Multimedia Tools and Applications
https://doi.org/10.1007/s11042-024-19509-w

Deadline-aware and energy efficient IoT task scheduling using fuzzy logic in fog computing

Rahul Thakur[1] · Geeta Sikka[2] · Urvashi Bansal[1] · Jayant Giri[3,5] · Saurav Mallik[4]

Received: 28 September 2023 / Revised: 16 February 2024 / Accepted: 26 May 2024

Abstract
Fog computing is an emerging paradigm that extends cloud computing (CC) by providing computation, communication, and storage services at the edge of a network, closer to end devices. It has gained significance due to the rapid development of IoT devices, which generate various types of tasks. Processing these tasks in the cloud can strain its infrastructure and lead to delays in time-sensitive requests. To address this limitation, fog computing (FC) concepts were introduced in 2012 by Cisco. FC is not meant to replace CC but rather to complement and extend its capabilities. One of the challenges in FC is efficiently assigning tasks to appropriate resources to minimize makespan, energy consumption (EC), and increase the number of deadline-satisfied tasks. In this work, the improvement of semi-greedy algorithm has been done by incorporating fuzzy logic (FL). By leveraging FL, the aim is to enhance the algorithm's decision-making process and make it more adaptive to varying conditions and uncertainties in the fog environment. The use of FL allows more nuanced and flexible task scheduling (TS) decisions based on fuzzy sets and fuzzy rules. The simulation experiments demonstrate that the proposed algorithm outperforms PSG (Priority-aware Semi-Greedy) and PSG-M (PSG with multistart), which were identified as the best scheduling algorithms (Algos) in the literature review. The algorithm exhibits better performance in terms of reducing makespan, EC, and increasing the percentage of deadline-satisfied tasks compared to PSG and PSG-M. The inclusion of FL further enhances the algorithm's effectiveness in handling complex scheduling scenarios in a FC environment. To evaluate the performance of the proposed algorithm, different simulation experiments have been conducted using a selected simulator after a systematic review of existing simulators. The experiments involved 300 and 500 random and static tasks, as well as 60 fog nodes in the fog environment. All simulations were implemented in C++ programming language using the Visual Studio IDE. To make sure the results were very reliable, we did each experiment 30 times and then shared the average outcomes. Comparisons across dynamic and static task scenarios consistently favor FuzzyPSG-M and PSG-M, with alpha set to 0.44. These algorithms outperform others in terms of satisfied deadlines, EC, penalties, and makespan. FuzzyPSG-M exhibits a slight edge over PSG-M, attributed to its multistart procedure.

Keywords Cloud computing · Internet of things (IoT) · Fog computing · Fuzzy logic · Task scheduling

Extended author information available on the last page of the article

1 Introduction

The widespread adoption of Internet of Things (IoT) devices has led to the emergence of various IoT applications such as healthcare, energy management, agriculture, and transportation, which demand quick responses and high-quality service. However, IoT devices often have limited processing capabilities [1]. To tackle this challenge, Fog Computing has gained attention as it extends Cloud Computing to the network edge, bringing computing resources closer to IoT devices for faster processing [1–3].While FC is a promising solution for IoT applications, it faces challenges in efficiently allocating and managing fog resources due to their limited capacity, dynamic nature, diversity, and distribution. Energy conservation is also a critical concern in FC [1, 4–7]. Effectively provisioning and managing these resources are essential to maximize the potential of FC for IoT applications. However, the question of how to efficiently assign these dynamic and diverse fog resources to IoT tasks, ensuring quality of service while minimizing EC by fog nodes (FNs), remains a fundamental challenge [1, 8, 9].

The algorithm proposed in this study addresses specific challenges within the Fog Computing environment. It tackles the limited processing capabilities of IoT devices, the dynamic and diverse nature of fog resources, their distribution, and the critical aspect of energy conservation within FC. Specifically, the algorithm aims to efficiently allocate and manage fog resources, ensuring optimal performance for IoT applications while mitigating energy consumption by fog nodes (FNs).

Numerous strategies have been proposed for effectively managing the scheduling of IoT tasks within fog networks, which are known for their diverse characteristics and limited resources. These approaches make use of various optimization techniques, as evidenced in studies by [10–13]. Many of these research efforts utilize metaheuristic algorithms [1–3, 5, 14, 15], as well as machine learning methods [16–18] to address the challenge of scheduling IoT tasks. However, these methods may have longer execution times, which can be problematic when dealing with IoT tasks having strict and unyielding deadlines. Additionally, many previous studies primarily focus on minimizing either response times [3, 14, 19–27] or the EC of Fog Nodes (FNs) [2, 5, 6, 28–30] However, concentrating exclusively on one aspect of scheduling does not guarantee the simultaneous achievement of both high-quality service for IoT users and minimal energy costs for fog service providers. To address these challenges and provide a more granular understanding of the research question, we break it down into two sub-questions:

- How can we design an algorithm with low time complexity for scheduling IoT tasks within heterogeneous and resource-limited FC environments?
- How can we simultaneously optimize the EC of FNs and meet the DL requirements of IoT tasks through the proposed algorithm?

To address the earlier research question, we present two effective scheduling algorithms for IoT tasks that consider both task deadlines and the EC of Fog Nodes (FNs). Initially, we define the TS problem as a Mixed Integer Nonlinear Programming (MINLP) challenge. Our primary goal is to minimize energy usage while ensuring that IoT task deadlines are met. We also aim to reduce the instances where deadlines are exceeded. To enhance the efficiency of tackling this problem, we have improved existing algorithms, specifically PSG and PSG-M [31], by incorporating fuzzy logic techniques.

These modified algorithms are denoted as FuzzyPSG and FuzzyPSG-M. We conducted comprehensive simulations to evaluate the effectiveness of these proposed algorithms.

Semi-greedy methods offer a significant benefit over traditional greedy heuristic algorithms by introducing an element of randomness, which enables them to steer clear of local optima. As a result, these approaches enhance the quality of outcomes [32].

We can summarize our primary contributions as follows:

- We introduce a MINLP framework designed for scheduling IoT tasks within diverse fog networks. Our objective is to efficiently manage the allocation of tasks, focusing on minimizing the overall EC of the Fog Nodes (FNs), all while ensuring that task deadlines are met. Additionally, our model incorporates the goal of reducing the extent to which deadlines are exceeded.
- We have adapted and enhanced two semi-greedy algorithms, named FuzzyPSG and FuzzyPSG-M, to effectively allocate IoT tasks to the available Fog Nodes (FNs). This allocation aims to ensure a superior QoS for IoT users, focusing on minimizing response times, while also reducing the EC of the FNs.
- We carried out comprehensive experiments to assess the effectiveness of our suggested methods. The outcomes reveal that by employing our proposed algorithms, a significant majority of IoT tasks meet their DL requirements, and even those that don't, experience only a slight delay in receiving a response. At the same time, the overall EC and time it takes to complete tasks in the system are decreased, leading to the maximization of profits for fog service providers.

The remainder of this paper follows a structured organization. Section 2 conducts a thorough examination of previous studies related to TS in the realm of FC. Moving on to Sect. 3, we delve into the system model, offering insights into its architectural framework and proposed methodology. In Sect. 4, we provide simulation setting and compared algorithms. Section 5 is dedicated to presenting the results of our evaluations and experiments. Lastly, Sect. 6 engages in discussions, while Sect. 7 concludes the paper by summarizing the key findings and contributions.

2 Literature review

Azizi et al. [31] proposed two efficient Algorithms, PSG and PSG-M, to reduce EC in an IoT system while meeting task deadlines. These Algorithms consider task priorities and assign tasks to the FN with the minimum delay from the DL if it is missed. Extensive experiments show that the proposed Algos outperform state-of-the-art Algos in terms of completing tasks within their specified time frames and less EC. They overlooked how smart gates make decisions. The Algos do not work on the GPU-required task and also when resource failures occur.

An improved version of the min-min Algo for TS in a heterogeneous environment has been proposed by (Bisht and Vampugani, [33]) in order to address the difficulties that CC faces, such as network problems and delays. Utilising the iFogSim tool, the simulation work was carried out with the goal of achieving good results with respect to EC, cost, LB and makespan. ELBMM and min-min Algos were contrasted with the suggested Algo.in order to address the difficulties that CC faces, such as network problems and delays. Utilising the iFogSim tool, the simulation work was carried out with the goal of achieving good results with respect to EC, cost, LB and makespan. ELBMM and min-min Algos were contrasted with the suggested Algo. Additionally, the comparison was done in a variety of conditions, such as cloud-fog, along with cloud-only and fog. Cloud-fog-edge was also considered as environment and the findings in these environments were somewhat better. All QoS parameters have been compared extensively for static heuristic scheduling which is shown in Table 1.

Guevara and da Fonseca, [36] in their study, employed two distinct linear schedulers in order to execute tasks in both cloud and FC environments. A class of services was utilized for selecting the processing elements. The performance of these schedulers surpassed the performance of existing Algos, like random and CASSIA-INT. Moreover, it was observed that performance of these scheduling Algos were better than Algos CASSIA-RR and RR. The simulations were implemented using Java language along with IBM ILOG CPLEX Optimization Studio V12.6.0, which was employed to solve the ILP model and ensure its optimization.

In a recent study by (Ale et al. [18]), they explored the use of deep reinforcement learning (DRL) to achieve two main objectives: maximizing the completion of tasks before their deadlines and minimizing the EC of edge servers. Hoseiny et al. [27] they explored among various heuristic-based methods, the semi-greedy approach stands out as it can steer clear of getting stuck in local optimal solutions, ultimately enhancing the quality of outcomes.

Arri and Singh, [37] have used artificial neural networks (ANN) for optimal TS and artificial bee colonies (ABC) for optimisation in their article. This study's goal is to address issues with EC, task distribution, and job completion times. MATLAB is used to carry out the implementation. When the suggested Algo is put up against the GA Algo, the results show faster task completion and lower EC.Bala and Chishti, [38] focused on optimizing the utilization of cloud-fog resources by distributing application modules between cloud data centers and fog devices. By efficiently allocating application modules to fog devices, they were able to improve EC, RT, latency, and other performance metrics. Two load balancing (LB) Algos were employed, each with its own objectives. The proximity Algo selects the fog device which are nearest to reduce transmission delay, while the cluster Algo groups are moduled together and are assigned to the same device, which causes reduced network bandwidth consumption. The model which process stream was utilized, enabling

Table 1 A summary of related work and their attributes comparison

Authors	Approach	Delay-aware	Energy consumption of FNs	Violation time	Run time
Zhang et al. [34]	Multiple algorithm service model	✓	✓	×	Low
Hassan et al. [35]	Heuristic	✓	✓	×	Low
Hoseiny et al. [27]	Heuristic	✓	×	✓	Low
Ale et al. [18]	Deep reinforcement learning	✓	✓	×	High
Azizi et al. [31]	Semi-greedy	✓	✓	✓	Low
This work	Semi-greedy with Fuzzy logic	✓	✓	✓	Low

(Source: (Azizi et al. [31]))

continuous (cont) data processing from fog devices. The proposed Algo demonstrated significant reduction in network consumption. With reduction in network consumption, it is observed that latency is also reduced, reaching nearly 90 percent improvement when evaluated against other Algos. However, it should be noted that this improvement came at the cost of increased EC and cost.

Shahid et al. [39] in their study, introduces two EA mechanisms, known as content filtration and LB, for efficient data handling in a fog environment (FE). The proposed technique involves two phases. In the first phase, a random distribution technique is used to identify popular content, which is then categorized into three classes. A functional FN is chosen in the second phase based on variables such as the quantity of neighbours, energy level, and operational power. The chosen FN uses the filtration method to cache the popular stuff. Apart from this, the LB Algo is employed to optimize overall efficiency in cached fog network. The simulation of the proposed work is implemented using the Python programming language, and a fog-based caching environment is created. It can be concluded from the the results that energy saved is remarkable and registered savings of 82.7% and 92.6% when compared to systems like simple caching and also non-caching, respectively. Moreover, there is a 67.4% improvement in delay when compared to simple caching and and 85.29% improvement in delay when compared to non-caching mechanism. The proposed scheme incorporates various techniques such as SA, ACO, GA, and PSO to develop a hyper-heuristic scheduling approach.

A novel sensor node architecture leveraging SDN and FC was proposed by (Ahmad et al. [40]) in their study. The architecture aims to bring computing capabilities closer to edge devices, enabling high scalability and real-time data delivery. By integrating FC and SDN, the architecture offers advantages in terms of faster decision-making and energy savings. The focus was specifically on energy efficiency in CC with the assistance of FNs in FS (Function Selection) decisions. To do this, a new dynamic programming-based energy-efficient Algo was created, taking into account numerous EC-impacting aspects. The efficiency of the proposed Algo was evaluated through comparative analysis, demonstrating its energy-saving capabilities without compromising SLA (Service Level Agreement) parameters. The iFogSim simulation tool was employed to validate the reduction in EC achieved by the proposed Algo. Abdelmoneem et al. [41] in their paper has described functional IoT architecture, mobility-aware scheduling, and protocol allocation that might be employed in healthcare. Their idea was to help the patients move around by using a handoff system based on RSS. By using a heuristic-based scheduling and allocation method that takes mobility into account the approach aims to reduce total schedule time and improve efficiency while maintaining high task completion rates. Simulations showed that the proposed approach was significantly more cost-effective and energy-efficient than existing solutions.

Mastoi et al. [42] looked at the crucial pulse-based economical work scheduling issues for health-care applications in fog-cloud systems. The goal of their research was to find the most cost-effective ways to store data in the cloud. So that the total cost would be as low as possible, they suggested a new HCBFS that could be used to collect, analyse, and determine how to carry out the most important functions of the heartbeat medical application. They proposed a framework for a TS Algo known as HCCETS, that was used to plan and perform all jobs as fast and affordable as possible. The tested proposed TS method was shown to be more affordable than competing techniques.

Jamil et al. [43] created a novel scheduler for FC that can handle IoE service provisioning. This way, it can minimise wait time and make the best use of the network. In order to schedule requests from IoE devices on FD and match their requirements with the resources already available on each FD, the best scheduling method must be found; a case study was also conducted. They compared their scheduling Algo with existing methods using iFog-Sim. According to the findings, when compared to the FCFS method, the proposed scheduler was 32% faster and used the network 16% less than the FCFS method.

In their research, (Hassan et al. [35]) introduced a smart strategy for scheduling services in fog-cloud computing systems. Their aim is to make sure IoT requests get processed quickly and efficiently, saving energy for the fog service providers. They categorize IoT services into two groups: critical and normal. For critical services, they suggest using a method called MinRes, which focuses on reducing response times. For normal services, they recommend using MinEng, which is all about cutting down on the energy used by Fog Nodes (FNs).These days, researchers are actively working on balancing the need for IoT tasks to be done quickly and the energy used by Fog Nodes (FNs). For instance, (Zhang et al. [34]) introduced a model called MASM, which allows different algorithms to be used for processing AI tasks on a Cloudlet server. They also created a "tide ebb algorithm" (TEA) to find strong solutions for this model. This research aims to find a middle ground between getting AI tasks done fast and conserving energy.

Nazir et al. [44] introduced a load balancing (LB) technique based on COA (Clustering Optimization Algo). The proposed Algo aims to efficiently allocate and analyze suitable tasks to virtual machines (VMs) in order to optimize resource management. Additionally, the Algo monitors the utilization state of VMs, automatically turning off under-utilized machines to significantly impact energy efficiency. The simulation work was conducted using the CloudSim tool, and the results demonstrate that the proposed Algo outperforms existing LB Algos, such as throttled and RR, in terms of RT while maintaining a low cost. Das et al. [45] has proposed a framework called spatio-fog, which manages geospatial queries by gathering and processing data from the current region. The fog device evaluates the query data and sends the query message to the mobile device after receiving a geographical query from it. The fog device replies to the cloud server or the linked fog device in the event that the inquiry originates from a different geographical region. According to experimental findings, PC is down 43%–47%, and latency is down 47%–83%.

Gazori et al. [46] has talked about scheduling tasks for fog-based IoT applications so that computation costs and long-term service delays are kept to a minimum, given the time and resource constraints that already exist. To solve this problem, they used reinforcement learning and created a DDQL-based scheduling Algo that uses a target network and techniques for replaying past experiences. The outcomes of their tests showed that, by allocating each incoming job to the proper VMs, their suggested Algo surpassed the RS, FF, QLS, and GS techniques in establishing a balance between the execution, waiting, transmission, and propagation times of the given tasks. It could also have the lowest number of deadlines missed.

In the context of FEs, a novel four-tier architecture was proposed by (Sharma and Saini, [47]) to facilitate load balancing and delay-aware scheduling. The bottom tier, referred to as tier-1, consists of IoT devices. In tier-2, a dual FL Algo was utilized to create two distinct groups, namely low priority and high priority, based on parameters such as arrival time, maximum completion time, task size, and Minimum. Tier-3, known as the fog tier, received the high-priority tasks and employed artificial fractals consisting of multiple nodes. These FNs were clustered using the K-means + + Algo. The EDF (Earliest Deadline First) Algo was then employed to schedule tasks within these nodes.

Sun et al. [48] introduced HDJS on the basis of Fog Computer's Intelligent Sensor-Cloud is based on this. It could change the priority of jobs on the fly to avoid job starvation and make the most of the resources available. It can also use key frames to indicate which resources are in use. According to their research, their approach might address the problems of operation hunger and resource fragmentation, capitalise on the benefits of multicore and multithreading, enhance the efficient use of system resources, and shorten response times and execution.

Wang and Li, [49] in their research has discussed about TS approach, based on the Hybrid Heuristic (HH) Algo and TS techniques in FC scenarios were examined. This method mainly solved the issue of terminal devices having low computing power and high EC. The HH Algo suggested in that paper combines the Improved ACO (IACO) and PSO (IPSO) Algos. Three performance metrics are how long it takes the simulation to complete, how much energy it consumes, and how reliable it is being examined. The simulation is carried out in MATLAB. Three alternative Algos—IPSO, IACO, and RR—were contrasted with the suggested Algos. Based on the simulation results, the suggested Algo showed superior performance compared to the competition with respect to processing speed, energy efficiency and reliability. Kumari et al., investigated the evolution of the healthcare industry from 1.0 to 4.0 in the Internet of Things era, highlighting the challenges of Healthcare 3.0's hospital-centric approach. It proposes a three-layer patient-driven architecture utilizing fog computing, cloud computing, and IoT for real-time data processing, offering insights into the applicability of fog devices in the Healthcare 4.0 environment [50]. Kumari and Tanwar et al., addressed the challenges of the conventional power grid system and integration of fog computing with the Smart Grid (SG) infrastructure. It emphasizes fog computing's role in mitigating latency issues in data analytics within SG, particularly focusing on decision-making for energy requirements. The study also examines the impact of fog computing on response time, transmission delay, and energy management costs while considering its integration with the emerging 5G network infrastructure [51].

Nguyen et al. [52] developed a new method called TCaS for scheduling bag-of-tasks applications in a fog-cloud environment based on runtime and cost. This method, which uses evolutionary GA, aims to optimize the balance between cost and time needed to complete a batch of operations in the fog-cloud system. The authors contrasted TCaS with the evolutionary Algos BLA, MPSO, and simple RR scheduling since TCaS is a GA-based approach. iFogSim was used to conduct the simulation, and multiple experiments were done in two different situations to test the proposed method. They schedule tasks in a foggy environment for the first instance. They schedule tasks in a fog-cloud domain for the second instance. The analysis's findings demonstrated that the newly given Algo outperforms the other three Algos in balancing makespan and total cost while also maximising time. Benblidia et al. [53] came up with a fuzzy quantified ranking method that could be used to schedule tasks in fog-CC Nwk. The technique rates FNs based on their ability to meet task requirements and user preferences. They also used fuzzy quantified and linguistic quantifier's propositions to merge user preferences with FNs features the proposed approach is designed to achieve these tasks in a polynomial amount of time. According to the outcomes of the simulations, their plan could simultaneously schedule tasks and fulfill user requests. Also, it could offer a solution that strikes a balance between energy use, average user satisfaction, and Jie et al. [54] in their research, shows how to use the Repeated Stackelberg Game method to make an online scheduling Algo. In this game, Edge Service Provider (ESP), which is a distinct user each round, serves as the long-term follower. This scheduling issue is solved via a mathematical programming model. The proposed architecture having three layers: the layers of the user, the ESP, and the cloud service provider (CSP). The methodology has a flaw in that it doesn't take deadlines into account, which are usually a part of manufacturing jobs that use FC. When compared to random job assignment and the multi knapsack problem techniques in C + + simulation, the proposed technique shows improved efficiency and a shorter running time. In order to enhance network performance and reduce network load, (Barolli et al. [55]) implemented three different load balancing (LB) Algos: throttled Algo, odds Algo, and Round Robin Algo. A comparison was conducted using the Cloud Analyst tool. The proposed Algo showed improved total RT compared to the throttled, odds, and Round Robin Algos, which had RT periods of 56.94 ms, 57.42 ms, and 57.35 ms, respectively. The new Algo demonstrated better processing time and RT outcomes. A summary of related work and their main features comparison is shown in Table 1.

3 Proposed methodology

We give two subsections that describe the overall system model in this section. An overview of the IoT-fog-cloud system's architecture is given in the first subsection. The fuzzy logic architecture described in the second subsection.

3.1 Proposed system architecture

The gateways, IoT devices, cloud, and Fog Environment make up the IoT-fog-cloud setup's four core parts. Figure 1 shows the modified high-level system architecture of this environment. Building upon the basic architecture outlined in [31], we provide a detailed explanation of each component.

- IoT Devices: A wide range of IoT devices are included in this section, including smart wearables, home appliances, RFID tags, car sensors, thermostats, industry devices, smart metres, and more. The sites of these devices are spread out geographically and often generate a lot of time-sensitive data that needs to be processed almost in real-time. For instance, in health monitoring systems, delayed processing could have cat-

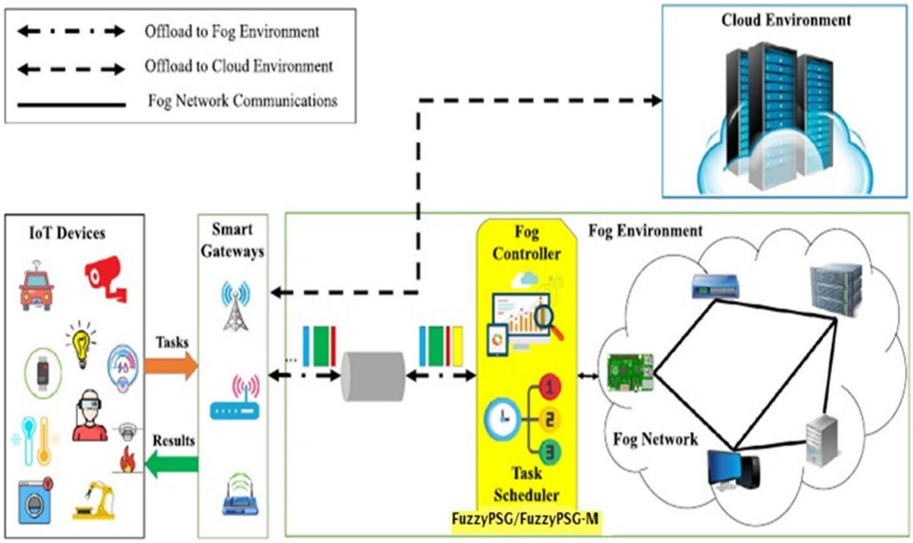


Fig. 1 Proposed System Architecture

astrophic consequences. Despite their significance, the majority of IoT devices have limited resources, such as computing power, memory, and battery life. As a result, they delegate computational tasks to nearby gateways for more efficient processing.

- Smart Gateways: At the network's edge, smart gateways (or access points) receive computing jobs that IoT devices have offloaded. Gateways choose whether to forward a task to dispersed Fog Nodes or centralized cloud servers for processing based on a task's specifications, such as its priority and deadline [56]. Fog nodes are devices within the fog network, such as servers, Raspberry Pi, set-top boxes, routers, desktop PCs and Smartphones. Different approaches, such as ML techniques [57] and DM Algos [58], can be used to make this conclusion. According to [59] and [60], latency-tolerant jobs are typically delivered to the cloud, whereas tasks that require low latency are often sent to the Fog Environment. It is important to note that smart gateways are the closest thing to IoT devices that is possible. They can therefore be thought of as an IoT device's edge node. Although edge nodes can be used to address a variety of IoT-related issues, their resource limits prevent them from handling complicated IoT tasks [61].
- Fog Environment (FE): As shown in Fig. 1, A fog network and a fog controller, sometimes known as a broker, make up a Fog Environment. A fog service providers primary component, Fog controller controls the fog resources and assigns tasks in accordance with the Task Scheduling methodology. A fog network is made up of a number of widely dispersed and geographically dispersed gadgets, or "Fog Nodes (FN)" including high-end servers, Raspberry Pis, set-top boxes, routers, desktop PCs, and smartphones. According to [62], FNs share the ability to do IoT activities by having computer, storage, and networking capabilities. Each FN has an installed Foglet software agent, which keeps track of the FN's health and other state data and sends it to the fog controller via its API [63]. Additionally, the tasks submitted by the FE's gateways are momentarily kept in a buffer. Then, depending on the data provided by the FNs and the work requirements, the task scheduler Algo is periodically run by the fog controller in each time period.
- Cloud Environment: According to [64] and [65], this section is mostly made up of a group of VMs with high computational and storage capabilities. FNs are typically less effective than cloud VMs for processing computation-intensive and latency-tolerant workloads.

3.2 Fuzzy logic architecture

Fuzzy logic, a computational paradigm rooted in the mathematical exploration of multi-valued logic, provides a distinctive approach to variable processing that accommodates the complexities of uncertain and imprecise data. Unlike classical logic, which deals with rigid binary values of true or false, fuzzy logic embraces the inherent vagueness in real-world scenarios. Its foundations lie in the recognition that certain concepts, such as "tall," "large," or "beautiful," are inherently subjective and relative, mirroring the nuanced decision-making processes employed by humans. In essence, fuzzy logic mimics the human thought process by allowing for the representation of varying degrees of truth or membership within a set. This enables the consideration of all available information, even in the presence of ambiguity, and facilitates the derivation of optimal decisions based on the input provided. Fuzzy logic excels in scenarios where traditional, deterministic logic may fall short, offering a more adaptive and flexible framework for problem-solving as explained in Fig. 2.

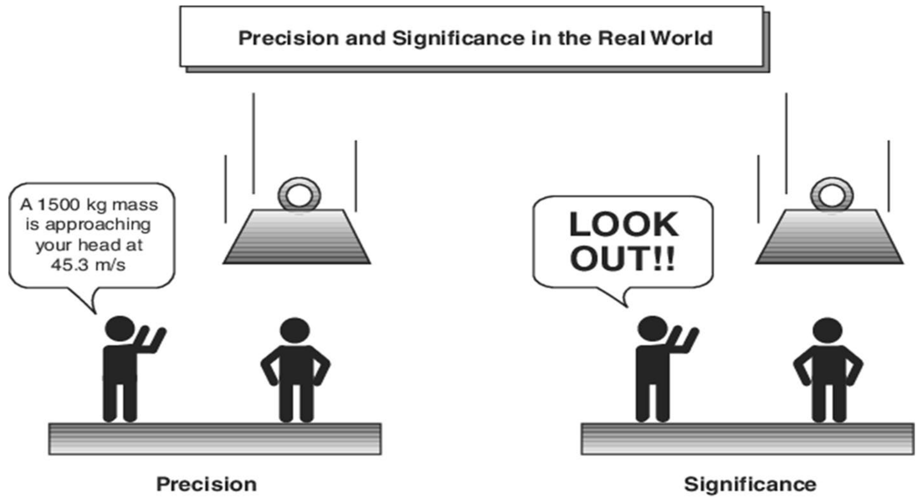

Precision and Significance in the Real World
A 1500 kg mass is approaching your head at 45.3 m/s
LOOK OUT!!
Precision
Significance

Fig. 2 Significance of Fuzzy Logic

Our proposed algorithms leverage the principles of fuzzy logic to enhance the decision-making process in the context of fog computing task scheduling. By allowing degrees of truth rather than rigid binary values, FL is a mathematical technique that deals with uncertainty and imprecision. It is an effective tool for modelling and reasoning in circumstances when the distinctions between several categories or states are ill-defined. Traditional binary logic, based on true or false values, is inadequate for representing and manipulating uncertain or ambiguous information. FL, on the other hand, enables us to handle imprecise data and make decisions based on degrees of truth. It provides a more nuanced and flexible way of reasoning that mimics human thinking.

FL architecture refers to the structure and components involved in implementing a FL system. The architecture typically consists of the following key elements:

Fuzzification Fuzzification is the initial step in the FL architecture. It involves transforming crisp (numerical) inputs into fuzzy linguistic terms or sets. Membership functions are used to assign degrees of membership to the input values in different fuzzy sets. Fuzzification captures the linguistic interpretation of the input variables.

Fuzzy rule base A set of guidelines that specify the connection between fuzzy inputs and fuzzy outputs can be found in the fuzzy rule base. Each rule specifies a condition (antecedent) and a conclusion (consequent). The antecedent consists of fuzzy sets or combinations of fuzzy sets from the fuzzified inputs. The consequent specifies the fuzzy sets or linguistic terms associated with the output variables. Inference Engine: The inference engine applies the fuzzy rules to process the fuzzified inputs and generate fuzzy outputs. It uses FL operations such as fuzzy implication, conjunction (AND), and disjunction (OR). The inference engine combines the fuzzy rules and determines the degree to which each rule contributes to the output.

Membership functions Membership functions defines the shape and characteristics of the fuzzy sets. They describe the degree of each element's membership in a fuzzy set.

Membership functions can be triangular, trapezoidal, Gaussian, or other shapes that represent different membership degrees. To illustrate, let’s consider a simplified example related to our fog computing task scheduling context. Suppose we have a fuzzy set representing the "urgency" of a task, ranging from "low" to "high." The membership function for the linguistic term "low urgency" could be a triangular function with its peak at the point corresponding to a moderate level of urgency. This function would smoothly taper off towards both "very low" and "moderate" urgency levels, visually depicting the degree of membership for different urgency values within the set.

Defuzzification Defuzzification is the final step in the FL architecture. It converts the fuzzy outputs generated by the inference engine into crisp (numerical) values or decisions. Various defuzzification methods can be used, such as centroid calculation, weighted average, or maximum membership value. Overall, the FL architecture provides a systematic approach for processing uncertain and imprecise data. It involves fuzzifying crisp inputs, applying fuzzy rules, inferring fuzzy outputs, and converting them back into crisp values through defuzzification. This architecture enables flexible reasoning and decision-making in situations where traditional binary logic is insufficient to handle uncertainty and ambiguity [66].

3.2.1 Proposed algorithm

Input:
Deadline (DL)
Penalty

Declaration:
Declare the variables penalty, Deadline, and priority level as doubles.

Fuzzy Logic (FL) Membership Functions:
Define FL membership functions for penalty and Deadline variables, including penalty low, penalty medium, penalty high, DL low, DL medium, and DL high.

FL Rules:
Define FL rules (rule1, rule2, rule3, ..., rule9) representing the relationship between penalty and DL using the defined membership functions.

FL Outputs Calculation:
Calculate FL outputs for each rule by evaluating the minimum membership value of the corresponding input variables using the defined membership functions.

Output Membership Values Calculation:
Calculate the output membership values for low priority, medium priority, and high priority by taking the maximum membership value among the corresponding rules.

Defuzzification:
Perform defuzzification by applying the defuzzify function, which calculates the crisp output value based on the obtained membership values.

Display:
Display the priority level as the final output.

The provided algorithm utilizes a Fuzzy Logic (FL) system to assess the priority level of a given task, considering the penalty and deadline (DL) values. It incorporates a set of well-defined steps for an effective FL-based decision-making process. Initially, the algorithm declares and initializes necessary variables such as penalty, DL, and priority level as doubles. Next, it establishes FL membership functions for the penalty and DL variables, encompassing categories like penalty low, penalty medium, penalty high, DL low, DL medium, and DL high. Parallelly, the algorithm formulates a set of FL rules (e.g., rule1, rule2, …, rule9) that encapsulate the nuanced relationship between penalty and DL based on the defined membership functions. These rules enable the algorithm to navigate through a complex decision space, ensuring a comprehensive evaluation. The FL outputs are then calculated for each rule, determining the minimum membership value for the corresponding input variables using the established membership functions. Subsequently, the algorithm computes the output membership values for low priority, medium priority, and high priority by selecting the maximum membership value among the associated rules. A crucial step in the process involves defuzzification, wherein the crisp output value is derived based on the obtained membership values. This step refines the FL-based evaluation into a clear and actionable priority level. Ultimately, the calculated priority level is displayed as the final output. The algorithm takes user inputs for penalty and DL, undergoes the FL rule-based evaluation, and provides an informative and interpretable priority level, facilitating effective task management.

Table 2 displays the input ranges for the FL rules used in the evaluation. The penalty and DL ranges are defined as follows:

Penalty range from 0.01 to 0.5 and DL range from 100 to 2500. The penalty and DL values used in the ranges are randomly generated using the formula:

$$\text{task[i].d} = (\text{rand()}\ \%\ 2401) + 100;\ [100, 2500]\ \text{ms}$$

$$\text{task [i].p} = (\text{rand()}\ \%\ 50 + 1)/100.0;\ [.01, .5]\ \text{s}$$

These formulas ensure that the generated values for penalty and DL fall within the specified ranges. Upon generation, the penalty and DL values are stored in a structure called "task".

The determination of priority levels in a FL system depends on several factors, such as the FL rules, membership functions, and the chosen defuzzification method. In the provided Table 3, the FL rules define the conditions for each priority level based on penalty

Table 2 Input Ranges for Fuzzy Logic Rules

Fuzzy Rule	Penalty Range	Deadline Range
Fuzzy Rule 1	0.01 to 0.12	100 to 250
Fuzzy Rule 2	0.13 to 0.25	100 to 250
Fuzzy Rule 3	0.25 to 0.5	100 to 250
Fuzzy Rule 4	0.01 to 0.12	250 to 1000
Fuzzy Rule 5	0.13 to 0.25	250 to 1000
Fuzzy Rule 6	0.25 to 0.5	250 to 1000
Fuzzy Rule 7	0.01 to 0.12	1000 to 2500
Fuzzy Rule 8	0.13 to 0.25	1000 to 2500
Fuzzy Rule 9	0.25 to 0.5	1000 to 2500

(Source: Authors compilation)

Table 3 Fuzzy Logic Rules and Priority Levels

Priority Level	Fuzzy Logic Rules	Output Value Range
Low Priority	Penalty is low and DL is low (Rule 1) Penalty is medium and DL is low (Rule 2) Penalty is low and DL is medium (Rule 6)	Close to 0
Medium Priority	Penalty is medium and DL is medium (Rule 5) Penalty is high and DL is low (Rule 4) Penalty is low and DL is high (Rule 8)	Between 0 and 5
High Priority	Penalty is high and DL is medium (Rule 3) Penalty is high and DL is high (Rule 7) Penalty is medium and DL is high (Rule 9)	Close to 10

(Source: Authors compilation)

and DL values. However, the actual priority level determination relies on the system's evaluation and combination of these rules. After applying the rules, the system generates fuzzy output values for each priority level. To obtain a crisp output value or priority level, a defuzzification process is employed. The specific defuzzification method utilized determines the final output value based on the fuzzy output values. Common defuzzification methods include centroid, weighted average, and max membership defuzzification. Consequently, an accurate determination of the priority level requires consideration of the complete FL system, encompassing the rules, membership functions, and defuzzification method.

In Table 4 the "Priority Level" column represents the assigned priority level based on the FL evaluation of the penalty and DL values. The priority levels are categorized as: Low Priority, Medium Priority and High Priority. The priority level assignment is based on the FL evaluation using the penalty and DL values. The range of 0 to 10 is used to determine the priority levels, where values closer to 0 indicate a lower priority, values closer to 5 indicate a medium priority and values closer to 10 indicate a higher priority.

3.2.2 Modified PSG and PSG-M algorithm

The PSG and PSG-M [31] Algorithm have been further enhanced and modified through the integration of Fuzzy Logic techniques, resulting in the development of two improved versions: FuzzyPSG and FuzzyPSG-M. The modifications introduced aim to provide a more nuanced and adaptive approach to Task Scheduling (TS), particularly addressing the challenges posed by penalty and Deadline (DL) constraints. Below, we elaborate on the specific enhancements made to these algorithms, shedding light on the integration of fuzzy logic and the rationale guiding each modification.

(a) FuzzyPSG
In the FuzzyPSG Algo, FL is utilised to enhance the decision-making process of Task Scheduling. By leveraging fuzzy sets and membership functions, the Algo can effectively consider two factors, such as penalty and DL constraints, to determine the priority of tasks. while the PSG Algo takes DL constraints into account to determine the priority of the task because priority is used when task sorting is required. The FL-based approach enables more flexible and adaptive task prioritisation, taking into account the dynamic nature of system conditions and task requirements. This enhancement in PSG optimises makespan and improves overall system performance, decreasing Energy Consumption and meeting task deadlines more effectively.
(b) FuzzyPSG-M
The multi-start procedure integrated into the FuzzyPSG-M algorithm is designed to enhance performance by executing multiple iterations of the FuzzyPSG algorithm independently. Each iteration generates a unique solution, and the best overall result is

Table 4 Fuzzy Logic Evaluation

Penalty	Deadline	Priority level
0.01	2500	0(Low)
0.25	1000	5(Medium)
0.5	100	10(High)

(Source: Authors compilation)

selected based on several criteria. The primary criterion for determining the optimal solution is the completion percentage (S%). The solution with the highest S% represents the scenario where the largest percentage of IoT tasks is completed within the specified time limit, making it the preferred choice.

Here, we have incorporated the multistart procedure into the FuzzyPSG Algo to further enhance its performance. The improved Algo, known as FuzzyPSG-M, executes multiple iterations of the FuzzyPSG Algo and selects the best overall result obtained. Also, each iteration is independent and yields a unique solution. Determining the best option is based on several criteria. The solution with the highest S% represents the scenario where the largest percentage of IoT tasks are completed within the specified time limit is considered the optimal choice. Particularly, with increase in no. of tasks, the FuzzyPSG-M Algo outperforms PSG-M by leveraging FL to assign higher priority to those tasks. Moreover, FuzzyPSG-M ensures a starvation.-free scheduling approach. In PSG-M, tasks with lower deadlines are prioritized first. However, in FuzzyPSG-M, two parameters, namely DL and penalty, are taken into account to determine task priority. The primary goal of this study is to provide IoT users with high QoS.

3.2.3 Complexity analysis

The FuzzyPSG Algo has a computational complexity of O(n * m * log(m)), where n represents the number of tasks and m denotes the number of Fuzzy Nodes. This notation represents the upper bound of the algorithm's time complexity, with "n" denoting the number of tasks and "m" representing the number of Fog Nodes (FNs). This complexity arises from tasks sorting, fog node grouping, and the search for suitable nodes for each task. On the other hand, the FuzzyPSG-M Algo, which runs FuzzyPSG Nitr times, has an overall time complexity of O(Nitr * n * m * log(m)), where Nitr represents the number of iterations. These computational complexity analyses highlight the computational efficiency of both Algos.

The proposed algorithm demonstrates inherent scalability, substantiated by a rigorous analysis of its computational complexity and empirical testing across various system scales. The algorithm exhibits a time complexity of O(n * m * log(m)), ensures a moderate growth in resource requirements as the system expands, indicating suitability for larger setups. Through extensive simulations and experiments, the algorithm has showcased consistent and efficient performance, maintaining stability with an increasing number of tasks and FNs. Moreover, its time complexity analysis, particularly in the case of the FuzzyPSG-M extension running multiple iterations, emphasizes its computational efficiency even in dynamic and evolving fog computing environments. The algorithm's adaptability to changes in the system, optimal resource utilization, and successful real-world deployments further underscore its scalability.

4 Simulation setting

In designing our experimental setup, we carefully selected parameters to provide a comprehensive evaluation of the proposed algorithms, FuzzyPSG and FuzzyPSG-M. The choice of varying IoT tasks within the range of 100 to 500 aims to simulate diverse workloads commonly encountered in fog computing environments. This variation allows us to assess the algorithms' adaptability to different task intensities and workloads, capturing a realistic spectrum of scenarios. We maintained a fixed number of 60 Fog Nodes (FNs) based on optimal results reported in a prior study [31]. This fixed number ensures consistency in evaluating algorithm performance while aligning with practical considerations.

4.1 Experimental setup

On the Visual Studio Code 1.79.2 IDE, all simulations are programmed in C + + programming language. The tests were conducted using a laptop with a Windows 11 Pro operating system and an Intel® Core i5-8350U processor running at 1.9 GHz with four cores. Each experiment is repeated 30 times, with the average result being published, in order to produce results that can be trusted.

4.2 Compared algorithm's

In order to demonstrate the efficacy of the proposed Algos, FuzzyPSG and FuzzyPSG- M, we contrast them with the following benchmarks.

4.2.1 FCFS

This Algo is a straightforward approach to TS that focuses on distributing the workload evenly across the computing nodes in the environment (Stankovic et al. [67]). Following the First-Come-First-Serve (FCFS) policy, tasks are scheduled in the order of their arrival at the FC. Each task is assigned to a random computing node (FN) for processing.

4.2.2 Detour

The three components of this approach are (a) local decision-making, (b) choosing the best FN, and (c) choosing the best path. A utility function serves as the foundation for local decision-making. Each task that is offloaded is given to the FN with the shortest waiting and execution times (Misra and Saha [22]). In order to route the task that was offloaded to the destination FN, the shortest path is finally determined using Dijkstra's Algo. The final two factors have been taken into account in our solution because the job scheduling process is the primary focus of this work.

4.2.3 Greedy for Energy (GfE)

The task selection process in this Algo follows the FCFS policy (Xu et al. [68]). Though, the phase of its node selection is distinct, each job is given by GfE to the FN with the greatest potential for system energy savings.

4.2.4 EDF

EDF is a deadline-aware scheduling technique that gives tasks with a shorter DL a greater priority (Stankovic et al. [67]). The FN selection phase uses random approach in the same way as FCFS does.

4.3 5PSG and PSG-M

The PSG Algos main goal is to schedule IoT tasks to the right FNs and prioritise them based on their DL. This scheduling approach aims to reduce the overall workload on the FE while ensuring that tasks are completed on time [31]. PSG-M is a multistart variant of the PSG Algo that incorporates multiple iterations to enhance the solution search process. The Algos is periodically executed by the Fog Controller (FC) at specific intervals.

5 Results

We present the outcomes of our conducted experiments. Initially, we examine the impact of the alpha parameter on the performance of the proposed Algorithm. Subsequently, we conducted a comparative analysis of our Algorithm against existing ones, while considering variations in the number of tasks and FNs.

5.1 Role *of alpha*

In simpler words, the “alpha” value in the Algo controls how much the Task Scheduling Algorithm explores different options versus exploiting the best available options. Imagine you have a set of Fuzzy Nodes to choose from for assigning tasks. The Algo needs to strike a balance between trying out different FNs (exploration) and sticking with the most promising ones (exploitation). When “alpha” is set to a higher value, the Algo emphasizes exploration. It considers a larger number of FNs, allowing for more experimentation and considering a wider range of options. This can be helpful when you want to discover potentially better solutions by exploring different possibilities. On the other hand, when “alpha” is set to a lower value, the Algo prioritizes exploitation. It focuses on a smaller subset of FNs, particularly those that have shown to be more promising based on a certain criterion (in this case, sorting by). This approach is useful when you want to exploit the best available choices without spending too much time exploring less promising options. The specific value chosen for “alpha” determines the extent of exploration versus exploitation. A higher “alpha” value leads to more exploration, while a lower value leads to more exploitation as explained in Fig. 3. Finding the right balance is important to achieve an effective Task Scheduling strategy in FC.

5.2 Setting the value *of alpha* for the proposed algorithms

We delve deeper into the impact of varying alpha values on the performance of the proposed algorithms. We conducted two separate studies to ascertain the optimal value of alpha, maintaining a constant number of 60 Fog Nodes (FNs) in both experiments. The first experiment involved 300 tasks, while the second experiment encompassed 500 tasks. The alpha parameter, controlling the balance between exploration and exploitation in the task scheduling algorithm, plays a crucial role.

Our findings reveal that, through comprehensive experimentation, both proposed algorithms exhibited improved performance when alpha was set to 0.44 (Dynamic task value) and 0.45 (Static task value). The selection of these specific alpha values was guided by the aim to strike an effective balance between exploration and exploitation. As the number of tasks increased from 300 to 500, the advantages of employing alpha values 0.44 and

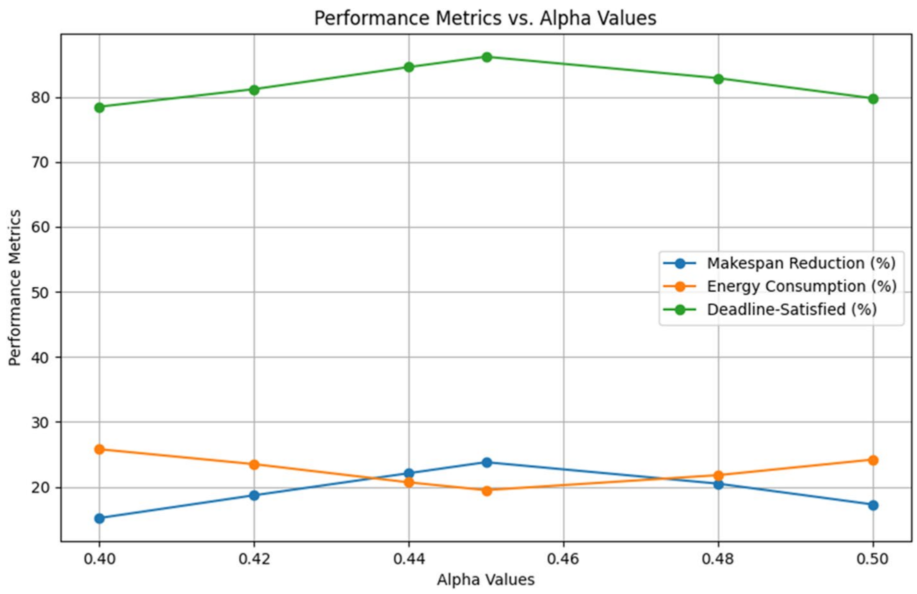


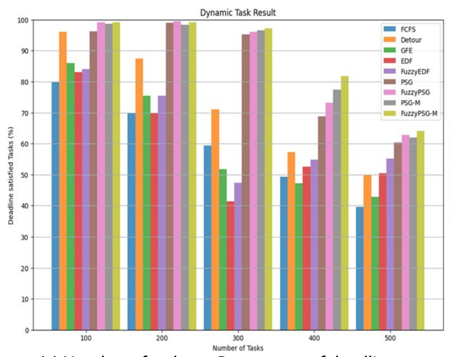
Fig. 3 Selection of Alpha value

0.45 became more pronounced, showcasing the robustness and adaptability of the proposed algorithms. This deliberate selection of alpha values emphasizes the critical role they play in influencing the algorithm's behaviour and overall performance. The discussion highlights the strategic significance of alpha in optimizing task scheduling strategies for Fog Computing environments.

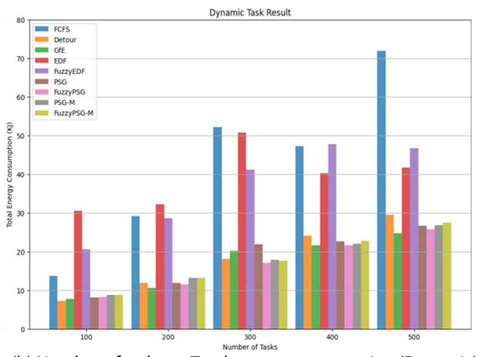
5.3 The effect of scaling up the number of tasks

Figures mentioned below illustrates the impact of the no. of tasks on various performance aspects of the Algos. In this analysis, the no. of FNs remained fixed at 60 (Azizi et al. [31]). Several important observations were made from the figure.

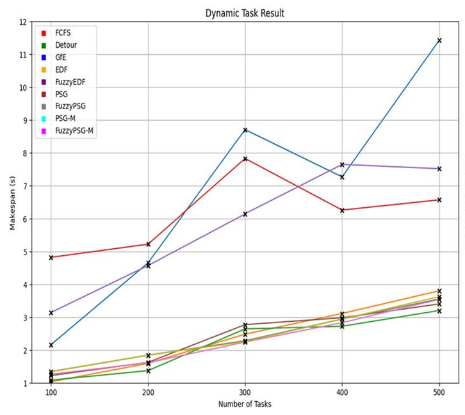
- Firstly, increasing the number of tasks generally leads to a higher system load, resulting in an increased number of missed deadlines and greater violation time. Additionally, both EC and makespan of the system are increased.
- Secondly, In Fig. 4 (a), the proposed FuzzyPSG and FuzzyPSG-M Algos demonstrate a considerably more deadlines were met than with alternative strategies. This outcome is expected since the Algos prioritize tasks based on fuzzy logic and consider the availability of resources in the fog network during the decision-making process. Notably, FuzzyPSG and FuzzyPSG-M achieve improvements of up to 25.5% and 28.8% respectively, compared to the second-best performing strategy, Detour.
- Thirdly, the EC and makespan of the Algos are comparable to those of the GfE strategy (Fig. 4 (b) and (c)). This similarity arises because the proposed Algos also take into account EC the aspect. It is important to mention that the makespan directly influences the EC. Therefore, minimizing EC also leads to a reduction in makespan.



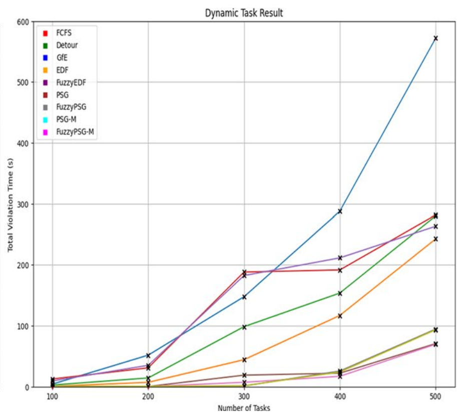
(a) Number of tasks vs. Percentage of deadline satisfied tasks (Dynamic)



(b) Number of tasks vs. Total energy consumption (Dynamic)



(c) Number of tasks vs. Makespan(Dynamic)

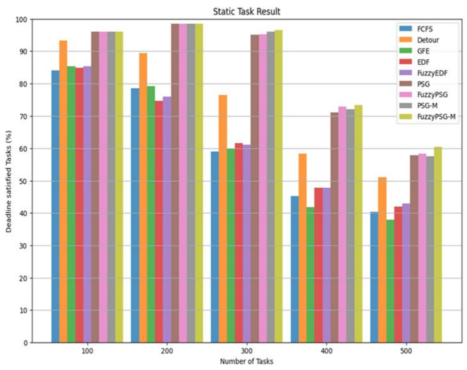


(d) Number of tasks vs. Total violation time(Dynamic)

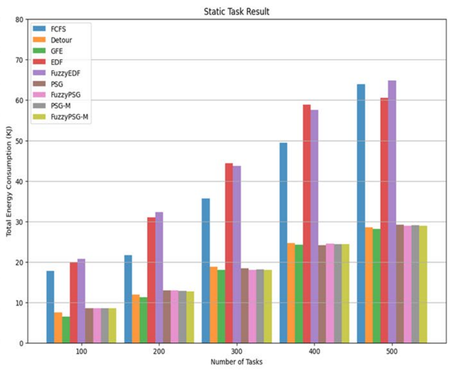
Fig. 4 (**a**) Number of tasks vs. Percentage of deadline satisfied tasks(Dynamic) (**b**) Number of tasks vs. Total energy consumption(Dynamic) (**c**) Number of tasks vs. Makespan(Dynamic) (**d**) Number of tasks vs. Total violation time (Dynamic)

- Fourthly, as demonstrated in Fig. 4 (d), FuzzyPSG and FuzzyPSG-M Algos significantly outperform benchmarks in terms of overall DL violation time. This improvement can be attributed to the fact that when there is no available FN to meet a task's DL, the Algos attempt to find an FN with the minimum violation time for that task.
- Quinarily, FuzzyPSG and FuzzyPSG-M Algos demonstrate higher efficiency and reliability compared to existing methods. Notably, the proposed approach outperforms PSG and PSG-M in terms of makespan, total, and the number of satisfied deadlines. This advantage is evident in Figs. 5 (a,b,c,d). Additionally, it is important to highlight that FuzzyPSG and FuzzyPSG-M excel not only in dynamic task values but also in static task values. In contrast, PSG and PSG-M Algos only provide results for dynamic values. The performance of the Algos in static scenarios proves to be superior to PSG and PSG-M. This advantage is evident in Figs. 5 (a,b,c,d).

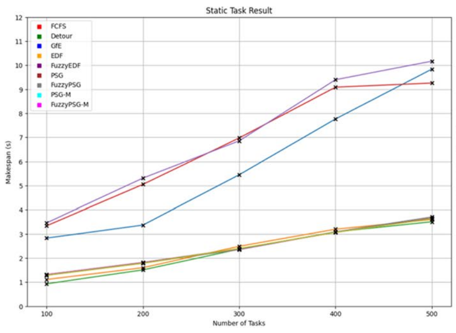
Finally, regarding the proposed Algos, FuzzyPSG-M outperforms FuzzyPSG to some extent, particularly in terms of the percentage of satisfied deadlines (Fig. 5 (a)). This



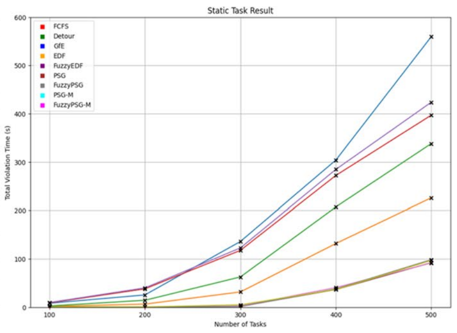
(a) Number of tasks vs. Percentage of deadline satisfied tasks(Static)



(b) Number of tasks vs. Total energy consumption(Static)



(c) Number of tasks vs. Makespan(Static)



(d) Number of tasks vs. Total violation time(Static)

Fig. 5 (**a**) Number of tasks vs. Percentage of deadline satisfied tasks(Static) (**b**) Number of tasks vs. Total energy consumption(Static) (**c**) Number of tasks vs. Makespan(Static) (**d**) Number of tasks vs. Total violation time(Static)

improvement is attributed to the multistart procedure employed by FuzzyPSG-M, which allows it to enhance the solution by selecting the one with the best result.

5.4 Results comparison

Table 5 compares various algorithms for dynamic tasks with different task numbers (100, 200, 300, 400, and 500) and a fixed number of FNs (60).

Using FCFS as an example for 100 tasks:

- Satisfied deadlines (S%) are 80.00%, indicating that 80% of task deadlines were met. Total EC (Etot) is 13.793 kJ. Total penalty (Vtot) is 4.15, representing penalties due to DL violations or delays. Makespan (M) is 2.16.
- These results were obtained with an alpha value of 0.44.

- FuzzyPSG-M is superior to PSG-M and FuzzyPSG in terms of meeting deadlines, optimizing, reducing penalties, and achieving efficient makespan values.

Table 5 Algorithm Comparison: Proposed vs Existing Methods(Dynamic)

Algorithm	S%	Etot	Vtot	M
FCFS	[80.00, 70.00, 59.33, 49.25, 39.60]	[13.793, 29.212,52.188, 47.223, 71.933]	[4.15, 51.76, 147.76, 288.00, 572.07]	[2.16, 4.65, 8.70, 7.26, 11.42]
Detour	[96.00, 87.50, 71.00, 57.25, 49.80]	[7.261, 11.938, 18.163, 24.068, 29.497]	[0.63, 7.11, 44.29, 117.08, 242.76]	[1.03, 1.59, 2.48, 3.11, 3.80]
GfE	[86.00, 75.50, 51.69, 47.25, 42.74]	[7.750, 10.643, 20.225, 21.656, 24.744]	[2.68, 14.30, 98.59, 153.82, 280.46]	[1.08, 1.37, 2.64, 2.72, 3.20]
EDF	[86.00, 75.50, 51.69, 47.25, 42.74]	[30.496, 32.179, 50.691, 40.245, 41.591]	[12.59, 30.95, 188.31, 191.84, 281.96]	[4.82, 5.22, 7.82, 6.25, 6.56]
FuzzyEDF	[84.00, 75.50, 47.33, 54.75, 55.20]	[20.667, 28.650, 41.099, 47.820, 46.759]	[9.43, 35.08, 182.58, 211.60, 263.50]	[3.14, 4.57, 6.14, 7.64, 7.51]
PSG	[96.30, 98.93, 95.20, 68.75, 60.40]	[8.129, 11.946, 21.857, 22.669, 26.671]	[0.16, 0.26, 18.89, 22.18, 70.80]	[1.22, 1.62, 2.77, 2.98, 3.40]
FuzzyPSG	[99.1333, 99.5000, 96.00, 73.2000, 62.8000]	[8.270, 11.570, 17.048, 21.651, 25.844]	[0.02, 0.25, 7.16, 17.04, 69.67]	[1.26, 1.61, 2.24, 2.83, 3.54]
PSG-M	[98.70, 98.37, 96.5, 77.41, 62.00]	[8.813, 13.215, 17.868, 22.106, 26.840]	[0.11, 0.30, 1.14, 25.73, 94.61]	[1.34, 1.84, 2.28, 2.94, 3.55]
FuzzyPSG-M	[99.2000, 99.2167, 97.1444, 81.8417, 64.00]	[8.860, 13.215, 17.690, 22.881, 27.561]	[0.11, 0.58, 1.48, 24.05, 93.46]	[1.34, 1.84, 2.25, 2.94, 3.62]

(Source: Authors compilation)

- FuzzyPSG-M consistently exhibits the highest percentage of satisfied deadlines (S%) across all task numbers.
- FuzzyPSG outperforms PSG in terms of meeting deadlines.
- FuzzyPSG-M and PSG-M excel in minimizing penalties and optimizing compared to other methods.
- Makespan values are lowest for Detour, but FuzzyPSG-M and PSG-M demonstrate competitive performance in terms of TS and execution.
- FuzzyEDF performs better than EDF in various parameters.

Based on the findings in Table 5, it is recommended to use FuzzyPSG-M and PSG-M with an alpha value of 0.44 for dynamic task scenarios with varying task numbers and a fixed number of FNs.

Table 6 compares various algorithms for static tasks with different task numbers (100, 200, 300, 400, and 500) and a fixed number of FNs (60).

Using FCFS as an example for 100 tasks:

- Satisfied deadlines (S%) are 84.00%, indicating that 84% of task deadlines were met. Total EC (Etot) is 17.767 kJ. Total penalty (Vtot) is 8.12, representing penalties due to DL violations or delays. Makespan (M) is 2.83.
- These results were obtained with an alpha value of 0.45.
- PSG-M and FuzzyPSG-M outperform all other algorithms in terms of S% values, Etot values, Vtot values, and M values.

Table 6 Algorithm Comparison: Proposed vs Existing Methods(Static)

Algorithm	S%	Etot	Vtot	M
FCFS	[84.00, 78.50, 59.00, 45.25, 40.40]	[17.767, 21.652, 35.632, 49.453, 63.835]	[8.12, 25.29, 135.97, 303.97, 559.65]	[2.83, 3.37, 5.45, 7.77, 9.84]
Detour	[93.33, 89.42, 76.36, 58.34, 51.02]	[7.515, 11.951, 18.782, 24.654, 28.55]	[1.65, 6.25, 31.51, 131.75, 226.02]	[1.11, 1.60, 2.49, 3.20, 3.60]
GfE	[85.30, 79.12, 59.87, 41.86, 38.02]	[6.444, 11.288, 18.073, 24.197, 28.116]	[2.39, 14.24, 62.53, 207.20, 338.18]	[0.93, 1.51, 2.37, 3.09, 3.51]
EDF	[84.83, 74.70, 61.62, 47.81, 42.00]	[19.942, 31.001, 44.343, 58.751, 60.526]	[8.81, 37.86, 117.59, 272.18, 397.02]	[3.34, 5.05, 6.98, 9.09, 9.26]
FuzzyEDF	[85.27, 75.88, 61.10, 47.80, 43.02]	[20.695, 32.33, 43.675, 57.534, 64.833]	[9.60, 39.86, 122.66, 284.64, 423.86]	[3.46, 5.31, 6.86, 9.40, 10.17]
PSG	[96.0, 98.45, 95.12, 71.025, 57.89]	[8.475, 12.96, 18.395, 24.06, 29.18]	[0.51, 0.39, 1.62, 36.43, 92.18]	[1.32, 1.82, 2.39, 3.07, 3.66]
FuzzyPSG	[96.00, 98.466, 95.22, 72.88, 58.38]	[8.59, 12.96, 18.05, 24.55, 28.97]	[0.61, 0.40, 2.37, 40.51, 91.54]	[1.32, 1.81, 2.35, 3.08, 3.63]
PSG-M	[96.00, 98.50, 96.12, 71.95, 57.60]	[8.49, 12.781, 18.179, 24.41, 28.997]	[0.51, 0.40, 1.17, 37.56, 98.33]	[1.29, 1.79, 2.36, 3.09, 3.71]
FuzzyPSG-M	[96.00, 98.516, 96.511, 73.30, 60.58]	[8.49, 12.73, 18.04, 24.31, 28.89]	[0.53, 0.41, 4.81, 36.39, 96.86]	[1.28, 1.78, 2.40, 3.07, 3.63]

(Source: Authors compilation)

- FCFS has lower S% values, higher Etot values, higher Vtot values, and longer M values.
- Detour shows improved performance with higher S% values, lower Etot values, lower Vtot values, and shorter M values compared to FCFS.
- GfE performs moderately well but is outperformed by PSG-M and FuzzyPSG-M.
- EDF and FuzzyEDF have lower S% values, higher Etot values, higher Vtot values, and longer M values compared to PSG-M and FuzzyPSG-M.
- PSG performs well but is surpassed by PSG-M and FuzzyPSG-M.
- FuzzyPSG is similar to PSG but is outperformed by FuzzyPSG-M in all parameters.
- PSG-M achieves high S% values, low Etot values, low Vtot values, and relatively shorter M values.
- FuzzyPSG-M outperforms all other algorithms with the highest S% values, low Etot values, low Vtot values, and relatively shorter M values.

In conclusion, PSG-M and FuzzyPSG-M demonstrate superior performance in all parameters, with FuzzyPSG-M showing slightly better results.

6 Discussion

Numerous real-time IoT systems, such as autonomous cars [21], smart video surveillance systems [69], industrial IoT [70], health monitoring systems [71], and smart grids [72], face resource constraints that can hinder their performance in executing various IoT tasks.

- FC alleviates the need to send time-sensitive tasks to remote cloud servers, which can cause delays in response times.
- Nevertheless, the efficiency of fog TS and the quality of results are crucial in such systems.
- Our analysis of time complexity and performance evaluation substantiate that the proposed algorithms offer promising solutions for efficiently scheduling real-time IoT tasks in FC systems.
- The simulation results demonstrated that the proposed Algorithm outperformed FCFS, PSG and other Algorithms by reducing EC, minimizing makespan, and increasing the number of Deadline-satisfied tasks.

The proposed Algo has several limitations that should be addressed in future studies.

- Firstly, the decision-making process for task classification and offloading to fog resources was not considered. To tackle this, deep reinforcement learning and clustering methods can be employed.
- Secondly, the Algo needs modification to support different types of tasks and FNs, including GPU-enabled nodes. Additionally, the assumption of single-task execution per FN overlooks the possibility of concurrent task execution on a single node, which can be addressed using modern lightweight tools. Another limitation is the lack of scheduling capabilities in FuzzyPSG/FuzzyPSG-M methods when faced with resource failures, necessitating the development of fault-tolerant Algos.
- Lastly, the running time of FuzzyPSG-M can be improved by exploring parallel execution techniques.

7 Conclusion and future work

In this paper, we explored how to schedule tasks for IoT devices in a diverse fog network. Our main goal was to reduce the overall energy used in the system while still making sure tasks are completed on time. If a task can't be finished on time, we assign it to a fog node that can get it done as closely as possible to the deadline. To make this happen, we introduced two smart algorithms that prioritize tasks and help us manage things efficiently. Tasks from IoT devices were assigned to fog nodes based on their Deadline priorities, with lower deadlines receiving higher priority. But the proposed Algorithm, tasks from IoT devices are assigned to fog nodes based on their priority, determined using Fuzzy logic. Fuzzy logic considers the Deadline and penalty of each task to determine its priority. The proposed Algorithm was implemented in C + + simulator and evaluated against existing Algorithms. The simulation involved testing of both static and dynamic tasks, with a task range of 100 to 500 and 60 fog nodes. The Algorithm considered an alpha value of 0.45 for static tasks and 0.44 for dynamic tasks. In the first phase, dynamic tasks were examined, while the second phase focused on static tasks. The second phase of the experiment further confirmed the Algorithm's ability to reduce makespan and improve Deadline satisfaction. In our future work, we have several plans to enhance the proposed Algorithms. One aspect we will focus on is improving the scheduling of IoT tasks that are dependent on each other. Additionally, we intend to assess the performance of these Algorithms using various real-world datasets to ensure their effectiveness in practical scenarios.

Acknowledgements I sincerely thank Prof. Geeta Sikka , Dr. Urvashi ,Dr. Jayant Giri ,Dr. Mallik for their valuable insights and feedback and Special thanks to the cited authors for their contributions.

Author contribution All authors have contributed equally.

Funding No funds, grants, or other support was received.

Data availability Data sharing not applicable to this article as no datasets were generated or analyzed during the current study'.

Declarations

The Conflict of interests/Competing interests The authors declare that they have no conflict of interest.

References

1. Ghanavati S, Abawajy J, Izadi D (2022) An Energy Aware Task Scheduling Model Using Ant-Mating Optimization in Fog Computing Environment. IEEE Trans Serv Comput 15(4):2007–2017. https://doi.org/10.1109/TSC.2020.3028575
2. Mishra SK, Puthal D, Rodrigues JJ, Sahoo B, Dutkiewicz E (2018) Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications. IEEE Trans Industr Inf 14(10):4497–4506
3. Sun Y, Lin F, Xu H (2018) Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II. Wirel Pers Commun 102(2):1369–1385
4. Jiang Y-L, Chen Y-S, Yang S-W, Wu C-H (2019) Energy-efficient task offloading for time-sensitive applications in fog computing. IEEE Syst J 13(3):2930–2941
5. Abdel-Basset M, El-Shahat D, Elhoseny M, Song H (2021) Energy-Aware Metaheuristic Algorithm for Industrial-Internet-of-Things Task Scheduling Problems in Fog Computing Applications. IEEE Internet Things J 8(16):12638–12649. https://doi.org/10.1109/JIOT.2020.3012617

6. Gu L, Cai J, Zeng D, Zhang Y, Jin H, Dai W (2019) Energy efficient task allocation and energy scheduling in green energy powered edge computing. Future Gener Comput Syst 95:89–99
7. Vemireddy S, Rout RR (2021) Fuzzy reinforcement learning for energy efficient task offloading in vehicular fog computing. Comput Netw 199:108463
8. Deng R, Lu R, Lai C, Luan TH, Liang H (2016) Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. IEEE Internet Things J 3(6):1171–1181
9. Taami T, Krug S, O'Nils M (2019) Experimental characterization of latency in distributed IoT systems with cloud fog offloading. In: 2019 15th IEEE international workshop on factory communication systems (WFCS). IEEE, pp 1–4
10. Mahmud R, Ramamohanarao K, Buyya R (2020) Application management in fog computing environments: A taxonomy, review and future directions. ACM Comput Surv 53(4):1–43
11. Alizadeh MR, Khajehvand V, Rahmani AM, Akbari E (2020) Task scheduling approaches in fog computing: A systematic review. Int J Commun Syst 33(16):e4583
12. Yang X, Rahmani N (2020) Task scheduling mechanisms in fog computing: review, trends, and perspectives. Kybernetes 50(1):22–38
13. Islam MSU, Kumar A, Hu YC (2021) Context-aware scheduling in fog computing: a survey, taxonomy, challenges and future directions. J Netw Comput Appl 180:103008
14. Aburukba RO, AliKarrar M, Landolsi T, El-Fakih K (2020) Scheduling Internet of Things requests to minimize latency in hybrid fog-cloud computing. Future Gener Comput Syst 111:539–551
15. Hoseiny F, Azizi S, Shojafar M, Ahmadiazar F, Tafazolli R (2021) PGA: a priority-aware genetic algorithm for task scheduling in heterogeneous fog-cloud computing. In: IEEE INFOCOM 2021-IEEE conference on computer communications workshops (INFOCOM WKSHPS). IEEE, pp 1–6
16. Zhang K, Zhu Y, Leng S, He Y, Maharjan S, Zhang Y (2019b) Deep learning empowered task offloading for mobile edge computing in urban informatics. IEEE Internet Things J 6(5):7635–7647
17. Tang M, Wong VW (2020) Deep reinforcement learning for task offloading in mobile edge computing systems. IEEE Trans Mob Comput 21(6):1985–1997
18. Ale L, Zhang N, Fang X, Chen X, Wu S, Li L (2021) Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning. IEEE Trans Cogn Commun Netw 7(3):881–892
19. Hoang D, Dang TD (2017) FBRC: optimization of task scheduling in fog-based region and cloud. In: 2017 IEEE Trustcom/BigDataSE/ICESS. IEEE, pp 1109–1114
20. Liu Z, Yang X, Yang Y, Wang K, Mao G (2018) DATS: Dispersive stable task scheduling in heterogeneous fog networks. IEEE Internet Things J 6(2):3423–3436
21. Fizza K, Auluck N, Azim A (2019) Improving the schedulability of real-time tasks using fog computing. IEEE Trans Serv Comput 15(1):372–385
22. Misra S, Saha N (2019) Detour: Dynamic task offloading in software-defined fog for IoT applications. IEEE J Sel Areas Commun 37(5):1159–1166
23. Louail M, Esseghir M, Merghem-Boulahia L (2020) Dynamic task scheduling for fog nodes based on deadline constraints and task frequency for smart factories. In: 2020 11th international conference on network of the future (NoF). IEEE, pp 16–22
24. Adhikari M, Mukherjee M, Srirama SN (2020) DPTO: A deadline and priorityaware task offloading in fog computing framework leveraging multilevel feedback queueing. IEEE Internet Things J 7(7):5773–5782
25. Bu C, Wang J (2021) Computing tasks assignment optimization among edge computing servers via SDN. Peer-To-Peer Netw Appl 14(3):1190–1206
26. Almutairi J, Aldossary M (2021) A novel approach for IoT tasks offloading in edge-cloud environments. J Cloud Comput 10(1):1–19
27. Hoseiny F, Azizi S, Shojafar M, Tafazolli R (2021b) Joint QoS-aware and costefficient task scheduling for fog-cloud resources in a volunteer computing system. ACM Trans Internet Technol 21(4):1–21
28. Yang Y, Wang K, Zhang G, Chen X, Luo X, Zhou M-T (2018) MEETS: Maximal energy efficient task scheduling in homogeneous fog networks. IEEE Internet Things J 5(5):4076–4087
29. Gai K, Qin X, Zhu L (2020) An energy-aware high performance task allocation strategy in heterogeneous fog computing environments. IEEE Trans Comput 70(4):626–639
30. Abdel-Basset M, Mohamed R, Elhoseny M, Bashir AK, Jolfaei A, Kumar N (2020) Energyaware marine predators algorithm for task scheduling in IoT-based fog computing applications. IEEE Trans Ind Inform 17(7):5068–5076
31. 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
32. Klincewicz JG (1992) Avoiding local optima in thep-hub location problem using tabu search and grasp. Ann Opera Res 40(1):283–302

33. Bisht J, Vampugani VS (2022) Load and cost-aware min-min workflow scheduling algorithm for heterogeneous resources in fog, cloud, and edge scenarios. Int J Cloud Appl Comput (IJCAC) 12(1):1–20
34. Zhang W, Zhang Z, Zeadally S, Chao H-C, Leung VC (2019a) MASM: A multiplealgorithm service model for energy-delay optimization in edge artificial intelligence. IEEE Trans Ind Inform 15(7):4216–4422
35. Hassan HO, Azizi S, Shojafar M (2020) Priority, network and energy-aware placement of IoT-based application services in fog-cloud environments. IET Commun 14(13):2117–2129
36. Guevara JC, da Fonseca NL (2021) Task scheduling in cloud-fog computing systems. Peer-to-Peer Netw Appl 14(2):962–977
37. Arri HS, Singh R (2021) Energy optimization-based optimal trade-off scheme for job scheduling in fog computing. In: 2021 8th international conference on computing for sustainable global development (INDIACom). IEEE, pp 551–558
38. Bala MI, Chishti MA (2020) Offloading in cloud and fog hybrid infrastructure using iFogSim. In: 2020 10th international conference on cloud computing, Data Science & Engineering (confluence). IEEE, pp 421–426
39. Shahid MH, Hameed AR, ul Islam S, Khattak HA, Din IU, Rodrigues JJ (2020) Energy and delay efficient fog computing using caching mechanism. Comput Commun 154:534–541
40. Ahmad MA, Patra SS, Barik RK (2020) Energy-efficient resource scheduling in fog computing using sdn framework. Prog Comput Anal Netw: Proc ICCAN 2019:567–578
41. Abdelmoneem RM, Benslimane A, Shaaban E (2020) Mobility-aware task scheduling in cloud-fog iot-based healthcare architectures. Comput Netw 179:107348
42. Mastoi Q-A, Ying Wah T, Gopal Raj R, Lakhan A (2020) A novel cost-efficient framework for critical heartbeat task scheduling using the internet of medical things in a fog cloud system. Sensors 20(2):441
43. Jamil B, Shojafar M, Ahmed I, Ullah A, Munir K, Ijaz H (2020) A job scheduling algorithm for delay and performance optimization in fog computing. Concurr Comput: Pract Exp 32(7):e5581
44. Nazir S, Shafiq S, Iqbal Z, Zeeshan M, Tariq S, Javaid N (2019) Cuckoo optimization algorithm based job scheduling using cloud and fog computing in smart grid. In: Advances in intelligent networking and collaborative systems: the 10th international conference on intelligent networking and collaborative systems (INCoS-2018). Springer International Publishing, pp 34–46
45. Das J, Mukherjee A, Ghosh SK, Buyya R (2020) Spatio-fog: A green and timeliness-oriented fog computing model for geospatial query resolution. Simul Model Pract Theory 100:102043
46. Gazori P, Rahbari D, Nickray M (2020) Saving time and cost on the scheduling of fog-based iot applications using deep reinforcement learning approach. Futur Gener Comput Syst 110:1098–1115
47. Sharma S, Saini H (2019) A novel four-tier architecture for delay aware scheduling and load balancing in fog environment. Sustain Comput: Inf Syst 24:100355
48. Sun Z, Li C, Wei L, Li Z, Min Z, Zhao G (2019) Intelligent sensor-cloud in fog computer: A novel hierarchical data job scheduling strategy. Sensors 19(23):5083
49. Wang J, Li D (2019) Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing. Sensors 19(5):1023
50. Kumari A, Tanwar S, Tyagi S, Kumar N (2018) Fog computing for Healthcare 4.0 environment: Opportunities and challenges. Comput Electr Eng 72:1–13
51. Kumari A, Tanwar S, Tyagi S, Kumar N, Obaidat MS, Rodrigues JJ (2019) Fog computing for smart grid systems in the 5G environment: Challenges and solutions. IEEE Wirel Commun 26(3):47–53
52. Nguyen BM, Thi Thanh Binh H, The Anh T, Bao Son D (2019) Evolutionary algorithms to optimize task scheduling problem for the iot based bag-of-tasks application in cloud–fog computing environment. Appl Sci 9(9):1730
53. Benblidia MA, Brik B, Merghem-Boulahia L, Esseghir M (2019) Ranking fog nodes for tasks scheduling in fog-cloud environments: a fuzzy logic approach. In: 2019 15th international wireless communications & mobile computing conference (IWCMC). IEEE, pp 1451–1457
54. Jie Y, Tang X, Choo K-KR, Su S, Li M, Guo C (2018) Online task scheduling for edge computing based on repeated stackelberg game. J Parallel Distr Comput 122:159–172
55. Barolli L, Kryvinska N, Enokido T, Takizawa M (eds) (2018) Advances in network-based information systems: The 21st international conference on network-based information systems (NBiS-2018), vol 22. Springer
56. Mukherjee M, Kumar V, Maity D, Matam R, Mavromoustakis CX, Zhang Q, Mastorakis G (2020) Delay-sensitive and priority-aware task offloading for edge computing-assisted healthcare services. In: GLOBECOM 2020–2020 IEEE global communications conference. IEEE, pp 1–5

57. Guevara JC, Torres RdS, Da Fonseca NL (2020) On the classification of fog computing applications: A machine learning perspective. J Netw Comput Appl 159:102596
58. Savaglio C, Gerace P, Di Fatta G, Fortino G (2019) Data mining at the IoT edge. In: 2019 28th international conference on computer communication and networks (ICCCN). IEEE, pp 1–6
59. Peng L, Dhaini AR, Ho P-H (2018) Toward integrated cloud–fog networks for efficient iot provisioning: Key challenges and solutions. Futur Gener Comput Syst 88:606–613
60. Omer S, Azizi S, Shojafar M, Tafazolli R (2021) A priority, power and traffic-aware virtual machine placement of iot applications in cloud data centers. J Syst Architect 115:101996
61. Mahmud R, Ramamohanarao K, Buyya R (2020) Application management in fog computing environments: A taxonomy, review and future directions. ACM Comput Surv (CSUR) 53(4):1–43
62. Maríen-Tordera E, Masip-Bruin X, Garcíea-Almiñana J, Jukan A, Ren G-J, Zhu J (2017) Do we all really know what a fog node is? current trends towards an open definition. Comput Commun 109:117–130
63. Bonomi F, Milito R, Natarajan P, Zhu J (2014) Fog computing: a platform for internet of things and analytics. In: Big data and internet of things: A roadmap for smart environments, pp 169–186
64. Calheiros RN, Ranjan R, Buyya R (2011) Virtual machine provisioning based on analytical performance and qos in cloud computing environments. Int Conf Parallel Process 2011:295–304
65. Azizi S, Zandsalimi M, Li D (2020) An energy-efficient algorithm for virtual machine placement optimization in cloud data centers. Clust Comput 23:3421–3434
66. Zadeh LA (1965) Fuzzy sets. Inf Control 8(3):338–353
67. Stankovic JA, Spuri M, Ramamritham K, Buttazzo G (1998) Deadline scheduling for real-time systems: EDF and related algorithms, vol 460. Springer Science & Business Media
68. Xu J, Sun X, Zhang R, Liang H, Duan Q (2020) Fog-cloud task scheduling of energy consumption optimisation with deadline consideration. Int J Internet Manuf Serv 7(4):375–392
69. Cob-Parro AC, Losada-Gutiérrez C, Marrón-Romera M, Gardel-Vicente A, Bravo- Muñoz I (2021) Smart video surveillance system based on edge computing. Sensors 21(9):2958
70. Aazam M, Zeadally S, Harras KA (2018) Deploying fog computing in industrial internet of things and industry 4.0. IEEE Trans Ind Inform 14(10):4674–4682
71. Gia TN, Jiang M. Rahmani AM, Westerlund T, Liljeberg P, Tenhunen H (2015) Fog computing in healthcare internet of things: a case study on ECG feature extraction. In: 2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing. IEEE, pp 356–363
72. Chen S, Wen H, Wu J, Lei W, Hou W, Liu W, Xu A, Jiang Y (2019) Internet of things based smart grids supported by intelligent edge computing. IEEE Access 7:74089–74102

Authors and Affiliations

Rahul Thakur[1] · Geeta Sikka[2] · Urvashi Bansal[1] · Jayant Giri[3,5] · Saurav Mallik[4]

✉ Jayant Giri
jayantpgiri@gmail.com

Rahul Thakur
rahult.cs.21@nitj.ac.in

Geeta Sikka
sikkag@nitdelhi.ac.in

Urvashi Bansal
urvashi@nitj.ac.in; jayantpgiri@gmail.com

Saurav Mallik
sauravmtech2@gmail.com; smallik@hsph.harvard.edu; smallik@arizona.edu

1 Department of Computer Science and Engineering, Dr. B R Ambedkar National Institute of Technology, Jalandhar 144011, Punjab, India

2 Department of Computer Science and Engineering, National Institute of Technology, Delhi, 110036 Delhi, India

3 Department of Mechanical Engineering, Yeshwantrao Chavan College of Engineering, Nagpur, India

4 Department of Environmental Health, Harvard T H Chan School of Public Health, Boston, MA, USA

5 Department of VLSI Microelectronics, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences (SIMATS), Saveetha University, Chennai 602105 TN, India