Received: 15 December 2019 | Revised: 5 April 2020 | Accepted: 23 April 2020

DOI: 10.1002/spe.2867

SPECIAL ISSUE PAPER

WILEY

FPFTS: A joint fuzzy particle swarm optimization mobility-aware approach to fog task scheduling algorithm for Internet of Things devices

Saeed Javanmardi[1] | **Mohammad Shojafar**[2] | **Valerio Persico**[1] | **Antonio Pescapè**[1]

[1]Department of Electrical Engineering and Information Technologies (DIETI), University of Napoli "Federico II", Napoli, Italy

[2]5G Innovation Centre, Institute for Communication Systems, University of Surrey, Guildford, UK

Correspondence
Antonio Pescapé, Department of Electrical Engineering and Information Technologies (DIETI), University of Napoli "Federico II", Napoli, Italy.
Email: pescape@unina.it

Summary

In the Internet of Things (IoT) scenario, the integration with cloud-based solutions is of the utmost importance to address the shortcomings resulting from resource-constrained things that may fall short in terms of processing, storing, and networking capabilities. Fog computing represents a more recent paradigm that leverages the wide-spread geographical distribution of the computing resources and extends the cloud computing paradigm to the edge of the network, thus mitigating the issues affecting latency-sensitive applications and enabling a new breed of applications and services. In this context, efficient and effective resource management is critical, also considering the resource limitations of local fog nodes with respect to centralized clouds. In this article, we present FPFTS, fog task scheduler that takes advantage of particle swarm optimization and fuzzy theory, which leverages observations related to application loop delay and network utilization. We evaluate FPFTS using an IoT-based scenario simulated within iFogSim, by varying number of moving users, fog-device link bandwidth, and latency. Experimental results report that FPFTS compared with first-come first-served (respectively, delay-priority) allows to decrease delay-tolerant application loop delay by 85.79% (respectively, 86.36%), delay sensitive application loop delay by 87.11% (respectively, 86.61%), and network utilization by 80.37% (respectively, 82.09%), on average.

KEYWORDS

fog computing, Internet of Things, Mamdani fuzzy inference system, particle swarm optimization algorithm, resource management, task scheduling

1 | INTRODUCTION

The proliferation of sensors, actuators, and handheld devices connected in a communicating-actuating network has generated the so-called Internet of Things (IoT). In this context, smart objects blend seamlessly with the environment around us, allowing the information to be shared across platforms to develop a common operating picture.[1] To support the requirements of these resource-constrained devices, the integration of IoT with the cloud paradigm is critical.[2,3] Indeed, cloud computing is an enabler of the utmost importance, providing the IoT devices with the means to gather, process, store, and distribute the huge amount of information generated by the IoT ecosystem. On the other hand, the integration

with cloud is not trivial. In fact, its inherent performance and availability limitations[4,5] make cloud unsuitable for specific classes of applications (eg, real-time and Industry 4.0 applications, gaming, medical apps).[6-10] Although cloud providers are deploying more and more capillary infrastructures closer to users' access networks, cloud is proven to be unsuitable for latency-sensitive applications whose performance is impacted by cloud-network infrastructural shortcomings.

In order to mitigate cloud limitations, fog computing has been proposed.[11] This paradigm extends cloud to the edge of the network, with location awareness and low latency infrastructure, also to support mobility. Accordingly, this allows to enable a new breed of streaming and real-time applications and services. Fog paradigm makes available nodes close to users to provide computing resources to run applications or store significant amounts of data.[12] In detail, user-application requests generate a set of tasks that can be run by fog devices. Fog task scheduling defines how to *properly* assign tasks to the fog devices to satisfy the needs of the specific applications or to reduce network utilization.[13] In fact, while fog computing provides significant performance benefits by design, it introduces resource constraints with respect to the cloud paradigm and, therefore, the need for effective and efficient resource management, that may results in even complex and sophisticated scheduling strategies.[14]

1.1 | Goal and contribution

In this article, we propose FPFTS, a fog task scheduler in which we jointly employ *fuzzy logic* in the fitness function of a *particle-swarm optimization (PSO)* algorithm to improve its performance. Due to its suitability for global searching and its guided randomness feature, PSO is expected perform well in dynamic environments such as fog computing. Moreover, it has fewer parameters than either genetic algorithm or swarm intelligence.[15-17] On the other hand, a fuzzy optimization has some unique characteristics that make it an appropriate choice for several control problems and suited for environments where the settings are not precisely defined or previously unknown.[18] By using Mamdani fuzzy rules, we can make a relationship between some parameters by adding priority to them.[19]

This research article presents a new approach to optimize the task scheduling problem in fog computing for both delay-sensitive and delay-tolerant applications. The goal of the proposed approach is to optimally use fog resources such to reduce network utilization and application loop delay. To this end, in our proposal we simultaneously consider the *computing features of the fog nodes*such as CPU processing capacity, RAM size, and bandwidth, together with *the features of the tasks* such as CPU need, as similar approaches proved to be suitable for task scheduling in previous works.[20-22] Moreover, our proposed approach is designed to work with both delay-sensitive and delay-tolerant applications: FPFTS benefits from the information about the class each application belongs to, to refine scheduling decisions in case of fog-layer overloading. We use fuzzy logic in PSO considering the features of resources and tasks to solve fog task scheduling problems without being trapped into a local minimum, and optimally use the fog resources.

The contributions of this article are as follows: (i) We design a new joint metaheuristic method called FPFTS combining PSO and fuzzy methods to tackle the fog task scheduling problem. (ii) We implement and test our method taking into account an IoT-based case study and considering a three-layered fog-computing architecture. (iii) We evaluate FPFTS leveraging *iFogSim*, a widely adopted and well-known fog simulator. To this end, to test our proposal we take into account both delay-tolerant and delay-sensitive applications. We test FPFTS against *first-come first served* (FCFS) and *delay-priority* strategies (which are classic methods in this area) by varying the number of users benefiting from the fog services, as well as the characteristics of the network infrastructure, in terms of the bandwidth and the latency of its links. The obtained results show that our proposal outperforms FCFS and delay-priority approaches in terms of application loop delay and also network utilization. In details, FPFTS improves by ≈86% application loop delay, on average, while concerning network utilization, FPFTS improves this metric in by ≈81%, on average.

The remainder of the article is reported hereinafter. Section 2 presents the related literature on fog scheduling and provides their pros and cons. Section 3 overviews the considered architecture and its main components. In addition, it presents the proposed methodology leveraging PSO and fuzzy algorithms. Section 4 details the evaluation of our proposal with experimental results. In Section 5, we discuss fog task scheduling challenges, which are related to this work. Finally, Section 6 concludes the article with some final remarks and future outlines.

2 | RELATED WORK

In this section, we delineate the existing task scheduling algorithms applied in the fog systems. Fog task scheduling approaches are divided into two categories: *dynamic algorithms* and *static algorithms*. Dynamic strategies calculate the

task priorities during their execution, while static approaches assume to have prior information about fog resources and tasks. Static scheduling algorithms can be further divided into two main families: *heuristic-based* and *metaheuristic-based* algorithms. To reduce the delay, heuristic algorithms may return a suboptimal resource allocation, but still proper to satisfy the users' requests. Metaheuristic algorithms perform a random search to find a good solution to an optimization scheduling problem.[23] In this section, we briefly review the static heuristic-based and metaheuristic-based algorithms.

Concerning heuristic-based strategies, Bittencourt et al[24] design a mobility-aware fog application scheduling framework, which uses application classification and mobility of the clients as the nature of decision making. They implement first come-first served (FCFS), concurrent strategy, and delay-priority as the scheduling algorithms of the framework. Afterward, Mahmud et al[25] introduce a latency-sensitivity application task approach that considers the different application delivery latency for fog applications. Gill et al[15] model a PSO-based resource management approach associating with scheduling which is suitable for smart home IoT device. Zeng et al[26] present a fog-supported resource management framework, which consists of a three-step task scheduling and a task image placement algorithm to minimize the application delay. The authors focus on computation time and task computation. Hoang et al[27] present a task scheduling approach called FBRC, which assigns user tasks to fog-based regions and cloud data centers to minimize the task execution time. The authors try to reduce the application delay by using a heuristic algorithm. By using a heuristic algorithm, the authors try to reduce the application delay.

For what concerns metaheuristic-based strategies, Sun et al[28] present a two-level task scheduling approach which consists of two steps. In the first step, the algorithm finds the most proper fog device cluster, and after that it finds the most proper fog device to execute the applications' tasks. This approach implements an improved nondominated sorting genetic algorithm II (NSGA-II) to reduce the delay, execution time, and also increase scalability. The algorithm designed by Bitam et al[29] implements an optimization algorithm inspired by bees swarm intelligence method, which targets run-time tasks and the allocated memory. Also, the algorithm presented by Luo et al[13] forms a fuzzy load balancing strategy associating with real-time scheduling. Rafique et al[30] describe a hybrid bioinspired algorithm, which is a combination of modified PSO and modified cat swarm optimization (MCSO) named *MPSO*. Their MPSO schedules the tasks among fog devices and manages the availability of resources at the fog device level.

Overview on background methods and comparison with FPFTS. The mentioned works refer to various IoT scenarios. Moreover, two main strategies for realizing collaboration among nodes can be identified:

peer-to-peer (P2P) and *master-slave*.[12] peer-to-peer (P2P) collaboration among the fog devices is very common in fog computing infrastructure. Through P2P collaboration, a fog device can use the processed output of the other fog device. Moreover, fog devices can share virtual computing instances between each other. In master-slave mode, a master fog device controls underlying slave fog devices such as processing load, resource management, data flow, and so on.[12] All of the mentioned methods except the methods presented by Bitam et al,[29] and Luo et al[13] implement P2P as the nodal collaboration strategy between the fog and IoT devices.

Regarding the nature of observations, differently from all of the mentioned works, we analyze the computing features of resources and tasks altogether. Therefore, for FCFS and delay-priority strategies,[24] if fog devices do not have free space capacity, they offload tasks to the cloud data center. They consider availability and CPU capacity of fog node resources for the scheduling. The presented algorithm by Mahmud et al[25] is based on the placement time and the percentage of task deadlines. The presented algorithm by Bitam et al[29] considers two computing criteria, namely, *time* and the *allocated memory* needed by applications for the algorithm decision. The algorithm which is presented by Luo et al[13] considers task arrival time, task deadline time, and task size metrics for their scheduling algorithm. The presented algorithm by Gill et al[15] uses various quality of service criteria such as response time, energy, latency, and network bandwidth. The algorithm presented by Sun et al[28] considers the CPU performance and the battery lifetime, the resource utilization of fog clusters, and also the distance between fog device clusters and users for decision making. The algorithm presented by Zeng et al[26] considers task completion time as the main metric for decision making. Their algorithm aims to minimize the computation latency of user requests. The algorithm presented by Hoang et al[27] considers the maximum delay as the constraint condition to set the upper bound of delay, which increases the complexity of the algorithm. In their work, computation time and transmission time are the main parameters for making a scheduling decision. Finally, the algorithm presented by Rafique et al[30] considers the least loaded device as a criterion of availability of resources for making a decision.

In our proposed approach, differently from the works by Gill et al[15] and Rafique et al[30] we use fuzzy theory in the PSO fitness function. These works use some weights to prioritize the components of the PSO fitness function. In this article—unlike the works by Sun et al,[28] Bitam et al,[29] Zeng et al,[26] and Hoang et al[27]— we use iFogSim for simulations and testing on various metrics. In addition, unlike the works by Mahmud et al,[25] Bitam et al,[29] and Gill et al,[15] FPFTS uses mobility-aware scenario in performance evaluation. The algorithm presented by Sun et al[28] considers the distance

TABLE 1 Comparison of existing scheduling approaches against FPFTS

Refs.	Algorithms	Tool	Scenario	Observations	NC	Adv.	Disadv.
24	FCFS/delay-priority	iFogSim	Mobile IoT	Resources availability and CPU capacity	P2P	Considering movement based scheduling at cloudlet level	No consider fog device processing capacity
25	Heuristic method	iFogSim	Smart home IoT	Deployment time, deadline, #fog node	P2P	Latency-aware IoT application Reducing the amount of deployment time Deals with varying application	No real case study No mobility
29	Bees swarm	C++	General IoT	CPU execution time, allocated memory	M-s	Managing allocated memory Low CPU execution time	Only fog devices are used tatic scheduling
13	Fuzzy-NN	iFogSim	Mobile health-care IoT	Arrival time, deadline time, task size	M-s	Reliable real-time applications	High cost
15	PSO	iFogSim	Smart home IoT	Response time, energy, latency, and network bandwidth	P2P	Optimizes energy, latency, response time, and network bandwidth	No reliability assurance due to a simple fog device
30	Hybrid MPSO	iFogSim	Mobile IoT	Least loaded fog device	P2P	Consider communication latency and computation latency	No network latency
28	NSGA-II	Matlab	Word count IoT	CPU performance Battery lifetime Distance between clusters and users	P2P	Low application delay High scalability	High cost
26	Heuristic-based algorithm	Gurobi tool	Image tasks IoT	Task completion time	P2P	Low computation complexity	High memory consumption
27	Heuristic-based algorithm	NA simulator	General IoT	Computation time Transmission time	P2P	Low latency rate	Low efficiency in service processing rate High time complexity
FPFTS	Hybrid (PSO-Fuzzy)	iFogSim	Mobile IoT	CPU, RAM bandwidth of fog devices and task CPU length	P2P	Mobility-aware scenario Considering computing capacity of resources Low application loop delay/network utilization	Reliability relies on fog gateway fault tolerance

Abbreviations: Adv., advantages; Ctr, centralized; Disadv, disadvantages; M-s, master-slave; NC, nodal collaboration; P2P, peer-to-peer.

between the fog resources and the service requester. They assume considering the principle of proximity effects on application delay. It means that for instance, if a fog device is far away from the requester, the application delay will become high. The demerit of this strategy is that they do not consider link bandwidth. Differently than this work, FPFTS considers link bandwidth, which is a more proper metric than the distance proximity parameter to reduce delay. Differently than the work by Zeng et al,[26] FPFTS has low memory consumption. Differently than the work by Hoang et al,[27] FPFTS has low time complexity and high efficiency in application processing rate. Finally, unlike the works by Bitam et al[29] and Luo et al,[13] in FPFTS, we use the P2P structure for nodal collaboration. Table 1 presents the comparison of existing task scheduling methods compared with our proposed FPFTS strategy in summary.

3 | FPFTS: FUZZY PSO FOG TASK SCHEDULER

In this section, we describe FPFTS. Before going into the details of the proposed scheduler, we discuss the reference architecture and provide the problem statement.

3.1 | Reference architecture

In this article, we refer to a cloud-fog-device structure,[31,32] which consists of three layers. The front-end layer (*device layer*) consists of user devices (such as smartphones, laptops, tablets, etc). They run user applications and may send requests to fog devices. The *fog layer*, is composed of a set of fog devices (ie, servers) located at the edge of the access network and spread across several fog regions. It receives and processes users' requests. Finally, the *cloud layer*, which consists of one or more cloud data centers, provides virtually unlimited resources to execute the tasks, which are offloaded from the fog devices.[33] The architecture is presented in Figure 1. Based on the architecture, there is only one fog gateway in each fog region.

In our architecture, we employ decentralized broker management strategy[34] for task scheduling. The broker is a node which has the information of the fog devices. It is located between the fog devices and the users, and acts like a portal for users applications. FPFTS runs in the broker that may be located at the fog gateway. However, based on different scenarios, the broker can be located in either the cloud gateway or the fog gateway, or any other intermediate location between the devices and the cloud. Changing the location of the broker may impact the performance of the architecture, which in turn depends on network performance. According to our model, each fog region has its own fog gateway, which is responsible to coordinate the fog devices located in fog region.

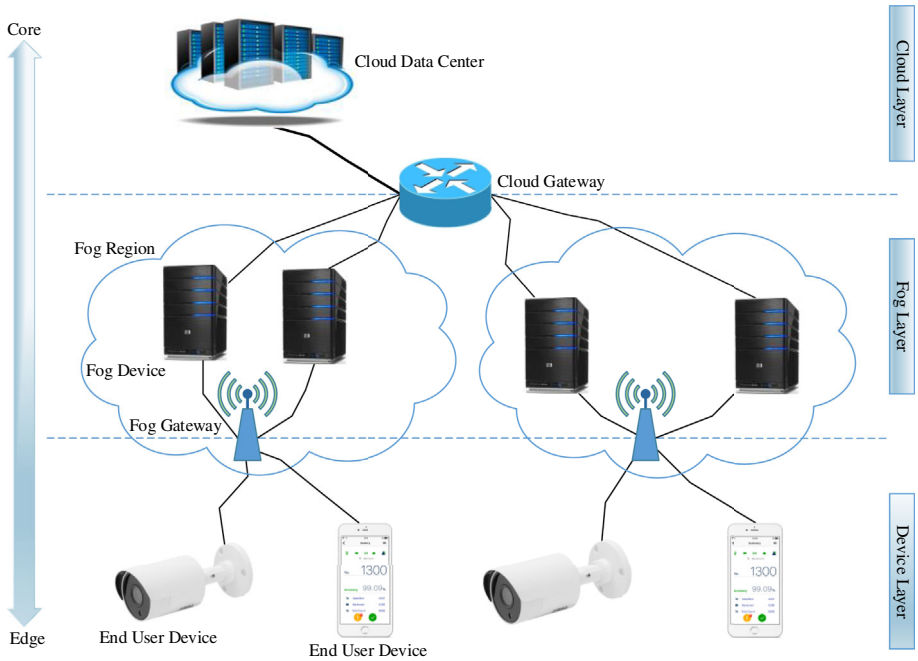


FIGURE 1 Reference architecture [Color figure can be viewed at wileyonlinelibrary.com]

As in practical scenarios the cloud data center is far away from the users' devices, FPFTS first uses fog devices for task scheduling. Only in case of fog-device overloading, FPFTS offloads tasks to the cloud data center by using *offloading agents* (located in fog devices) to overcome the fog-devices computing limitations. We provide additional details about offloading agents in Section 3.3.

3.2 | Problem statement

In this article, we focus on the task scheduling problem which has to be effectively and efficiently addressed to distribute applications' tasks among fog devices. Indeed, task scheduling is a critical part in both cloud and fog resource management and significantly impacts the performance of the overall fog/cloud-based architecture.[34] The aim of task scheduling is assigning tasks generated by users' applications to resources available at the fog layer. Resource management strategies may also implement cloud data offloading so as to overcome resource constraints (eg, limited computational capacity at fog nodes): in case of fog device overloading, cloud resources can be leveraged. In more detail, jobs generated by applications/services are decomposed into a set of atomic tasks to be assigned to fog devices. If the tasks composing a job are mutually independent, they can be executed on separate fog devices, with no need of explicit synchronization.[29] The goal is to find the most adequate fog devices to run each application task so as to have a proper outcome in terms of application loop delay and also network utilization.

3.3 | Proposed scheduler

This subsection describes the presented approach, and the way it assigns applications' tasks to the resources. FPFTS implements a bioinspired approach that uses fuzzy logic in the PSO fitness function[35] for fog task scheduling. When the scheduler assigns a set of applications' tasks, it returns the most adequate fog devices for executing them, based on both the characteristics of the tasks and of the fog devices. In detail, FPFTS uses task CPU need, fog-device processing capacity, fog-device RAM capacity, and fog-device bandwidth as the input parameters of the fuzzy-based fitness function. In other words, FPFTS considers the features of resources and tasks *simultaneously*. This enables FPFTS to achieve a proper distribution of the task across devices such to reduce network utilization and application loop delay.

Fuzzy-based fitness function. Most of the PSO-based schedulers use some predefined weights to prioritize the parameters in the fitness function.[15] Using fuzzy logic is an efficient alternative. We use a Mamdani fuzzy inference system to reap the benefits deriving from fuzzy logic in the prioritization of parameters. Mamdani fuzzy inference engine is one of the common fuzzy inference engines,[36] which uses some fuzzy rules. These fuzzy rules can be based on either former experience or some assumptions.[19,37] Concerning Mamdani fuzzy inference engine, we define some overlapping fuzzy sets—which make the input parameters possibly lie in several sets at the same time. Accordingly, an input parameter can have more than one value with different membership degrees. For example, an input parameter can be defined both as *low* and *medium* with a different membership fitness value (that is a number between 0 and 1 which is obtained via membership fitness functions). As a consequence, the input parameters may trigger multiple fuzzy rules. As each fuzzy rule is associated to a specific priority, it increases the degree of prioritization. For instance, for task CPU length parameter, a value equal to 1000 belongs to low, medium, high fuzzy sets with the membership value 0.8, 0.4, and 0.2 respectively. Figure 2A-D reports fuzzy sets for the fog device processing capacities and RAM and bandwidth and task CPU length parameters, respectively. Figure 2E indicates the fuzzy set for the result parameter.

Table 2 reports the rules we used. Based on the input parameters of the fitness function, the fuzzy inference engine triggers some of them. We refer to them as the *fired rules*. These fuzzy rules are then integrated (creating a fuzzy output graph). Finally, the the fuzzy inference engine defuzzifies this fuzzy output into a numeric value via the centroid method.[37] The obtained value is the output of fitness function, which determines the suitability degree of resources for assigning them to the tasks.

PSO-based fog task scheduling algorithm. In our approach we use the PSO algorithm[38-40] to find a proper solution. Each particle is an instance in the search space, and in this work we consider tasks as particles. To start this algorithm, a population of particles is generated randomly. The position of the particles (fog devices in this work) represents a potential solution. Each particle has a position vector and a velocity vector, which determines the direction of movements in the search space. The search space represents the fog regions, which contain fog devices. We define *pbest* and *gbest* as the best position for each task and the best position for the tasks among group based on the fitness value of all the tasks,

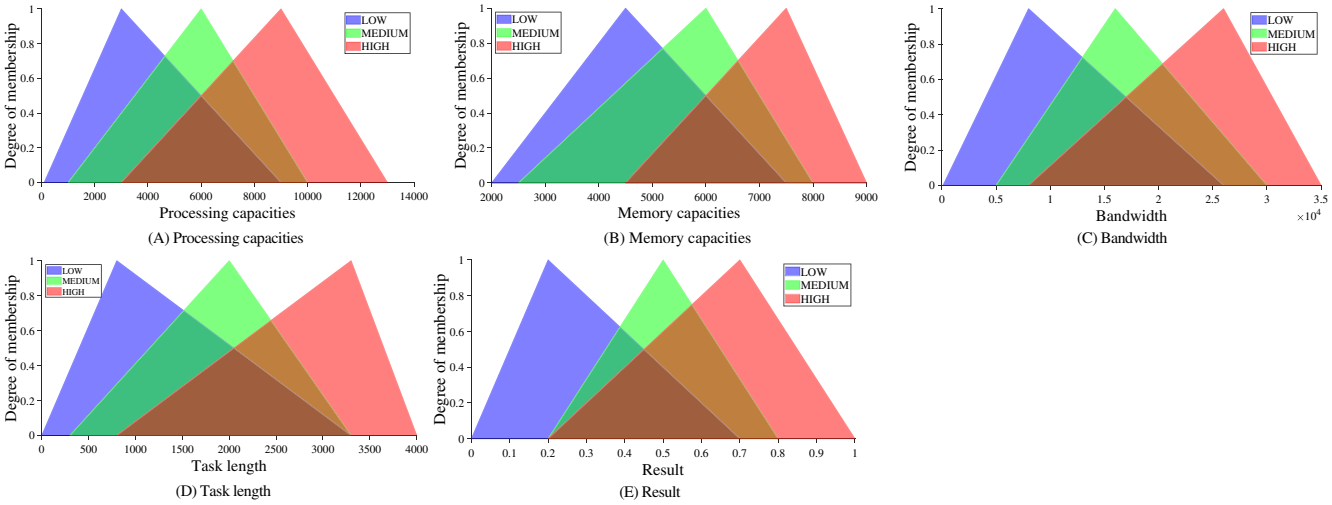


FIGURE 2 Fuzzy set parameters [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 2 Mamdani fuzzy rules

Computing capacity	Memory	Bandwidth	Task length	Result
Low	Low	Low	Low	High
Low	Medium	Medium	Medium	High
Medium	High	Low	High	Medium
Medium	Low	High	Low	Medium
High	Medium	Medium	Medium	High
High	High	Low	High	High
Medium	medium	High	medium	Medium
High	Low	Low	High	Low
High	Medium	Medium	High	Medium

respectively. The PSO algorithm uses Equation (1) for updating position vector, and Equation (2) for updating velocity vector.

$$X_{k+1}^{i} = X_{k}^{i} + V_{k+1}^{i}, \tag{1}$$

$$V_{k}^{i+1} = W_k V_{k}^{i} + c_1 r_1 (\text{pbest}_{k}^{i} - X_{k}^{i}) + C_2 r_2 (\text{gbest} - X_{i}^{k}), \tag{2}$$

C_1 and C_2 are the acceleration coefficients, and PSO considers them as learning factors. Moreover, PSO uses r_1 and r_2 to control the randomness of the particles movements in the search space. In addition, W is inertia weight, which indicates the balance between local and global search. This parameter determines the repeat rate for finding the best solution. In FPFTS number of iterations is equal in value to the number of fog devices so as to keep the computation time low. FPFTS improves the fitness value of the applications' tasks in every generation. It rejects a new solution if its fitness value is less than the current solution. The final result is the best value *gbest*, which is calculated by the fuzzy-based fitness function.

FPFTS Algorithm Algorithm 1 description. In Algorithm 1, first, FPFTS determines the initial positions and velocity for each particle in the search area randomly (lines 1 and 2). We save the current solution in *pbest* (line 3). Then FPFTS computes the fitness function for all particles (line 4), and we save it as the best solution (line 5). Then, after defining the fitness value for all of the particles, FPFTS compares them, and if the fitness value of the current solution is better than *pbest*, it replaces it with the *pbest* (lines 6 till 8). In FPFTS, each task's movement is influenced by its best known position (pbest). In addition, it is guided toward the best known positions (gbest), which are found by other particles. To this end, FPFTS uses Equations (1) and (2). FPFTS updates *gbest* value by comparing the *pbest* value of the current iteration to the value, which is stored in the *gbest*, and if its satisfaction is more than *gbest*, the PSO algorithm replaces it (lines 9 till 24).

We repeat these steps to reach the maximum number of iterations (line 25). Finally, the updated *gbest* value is the final solution and output of this algorithm (line 26).

Algorithm 1. FPFTS: proposed algorithm

INPUT: M, P, v
M: population size
ITE: Iteration number
P: population of particles
v: speed of each particle
OUTPUT: *gbest*

1: **for** $i = 1$ to *ITE* **do**
Initialize $P[i]$ randomly
2: **Initialize** $V[i] = 0$;
3: $pbest[i] = P[i]$
4: $F = ComputeFitness(P[i])$;
5: $gbest =$ best particle found in F;
6: **for** $i = 1$ to M **do**
7: $pbest[i] = F$;
8: **end for**
9: **repeat**
10: **for** $i = 1$ to M **do**
11: $V[i] = W \times V[i] = C_1 \times rand1(pbest[i] - P[i] + C_2 \times rand2(gbest[i] - P[i])$;
12: **update** $V[i]$
13: Set W, $C_1 <= 0$ and $C_2 <= 0$;
14: $P[i] = P[i] + V[i]$;
15: **if** a particle goes outside the predefined hypercube **then**
16: it is reintegrated to its boundaries;
17: **end if**
18: $F = Computefitness(P[i])$;
19: **if** new population is **better then**
20: $pbest[i] = F$;
21: **end if**
22: *gbest*=Best particle found in $P[i]$;
23: **end for**
24: **until** stopping criterion is satisfied
25: **end for**
26: **return** *gbest*

FPFTS process diagram. Figure 3 indicates the process diagram of the presented approach. In the beginning, users (devices such as smartphones or cameras) send their requests (applications) to the nearest broker (fog gateway), which is located in the fog layer and is responsible for task scheduling. FPFTS instances are located in fog gateways, which are responsible to decompose the applications into a set of tasks, schedule the tasks, and assign the tasks to the fog devices inside the fog regions. Fog devices execute the tasks, and in case of fog device overloading, fog devices move tasks to the cloud data center, which is located in the cloud layer (cloud data offloading). Fog devices and cloud data center execute the tasks. They send results to the instances of FPFTS, and these instances send the results back to the users. When a fog device becomes overloaded, two different possible data offloading scenarios can take place: *fog device to fog device data offloading*, and *fog device to cloud data center data offloading*.[41] In this work, we only use fog device to cloud data center data offloading. Task scheduling and task offloading are the major parts of resource management approaches. Task schedulers need data offloading to deal with system failure in case of fog device overloading.[34] In this article, we implement task scheduling unit and data offloading unit, separately. In order to move tasks to the cloud data center, first, *offloading agents* sends delay-tolerant applications to the cloud data center. Then, they offload delay-sensitive applications to the cloud data center.

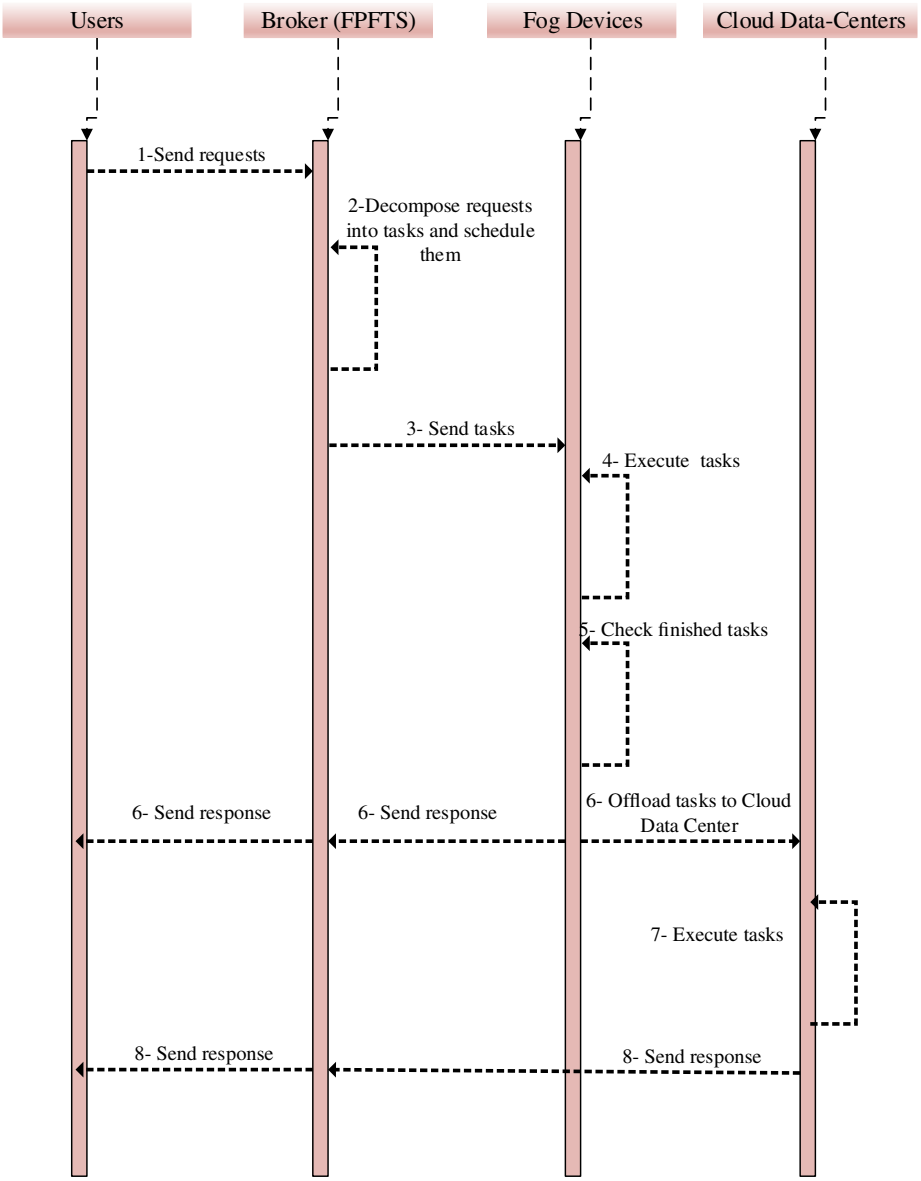


FIGURE 3 The sequence diagram of FPFTS activities [Color figure can be viewed at wileyonlinelibrary.com]

Computational complexity of FPFTS. Focusing on the computational complexity of FPFTS, we refer the readers to Reference 42, which is a mathematical model to illustrate proofs for computational complexity. It indicates the number of resources that we require to run the algorithm. Hence, in FPFTS, we use the complexity of fitness function and PSO separately and then multiply them. To be simplified, the complexity of PSO is in the order of $\mathcal{O}(I \times F \times T)$, where I is the iteration, F is the number of fog nodes, and T is the number of tasks. Also, FPFTS uses Mamdani fuzzy inference engine, so the complexity of the fitness function is of order $\mathcal{O}(1)$.[43] It is because of all the Mamdani fuzzy rules process in a parallel manner. In addition, searching the appropriate Mamdani fuzzy rules performs in a parallel manner, so the complexity of it is in order of $\mathcal{O}(1)$. As a result, the total computational complexity of the FPFTS is in order of $\mathcal{O}(I \times F \times T)$.

Space complexity of FPFTS. For calculating the space complexity we consider the space FPFTS instances require for task scheduling. We name this parameter S, and we consider it (S parameter) in the computational complexity formula. As a result, in general, the total space complexity of the FPFTS is in order of $\mathcal{O}(S \times (I \times F \times T))$. For a small scale of networks, in which there is only one fog region (there is only one instance of FPFTS), S parameter is 1, and as a result, the total space complexity of FPFTS is in order of $\mathcal{O}(I \times F \times T)$. For a large scale of networks, in which there is one instance of FPFTS in each fog region, S parameter is considered equal in value to the number of fog regions.

3.4 | Case study: Smart city

In this subsection, we provide a simple smart-city case study to exemplify the task scheduling strategies. The presence of IoT devices allows the monitoring of different activities of the smart city system. In a smart city, improper task scheduling strategy results in a high delay that is unacceptable to user satisfaction. Using a proper task scheduling strategy, a significant number of smart city requests can be performed by nearby Fog devices, resulting in lower delay and much more user satisfaction.[44] In a smart city, some of the applications are delay-sensitive, while the others are delay-tolerant. For example, a set of smart camera observations data is collected and saved so that we can prepare a database about urban traffic. We consider data saving and retrieval activities as delay-tolerant applications because delay does not harm their results. On the other hand, in some activities like online gaming, fast processing, and low response times are required to generate real-time experience for users. We consider these real-time interactions as delay-sensitive applications, as user satisfaction is impacted by the delay user perceives. As there is no priority in FCFS, this strategy assigns applications to the resources based on the arrival time of the related requests. As this strategy is not oriented to preserve real-timeliness, users may experiment with bad quality of experience in real-time gaming. Suppose some online gaming and smart camera applications arrive at the fog gateway, in this way, priority aware strategies (ie, delay-priority and FPFTS) first consider online gaming in both assigning tasks to the resources and data offloading steps. After that, they consider smart camera applications in both mentioned steps. These strategies enhance online gaming real-time experience.

To give a better insight of various strategies, let T=$\{T_1,T_2,\dots,T_n\}$ and S=$\{S_1,S_2,\dots,S_n\}$ be *two* sets of delay-tolerant and delay-sensitive applications, respectively. We assume T_1 and T_2 arrive at time t_1 and t_2 to the fog gateway, respectively. In addition, we assume S_1 arrives at time t_3 to the fog gateway. Finally, we assume each of the applications has three tasks as follows: $T_1(1)$, $T_1(2)$, $T_1(3)$, $T_2(1)$, $T_2(2)$, $T_2(3)$, $S_1(1)$, $S_1(2)$, $T_1(3)$. FCFS strategy assigns applications' tasks to the nearest fog device to the fog gateway with no priority based on their arrival time. It also does not consider other fog devices in the fog region. Although S_1 is a delay-sensitive application, this strategy does not consider its priority over T_1 and T_2. FCFS, first, schedules T_1 tasks and after that T_2 tasks, and finally it schedules S_1 tasks. Accordingly, based on this strategy, the nearest fog device executes tasks as follows: $T_1(1)$, $T_1(2)$, $T_1(3)$, $T_2(1)$, $T_2(2)$, $T_2(3)$, $S_1(1)$, $S_1(2)$, $T_1(3)$. In the case of fog device overloading, it moves applications to the cloud data center based on their arrival time. Suppose in the middle of $T_2(1)$ execution, fog device overloads. Hence, this strategy moves the rest of the tasks to the cloud data center as follows: $T_2(2)$, $T_2(3)$, $S_1(1)$, $S_1(2)$, $T_1(3)$. Delay-priority strategy schedules applications based on their priority, but it does not consider other fog devices. This strategy schedules T_1 tasks, and after that, it starts to schedule T_2 tasks. Suppose in the middle of T_2 tasks execution, S_1 arrives at the fog gateway, and we require to schedule it. Since it is a delay-sensitive application, fog device halts the execution of the rest of T_2 tasks and starts to execute the arrived delay-sensitive application tasks. After executing S_1 tasks, this strategy continues to execute the rest of T_2 tasks. Hence, in this strategy, the nearest fog device executes tasks as follows: $T_1(1)$, $T_1(2)$, $T_1(3)$, $T_2(1)$, $S_1(1)$, $S_1(2)$, $T_1(3)$, $T_2(2)$, $T_2(3)$. For cloud data offloading, suppose in the middle of $T_2(1)$ execution, fog device overloads; this strategy moves the rest of tasks to the cloud data center as follow: $S_1(1)$, $S_1(2)$, $T_1(3)$, $T_2(2)$, $T_2(3)$. When it comes to the FPFTS, unlike the delay-priority strategy, FPFTS considers all of the fog devices in the same fog region for applications' task scheduling. Like delay-priority strategy, FPFTS considers application priority for both task scheduling and cloud data center data offloading steps. Suppose there are two fog devices in the fog region, FPFTS decomposes applications into a set of tasks as follow: $T_1(1)$, $T_1(2)$, $T_1(3)$, $T_2(1)$, $T_2(2)$, $T_2(3)$, $S_1(1)$, $S_1(2)$, $T_1(3)$. Suppose FPFTS makes the first fog device responsible for executing $T_1(1)$, $T_1(2)$, $T_2(1)$, $S_1(1)$; suppose the first fog device is in the middle of executing $T_1(2)$; at this time FPFTS assigns $T_2(1)$ and $S_1(1)$ to the first fog device. As the priority of $S_1(1)$ is higher than $T_2(1)$, the first fog device execute it before $T_2(1)$. So, the first fog device execute the tasks as follow: $T_1(1)$, $T_1(2)$, $S_1(1)$, $T_2(1)$. Suppose FPFTS makes the second fog device responsible for executing $T_1(3)$, $T_2(2)$, $T_2(3)$, $S_1(2)$, $S_1(3)$; suppose the second fog device is in the middle of executing $T_2(2)$; at this time FPFTS assigns $T_2(3)$, $S_1(2)$ and $S_1(3)$ to the second fog device. As the priority of $S_1(2)$ and $S_1(3)$ are higher than $T_2(3)$, it executes the tasks as follow: $T_1(3)$, $T_2(2)$, $S_1(2)$, $S_1(3)$, $T_2(3)$. Suppose in the middle of executing $S_1(2)$, the second fog device overloads; *Offloading agent* which is located in the second fog device moves $S_1(3)$ and $T_2(3)$ to the cloud data center. Figure 4 illustrates tasks arrival and assignment to fog devices based on FCFS strategy and priority-based strategy. Execution time is the time required by a task for executing a fog device. We assume T_1 comes at time 0, T_2 comes at time 3, and finally S_1 comes at time 9 to the fog gateway.

Tasks	Arrival times (ms)	Execution time (ms)	Priority
$T_1(1)$	0	1	2
$T_1(2)$	0	3	2
$T_1(3)$	0	2	2
$T_2(1)$	3	3	2
$T_2(2)$	3	2	2
$T_2(3)$	3	2	2
$S_1(1)$	9	3	1
$S_1(2)$	9	1	1
$S_1(3)$	9	1	1

FCFS strategy:

$T_1(1)$	$T_1(2)$	$T_1(3)$	$T_2(1)$	$T_2(2)$	$T_2(3)$	$S_1(1)$	$S_1(2)$	$S_1(3)$
0	1	4	6	9	11	13	16	17	18

Priority-based strategy:

$T_1(1)$	$T_1(2)$	$T_1(3)$	$T_2(1)$	$S_1(1)$	$S_1(2)$	$S_1(3)$	$T_2(2)$	$T_2(3)$
0	1	4	6	9	12	13	14	16	18

FIGURE 4 An example of task arrival and assignment to fog devices [Color figure can be viewed at wileyonlinelibrary.com]

4 | PERFORMANCE EVALUATION

In this section, we utilize IoT scenarios to evaluate our proposed algorithm. The following subsections detail the pursued scenarios, methodology, comparing metrics and present the results.

4.1 | Simulation setup

To analyze the performance of the FPFTS algorithm, we run the simulation under the iFogSim testbed.[45,46] In addition, we use JFuzzyLogic[47] for implementing fuzzy-based fitness function of FPFTS. We evaluate the proposed method via a city automation experiment case study[24] in which we reuse delay tolerant and delay sensitive applications. For the former, we use a video application video surveillance/object tracking (VSOT) application, and for the latter, we use a game application (electroencephalography [EEG] tractor beam game) or EEGTBG.

Performance comparison. To evaluate the performance of the proposed FPFTS, we compared it against first come-first served (FCFS) and delay-priority strategies.[24] In detail, FCFS is a classic method, which is located in the second fog device. In FCFS, the system schedules the tasks based on their arrival time till the fog device becomes overloaded. When it becomes overloaded it offloads the applications to the cloud data center. On the other hand, delay-priority strategy (which is located in the second fog device) first schedules the delay sensitive applications. Then, it schedules the delay tolerant applications. In addition, for cloud data offloading, delay sensitive applications we consider more priorities for scheduling.

Mobility scenario. In an urban mobility scenario, inside the smart city model, several mobile users move to other places. These mobile users send delay-sensitive applications to fog resources. Initially, the delay-sensitive applications run on the nearest fog devices. We assume in a city center several smart cameras run the delay-tolerant applications; we also assume a fog gateway is located in the city center. These smart cameras send delay-tolerant applications to the fog gateway. Moreover, mobile users move to the city center and send delay-sensitive applications to the fog gateway. We put the FPFTS scheduler in the fog gateway. FPFTS schedules the tasks of applications that arrive at the city center. Figure 5 presents the mobility scenario.

4.2 | Problem statement

This scenario helps us to study the effects of different scheduling strategies for a mobile scenario. We assume during rush hours, mobile users move toward the city center. First, we consider application instances in the fog devices. Then, we

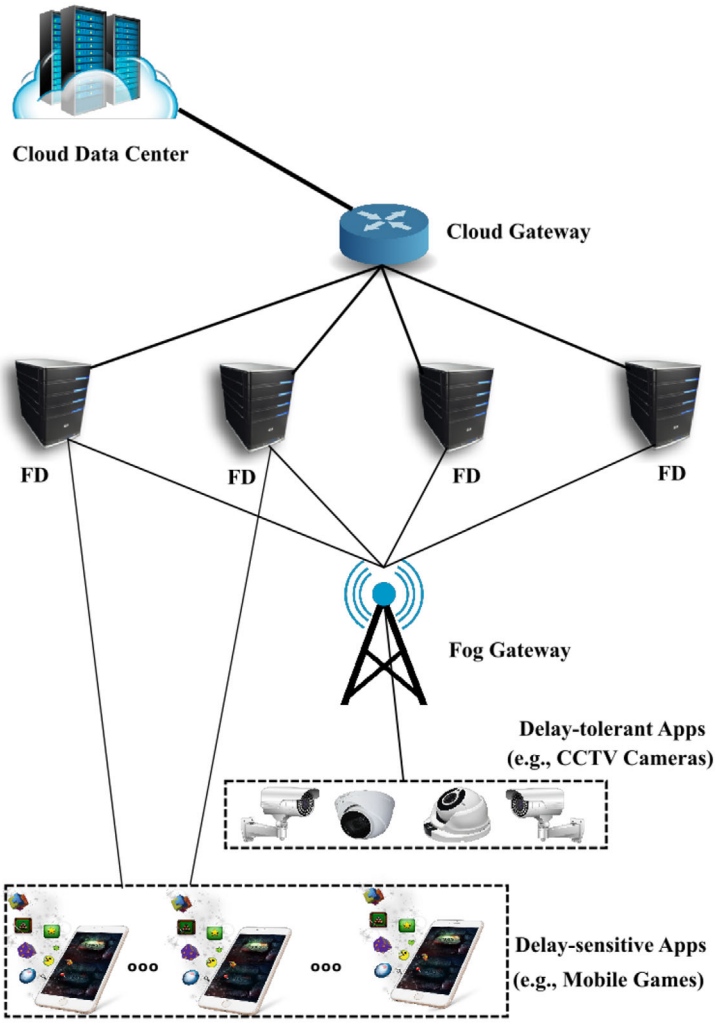


FIGURE 5 Mobility scenario. FD, fog device. It consists of the delay-tolerant applications (eg, apps instantiated on each CCTV device) that are connected to fog gateway and delay-sensitive apps, which are directly connected to FDs [Color figure can be viewed at wileyonlinelibrary.com]

move the users from the fog devices outside the city center to the fog devices inside the city center. Therefore, the fog devices outside the city center become underutilized, while the fog devices inside the city center become overloaded. To overcome this problem, When mobile users arrive at the city center, they connect to the fog gateway and FPFTS assigns the incoming tasks to the fog devices in the entire fog region. In this study, we locate the user interface in the cloud data center, while the client and motion detector modules in mobile devices are located in the device layer. In addition, we place the object detector, object tracker, concentration calculator and coordinator in fog device or cloud data center based on the scheduling strategy. Table 3 presents the features of application modules.

Fog device setup and their communications. In this article, we consider *three* fog devices.[24] According to Reference 24, the first fog device has a 5000 millions of instructions per second (MIPS) computing capacity, 5000 GB RAM, and 9000 Kbps bandwidth. The second fog device has a 4000 MIPS computing capacity, and 4000 GB RAM, and 10 000 Kbps bandwidth. The third fog device has a 5000 MIPS computing capacity, and 5000 GB RAM, and 11 000 Kbps bandwidth. The link latency between the cloud gateway and cloud data center, fog devices and cloud gateway, fog gateway and second fog device, fog gateway to the first/third fog device and devices connected to the first/third fog device and fog gateway are 100, 4, 2, 12, and 2 milliseconds, respectively. Table 4 indicates simulation setup of fog devices.

TABLE 3 Maximum CPU requirements of the application modules in MIPS

	VSOT				EEGTBG		
Module name	OD	MD	OT	UI	Client	CC	CO
Maximum CPU requirements	550	300	300	200	200	350	100

Abbreviations: CC, concentration calculator; CO, coordinator; OD, object detector; MD, motion detector; OT, object tracker; UI, user interface.

TABLE 4 Simulation setup for each fog device and cloud data center

Fog devices	CPU capacity (MIPS)	RAM (GB)	Bandwidth (Kbps)	Latency to cloud gateway (ms)
First	5000	5000	9000	4
Second	4000	4000	10 000	4
Third	5000	5000	11 000	4
Cloud data center	44 800	40 000	10 000	100

We select these fog devises for our experiments due to some reasons. First, we set the CPU and RAM capacity of the first/third fog devices higher than the second device to study the situation in which the second fog device becomes overloaded and the others are getting underutilized. Second, we set the link bandwidth of the first fog device lower than the second one. Also, we set the link bandwidth of the third fog device higher than the second fog device. The reason is that we want to study the effects of different fog devices links bandwidth. It also causes fog devices features to be more different to each other. The benefit of this configuration is to *fire more Mamdani fuzzy rules in PSO fitness function*. Third, the delay of communication between the fog gateway to the first/third fog device and fog devices among each other are about 2 and 12 milliseconds, respectively. It is because of the case by moving from the fog gateway to the first/third fog nodes it requires at least $2 + 4 + 4$ milliseconds, which equals 10 milliseconds link latency. In other words, it takes 2 milliseconds to communicate between fog gateway and second fog device, 4 milliseconds for communication between second fog device and cloud gateway, and finally takes 4 milliseconds to communicate between cloud gateway and second gateway. As these two links are only used in FPFTS, to make the simulation setup comparable and fair, we set 12 milliseconds for these links latency.

4.2.1 | Evaluation metrics

To provide a comprehensive evaluations of our algorithm, we use the following metrics:

- *Application loop delay:* There is a processing loop for running the tasks of each application (here application is a collection of the same task types), and the time for executing the application tasks is called *the application loop delay*. As the application modules are in user devices, fog devices and cloud data center, implementing a proper scheduling strategy reduces this metric. Link latency, link bandwidth, and also computing abilities of the resources impact on the application loop delay.
- *Network utilization:* It indicates total amount of data, which are transmitted in the links between network nodes. For calculating this parameter, we use network latency between the devices that are the origin and the destination of the requests multiplied by the size of the tasks. *Task size* is the file size (in byte) of the task. Each task is characterized by two attributes as follow: first, processing requirements which is defined as million instructions (MI); second, the length of data encapsulated in the task. Task size is the length of data encapsulated in each task. The network utilization is obtained as the sum of the network usage generated by the requests which is sent during the simulation time. Equation (3) illustrates how iFogSim calculates network utilization metric.

$$\text{Network utilization} = \sum_{x=1}^{\text{Requests}} (\text{Latency (milliseconds)} * \text{Task size (byte)})/\text{Simulation time (milliseconds)}. \quad (3)$$

4.3 | Results

In this section, we test our presented algorithm against delay-priority and FCFS methods over various moved users, variant of the bandwidth, and various link latency.

4.3.1 | Comparing methods based on number of users moved

This experiment indicates the mobility-aware scenario of performance evaluation for various number of moved users (IoT tier) in the network. We assume in the second fog device, there are four delay-tolerant applications. We also assume 10

mobile delay-sensitive applications running on the first fog device and 10 mobile delay-sensitive applications running on the third fog device, move one by one to the second fog device.[24]

Loop delay vs number of users moved: Figure 6A,B indicates VSOT and EEGBTG applications loop delay for the FPFTS approach compared task scheduling strategies based on the number of users moved from the first/third fog device with the second fog device, respectively. Both FCFS and delay-priority implement a module merging mechanism in which all of the same kinds of modules are scheduled in the same device. It means they cannot schedule the same kind of modules on different devices. From the arrival of the second EEGBTG player till the 11th, delay-priority has the highest delay in the VSOT loop, while it has a low delay in EEGTBG loop. On the contrary, FCFS has the highest delay in the EEGBTG loop, while it has a low delay in the VSOT loop. When 12th EEGBTG player arrives, because second fog device does not have enough processing capacity, the delay-priority strategy moves EEGTBG modules to the cloud data center and keeps the VSOT modules in the second fog device. The reason for this behavior is that delay-priority schedules all the same kind modules in a specific device. It cannot keep some of the same kind of modules in the second fog device and move the rest of them to the cloud data center. At this point, both FCFS and delay-priority have the same outcomes for both applications. From the arrival of 13th EEGBTG player till the last one, both FCFS and delay-priority have almost the same results. Both these strategies, move the EEGBTG modules to the cloud data center, while they keep the VSOT modules in the second fog device. As the bandwidth of the second fog device links is low for moving this amount of data to the cloud data center, both applications loop are increased dramatically. We study the effects of bandwidth on application loop delay in the next experiments. Despite the fact that from the arrival of 13th EEGBTG player to the last one, both the strategies keep object tracer and object detector VSOT modules in the second fog device. Because *user interface* VSOT modules are located in cloud data center, it increases the VSOT application loop delay. When it comes to FPFTS, as it uses the features of resources and tasks, it has good results in both application loops. Error bar is a representation of the variability of data and used on graphs to show the uncertainty in a reported measurement. As FPFTS uses the randomness features of the PSO algorithm, we run the FPFTS on average 10 times, for several users moved variable values from 0 to 20. We observed from the arrival of 10th EEGBTG player to the last one, it moves modules to the cloud data center, on average, about 10% of simulator running, which causes having a more standard deviation. To put it another way, because of the randomness features of PSO, for each number of users moved, FPFTS requires a simulator to run in different numbers to offload modules to the cloud data center. On average, it takes one time cloud data offloading process per 10 times running the simulator.

Network utilization vs number of users moved: In Figure 7, we present network utilization among the scheduling algorithms. As FPFTS has the least amount of data transmitted to the cloud data center, it has a reasonable outcome for network utilization. In other words, FPFTS transmits more data in the second layer (fog layer) rather than the first layer (cloud layer), decreasing the network utilization. Focusing on FCFS and delay-priority methods, these approaches could schedule tasks either in the second fog device or cloud data center, without considering the another fog device, hence, they increase network utilization. Also, when number of moved users ≥ 2, FCFS algorithm moves the EEGBTG modules to the cloud data center. Hence, it increases the network utilization gradually. Moreover, from the arrival of the third

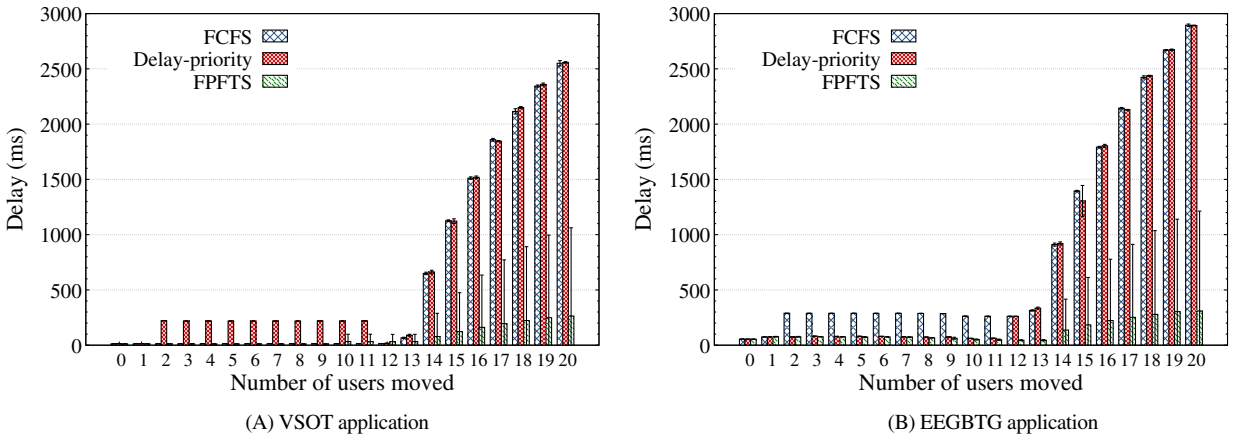


FIGURE 6 Application loop delay in milliseconds for, A, VSOT applications and, B, EEGBTG applications for various number of moved users among FCFS, delay-priority, and FPFTS algorithms [Color figure can be viewed at wileyonlinelibrary.com]

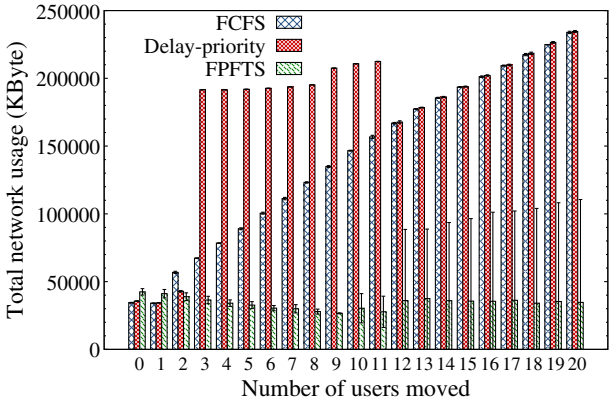


FIGURE 7 Network utilization in KBytes for various number of moved users among FCFS, delay-priority, and FPFTS algorithms [Color figure can be viewed at wileyonlinelibrary.com]

EEGBTG player to the 11th moved user, delay-priority strategy has a high total network usage. The reason is that it moves the VSOT modules to the cloud data center and raises the network usage in this algorithm. Finally, from the arrival of the 12th EEGBTG user to the last one (20th moved user), it keeps *objecttracker* and *objectdetector* VSOT modules in the second fog device, it has the same results as FCFS.

Precisely, when the second EEGBTG application arrives, the delay-priority method moves four *objecttracker* of VSOT application modules to the cloud data center. Then, by entering the third VSOT application until the 11th one, it maintains four *objecttracker* and four *objectdetector* VSOT application modules in cloud data center. Moreover, by arriving at the ninth EEGTBG application until the 11th one, because of the limitation in second device computing capacity, schedules the *concentrationcalculator* modules in the second fog device, and offloads *coordinator* modules to the cloud data center. It results from having around 34 330 and 42 820 KBytes network utilization by arriving the first and the second one respectively, and around 191 600 KBytes network utilization by arriving the third one till the eighth one. In addition, the network utilization by arriving the ninth one till 11th one is around 210 700 KBytes.

4.3.2 | Comparing methods based on various bandwidth ranges

This experiment indicates the mobility-aware scenario of performance evaluation for various fog bandwidth capacities. Bandwidth indicates the maximum data transfer rate of network links. It shows how much data can be sent over the links.

Application loop delay vs various bandwidth ranges: In this part, we discuss the effects of bandwidth on application loop delay, which we detailed in the following experiment. In this experiment, we move 20 users from the first and third fog device to the second one. Figure 8A,B indicates how bandwidth effects on both VSOT and EEGBTG applications

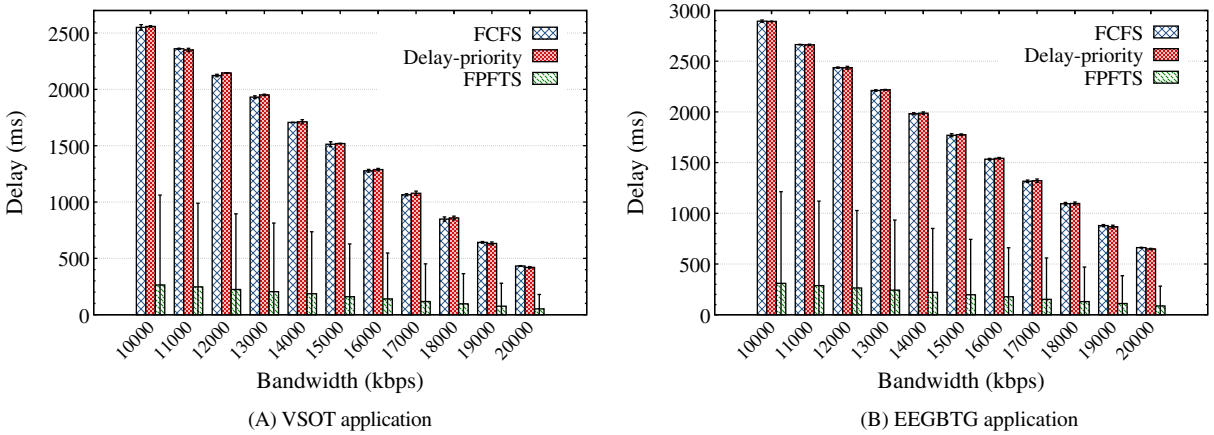


FIGURE 8 Application loop delay in milliseconds for, A, VSOT applications and, B, EEGBTG applications for various bandwidth ranges in Kbps among delay-priority, FCFS, and FPFTS algorithms [Color figure can be viewed at wileyonlinelibrary.com]

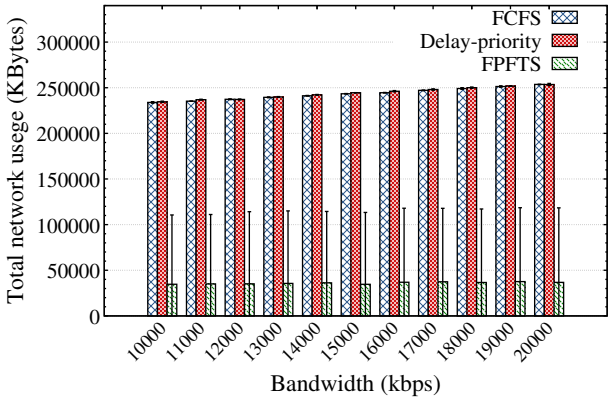


FIGURE 9 Total network usage in KByte for various bandwidth ranges in Kbps among delay-priority, FCFS, and FPFTS algorithms [Color figure can be viewed at wileyonlinelibrary.com]

loop delay, respectively. By changing the second fog device link bandwidth from 10 000 to 20 000 Kbps, both VSOT and EEGBTG applications loop delay in both FCFS and delay-priority strategies decrease significantly. The reason is that 19 modules of *concentrationcalculator* and 19 modules of *connector*(*Coordinator*) are offloaded to the cloud data center. Thus, increasing the bandwidth causes to decrease the amount of time for moving them to the cloud data center. When it comes to FPFTS, we can observe that the effects of bandwidth on FPFTS are much lower than other methods. As FPFTS uses all fog devices, and only in case of fog device overloading it moves modules to the cloud data center, link bandwidth has the lowest effects on FPFTS. It experienced a modest decrease in both VOST and EEGTBG loop delay.

Network utilization vs various bandwidth ranges: Network utilization indicates how much bandwidth is used in a specific time period. Thus, changing link bandwidth has a slight effect on network utilization. Figure 9 indicates how bandwidth variances influence on network utilization. From this figure we conclude that there is a moderate rise for both FCFS and delay-priority strategies, while there is a very slight rise for FPFTS. We run the FPFTS on average nine times for variable bandwidth values from 10 000 to 20 000 Kbps. Based on our observations, FPFTS could offload modules to the cloud data center on average 11% of simulator running, which causes having a more standard deviation. To put it another way, because of the randomness features of PSO, for each bandwidth value, FPFTS requires a simulator to run in different numbers to offload modules to the cloud data center.

4.3.3 | Comparing methods based on various link latency ranges

This experiment indicates the mobility-aware scenario of performance evaluation for various link latency thresholds. Link latency shows the total amount of time it takes to send data.

Application loop delay vs various link latency ranges: In this experiment, we consider a situation in which 20 users moved from the first and third fog device to the second one. This experiment aims to study the effects of changing fog device link latency on application loop delay. In addition, the second fog device link bandwidth is 20 000 KByte. In this way, Figure 10A,B illustrates the effects of link latency on both VSOT and EEGBTG applications loop delay, respectively. By arriving 20 users, both FCFS and delay-priority keep VSOT applications in the second fog device, and they only move EEGTBG modules to the cloud data center. Considering this experiment, we realize that changing link latency only affects on EEGTBG loop delay. The effect of a link latency from 4 to 10 milliseconds on the EEGTBG loop delay is almost negligible, while one from 10 to 50 milliseconds has almost a considerable effect. Moreover, a link latency ranging from 50 to 100 milliseconds, has a more considerable effect on the EEGTBG loop delay, which then sees almost a sharp increase. During this period (see link latency 4 to 100 milliseconds in the figures), the VSOT application loop delay remains constant. However, as FPFTS offloads modules to the cloud data center in case of fog device overloading, changing latency has negligible effects on both VSOT and EEGTBG loops delay for the proposed approach.

Network utilization vs various link latency ranges: Link latency has a considerable effect on network utilization. In Figure 11, we evaluate this matter among the scheduling algorithms. In this figure, by arriving the 20th EEGTBG user, both FCFS and delay-priority algorithms move EEGBTG modules to the cloud data center, so considerably raises the network utilization. On the contrary, as FPFTS has the lowest amount of cloud data offloading, changing link latency has the smallest effect on it, and it increases network utilization marginally. We run FPFTS on average 16 times for variable link latency values from 4 to 100 milliseconds. As a result, we observe that FPFTS offloads modules to the cloud data center

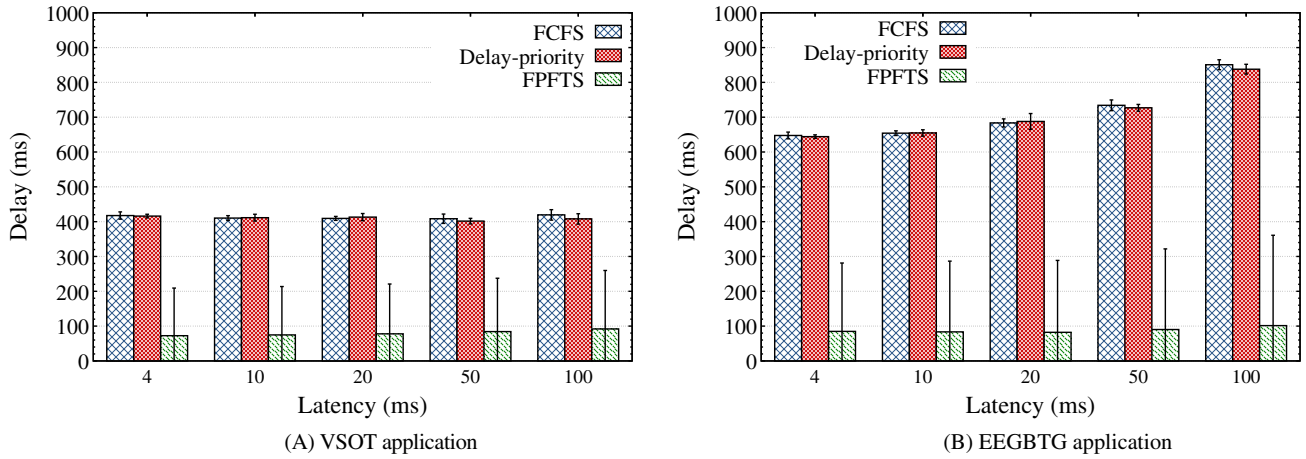
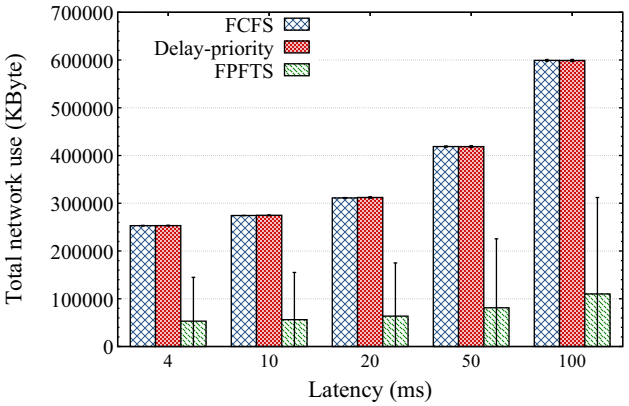


FIGURE 10 Application loop delay in milliseconds for, A, VSOT applications and, B, EEGBTG applications for various latency ranges in milliseconds among delay-priority, FCFS, and FPFTS algorithms [Color figure can be viewed at wileyonlinelibrary.com]

FIGURE 11 Total network usage in KByte for various latency ranges in milliseconds among delay-priority, FCFS, and FPFTS algorithms [Color figure can be viewed at wileyonlinelibrary.com]



on average, 12.5% of simulator running, which causes having a more standard deviation. It means for 16 times simulator running, FPFTS offloads modules to the cloud data center two times. These twice cloud data offloading processing selects a high application loop delay compared with the 14 times FPFTS used fog devices. As there is a huge difference between them, standard deviation of the average of them are high. To put it another way, because of the randomness features of PSO, for each link latency value, FPFTS requires simulator to run in different numbers to offload modules to the cloud data center. On average it takes twice Cloud data offloading per 16 times simulator running.

5 | DISCUSSION

The major reason for developing fog computing is to reduce delay, network utilization, and amount of data transferred to the cloud data center for processing. Resource management plays an essential role in achieving these goals. The process of scheduling and allocating fog resources to users' applications is called *resource management*. Adopting proper task scheduling is the main challenge of fog computing, which causes improving efficiency and also user satisfaction. In this work, we consider application classification and user mobility features for devising the proposed fog task scheduler. Based on the scenario that we used, we only consider two types of user applications and scheduled them to run users' requests. We argue that the task scheduler should strike a balance between the mentioned two types of applications. It is of interest we indicate that FPFTS can use in different scenarios by considering different policies. We can place some instances of the FPFTS method to locally manage the service requests for various applications, which can enhance abilities such as computing capacity, reliability, and availability in the network. In this article, we paid close attention to the user mobility, and the designed algorithm mainly addresses the task scheduling on moving applications. The extended case can consider

local scheduling. FCFS and delay-priority methods do not consider the computing capacities of the other fog devices, and the difference is that FCFS does not consider application classification while delay-priority does. We map and run the instances of applications, which arrived at the first and third fog device, respectively. We put one instance of FPFTS in the fog gateway, which is close to the second fog device, and schedules the tasks to all fog devices to use the computing capacities of fog layer resources in a fog region. Only in case of fog device overloading, *Offloading agents* moves data to the cloud data center. Although we used a small scale network scenario in the experiments, we argue that we can use FPFTS in large scale networks with the aid of peer-to-peer (P2P) technology. P2P technology could utilize distributed systems like Web Operating Systems,[48] Semantic P2P Grids,[43] Cloud environment,[49] and also fog computing,[50] and Internet of Things environment.[51] It has two major benefits. First, FPFTS instances can collaborate to assign tasks to the fog devices in the other fog regions. Second, in the case of fog device overloading, we can use a fog device to data offloading either in the same fog region or in the different fog regions.

In this work, we only consider fog device to cloud data center data offloading. The reason is that the resource capacities of the fog devices are much lower than the cloud data center, and the approach requires checking the free space of the other fog devices. It needs some suitable strategies indeed, which causes the proposed method more complicated.[41] Moreover, VM live migration is another good approach to overcome the computational limitation of the fog devices to improve the load balancing of the network. Because it consumes a large number of CPU cycles and network bandwidth, it is considered as a resource-intensive strategy. Even though it brings some advantages in terms of load balancing, transparent mobility, proactive fault tolerance, and green computing, it requires sufficient network bandwidth capacity to move the virtual machines from one fog device to another.[52] If we consider fog device to fog device data offloading, and VM live migration to the FPFTS, it will be changed as follow: first, the scheduler should be aware of the real-time changes in other fog devices and the links. It includes the changes in the available computing capacities, links latency and the availability of links bandwidth. The instances of the scheduler in different virtual organizations should collaborate to find the best fog device for fog device to fog device data offloading/VM live migration. Second, the instances of the scheduler should decide which fog device is the most suitable target for data offloading/VM live migration. This decision is made by the availability degree of the fog devices computing capacity, link latency, and link bandwidth.

The IoT is the foundation of so many states of the art technologies. Adopting a better resource management strategy in relevant technologies increases their performance and user satisfaction. For instance, the Industrial Internet of Things (IIoT)[53] presents novel applications for powerful industrial systems. It is the implementation of IoT technology in manufacturing systems. IoT enables interaction and cooperation of physical objects such as sensors, machines, cars, buildings, and other items to reach specific goals in the industry. Fog computing provides near user processing capability, which has a low delay to actuators, sensors, and robots in the manufacturing industry. In addition, as most of the time, industrial big data are unstructured, near user resources perform the process they need before sending it to the cloud data center.[54] As IIoT machines must connect and communicate in a real-time manner, we can use a proper fog task scheduling approach to reduce application delays. To take another example, we can point to smart cities.[55] The smart city uses different types of Internet of Things devices to collect data and then use them to manage the resources and services efficiently. To do so, it requires related skills and the capacity to make the users satisfied. In IoT based smart cities, the devices (street cameras for observation street traffic, sensors for urban transportation systems, etc) interconnect and communicate with each other through a network infrastructure.[56] *Smart surveillance'* is another usage of smart cities that provides the ability to monitor human activities. The strength of real-time video analysis in surveillance applications is the necessity of this goal.[57] To this end, adopting a proper fog task scheduling approach decreases the network utilization and application loop delay. Cognitive IoT (CIoT) is another example of the usage of IoT networks in relevant cutting-edge technologies. Nowadays, so many health care, agriculture, environment monitoring, and smart metering scenarios are implemented by using cognitive networks (CNs).[58] The CIoT is the combination of current IoT and cognitive and cooperative mechanisms to achieve intelligence. Intelligence sensing is a novel research field. We implement the intelligence sensing information to make people able to contribute data samples for CIoT captured by sensors, which often performs by smartphones. As in CIoT, sensing information shortages/absences results in loss of human life and social unrest, it needs a sufficient fog resource management approach.[59,60]

6 | CONCLUSIONS AND FUTURE DIRECTIONS

In this article, we proposed FPFTS, a mobility-aware fog task scheduler designed to efficiently assign IoT user's tasks to fog devices. FPFTS considers the computing capacity of the resources as well as user's tasks requirements as input parameters

and implements an algorithm jointly based on PSO and fuzzy theory to assign applications' tasks to fog resources. We apply FPFTS within an IoT-based scenario. To evaluate the proposed approach, we assess the performance of the FPFTS using the iFogSim simulator, considering different numbers of moved users and several configurations for fog-device link bandwidth and link latency. Results show that FPFTS outperforms both FCFS and delay-priority algorithm. Compared against the former, FPFTS improves by 85.79% (respectively, 87.11%) delay-tolerant (respectively, delay-sensitive) application loop delay and network utilization by 80.37%. On the other hand, FPFTS improves the performance of delay-priority algorithm, in terms of delay-tolerant application loop delay by 86.36%, delay-sensitive application loop delay by 86.61%, and network utilization by 82.09%.

In the future, we would like to adapt FPFTS to meet the requirements of 5G mobile edge computing. Moreover, in each fog device, we can design a mobility-aware task scheduling algorithm by modifying the related fitness function. As fog computing can be used in 5G mobile edge computing like Reference 61, we want to modify FPFTS and set priority in Mamdani fuzzy rules to decide which tasks should be offloaded to the cloud data center. In addition, we want to consider fog device to fog device data offloading and VM live migration for 5G mobile edge computing resource management.

ORCID

Mohammad Shojafar https://orcid.org/0000-0003-3284-5086
Valerio Persico https://orcid.org/0000-0002-7477-1452

REFERENCES

1. Mass J, Chang C, Srirama SN. Edge process management: a case study on adaptive task scheduling in mobile IoT. *Internet of Things*. 2019;6:100051.
2. Botta A, De Donato W, Persico V, Pescapé A. Integration of cloud computing and internet of things: a survey. *Futur Gener Comput Syst*. 2016;56:684-700.
3. Abbasi AA, Abbasi A, Shamshirband S, Chronopoulos AT, Persico V, Pescapè A. Software-defined cloud computing: a systematic review on latest trends and developments. *IEEE Access*. 2019;7:93294-93314.
4. Persico V, Marchetta P, Botta A, Pescapè A. Measuring network throughput in the cloud: the case of Amazon EC2. *Comput Netw*. 2015;93:408-422.
5. Palumbo F, Aceto G, Botta A, Ciuonzo D, Persico V, Pescapé A. Characterizing cloud-to-user latency as perceived by AWS and azure users spread over the globe, 2019. IEEE Global Communications Conference (GLOBECOM); 2019. https://doi/org/10.1109/GLOBECOM38437.2019.9013343.
6. Aceto G, Persico V, Pescapé A. A survey on information and communication technologies for industry 4.0: state-of-the-art, taxonomies, perspectives, and challenges. *IEEE Commun Surv Tutor*. 2019;21(4):3467-3501.
7. Choy S, Wong B, Simon G, Rosenberg C. *The Brewing Storm in Cloud Gaming: A Measurement Study on Cloud to End-User Latency*. Venice, Italy: IEEE; 2012:1-6.
8. Dainotti A, Pescape A, Ventre G. *A Packet-Level Traffic Model of Starcraft*. San Diego, CA, USA: IEEE; 2005:33-42.
9. Aceto G, Persico V, Pescapé A. The role of Information and Communication Technologies in healthcare: taxonomies, perspectives, and challenges. *J Netw Comput Appl*. 2018;107:125-154.
10. Aceto G, Persico V, Pescapé A. Industry 4.0 and health: internet of things, big data, and cloud computing for healthcare 4.0. *J Ind Inf Integr*. 2020;18:100129. https://doi.org/10.1016/j.jii.2020.100129.
11. Bonomi F, Milito R, Zhu J, Addepalli S. *Fog Computing and Its Role in the Internet of Things*. ACM; 2012:13-16.
12. Mahmud R, Kotagiri R, Buyya R. Fog Computing: A Taxonomy, Survey and Future Directions. In: Di Martino B, Li KC, Yang L, Esposito A, (eds). *Internet of Everything. Internet of Things (Technology, Communications and Computing)*. Singapore: Springer; 2018.
13. Talaat FM, Ali SH, Saleh AI, Ali HA. Effective load balancing strategy (elbs) for real-time fog computing environment using fuzzy and probabilistic neural networks. *J Netw Syst Manag*. 2019;27(4):883-929.
14. Puliafito C, Mingozzi E, Longo F, Puliafito A, Rana O. Fog computing for the internet of things: a survey. *ACM Trans Internet Technol (TOIT)*. 2019;19(2):18.
15. Gill SS, Garraghan P, Buyya R. ROUTER: fog enabled cloud based intelligent resource management approach for smart home IoT devices. *J Syst Softw*. 2019;154:125-138.
16. Butt AA, Khan S, Ashfaq T, Javaid S, Abdul SN, Nadeem J. *A Cloud and Fog Based Architecture for Energy Management of Smart City by Using Meta-Heuristic Techniques*. Tangier, Morocco: IEEE; 2019:1588-1593.
17. Le Le T, Nguyen H, Dou J, Zhou J. A comparative study of PSO-ANN, GA-ANN, ICA-ANN, and ABC-ANN in estimating the heating load of buildings' energy efficiency for smart city planning. *Appl Sci*. 2019;9(13):2630. https://doi.org/10.3390/app9132630.
18. Benblidia MA, Brik B, Merghem-Boulahia L, Esseghir M. *Ranking Fog Nodes for Tasks Scheduling in Fog-Cloud Environments: A Fuzzy Logic Approach*. Tangier, Morocco: IEEE; 2019:1451-1457.
19. Pourjavad E, Mayorga RV. A comparative study and measuring performance of manufacturing systems with Mamdani fuzzy inference system. *J Intell Manuf*. 2019;30(3):1085-1097.

20. Shojafar M, Javanmardi S, Abolfazli S, Cordeschi N. FUGE: a joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. *Clust Comput*. 2015;18(2):829-844.
21. Javanmardi S, Shojafar M, Amendola D, Cordeschi N, Liu H, Abraham A. *Hybrid Job Scheduling Algorithm for Cloud Computing Environment*. Switzerland: Springer; 2014:43-52.
22. Ben Alla H, Ben Alla S, Ezzati A, Mouhsen A. *A novel architecture with dynamic queues based on fuzzy logic and particle swarm optimization algorithm for task scheduling in cloud computing*. In: El-Azouzi R, Menasche D, Sabir E, De Pellegrini F, Benjillali M, (eds). Advances in Ubiquitous Networking 2. UNet 2016. Lecture Notes in Electrical Engineering, vol 397. Singapore: Springer; 2017.
23. Hosseinioun P, Kheirabadi M, Kamel Tabbakh SR, Ghaemi R. aTask scheduling approaches in fog computing: A survey. *Transactions on Emerging Telecommunications Technologies*. 2020. http://dx.doi.org/10.1002/ett.3792.
24. Bittencourt LF, Diaz-Montes J, Buyya R, Rana OF, Parashar M. Mobility-aware application scheduling in fog computing. *IEEE Cloud Comput*. 2017;4(2):26-35.
25. Mahmud R, Ramamohanarao K, Buyya R. Latency-aware application module management for fog computing environments. *ACM Trans Internet Technol (TOIT)*. 2019;19(1):9.
26. Zeng D, Gu L, Guo S, Cheng Z, Yu S. Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. *IEEE Trans Comput*. 2016;65(12):3702-3712.
27. Hoang D, Dang TD. *FBRC: Optimization of Task Scheduling in Fog-Based Region and Cloud*. Sydney, NSW, Australia: IEEE; 2017:1109-1114.
28. Sun Y, Lin F, Xu H. Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II. *Wirel Pers Commun*. 2018;102(2):1369-1385.
29. Bitam S, Zeadally S, Mellouk A. Fog computing job scheduling optimization based on bees swarm. *Enterpr Inf Syst*. 2018;12(4):373-397.
30. Rafique H, Shah MA, Islam SU, Maqsood T, Khan S, Maple C. A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing. *IEEE Access*. 2019;7:115760-115773.
31. Hu P, Dhelim S, Ning H, Qiu T. Survey on fog computing: architecture, key technologies, applications and open issues. *J Netw Comput Appl*. 2017;98:27-42.
32. Naranjo PGV, Pooranian Z, Shojafar M, Conti M, Buyya R. FOCAN: a fog-supported smart city network architecture for management of applications in the internet of everything environments. *J Parall Distrib Comput*. 2019;132:274-283.
33. Dastjerdi AV, Gupta H, Calheiros RN, Ghosh SK, Buyya R. *Fog Computing: Principles, Architectures, and Applications*. Elsevier; 2016:61-75. https://doi.org/10.1016/B978-0-12-805395-9.00004-6.
34. Ghobaei-Arani M, Souri A, Rahmanian AA. Resource management approaches in fog computing: a comprehensive review. *J Grid Comput*. 2020;18:1-42.https://doi.org/10.1007/s10723-019-09491-1.
35. Aburukba RO, AliKarrar M, Landolsi T, El-Fakih K. Scheduling internet of things requests to minimize latency in hybrid fog–cloud computing. *Futur Gener Comput Syst*. 2020.111:539–551.
36. Zadeh LA. Fuzzy logic–a personal perspective. *Fuzzy Sets Syst*. 2015;281:4-20.
37. Kosko B. Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence. No. QA76. 76. E95 K861992.
38. Sidhu MS, Thulasiraman P, Thulasiram RK. *A Load-Rebalance PSO Heuristic for Task Matching in Heterogeneous Computing Systems*. Singapore: IEEE; 2013:180-187.
39. Abdi S, Motamedi SA, Sharifian S. *Task Scheduling using Modified PSO Algorithm in Cloud Computing Environment*. International Conference on Machine Learning, Electrical and Mechanical Engineering (ICMLEME'2014) Jan. 8-9, 2014 Dubai (UAE); 2014:8-9.
40. Alkhashai HM, Omara FA. BF-PSO-TS: hybrid heuristic algorithms for optimizing task schedulingon cloud computing environment. *Int J Adv Comput Sci Appl*. 2016;7(6):207-212.
41. Aazam M, Zeadally S, Harras KA. Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities. *Futur Gener Comput Syst*. 2018;87:278-289.
42. Arora S, Barak B. *Computational Complexity: A Modern Approach*. Cambridge, MA: Cambridge University Press; 2009.
43. Javanmardi S, Shojafar M, Shariatmadari S, Ahrabi SS. FRTRUST: a fuzzy reputation based model for trust management in semantic P2P grids; 2014. arXiv preprint arXiv:1404.2632.
44. Deng Y, Chen Z, Yao X, Hassan S, Wu J. Task scheduling for smart city applications based on multi-server mobile edge computing. *IEEE Access*. 2019;7:14410-14421.
45. Mahmud R, Buyya R. Modelling and simulation of fog and edge computing environments using iFogSim toolkit. *Fog and Edge Computing: Principles and Paradigms*; 2019:1-35. https://doi.org/10.1002/9781119525080.ch17.
46. Gupta H, Vahid DA, Ghosh SK, Buyya R. iFogSim: a toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments. *Softw Pract Exp*. 2017;47(9):1275-1296.
47. Cingolani P, Alcalá-Fdez J. jFuzzyLogic: a java library to design fuzzy logic controllers according to the standard for fuzzy control programming. *Int J Comput Intell Syst*. 2013;6(sup1):61-75.
48. Javanmardi S, Shojafar M, Shariatmadari S, Abawajy JH, Singhal M. PGSW-OS: a novel approach for resource management in a semantic web operating system based on a P2P grid architecture. *J Supercomput*. 2014;69(2):955-975.
49. Sun SY, Yao WB, Qiao BJ, Zong M, He X, Li XY. RRSD: a file replication method for ensuring data reliability and reducing storage consumption in a dynamic cloud-P2P environment. *Futur Gener Comput Syst*. 2019;100:844-858.
50. Shojafar M, Pooranian Z, Naranjo PGV, Baccarelli E. FLAPS: bandwidth and delay-efficient distributed data searching in Fog-supported P2P content delivery networks. *J Supercomput*. 2017;73(12):5239-5260.
51. Park S, Cha BR, Chung CK, Kim JW. Mobile IoT device summarizer using P2P web search engine and inherent characteristic of contents. *Peer-to-Peer Networking and Applications*. 13. 2020:684-693. https://doi.org/10.1007/s12083-019-00780-w.

52. Osanaiye O, Chen S, Yan Z, Lu R, Choo K-KR, Dlodlo M. From cloud to fog computing: a review and a conceptual live VM migration framework. *IEEE Access*. 2017;5:8284-8300.
53. Jeschke S, Brecher C, Meisen T, Özdemir D, Eschert T. *Industrial internet of things and cyber manufacturing systems*. In: Jeschke S., Brecher C., Song H., Rawat D. (eds) Industrial Internet of Things. Springer Series in Wireless TechnologyCham: Springer; 2017.
54. Aazam M, Zeadally S, Harras KA. Deploying fog computing in industrial internet of things and industry 4.0. *IEEE Trans Ind Inform*. 2018;14(10):4674-4682.
55. Song H, Srinivasan R, Sookoor T, Jeschke S. *Smart Cities: Foundations, Principles, and Applications*. USA: John Wiley & Sons; 2017.
56. Arasteh H, Hosseinnezhad V, Loia V, et al. *Iot-Based Smart Cities: A Survey*. Florence, Italy: IEEE; 2016:1-6.
57. Memos VA, Psannis KE, Ishibashi Y, Kim B-G, Gupta BB. An efficient algorithm for media-based surveillance system (EAMSuS) in IoT smart city framework. *Futur Gener Comput Syst*. 2018;83:619-628.
58. Zhu J, Song Y, Jiang D, Song H. A new deep-Q-learning-based transmission scheduling mechanism for the cognitive Internet of Things. *IEEE IoT J*. 2017;5(4):2375-2385.
59. Liu Y, Liu A, Wang T, Liu X, Xiong NN. An intelligent incentive mechanism for coverage of data collection in cognitive Internet of Things. *Futur Gener Comput Syst*. 2019;100:701-714.
60. Sangaiah AK, Thangavelu A, Sundaram VM. Cognitive computing for big data systems over IoT. *Gewerbestrasse*. 2018;11:6330.
61. Balasubramanian V, Otoum S, Aloqaily M, Al Ridhawi I, Jararweh Y. Low-latency vehicular edge: a vehicular infrastructure model for 5G. *Simul Model Pract Theory*. 2020;98:101968.

How to cite this article: Javanmardi S, Shojafar M, Persico V, Pescapè A. FPFTS: A joint fuzzy particle swarm optimization mobility-aware approach to fog task scheduling algorithm for Internet of Things devices. *Softw Pract Exper*. 2020;1–21. https://doi.org/10.1002/spe.2867