic resource allocation and computation offloading for IoT fog computing system, IEEe Trans. Industr. Inform. (2020).
[6] T. Wang, et al., Edge-based differential privacy computing for sensor-cloud systems, J. Parallel. Distrib. Comput. 136 (2020) 75–85.
[7] R. Kemp, et al., Cuckoo: a computation offloading framework for smartphones, in: International Conference on Mobile Computing, Applications, and Services, Springer, 2010.
[8] Q. Wang, S. Chen, Latency-minimum offloading decision and resource allocation for fog-enabled Internet of Things networks, Transac. Emerg. Telecommun. Technol. 31 (12) (2020) e3880.
[9] M. Yang, et al., Machine learning differential privacy with multifunctional aggregation in a fog computing architecture, IEEe Access. 6 (2018) 17119–17129.
[10] S. Rani, P. Saini, *Fog computing: Applications and Secure Data aggregation*, in *Handbook of Computer Networks and Cyber Security*, Springer, 2020, pp. 475–492.
[11] J. Pan, et al., A Novel Fog Node Aggregation Approach for Users in Fog Computing Environment, in: 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), IEEE, 2020.
[12] K. Sarwar, et al., Lightweight, Divide-and-Conquer privacy-preserving data aggregation in fog computing, Future Generat. Comput. Syst. (2021).
[13] F. Gu, et al., Partitioning and offloading in smart mobile devices for mobile cloud computing: state of the art and future directions, J. Netw. Comput. Applic. 119 (2018) 83–96.
[14] A. Bhattacharya, P. De, A survey of adaptation techniques in computation offloading, J. Netw. Computer Applic. 78 (2017) 97–115.
[15] J. Bellendorf, Z.Á. Mann, Classification of optimisation problems in fog computing, Future Generat. Computer Syst. 107 (2020) 158–176.
[16] A. Yousefpour, G. Ishigaki, J.P. Jue, Fog computing: towards minimising delay in the internet of things, in: 2017 IEEE international conference on edge computing (EDGE), IEEE, 2017.
[17] A. Yousefpour, et al., FogPlan: a lightweight QoS-aware dynamic fog service provisioning framework, IEEe Internet. Things. J. 6 (3) (2019) 5080–5096.
[18] A. Yousefpour, et al., On reducing IoT service delay via fog offloading, IEEe Internet. Things. J. 5 (2) (2018) 998–1010.
[19] R. Deng, et al., Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption, IEEe Internet. Things. J. 3 (6) (2016) 1171–1181.
[20] L. Liu, et al., Multi-objective optimisation for computation offloading in fog computing, IEEe Internet. Things. J. 5 (1) (2017) 283–294.
[21] Y.-L. Jiang, et al., Energy-efficient task offloading for time-sensitive applications in fog computing, IEEe Syst. J. 13 (3) (2018) 2930–2941.
[22] S.A.A. Naqvi, et al., Metaheuristic optimisation technique for load balancing in cloud-fog environment integrated with smart grid, in: International Conference on Network-Based Information Systems, Springer, 2018.
[23] H.T.T. Binh, et al., An evolutionary algorithm for solving task scheduling problem in cloud-fog computing environment, in: Proceedings of the Ninth International Symposium on Information and Communication Technology, 2018.
[24] M.K. Hussein, M.H. Mousa, Efficient Task Offloading for IoT-Based Applications in Fog Computing Using Ant Colony Optimization, IEEe Access. 8 (2020) 37191–37201.
[25] K. Peng, et al., An energy-and cost-aware computation offloading method for workflow applications in mobile edge computing, EURASIP. J. Wirel. Commun. Netw. 2019 (1) (2019) 207.
[26] M. Aruldoss, T.M. Lakshmi, V.P. Venkatesan, A survey on multi criteria decision making methods and its applications, Am. J. Informat. Syst. 1 (1) (2013) 31–43.
[27] L. Cui, et al., Joint optimisation of energy consumption and latency in mobile edge computing for Internet of Things, IEEe Internet. Things. J. 6 (3) (2018) 4791–4803.
[28] A. Afshari, M. Mojahed, R.M. Yusuff, Simple additive weighting approach to personnel selection problem, Int. J. Innovat., Manag. Technol. 1 (5) (2010) 511.
[29] Y. Mao, J. Zhang, K.B. Letaief, Dynamic computation offloading for mobile-edge computing with energy harvesting devices, IEEE J. Selected Areas Commun. 34 (12) (2016) 3590–3605.

[30] G. Javadzadeh, et al., Mathematical model for the scheduling of real-time applications in IoT using Dew computing, J. Supercomput. 78 (2022) 7464–7488, https://doi.org/10.1007/s11227-021-04170-z.
[31] S. Abolhassani Khajeh, et al., Real-Time Scheduling in IoT Applications: a Systematic Review, Sensors 23 (1) (2023) 232, https://doi.org/10.3390/s23010232.
[32] A.A. Sadri, et al., Data reduction in fog computing and internet of things: a systematic literature survey, Internet of Things 20 (2022) 100629, https://doi.org/10.1016/j.iot.2022.100629. ISSN 2542-6605.