Computing
https://doi.org/10.1007/s00607-023-01215-4

REGULAR PAPER

An energy-efficient and deadline-aware workflow scheduling algorithm in the fog and cloud environment

Navid Khaledian[1,5] · Keyhan Khamforoosh[1] · Reza Akraminejad[2] · Laith Abualigah[3] · Danial Javaheri[4]

Received: 19 May 2023 / Accepted: 14 August 2023

Abstract

The Internet of Things (IoT) is constantly evolving. The variety of IoT applications has caused new demands to emerge on users' part and competition between computing service providers. On the one hand, an IoT application may exhibit several important criteria, such as deadline and runtime simultaneously, and it is confronted with resource limitations and high energy consumption on the other hand. This has turned to adopting a computing environment and scheduling as a fundamental challenge. To resolve the issue, IoT applications are considered in this paper as a workflow composed of a series of interdependent tasks. The tasks in the same workflow (at the same level) are subject to priorities and deadlines for execution, making the problem far more complex and closer to the real world. In this paper, a hybrid Particle Swarm Optimization and Simulated Annealing algorithm (PSO–SA) is used for prioritizing tasks and improving fitness function. Our proposed method managed the task allocation and optimized energy consumption and makespan at the fog-cloud environment nodes. The simulation results indicated that the PSO–SA enhanced energy and makespan by 5% and 9% respectively on average compared with the baseline algorithm (IKH-EFT).

Keywords Fog computing · Workflow scheduling · Metaheuristic algorithm · Particle swarm optimization · Energy efficiency · Deadline

Mathematics Subject Classification 68M20

1 Introduction

As inexpensive computer chips develop and widespread Internet access is provided, the Internet of Things (IoT) has become an important part of the digital world. The applications of IoT encompass virtually all crucial aspects of modern life, such as

Extended author information available on the last page of the article

Published online: 26 August 2023

healthcare [1], underwater exploration [2], transportation and smart homes [3], and industrial and mobile systems [4]. According to Cisco [5], 15 billion IoT devices will be connected to the Internet by 2023, 21 billion by 2025, and 41 billion by 2027. Statistics indicate that 18.3 zettabytes of data were generated in 2019 by IoT devices. Meanwhile, the International Data Corporation predicts an increase of 400 percent in the forthcoming years, denoting a value of 73 zettabytes by 2025 [6]. However, most IoT devices have constrained computational power and storage space, making deploying apps and conducting data analysis difficult. Therefore, data must be sent to alternative servers for processing and storage. There should be a particular concern for processing, storage, and transfer of this huge amount of IoT data when different IoT applications are considered. Despite the high storage and processing capability of cloud computing, it causes long delays in data transfer and response time due to the centralized nature of the devices and the long distances between them. Consequently, implementing cloud-based IoT applications may fail to meet a wide variety of their requirements, particularly for delay-sensitive ones. Therefore, it is proposed that fog computing should be applied to complement cloud computing [7, 8].

Due to the proximity of IoT devices to fog servers, they can provide services similar to those of a cloud with less delay. Therefore, fog computations efficiently meet the requirements of delay-sensitive applications in IoT. Moreover, a fog server enables temporary data storage, which helps make real-time decisions. Thus, the fog mitigates processing time, delay, and cost of service provision. Fog computations can also help reduce the energy consumed by IoT devices supplied by limited batteries since data processing requires energy consumption and will decrease the energy consumed by an IoT device if carried out in the fog [9]. Therefore, IoT processing requests are sent to the fog to mitigate delay and energy. If a request requires further processing that the fog device cannot carry out, it will be sent to the cloud to be processed there [3, 10]. In fog-cloud computing, each task has different bandwidths, processing and storage resources, resource costs, and response times [11]. Therefore, task scheduling involves a large number of challenges in these environments. Paying attention to the delay in scheduling alone cannot meet the needs of IoT applications. Since every application demands to be executed within a specific time limit, if it is conducted after the time limit, it may cause the application to fail. Therefore in this research, attention is paid to energy and deadline simultaneously.

Scheduling has been turned into a critical issue by challenges including load balancing, scalability, reliability, reallocation, concern for performance, dynamicity of processing resources, etc. [12, 13]. Scheduling aims to allocate tasks to resources to optimize resource utilization. In many IoT applications, tasks are addressed as workflows. A workflow considers tasks assets of interdependent ones that need to be performed in a particular order and with specific priorities [14]. In a workflow, task scheduling is represented as a directed acyclic graph (DAG). Each node indicates a task, and the weight denotes runtime or computational cost. The graph edges show the prerequisite relationships between work units and represent workflow [14, 15]. Most workflows in an application are scheduled with a particular purpose. The aim can be to reduce cost, data transfer time, utilized storage space, energy consumption, ultimate runtime, or a combination thereof. Among the criteria, makespan is

of great significance, as it increases the productivity of computational resources and user satisfaction. Therefore, the algorithms attempt to mitigate the times required for processing and workflow implementation as far as possible or to finish workflow implementation before a particular deadline Workflow scheduling also has the goal of lowering energy usage, which lowers costs and protects the environment [16, 17]. Energy productivity techniques can be classified into two groups [18]. Energy productivity is achieved at both software and hardware levels. Optimization at the hardware level is carried out by replacing logic gates with low consumption, leading to an optimal architecture. Energy is decreased at the software level by optimizing task scheduling and resource consumption [19–21]. An idle server consumes 70 percent of its energy at the data center concerning its peak consumption [22, 23]. Researchers have proposed many methods and approaches to reduce energy consumption at data centers. One of these methods is to efficiently use the resources available at the data center using scheduling algorithms to organize the allocation of tasks to servers. Deadline and priority are other important criteria that need to be considered in the scheduling problem since some tasks and requests should be performed quickly within the set period in the real world. Their implementation afterward will be useless [24].

Given the importance of the examined criteria, the problem of energy consumption concerning huge numerous IoT devices will lead to unrecoverable damages to the environment as stated Co_2 gas emission is increasing the energy consumption of datacenters. Thus, managing incoming requests from the data centers is important to the environment and also the budget for both business owners and users. Efficient scheduling, considering deadlines of workflows with parameters of energy and cost will overcome this issue up to a significant value.

This research seeks to consider the criteria of energy consumption, deadline, and task priority simultaneously in the workflow. This will increase satisfaction on the users' and companies' part, since users like their requests to be responded to within the shortest time and without delay, and companies providing services like their costs to be reduced. In contrast, huge costs are imposed on them by energy consumption. We suggest a successful IoT task scheduling technique that minimizes energy consumption and meets the deadline and task priority prerequisites to confront the above problems. For that purpose, we consider task scheduling as a mixed-integer linear programming (MILP) problem to have IoT tasks performed before the set deadlines considering priority and minimizing energy consumption. We also consider deadline violation time minimization and priority in our model and combine the improved PSO and SA algorithms to obtain the optimal solution. The main contributions of this paper are mentioned in the following:

- Presenting a mathematical model for simultaneously minimized deadline, makespan and optimizing energy consumption in fog nodes.
- Solving the proposed model using a hybrid two-step meta-heuristic algorithm (smart tasks prioritization is done by PSO and allocation by SA).
- Improvement of PSO algorithm to efficiently map the workflow to available fog and cloud nodes.

The rest of the paper is organized as follows. In Sect. 2, we review the research literature. In Sect. 3, we discuss the problem architecture and formula. Section 4 examines the proposed method to minimize energy consumption and reduce makespan, and deadline violation gave task priority. Finally, the experimental results and conclusions appear in Sects. 5 and 6, respectively.

2 Literature review

In [25], a task-loading algorithm has been proposed using smart ant colony optimization (SACO) inspired by the nature of metaheuristic programming for task offloading in IoT sensor applications in a fog environment. In [26], an alternative task scheduling method called AEOSSA has been suggested for IoT requests in a fog-cloud environment. This method is built on an artificial ecosystem (AEO). The authors focus on improving makespan and throughput but disregard energy consumption and deadline. As stated earlier, IoT tasks are loaded in fog computing rather than the cloud to raise the quality of service (QoS) required by many applications. However, IoT applications are limited by the constant availability of computational resources on fog computing servers since the large volume of data created by IoT devices causes network traffic and increases overhead. As a result, task scheduling is a significant problem that requires effective resolution. AOAM model is an energy-aware model developed to solve the problem of task scheduling in fog computing [27]. The goal of AOAM is to maximize time intervals and improve user QoS. The model uses the search operators in the Marine Predator Algorithm (MPA) to generate various solutions and address local optimization problems. AOAM is validated using several parameters, including clients, data centers, hosts, virtual machines, tasks, and standard evaluation criteria such as energy and lifetime. The model is designed to be efficient and effective in addressing the task scheduling problem in fog computing, and it has been evaluated using rigorous testing methods.

Management and placement of smart tasks on fog platforms are challenging on a large scale due to the essential nature of the modern workflow and the requirements of a user sensitive to low energy consumption and response time. New platforms have emerged to mitigate the drawbacks of earlier methods, either using heuristic methods to achieve scheduling decisions rapidly or using methods based on artificial intelligence, such as reinforcement learning and evolutionary approaches to adapt to dynamic scenarios[28, 29]. The former methods are primarily incapable of fast adaptation in a highly dynamic environment, while the latter is characterized by low runtimes, negatively affecting response time. Thus, scheduling policies are required to be reactive and, therefore, efficient in an unstable environment while exhibiting low scheduling overhead costs. For this purpose, a gradient-based optimization strategy has been proposed in [30] using gradient backpropagation given the input (GOBI). The experiments conducted using GOBI methods with real data on fog applications exhibit considerable improvements over the state-of-the-art algorithms regarding energy consumption, response time, service level, and scheduling. In fog computing, workflow scheduling is an NP-hard issue that aims to assign the best

possible set of resources to the workflow while considering various goals, including deadlines, costs, energy, and QoS.

Fog providers, however, can experience high IoT network loads, breaking the service-level agreement (SLA). A hidden Markov model (HMM) has been used in [31] to predict the availability of each fog computing provider given variables like the volume of requests received at each node, missed workflow deadlines, and fog task loading in cloud computing. The architecture is made up of several fog computing providers. The unsupervised Baum-Welch approach trains the HMM, and the Viterbi algorithm determines the likelihood that each node is reachable. Then, the probability of access to fog computing providers is used to select one to schedule IoT workflows. The rigorous testing done using iFogSim shows that the proposed plan can perform better than the state-of-the-art by drastically lowering the number of tasks loaded in cloud computing, missed workflow deadlines, and SLA breaches.

The fog computing paradigm has evolved as a distributed computational approach for application presentation using fog nodes nearby IoT devices [28]. The amount of data needed to process IoT services has exponentially increased due to the quick development of IoT-based applications and the advent of 5G networks. Since IoT applications are designed as multiple IoT services with various QoS requirements that can be deployed at fog nodes with various resource capabilities in the fog ecosystem, it is difficult to devise a plan for optimal service placement. [32] makes a good case for an effective approach to the placement of IoT services based on autonomous IoT application establishment in fog infrastructure. The suggested method uses the metaheuristic whale optimization algorithm to design a strategy for effective service placement while monitoring the IoT QoS requirements and the capabilities of the available fog nodes (WOA). To further achieve the required IoT service placement plan while satisfying the QoS criteria of each service, an evolutionary mechanism uses throughput and energy usage as objective functions. In order to deliver useful services at the network edge and complement the cloud computing paradigm, fog computing has just been created. Placing applications in fog infrastructures is difficult due to the heterogeneity of the fog computing nodes, which calls for effective management to suit the application's needs. A bi-objective task allocation mechanism for fog computing environments has been suggested in [29]. This approach aims to appropriately position the task modules on the underlying fog devices, considering the security constraints and critical application levels. The non-dominated sorting genetic algorithm (NSGA-II) is used to solve the allocation problem, which is presented as a bi-objective knapsack algorithm. The simulation findings show that, in comparison to previous metaheuristic-based processes, the suggested solution boosts resource consumption and the ratio of service acceptance while decreasing service delay and energy usage. The simulation results show how the suggested algorithm may be used to deploy applications in a fog computing environment in such a way as to maximize their performance, throughput, and rate of security satisfaction.

Recent researchers have focused on efficient methods involving the capabilities of an edge network to run and support IoT applications and the relevant requirements. To meet the application requirements successfully while using the power of cloud computing efficiently, smart scheduling approaches are needed to optimize

IoT application task scheduling in computational resources. In [30], The ideal solution for IoT application job scheduling has been specified. However, when application components were set up improperly on the fog computing infrastructure, bandwidth and resource loss would be increased, resulting in high energy usage and poor QoS. In [33], the authors considered reducing bandwidth loss given the dependence of application components in their distributed establishment. On the other hand, service reliability will decrease if application components are established at a single node due to a power consumption management perspective. In order to address this point of failure and improve the application's resistance to it, a mechanism is described. After that, a multi-objective optimization algorithm's component setup is developed with the goal of minimizing energy consumption and overall delay for each pair of application-relevant components. A multi-objective cuckoo search algorithm (MOCSA) is suggested to resolve the hybrid optimization issue. The algorithm is tested under various circumstances compared to several contemporary techniques for validation reasons. Optimal use of the node energy has always been one of the greatest challenges in a wireless sensor network (WSN). Another crucial issue is lengthening network life, given the limited lifetimes of nodes on these networks and energy management. Two computational distributions have been presented for a dynamic wireless sensor network [34, 35]. An optimistic and a blind strategy are used in this fog-based system to divide the computational load among fog networks. The distribution, map, transfer, and combination (DMTC) processes comprise the four main steps of the approach given. Based on the suggested distribution methods, fuzzy multiple-attribute decision-making (fuzzy MADM) is also utilized for clustering and network routing. The outcomes show that the optimistic method outperforms the blind method and consumes less energy, especially across wide-area networks.

When many of an IoT user's new applications are connected to fog nodes, load examination becomes an issue in fog computing. The huge amount of data that transfers from IoT devices to remote cloud servers causes delay and excessive use of bandwidth [32]. An Internet-based distributed computational model called fog computing has emerged for the purpose of storing datasets produced by IoT devices around the user. The method facilitates the allocation of the required resources by removing inactive services. Since an IoT device constantly generates large amounts of data, it is challenging to place it at a storage fog node with different capabilities of decreasing delay and data access cost and increasing dataset reliability and accessibility while meeting the QoS requirements as a task. A data placement mechanism based on a metaheuristic algorithm has been proposed in [36] for data-based IoT applications in the fog environment using biogeography-based optimization (BBO). Additionally, for the issue of data placement in the fog ecosystem, a different framework is created to represent the transmission of data copies between IoT devices and storage fog nodes. Energy consumption is an essential factor that can directly affect the costs of CO_2 storage and emission in a fog environment. It can be reduced using efficient scheduling approaches where tasks are drawn on the best resources relevant to several contradictory objectives [37]. To address these issues, In [38], an opposition-based hybrid discrete optimization algorithm referred to as DMFO-DE, a discrete version of the moth-flame optimization (MFO) algorithm based on the opposition (OBL) is presented and combined with the differential evolution (DE)

algorithm to improve convergence speed. The problem of local optima in DMFO-DE for workflow scheduling in a fog computing environment is resolved using the methods of dynamic voltage and frequency scaling (DVFS). The order in which activities are completed in a workflow is determined using the heterogeneous earliest finish time (HEFT) method. By using fewer virtual machines (VMs) and establishing connections between related processes, the workflow scheduling strategy aims to reduce energy usage in the scheduling process.

The authors of [39] have solved the workflow scheduling problem using a hybrid algorithm involving plant growth optimization, simulated annealing, and water cycle optimization to reduce cost, delay, and energy consumption in the cloud and fog clusters. First, the plant growth optimization algorithm obtains the appropriate clusters based on cost and energy. Resources are then selected to be allocated to tasks based on the simulated annealing algorithm. Next, tasks immigrate between nodes to balance the load based on the water cycle optimization algorithm. Workflow scheduling has been formulated in [40] as a bi-objective problem in cloud computing. A hybrid algorithm involving simulated annealing and the iteration technique is used to solve the problem. Task priority is based on the initial population on the HEFT approach, including upward, downward, and level ranking, which are iterated. The authors of [41] have addressed the multi-objective workflow scheduling problem with a focus on the objectives of finish time, response time, and cost. They propose the multi-agent system technique based on the Genetic Algorithm to solve the problem. The approach involves the conversion of the workflow graph into a set of sub-workflows, which can be implemented using unique fog and cloud nodes, each identified as an agent. In [42], a heuristic algorithm involves displaying task priorities and selecting resources to solve the multi-objective workflow scheduling problem in the cloud computing environment to reduce cost and finish time. Upward ranking prioritizes tasks, and resources are selected based on Pareto ranking and crowding distance. Navid et al. have also presented a hybrid improved krill herd algorithm and an earliest finish time technique called IKH-EFT. Different navigations are carried out there in the workflow task prioritization step based on the krill movements and the DCD, SCD, and LCD methods, leading to the smart generation of dependent tasks. The authors use the earliest finish time technique in the task allocation step and the DVFS technique to reduce energy consumption. The results indicate simultaneous decreases in energy consumption makespan and monetary costs in the fog-cloud environment [43].

For instance-intensive workflows in the cloud environment, a heuristic algorithm named (HDECO) has been developed to optimize energy and cost [44]. An intelligent threshold detector was created to detect CPU consumption and alter the threshold of the CPU in order to prevent the powering on or off of several servers. Despite the threshold detector's distinctiveness, neither a meta-heuristic methodology nor scientific procedures were studied. Researchers in [45] investigated a lossless electroencephalogram (EEG) data compression technique. By employing lossless compression, k-means clustering, and Huffman encoding, the suggested method lowered the amount of EEG data provided to the fog gateway. It minimized the size of the IoMT EEG data and used Naive Bayes to determine the patient's epileptic seizure state. Through measurements and comparative data, the approach's efficacy was

demonstrated, ensuring accurate epileptic seizure detection. Despite their novelty method in a fog environment case study was not in the scheduling field.

In [45] sensor devices clustered using DBSCAN method applied Cuckoo Scheduling Algorithm (CSA) at cluster heads. CSA scheduled sensor nodes for optimal coverage, with cluster head pooling and CSA executed periodically. They studied energy efficiency without cost parameters. In order to maintain coverage and extend the lifetime of Wireless Sensor Networks (WSNs), an Energy-efficient Particle Swarm Optimization for Lifetime Coverage Prolongation (EPSOLCOP) protocol was presented. The protocol broadcasted the protocol among sensor nodes and subdivided the target sensing field into smaller subfields. In order to choose the optimum sensor nodes to cover the sector, ensure low energy consumption, and maximize WSN lifetime, the cluster head used PSO. According to the simulation results, EPSOLCOP was competitive and reached good coverage ratios while using less energy [46].

To increase the lifetime and coverage of wireless sensor networks, [45] presented the distributed genetic algorithm for lifetime coverage optimization (DiGALCO). It incorporated genetic algorithm optimization based on sensor activity scheduling, distributed cluster head selection, and virtual network subdivision. Cluster heads pick the active sensors to be monitored in rounds of DiGAfLCO work. Experimental results showed DiGALCO can prolong WSN lifetime and improve coverage performance. To improve the transmission and lifespan of IoT sensor networks, an Energy-efficient Transmission Optimization Protocol (ETOP) was presented as a solution to problems with transmitted readings, energy use, and network longevity in a growing number of sensor nodes in IoT. Before transmission, ETOP used correlation clustering based on reduction algorithms to eliminate redundant data. As a result, ETOP outperforms other techniques [47]. Authors in [48–51] also studied different problems in IoT and fog for optimizing energy with different methods.

Researchers looked at the layered IoT architecture, assessment criteria, and applications of fog computing over the previous 4 years. They discussed heterogeneous vehicular fog networks and vehicular fog frameworks and strategies. The study examined issues with interoperable processing and communication in fog networks as well as potential remedies [46]. In [47] study on multi-agent systems (MAS) in IoT applications and utilizing them for different purposes was done. They proposed that MAS can be embedded into IoT layers and optimize energy consumption. A new Energy-aware Data Offloading algorithm (EaDO) was proposed to improve latency and energy. Authors used scheduling based on queuing rules and Hall's graph theory [48].

3 Preliminaries

IoT devices are geographically distributed at different places, generate requests requiring plenty of computation, and are sensitive to delay. These devices send their data to the higher layer to be processed at that level [47]. Delay-sensitive requests call for (virtually) real-time processing. For instance, analyzing the data generated by a health monitoring system has to be urgent, as delays in such

systems can have disastrous consequences. An application is a collection of related tasks that must be completed in a given sequence and with a specific order of priority. The fog layer is an intermediary one that is situated near the terminal layer devices, between IoT devices and cloud data centers. There is also a particular node in this layer, known as the fog-breaker, responsible for central management and task scheduling. It is responsible for collecting user requests and managing resources at the fog-cloud nodes. This node also generates the most appropriate scheduling for workflows. The cloud, which uses powerful computers for processing and storage and demonstrates tremendous efficiency, is the top layer in this design. It offers services for heavy workloads, computers, and equipment for fast, secure data storage. Given those cloud layer servers are far away from data resources, processes may be sent with some delay. Therefore, it is suggested that delay-insensitive tasks with high complexity but no particular deadline should be performed in this layer [48]. Delay-tolerant tasks are typically delivered to the cloud environment, whereas delay-sensitive tasks are typically sent to the fog environment.

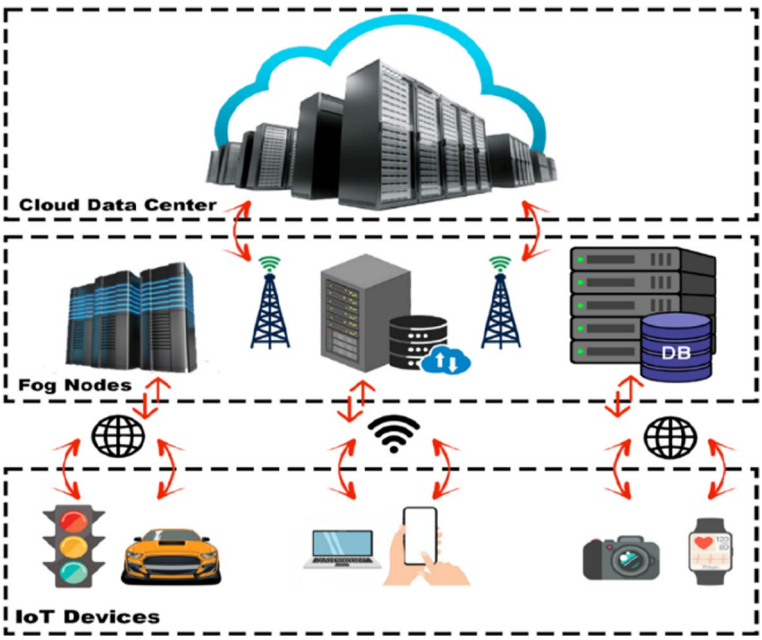


Fig. 1 Three-layer structure

Table 1 Notations used in the formulae and method [24]

	Notation	Description	Notation	Description
Input variables	e_{ij}	The link between nodes i and j	Ta_k^s	Size of task k
	$\mathbb{FN}_i^c$	CPU processing power at fog node i	Ta_k^d	Deadline required by task k
	$\mathbb{FN}_i^{act}$	Energy consumption at fog node i in the active mode	Ta_k^p	Priority of task k
	$\mathbb{FN}_i^{idl}$	Energy consumption at fog node i in the idle mode	Ta_k^{in}	Input file size of task k
	e_{ij}^p	Propagation delay between nodes i and j	Ta_k^{out}	Output file size of task k
	e_{ij}^b	Bandwidth between nodes i and j	Ta_k^f	Number of the parent of task k
Formulation variables	d_k^{prp}	Propagation delay experienced by task k	$S^{p\%}$	Percentage of satisfied IoT tasks given the priorities
	d_k^{tx}	Transfer time of task k	n^p	Sum of the priorities of all the tasks
	d_k^{exe}	Execution time of task k	ν_k	Deadline violation time of task k
	d_k^{wt}	Waiting time of task k	V^{tot}	Total violation time of the tasks given the priorities
	R_k	Response time of task k	A_i	Active time of fog node i for processing all the allocated tasks
	s_k	Satisfied deadline of task k	MS	Maximum fog node active time
	N_s	Number of IoT tasks with satisfied deadlines	ι_i	Idle time of fog node i
	N_s^p	Number of IoT tasks with satisfied deadlines given the priorities	ε_i	Estimated energy consumption of fog node i
	$S^{\%}$	Percentage of IoT tasks with satisfied deadlines	ε^{tot}	Total system energy consumption
Decision variables	x_{ik}	Allocation of task k to fog node i		
	y_{ij}^k	Selection of e_{ij} to route task k		
	z_{kl}	Performance priority of task i over j		

Figure 1 shows the architecture of the proposed method. It involves three layers: IoT, fog, and cloud [49]. The IoT layer includes smart devices (for example, sensors, smart cars, home appliances, and smartphones).

Table 1 shows the notations used in this paper. Attempts have been made to use standard notations.

3.1 Problem formulation

Based on the architecture described above, we formulate each part and then describe the mathematical model of the problem by combining each part.

3.1.1 Fog layer

The fog layer is composed of several heterogeneous devices. These devices and the relations between them can be considered as a graph $G = (\mathbb{FN};L)$, where $\mathbb{FN} = \left\{\mathbb{FN}_1, \mathbb{FN}_2, \ldots, \mathbb{FN}_m\right\}$ is a set of m fog devices, and $L = \left\{e_{ij} | i,j \in \mathbb{FN}\right\}$ is a set of links between the $\mathbb{FN}$ s. Each $\mathbb{FN}_i \in \mathbb{FN}$ involves the following features[24]:

- CPU processing power of $\mathbb{FN}_i^c$ MIPS (million instructions per second),
- Energy consumption of $\mathbb{FN}_i^{act}$ Watts in the active mode,
- Energy consumption of $\mathbb{FN}_i^{idl}$ Watts in the idle mode.

Moreover, each link $e_{ij} \in L$ is associated with two major features:

- Bandwidth of e_{ij}^b Megabits per second,
- Propagation delay of e_{ij}^p Milliseconds.

3.1.2 IoT layer

A workflow can be considered a set of tasks loaded in a particular period from IoT devices onto a fog controller (FC). The $Ta_k = Ta_k^s,\ Ta_k^d,\ Ta_k^p,\ Ta_k^{in},\ Ta_k^{out},\ Ta_k^f$ quadruple defines each task, where Ta_k^s is task size in MIPS, Ta_k^d is the required deadline in milliseconds, Ta_k^p is task priority (with a larger number denoting a higher priority), Ta_k^{in} is the input file size in kilobytes, Ta_k^{out} is output file size in kilobytes, and Ta_k^f is parent task number (zero for initial tasks) [24].

3.1.3 Decision variables

Three decision variables are used to allocate tasks to fog devices. The first one is a binary matrix $X_{m\times n}$ that indicates how tasks are assigned to fog devices and is defined as follows:

$$x_{ik} = \begin{cases} 1 \text{ if task } Ta_k \text{ is assigned to } \mathbb{FN}_i \\ 0 \text{ otherwise} \end{cases} \quad \forall i \in \mathbb{FN},\ \forall k \in Ta \tag{1}$$

The second variable specifies if a task is processed at the same fog device as those depending on it. In that case, the transfer cost will be zero; otherwise, this cost will also need to be considered. For each task Ta_k, therefore, a binary variable $y_{ij}^k \in \{0, 1\}$ indicates whether the link $e_{ij} \in L$ is selected to transfer the task. This variable is defined as follows:

$$y_{ij}^k = \begin{cases} 1 \text{ if link } \mathrm{e}_{ij} \text{ is chosen for routing } Ta_k \\ 0 \text{ otherwise} \end{cases} \quad \forall i,\ j \in \mathbb{FN},\ \forall k \in Ta \tag{2}$$

Moreover, a binary variable $z_{kl} \in \{0, 1\}$ is used to compare the priorities of two tasks Ta_k and Ta_l. For this purpose, we set $z_{kl} = 1$ if task Ta_k is prioritized over Ta_l, and we set $z_{kl} = 0$ otherwise.

3.1.4 Energy consumption model

In most research, a significant measure is an energy that FNs use to do all tasks. It is computed based on the amount of energy used in both active and idle modes. The active period of FN_i is the time that it requires to process all the tasks allocated to it, calculated using the following formula.

$$A_i = \sum_{\forall k \in Ta} \left(d_k^{exe} \times x_{ik}\right), \forall i \in \mathbb{FN} \tag{3}$$

The idle time $\mathbb{FN}_i$ can be calculated given the maximum runtime of each $\mathbb{FN}$ and makespan (MS), as follows.

$$\begin{aligned} MS &= \max_{\forall i \in \mathbb{FN}} \left(A_i\right) \\ \iota_i &= \left(MS - A_i\right), \forall i \in \mathbb{FN} \end{aligned} \tag{4}$$

Therefore, the estimated consumed energy FN_i is as follows based on the power consumption specifications and the active and inactive periods.

$$\varepsilon_i = \left(A_i \times \mathbb{FN}_i^{act} + \iota_i \times \mathbb{FN}_i^{idl}\right), \quad \forall i \in \mathbb{FN} \tag{5}$$

The following Equation can calculate the system's overall energy consumption as the sum of the energy used by each $\mathbb{F}$N.

$$\varepsilon^{tot} = \sum_{\forall i \in F} \varepsilon_i \tag{6}$$

3.1.5 Optimization objectives

$Ta = \{Ta_1, Ta_2, \ldots, Ta_n\}$ is a set of tasks concerning a flow received from IoT devices, and there is a set of heterogeneous $\mathbb{F}$N s $\mathbb{F}\mathrm{N} = \{\mathbb{F}\mathrm{N}_1, \mathbb{F}\mathrm{N}_2, \ldots, \mathbb{F}\mathrm{N}_m\}$. The problem involves a mapping of the tasks to the $\mathbb{F}$N s ($\psi : T \rightarrow F$), while energy consumption is optimized and the task execution deadlines are satisfied. Therefore, the aim is to minimize ε^{tot} provided that $R_k \leq Ta_k^d$ for each task k. Scheduling should occur so that each task is performed by one and only one $\mathbb{F}$N, and the algorithm ensures task deadline satisfaction. In the proposed method, we should seek to minimize deadline violation time upon its occurrence, *i.e.,* when the deadline of a task is not observed, considering priority. Thus, tasks with high priorities should preferably not exceed the set deadlines, and the violation should be minimal if unavoidable. Consequently, we add V^{tot} minimization to our optimization problem, where the violations of the set deadlines are considered based on priority. Therefore, the objective function is defined as follows. Where α and β are weights of two objective parameters of energy and deadline. These weights are modified dynamically during simulations.

$$\min(\alpha * V^{tot} + \beta * \varepsilon^{tot}) \tag{7}$$

3.1.6 Response time

The response time for the task Ta_k sent to the device $\mathbb{F}\mathrm{N}_i$ is influenced by factors such as propagation delay, transfer time, execution time, and waiting time in the queue. The propagation delay upon the transfer of task Ta_k to the device $\mathbb{F}\mathrm{N}_i$ is obtained as follows.

$$d_k^{prp} = \sum_{\forall e_{ij} \in L} \left(2 \times e_{ij}^p \times y_{ij}^k\right), \quad \forall k \in Ta_k \tag{8}$$

Transfer time, including the time spent to transfer the input file of size Ta_k^{in}, from FC to $\mathbb{F}$N ($\mathbb{F}\mathrm{N}_i \in \mathbb{F}\mathrm{N}$) and to resend the output file of size Ta_k^{out}, from $\mathbb{F}\mathrm{N}_i$ to FC can be calculated as follows.

$$d_k^{tx} = \sum_{\forall e_{ij} \in L} \frac{Ta_k^{in} + Ta_k^{out}}{e_{ij}^b} \times y_{ij}^k, \quad \forall k \in Ta \tag{9}$$

A major factor affecting the system response time is the CPU execution time. The execution time of Ta_k is specified through the division of task size (Ta_k) by the processing power of the fog device CPU.

$$d_k^{exe} = \sum_{\forall i \in \mathbb{FN}} \frac{Ta_k^s}{\mathbb{FN}_i^c} \times x_{ik}, \quad \forall k \in Ta \tag{10}$$

We consider two limitations of the devices. Firstly, each $\mathbb{FN}$ can process only one task at a time. We also assume that the tasks are non-preemptive. A task is continuously executed until finished once placed on a particular $\mathbb{FN}$. When a task arrives at an $\mathbb{FN}$, therefore, it needs to wait in the $\mathbb{FN}$ queue until the previous tasks allocated to the $\mathbb{FN}$ are completed. On that basis, the waiting time for the task Ta_k is as follows.

$$d_k^{wt} = \sum_{\forall l \in Ta} \sum_{\forall i \in \mathbb{FN}} \left(d_l^{exe} \times z_{kl} \times x_{ik} \times x_{il} \right), \quad \forall k \in Ta \tag{11}$$

Therefore, the response time of the task Ta_k is as follows.

$$R_k = d_k^{prp} + d_k^{tx} + d_k^{exe} + d_k^{wt}, \quad \forall k \in Ta \tag{12}$$

Next, we calculate the number of IoT tasks that meet the relevant deadlines (N_s) and then the total deadline violation time for the set of n tasks sent to FC in a particular period.

$$N_s = \sum_{\forall k \in Ta} s_k, \quad \forall k \in Ta \tag{13}$$

We use $s_k \in \{0, 1\}$ to indicate whether the deadline of the task Ta_k is satisfied.

$$s_k = \begin{cases} 1 \text{ if } R_k < Ta_k^d \\ 0 \text{ otherwise} \end{cases}, \quad \forall l, \ k \in Ta \tag{14}$$

Therefore, the tasks with satisfied deadlines and their percentage can be calculated.

$$S^{\%} = \frac{N_s}{n} \tag{15}$$

$$S^{p\%} = \frac{N_s^p}{n^p} \tag{16}$$

$S^{\%}$ represents the percentage of IoT tasks that satisfy the deadline requirement, and $S^{p\%}$ indicates the percentage of those considering the priorities as well, where N_s^p and N^p are as follows

$$N_s^p = \sum_{\forall k \in Ta} s_k \times Ta_k^p, \quad \forall k \in Ta \tag{17}$$

$$N^p = \sum_{\forall k \in T} Ta_k^p, \tag{18}$$

N_s^p denotes the deadline violation time of the task Ta_k considering the priorities, and N^p measures the quality of task scheduling by calculating deadline violation time considering priority. Finally, the total violation time of the tasks given the priorities can obtain by Eq. (20).

$$v_k = \left(max\left(0, R_k - Ta_k^d\right) \times Ta_k^p\right), \quad \forall k \in Ta \tag{19}$$

$$V^{tot} = \sum_{\forall k \in Ta} v_k \tag{20}$$

4 Proposed method

The proposed method uses a hybrid of the simulated annealing (SA) optimization algorithm and the particle swarm optimization (PSO) algorithm, which are examined below.

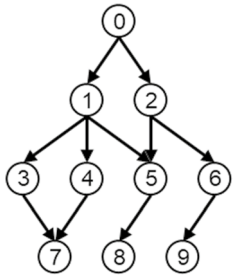
4.1 Improved PSO

Particle swarm optimization is a smart optimization algorithm based on particles' collective intelligence. To obtain the best position of each particle, this algorithm uses its best position in the past and its best global position. Thus, all particles help each other to achieve the best position. It has been found in many problems to be a better solution than heuristic and other smart metaheuristic algorithms, exhibiting better performance. It is a metaheuristic method used in optimization problems with continuous functions[50]. There are three vectors for each particle in the particle swarm optimization algorithm: the current position of the particle (P), particle speed (v), and the best position experienced so far by the particle (P^{best}). Moreover, the best position found by all particles is represented by P^{Gbest}. Movement of a particle combining these vectors helps to explore the problem space further, and consideration of P^{Gbest} causes it to converge over time. The following positions of the particles are specified using the particle velocity equation. There are three important parts to the Equation: current particle velocity ($v(t)$), change in particle velocity and its direction toward the best personal experience $C_1 \times rand \times \left(P^{best}(h,t) - P(h,t)\right)$,

Table 2 A sample particle considered for task scheduling of the DAG (Bolded values are fog nodes selected based on the maximum particle value)

FOG node	TA0	TA1	TA2	Ta3	TA4	TA5	TA6	TA7	TA8	TA9
FN1	**9.131**	−9.310	−4.720	−8.331	−4.730	6.569	−8.316	3.191	**9.356**	**9.628**
FN2	−8.226	−3.874	0.490	−6.046	1.357	−4.242	−1.186	**4.310**	−7.517	5.853
FN3	2.929	−7.522	**1.770**	**7.766**	−6.340	0.517	**9.039**	−3.513	−8.323	9.361
FN4	6.023	**8.975**	−7.315	2.149	**2.537**	**7.229**	6.514	−6.350	6.716	−6.396
FN5	−0.997	−0.124	−3.380	−3.458	−7.391	−7.439	−3.189	0.072	−7.792	2.126

Fig. 2 A sample DAG



and change in particle velocity and its direction toward the best group experience $C_2 \times rand \times \left(P^{Gbest}(t) - P(h,t)\right)$. Parameters C_1 and C_2 specify the importance and weight of collective intelligence, The fact that the above parameters are constant is a very important factor. They stand for the significance of particle velocity change and its direction with respect to the best individual experience and the significance of particle velocity change and its direction concerning the best collective experience, respectively. Therefore, we can calculate importance instead of applying a constant value. Thus, we update the Equation as follows:

$$v(h,t+1) = C_1' \times v(h,t) + C_2' \times \left(P^{best}(h,t) - P(h,t)\right) + C_3' \times \left(P^{Gbest}(t) - P(h,t)\right) \tag{21}$$

where C_1' is the probability that the particle undergoes no change in direction and moves along its own path, C_2' is the probability that the particle undergoes some change in direction and is directed toward the best personal experience, C_3' is the probability that the particle undergoes some change in direction and is directed toward the best group experience. In fact, the C_1', C_2', and C_3' factors are probabilities that are set to the possibility that the optimal particle is located at different positions given the objective function. They indicate the rate of maintaining the current velocity, the rate of movement toward the local optimum, and the rate of movement toward the global optimum. Therefore, these probabilities are expressed by the following equations.

$$C_1' = \frac{\left(\varepsilon^{tot}_{Position} + \varepsilon^{tot}_{Best\ Position} + \varepsilon^{tot}_{Best\ Global\ Position}\right)}{\varepsilon^{tot}_{Position}} \tag{22}$$

$$C_2' = \frac{\left(\varepsilon^{tot}_{Position} + \varepsilon^{tot}_{Best\ Position} + \varepsilon^{tot}_{Best\ Global\ Position}\right)}{\varepsilon^{tot}_{Best\ Position}} \tag{23}$$

$$C_3' = \frac{\left(\varepsilon^{tot}_{Position} + \varepsilon^{tot}_{Best\ Position} + \varepsilon^{tot}_{Best\ Global\ Position}\right)}{\varepsilon^{tot}_{Best\ Global\ Position}} \tag{24}$$

Table 3 Matrix x for a particle

FOG NODE	TA0	TA1	TA2	TA3	TA4	TA5	TA6	TA7	TA8	TA9
FN1	1	0	0	0	0	0	0	0	1	1
FN2	0	0	0	0	0	0	0	1	0	0
FN3	0	0	1	1	0	0	1	0	0	0
FN4	0	1	0	0	1	1	0	0	0	0
FN5	0	0	0	0	0	0	0	0	0	0

Given the above probabilities, particles move toward more probable positions, *i.e.,* they involve better objective functions, thereby facilitating the achievement of optimality for the conditions set by constant factors.

4.2 Simulated annealing algorithm

Simulated annealing (SA) is a simple, efficient heuristic optimization algorithm for solving optimization problems in large search spaces. It is preferable in problems where the search space is discrete. Obtaining an approximate solution for the global optimum is more important than an exact solution for the local optimum within a limited period [40]. This algorithm has been developed to enable escape from local optima. It may initially prefer fewer proper solutions to better ones but converges to global optima over time by avoiding local ones.

4.3 Description of the proposed method

In the proposed method, each particle is considered as a matrix for task-resource assignment. The matrix consists of m rows (equal to virtual machines) and n columns (equal to Tasks). Each task is assigned to the VM with the highest pi value for that column. $\mathbb{F}\mathrm{N}_i$ is allocated to the allowed task T_a via the following Equation.

$$x_{i,Ta} = \begin{cases} 1 \text{ if } i = \underset{j \in FN}{argmax}\, P_{j,Ta} \\ 0 \text{ otherwise} \end{cases} \quad \forall i \in \mathbb{F}\mathrm{N} \tag{25}$$

T_a, the selected allowed task is obtained via the following Equation:

$$T_a = \underset{k \in TaAllowed}{argmax} \left(\underset{j \in FN}{max} P_{j,k} \right) \tag{26}$$

where $TaAllowed = \left\{ k \in Ta | Ta_k^f = 0 \; or \; \sum_{i \in \mathbb{F}\mathrm{N}} x_{i,Ta_k^f} = 1 \right\}$. That is, a fog resource can be allocated to a task while observing the workflow hierarchy only if it is located at the root or its parent has undergone resource allocation. Once a task is performed,

all the tasks that depend on it are located in *TaAllowed*, and those that are not or have other parents besides this one (multi-parent tasks) wait for the other parents to be performed. For instance, let us consider the DAG in Fig. 2, which will be implemented on a network with five fog nodes. If we randomly generate a particle with components lying within the range from -10 to $+10$, it can assume the following values which is shown in Table 2.

The proposed algorithm first places Ta0, the only parentless node, on the list of allowed tasks and allocates it to FN1, for which the particle value is maximum in the column for this task. Therefore, $x_{1,0} = 1$ and $TaAllowed = \{1, 2\}$. Next, the maximum values in the Ta1 and Ta2 columns are calculated, and the node with the highest value is assigned to the relevant task. Thus, Task 1 is selected, for which FN4 is considered. Therefore, $x_{4,1} = 1$ and $TaAllowed = \{2, 3, 4\}$. The procedure continues until no more task remains. Consequently, the matrix x is displayed in Table 3.

Dijkstra's algorithm forms the matrix y, and the matrix z is completed in order of assignment. That is, the task to which FN is allocated is prioritized over those that are still waiting for their turns ($z_{ij} = 1$, if i is allocated to a node, and j is still in the waiting queue). Therefore, the matrix z is indicated in Table 4.

According to matrix z, the tasks are performed in this order: 0, 1, 3, 4, 7, 2, 6, 9, 5, and 8.

After the implementation of the PSO algorithm, it is time to implement the SA algorithm. At each step of the PSO, if the global optimum changes, for minimizing V_{tot}, which also leads to the improvement of the termination time, the steps of the SA algorithm are performed as follows:

1. The candidate solution (sol') is created based on the current location of P^{Gbest}.
2. If V^{tot} is less than the candidate solution and there is no increase in ε^{tot}, this solution replaces P^{Gbest} and if V^{tot} is less than V^{tot} with a probability, it will replace P^{Gbest}.
3. Return to the second step if the internal stop condition is not reached.
4. Decrease in temperature.
5. Return to the second step if the external stop condition is not reached.

Table 4 Matrix z for a particle

	0	1	2	3	4	5	6	7	8	9
0	0	1	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	1	1	1
2	0	0	0	0	0	1	1	0	1	1
3	0	0	1	0	1	1	1	1	1	1
4	0	0	1	0	0	1	1	1	1	1
5	0	0	0	0	0	0	0	0	1	0
6	0	0	0	0	0	1	0	0	1	1
7	0	0	1	0	0	1	1	0	1	1
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	1	0	0	1	0

Here Pseudocode for proposed PSO–SA algorithm is explained.

Algorithm 1 PSO-SA Algorithm:

1. Initialize popSize, T, FN, L, Ta, stopCriteriaSA, stopCriteriaPSO
2. Generate the population randomly
3. Obtain x, y and z for each particle using formulas (25) and (26) and Dijkstra algorithm
4. Calculate ε^{tot} for each particle using formula (6)
5. Detemine the P^{best} and P^{Gbest}
6. If P^{Gbest} was updated then sol= P^{Gbest} and:
    6.1 Generate sol'
    6.2 Calculate ε^{tot} and V^{tot} for sol' using formulas (6) and (19)
    6.3 If (V^{tot} for sol'< V^{tot} for sol) and (ε^{tot} for sol'<= ε^{tot} for sol)
        P^{Gbest}= sol'
    6.4 Else if p>rand:
        P^{Gbest}= sol'
    6.5 Reduce T
    6.6 If not(stopCriteriaSA)
        Return to 6.1
7. Calculte the velocity of particles using formula (21)
8. Calculate the position of particles based on original PSO
9. If not(stopCriteriaPSO)
    Return to 3
10. Calculate evaluation criteria for P^{Gbest} as the final solution

5 Performance evaluation

To demonstrate the effectiveness of the proposed algorithm, we compared it to FCFS, EDF, GFE, Detour, PSG [24], and PSG-M [24]. It should be noted that these algorithms are slightly modified concerning their initial conditions to be capable of implementing workflow tasks. In other words, scheduling is applied only to tasks first in the execution queue or whose parents had been executed.

For experimentation, as in [20], we took into account a fog environment made up of numerous heterogeneous, interconnected FNs with unpredictable network topology and a variety of tasks imported from IoT devices to schedule the Fs. The number of Fs varies from 30 to 90 for various trials, and the number of IoT jobs varies from

100 to 500. The CPU processing capacity of each is uniformly spread from 2000 to 6000 MIPS, and its active mode power consumption is expected to range between 80 and 200 watts arbitrarily to maintain the heterogeneity of the Fs. While the links' capacity is set to 1000 megabits per second, the propagation latency between the Fs is anticipated to range from 1 to 3 ms. It is believed that two different IoT task kinds assume a strong relationship between their sizes and necessary timeframes [51]. For one type, the task size is randomly set from 100 to 372 MI, while the task deadline is assumed to range uniformly from 100 to 500 ms. For the other type, the values range from 1028 to 4280 MI and 500 to 2500 ms, respectively. For both types, input and output file sizes are randomly set to range from 100 to 10,000 KB and 1 to 1000 KB, respectively. Task priority is assumed to be a random value between zero and one. For the proposed algorithm, we need to set the number of PSO iterations, number of particles, number of internal SA iterations, temperature reduction factor, and initial temperature by conducting extensive experimentation. The values bringing about proper results include 1000, 200, 15, 0.99, and 0.025, respectively. All the simulations are coded in MATLAB R2022a. The experiments are conducted on a computer with a dual Intel Xeon X5650 2.66 GHz processor, 64 gigabytes of RAM, and the Windows 10 operating system. To present the results with high reliability, each experiment is replicated twenty times, and the mean values are reported. It should be noted that the values of the tunable parameters of PSG and PSG-M are adopted from [24] to enable examination of the performance of the algorithms. For evaluation, therefore, we use criteria such as energy consumption (ε^{tot}), total deadline violation time given the V^{tot} priority, makespan, and the percentage of IoT tasks satisfying their deadline requirements given the priority ($S^{p\%}$).

5.1 Impact of an increase in the number of tasks

Figure 3 shows how the number of tasks affects algorithm performance. Here, we assume the number of $\mathbb{F}$Ns to be constantly 60. As can be observed in Fig. 3a, the simulation results demonstrate that an increase in the number of tasks imposes a heavier load on the system in general. Therefore, a larger number of tasks miss their deadlines, which raises task deadline violation time. As a result, the energy consumption and makespan of the system also increase. Of course, the proposed algorithm observes deadlines more properly than the other strategies. This is more evident when task priority is considered, as the proposed algorithm involves consideration. Figure 3b shows the percentage of tasks that observe deadlines, while Fig. 3c shows the number of tasks with violated deadlines. As observed, the criterion assumes smaller values in the proposed algorithm than in the competitors, even smaller than one-tenth of the total number of tasks. Another interesting observation concerning our algorithm is that it significantly improves makespan over the other methods. This is shown in Fig. 3d. The proposed algorithm seeks to obtain $\mathbb{F}$N so as to present minimum violation time for a particular task. Figure 3e shows

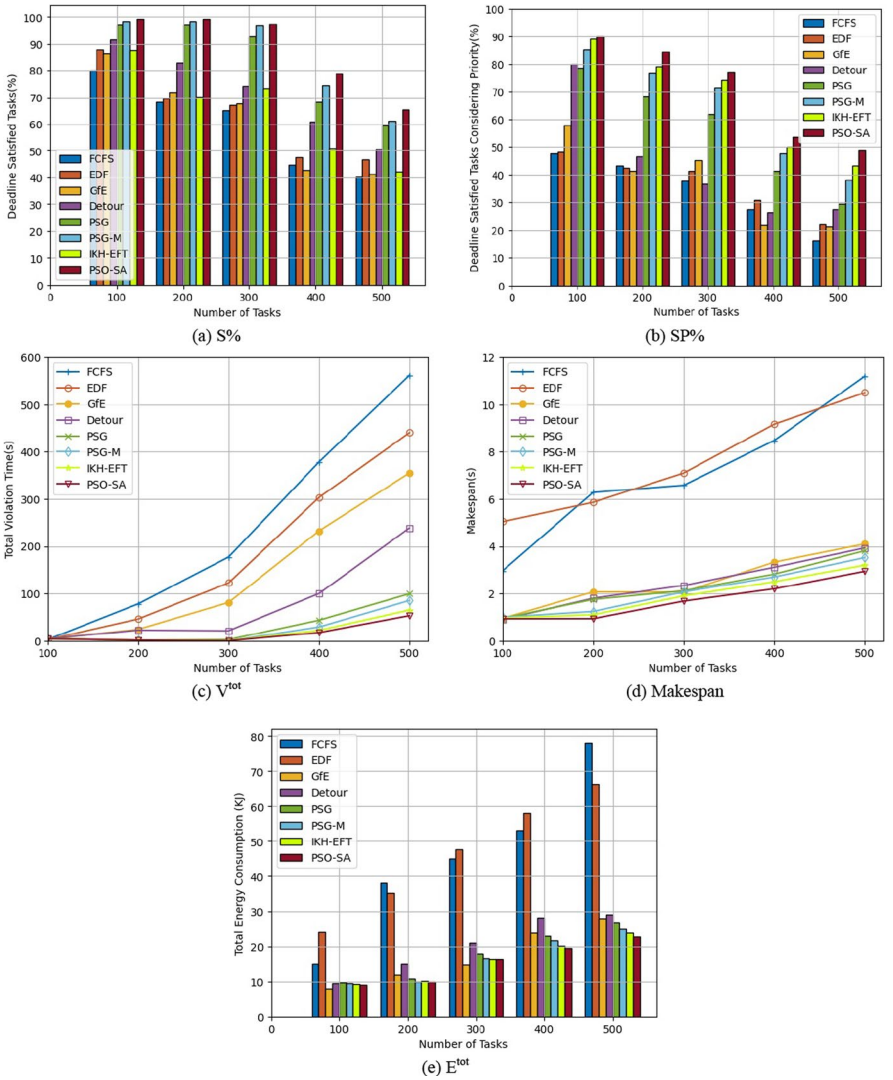


Fig. 3 Comparison of the performance of the algorithms with a varying number of tasks; simulation results for 60 $\mathbb{F}$Ns: **a** deadline satisfaction, **b** deadline satisfaction percentage with priority considerations, **c** task deadline violation, **d** makespan, **e** total energy consumption

the system energy consumption. It should be noted that the makespan of a system directly affects energy consumption. Therefore, makespan is expected to be minimized once energy is.

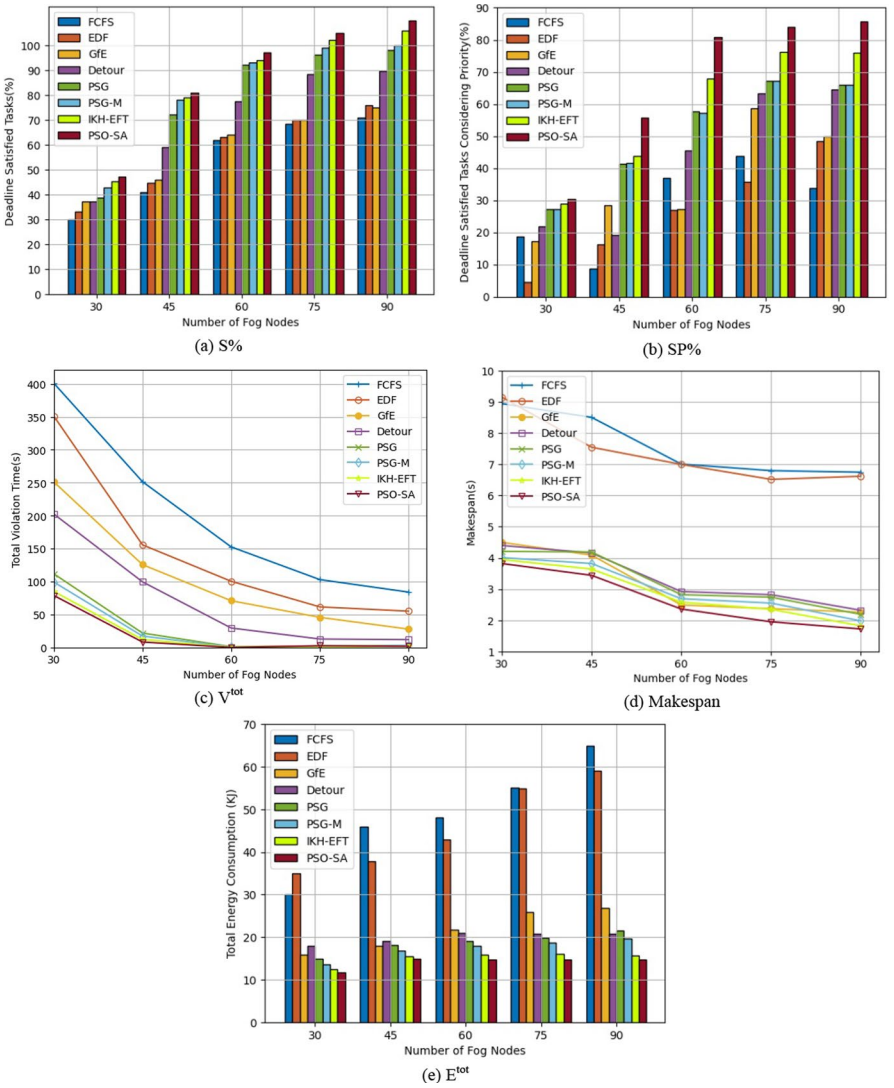


Fig. 4 Comparison of the performance of the algorithms with a varying number of fog nodes; simulation results for 300 tasks: **a** deadline satisfaction, **b** deadline satisfaction percentage with priority considerations, **c** task deadline violation, **d** makespan, **e** total energy consumption

5.2 Impact of an increase in the number of fog nodes

This experiment aims to discover how the number of FNs affects the efficiency of the compared algorithms. In this case, there are 300 tasks in total, which is a constant value. Figure 4 shows the simulation results. Part *c* of the figure indicates that more tasks are executed before their deadlines as the number of $\mathbb{F}$N s increases, which

Table 5 Average percentage optimization of %S for different number of tasks

Tasks	100	200	300	400	500	Percentage of optimization S%				
FCFS	80	68.23	64.95	44.89	40.21	24.09	45.24	49.48	76.01	62.37
EDF	87.58	69.5	66.93	47.71	46.8	13.35	42.59	45.06	65.60	39.51
Gfe	86.31	71.75	67.49	42.77	41.33	15.02	38.12	43.86	84.73	57.97
Detour	91.52	82.89	73.97	60.82	50.63	8.47	19.56	31.26	29.91	28.96
PSG	96.88	96.85	92.72	68.15	59.52	2.47	2.32	4.71	15.94	9.69
PSG-M	98.43	98.25	96.53	74.35	61.21	0.85	0.87	0.58	6.27	6.67
IKH-EFT	87.3	70.2	73.2	51	42.1	13.71	41.17	32.64	54.92	55.08
PSO–SA	99.27	99.1	97.09	79.01	65.29					
					Average	18.90	43.21	41.06	65.46	58.73

Table 6 Average percentage optimization of %SP for different number of tasks

Tasks	100	200	300	400	500	Percentage of optimization SP%				
FCFS	47.7	43.2	37.7	27.6	16.2	88.89	95.83	104.24	94.57	201.85
EDF	48.4	42.5	41.3	30.9	22	86.16	99.06	86.44	73.79	122.27
Gfe	57.8	41.2	45.4	21.9	21.2	55.88	105.34	69.60	145.21	130.66
Detour	80	46.7	36.7	26.5	27.6	12.63	81.16	109.81	102.64	77.17
PSG	78.7	68.5	61.9	41.3	29.3	14.49	23.50	24.39	30.02	66.89
PSG-M	85.5	76.7	71.5	47.8	38.2	5.38	10.30	7.69	12.34	28.01
IKH-EFT	89.3	79.2	74.2	50	43.1	0.90	6.82	3.77	7.40	13.46
PSO–SA	90.1	84.6	77	53.7	48.9					
					Average	37.76	60.29	57.99	66.57	91.47

is expected as addition of more $\mathbb{F}$Ns to the system raises the number of available resources. Given task prioritizing and resource awareness, the suggested algorithm has a significantly better percentage of tasks meeting their deadline criteria than the alternatives. Specifically, the algorithm satisfies the deadlines of all the tasks when the number of $\mathbb{F}$Ns is 60. Figure 4b shows this in the case where priority is considered, complicating the problem further. Our algorithm exhibits considerable performance here compared to the others since it focuses on task deadlines in the second step of fitness improvement. In part *c* of Fig. 4, it can be observed that total violation time considerably decreases for all the algorithms as the number of $\mathbb{F}$Ns increases. It is again clear that the proposed algorithm achieves the minimum number of deadline violations faster than the compared ones. Firgure 4d and Fig. 4e show the simulation results regarding energy consumption and makespan, where the proposed algorithm is far better than the others.

Experimental results for different number of tasks are listed in Tables 5, 6, 7, 8, 9.

Table 7 Average percentage optimization of $\%V_{tot}$ for different number of tasks

Tasks	100	200	300	400	500	Percentage of optimization V_{tot}				
FCFS	6.1	78.06	177.2	378.48	562.02	71.31	97.57	99.55	95.43	90.54
EDF	5.8	45.98	122.4	303.37	440.5	69.83	95.87	99.35	94.30	87.93
Gfe	4.7	24.05	81.02	232.49	355.27	62.77	92.10	99.01	92.56	85.04
Detour	4.1	22.36	21.09	100	238.82	57.32	91.50	96.21	82.70	77.74
PSG	3.2	4.5	2.52	43.46	99.57	45.31	57.78	68.25	60.19	46.61
PSG-M	2.9	3.1	1.75	29.11	85.25	39.66	38.71	54.29	40.57	37.64
IKH-EFT	2.5	2.5	1.28	22	65	30.00	24.00	37.50	21.36	18.22
PSO–SA	1.75	1.9	0.8	17.3	53.16					
					Average	53.74	71.07	79.16	69.59	63.39

Table 8 Average percentage optimization of Makespan for different number of tasks

Tasks	100	200	300	400	500	Percentage of optimization Makespan				
FCFS	2.96	6.28	6.58	8.48	11.2	68.55	85.11	74.47	74.06	73.75
EDF	5.04	5.86	7.1	9.17	10.5	81.53	84.04	76.34	76.01	72.00
Gfe	0.948	2.07	2.06	3.33	4.1	1.79	54.83	18.45	33.93	28.29
Detour	0.897	1.8	2.34	3.11	3.93	− 3.79	48.06	28.21	29.26	25.19
PSG	0.948	1.75	2.12	2.82	3.81	1.79	46.57	20.75	21.99	22.83
PSG-M	0.999	1.24	2.07	2.7	3.52	6.81	24.60	18.84	18.52	16.48
IKH-EFT	0.965	1.1	1.9	2.5	3.2	3.52	15.00	11.58	12.00	8.13
PSO–SA	0.931	0.935	1.68	2.2	2.94					
					Average	22.89	51.17	35.52	37.97	35.24

Table 9 Average percentage optimization of energy for different number of tasks

Tasks	100	200	300	400	500	Percentage of optimization E_{tot}				
FCFS	15	38	45	53.1	78.1	39.80	73.68	63.78	63.47	70.81
EDF	24.1	35.2	47.7	58	66.2	62.53	71.59	65.83	66.55	65.56
Gfe	7.91	12	14.9	23.8	27.9	− 14.16	16.67	− 9.40	18.49	18.28
Detour	9.38	15	21.1	28.1	29.1	3.73	33.33	22.75	30.96	21.65
PSG	9.71	10.8	17.8	23	26.7	7.00	7.41	8.43	15.65	14.61
PSG-M	9.37	10	16.6	21.7	24.9	3.63	0.00	1.81	10.60	8.43
IKH-EFT	9.25	10.1	16.4	20.2	23.8	2.38	0.99	0.61	3.96	4.20
PSO–SA	9.03	10	16.3	19.4	22.8					
					Average	14.99	29.10	21.97	29.95	29.08

6 Conclusion

Fog computing has facilitated the relations between IoT and cloud centers and increased data transfer speed between the two parts. Another achievement is that it can process IoT requests if required. This study focused on task scheduling in the fog computing environment to reduce service delay time and energy consumption while observing task deadlines and priorities and maintaining workflow for IoT devices. We examined IoT task scheduling over a heterogeneous fog network. For this purpose, we first discussed the notions and problem formulation in the workflow mode, which had not been considered before at the same time as maintenance of deadline and priority. We presented an architecture to address these issues. Given the multi-objective nature of the problem, we used an algorithm based on PSO and SA to identify the most appropriate solution among the possible ones. The algorithm reduces energy consumption and task finish time in two steps. We compared the proposed algorithm to the others in modes with varying numbers of fog nodes and tasks to evaluate its performance. The simulation results demonstrated that the proposed algorithm performs more successfully in all the evaluations. It avoids entrapment in local optima by generating neighboring solutions. Although the key idea in this paper was to optimize the system's total energy consumption while observing task deadlines, workflow and priority were also considered in the algorithm, making it far more complex.

Author contributions NK: Conceptualization, methodology, writing—original draft preparation. KK: Supervising, Data curation, writing—reviewing and editing, validation. RA: Software, visualization, investigation. LA: Writing—reviewing and editing, validation. DJ: Writing—reviewing and editing.

Funding No funding was obtained this study.

Data availability The dataset used and analyzed during the current study is available from the corresponding author upon reasonable request.

Declarations

Conflict of 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.

References

1. Nazari A, Kordabadi M, Mohammadi R, Lal C (2023) EQRSRL: an energy-aware and QoS-based routing schema using reinforcement learning in IoMT. Wireless Netw 24:1–15
2. Mohammadi R, Nazari A, Daneshmand B (2023) An efficient routing schema for internet of underwater things/ocean of things. In: 2023 Wave electronics and its application in information and telecommunication systems (WECONF), pp. 1–8. IEEE

3. Nazari A, Tavassolian F, Abbasi M, Mohammadi R, Yaryab P (2022) An intelligent sdn-based clustering approach for optimizing iot power consumption in smart homes. Wireless Commun Mobile Comput. https://doi.org/10.1155/2022/8783380
4. Samadi R, Nazari A, Seitz J (2023) Intelligent energy-aware routing protocol in mobile IoT networks based on SDN. IEEE Trans Green Commun Network. https://doi.org/10.1109/TGCN.2023.3296272
5. Cisco U (2020) Cisco annual internet report (2018–2023) white paper. Cisco: San Jose, CA, USA. 10(1):1–35
6. Goudarzi M, Wu H, Palaniswami M, Buyya R (2020) An application placement technique for concurrent IoT applications in edge and fog computing environments. IEEE Trans Mob Comput 20(4):1298–1311
7. Nazari A, Mohammadi R, Niknami N, Jazaeri SS, Wu J (2023) The fuzzy-IAVOA energy-aware routing algorithm for SDN-based IoT networks. Int J Sensor Netw 42(3):156–169
8. Qiu H, Zhu K, Luong NC, Yi C, Niyato D, Kim DI (2022) Applications of auction and mechanism design in edge computing: a survey. IEEE Trans Cognit Commun Netw 8(2):1034–1058
9. Sadri AA, Rahmani AM, Saberikamarposhti M, Hosseinzadeh M (2022) Data reduction in fog computing and internet of things: a systematic literature survey. Internet of Things 13:100629
10. Kumari N, Yadav A, Jana PK (2022) Task offloading in fog computing: a survey of algorithms and optimization techniques. Comput Netw 214:109137
11. Bansal S, Aggarwal H, Aggarwal M (2022) A systematic review of task scheduling approaches in fog computing. Trans Emerg Telecommun Technol 33(9):e4523
12. Nayak SC, Parida S, Tripathy C, Pattnaik PK (2022) An enhanced deadline constraint based task scheduling mechanism for cloud environment. J King Saud Univ Comput Inf Sci 34(2):282–294
13. Zhou G, Tian W, Buyya R (2023) Multi-search-routes-based methods for minimizing makespan of homogeneous and heterogeneous resources in Cloud computing. Future Gener Comput Syst 141:414–432
14. Versluis L, Iosup A (2021) A survey of domains in workflow scheduling in computing infrastructures: community and keyword analysis, emerging trends, and taxonomies. Future Gener Comput Syst 123:156–177
15. Chen G, Qi J, Sun Y, Hu X, Dong Z, Sun Y (2023) A collaborative scheduling method for cloud computing heterogeneous workflows based on deep reinforcement learning. Future Gener Comput Syst 141:284–297
16. Ghafari R, Kabutarkhani FH, Mansouri N (2022) Task scheduling algorithms for energy optimization in cloud environment: a comprehensive review. Cluster Comput 25:1035
17. Ijaz S, Munir EU, Ahmad SG, Rafique MM, Rana OF (2021) Energy-makespan optimization of workflow scheduling in fog–cloud computing. Computing 103(9):2033–2059
18. Ajmal MS, Iqbal Z, Khan FZ, Bilal M, Mehmood RM (2021) Cost-based energy efficient scheduling technique for dynamic voltage and frequency scaling system in cloud computing. Sustain Energy Technol Assess 45:101210
19. Xu M, Buyya R (2020) Managing renewable energy and carbon footprint in multi-cloud computing environments. J Parallel Distrib Comput 135:191–202
20. Dayarathna M, Wen Y, Fan R (2015) Data center energy consumption modeling: a survey. IEEE Commun Surv Tutor 18(1):732–794
21. Hussain M, Wei L-F, Rehman A, Abbas F, Hussain A, Ali M (2022) Deadline-constrained energy-aware workflow scheduling in geographically distributed cloud data centers. Future Gener Comput Syst 132:211–222
22. Li H, Xu G, Wang D, Zhou M, Yuan Y, Alabdulwahab A (2022) Chaotic-nondominated-sorting owl search algorithm for energy-aware multi-workflow scheduling in hybrid clouds. IEEE Trans Sustain Comput 7:595
23. Saurav SK, Benedict S (2021) A taxonomy and survey on energy-aware scientific workflows scheduling in large-scale heterogeneous architecture. In: 2021 6th international conference on inventive computation technologies (ICICT), 2021: IEEE, pp. 820–826

24. Azizi S, Shojafar M, Abawajy J, Buyya R (2022) Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: a semi-greedy approach. J Netw Comput Appl 201:103333
25. Kishor A, Chakarbarty C (2022) Task offloading in fog computing for using smart ant colony optimization. Wireless Pers Commun 127(2):1683–1704
26. Abd Elaziz M, Abualigah L, Attiya I (2021) Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments. Future Gener Comput Syst 124:142–154
27. Abd Elaziz M, Abualigah L, Ibrahim RA, Attiya I (2021) IoT workflow scheduling using intelligent arithmetic optimization algorithm in fog computing. Comput Intell Neurosci. https://doi.org/10.1155/2021/9114113
28. Sellami B, Hakiri A, Yahia SB, Berthou P (2022) Energy-aware task scheduling and offloading using deep reinforcement learning in SDN-enabled IoT network. Comput Netw 210:108957
29. Jayanetti A, Halgamuge S, Buyya R (2022) Deep reinforcement learning for energy and time optimized scheduling of precedence-constrained tasks in edge–cloud computing environments. Future Gener Comput Syst 137:14–30
30. Tuli S, Poojara SR, Srirama SN, Casale G, Jennings NR (2021) COSCO: Container orchestration using co-simulation and gradient based optimization for fog computing environments. IEEE Trans Parallel Distrib Syst 33(1):101–116
31. Javaheri D, Gorgin S, Lee J-A, Masdari M (2022) An improved discrete harris hawk optimization algorithm for efficient workflow scheduling in multi-fog computing. Sustain Comput Inform Syst 36:100787
32. Ghobaei-Arani M, Shahidinejad A (2022) A cost-efficient IoT service placement approach using whale optimization algorithm in fog computing environment. Expert Syst Appl 200:117012
33. Ramzanpoor Y, Hosseini Shirvani M, Golsorkhtabaramiri M (2022) Multi-objective fault-tolerant optimization algorithm for deployment of IoT applications on fog computing infrastructure. Complex Intell Syst 8(1):361–392
34. Al-Araji ZJ, Ahmad SSS, Kausar N, Farhani A, Ozbilge E, Cagin T (2022) Fuzzy theory in fog computing: review, taxonomy, and open issues. IEEE Access 10:126931–126956. https://doi.org/10.1109/ACCESS.2022.3225462
35. Varmaghani A, Matin Nazar A, Ahmadi M, Sharifi A, Jafarzadeh Ghoushchi S, Pourasad Y (2021) DMTC: optimize energy consumption in dynamic wireless sensor network based on fog computing and fuzzy multiple attribute decision-making. Wireless Commun Mobile Comput. https://doi.org/10.1155/2021/9953416
36. Taghizadeh J, Ghobaei-Arani M, Shahidinejad A (2021) An efficient data replica placement mechanism using biogeography-based optimization technique in the fog computing environment. J Ambient Intell Humaniz Comput 14:3691
37. Iftikhar S et al (2023) HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments. Internet of Things 21:100667
38. Ahmed OH, Lu J, Xu Q, Ahmed AM, Rahmani AM, Hosseinzadeh M (2021) Using differential evolution and Moth-Flame optimization for scientific workflow scheduling in fog computing. Appl Soft Comput 112:107744
39. Kaur M, Aron R (2022) An energy-efficient load balancing approach for scientific workflows in fog computing. Wireless Person Commun 125:3549
40. Hosseini Shirvani M, Noorian Talouki R (2022) Bi-objective scheduling algorithm for scientific workflows on cloud computing platform with makespan and monetary cost minimization approach. Complex Intell Syst 8(2):1085–1114
41. Mokni M, Yassa S, Hajlaoui JE, Chelouah R, Omri MN (2022) Cooperative agents-based approach for workflow scheduling on fog-cloud computing. J Ambient Intell Humaniz Comput 13(10):4719–4738
42. Han P, Du C, Chen J, Ling F, Du X (2021) Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique. J Syst Archit 112:101837
43. Khaledian N, Khamforoosh K, Azizi S, Maihami V (2023) IKH-EFT: an improved method of workflow scheduling using the krill herd algorithm in the fog-cloud environment. Sustain Comput Inform Syst 37:100834
44. Delavar AG, Akraminejad R, Mozafari S (2022) HDECO: a method for Decreasing energy and cost by using virtual machine migration by considering hybrid parameters. Comput Commun 195:49–60
45. Idrees AK, Al-Yaseen WL (2021) Distributed genetic algorithm for lifetime coverage optimisation in wireless sensor networks. Int J Adv Intell Paradig 18(1):3–24

46. Hazra A, Rana P, Adhikari M, Amgoth T (2023) Fog computing for next-generation internet of things: fundamental, state-of-the-art and research challenges. Comput Sci Rev 48:100549
47. Laroui M, Nour B, Moungla H, Cherif MA, Afifi H, Guizani M (2021) Edge and fog computing for IoT: a survey on current research activities & future directions. Comput Commun 180:210–231
48. Guevara JC, da Fonseca NL (2021) Task scheduling in cloud-fog computing systems. Peer-to-Peer Netw Appl 14(2):962–977
49. Peng L, Dhaini AR, Ho P-H (2018) Toward integrated cloud-fog networks for efficient IoT provisioning: key challenges and solutions. Future Gener Comput Syst 88:606–613
50. Nabi S, Ahmed M (2022) PSO-RDAL: particle swarm optimization-based resource-and deadline-aware dynamic load balancer for deadline constrained cloud tasks. J Supercomput 78:4624
51. Auluck N, Azim A, Fizza K (2019) Improving the schedulability of real-time tasks using fog computing. IEEE Trans Serv Comput 15:372

Authors and Affiliations

Navid Khaledian[1,5] · Keyhan Khamforoosh[1] · Reza Akraminejad[2] · Laith Abualigah[3] · Danial Javaheri[4]

✉ Navid Khaledian
navid.khaledian@uni.lu

Keyhan Khamforoosh
k.khamforoosh@iausdj.ac.ir

Reza Akraminejad
r.akraminejad@gmail.com

Laith Abualigah
aligah.2020@gmail.com

Danial Javaheri
javaheri@korea.ac.kr

[1] Department of Computer Engineering, Islamic Azad University, Sanandaj Branch, Sanandaj, Iran

[2] Department of Computer Engineering and Information Technology, Payame Noor University, Tehran, Iran

[3] Hourani Center for Applied Scientific Research, Al-Ahliyya Amman University, Amman 19328, Jordan

[4] Department of Computer Science and Engineering, Korea University, Seoul 02841, Republic of Korea

[5] Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Esch-sur-Alzette, Luxembourg