Journal of Network and Computer Applications 201 (2022) 103333

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach

Sadoon Azizi [a,*], Mohammad Shojafar [b], Jemal Abawajy [c], Rajkumar Buyya [d]

[a] Department of Computer Engineering and IT, University of Kurdistan, Sanandaj, Iran
[b] 5GIC & 6GIC, Institute for Communication Systems (ICS), University of Surrey, Guildford, GU27XH, United Kingdom
[c] School of Information Technology, Deakin University, Geelong, VIC 3220, Australia
[d] CLOUDS lab, School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia

ARTICLE INFO

Keywords:
Internet of Things
Fog computing
Cloud computing
Task scheduling
Semi-greedy algorithm
Deadline-aware
Energy consumption

ABSTRACT

With the rapid advancement of Internet of Things (IoT) devices, a variety of IoT applications that require a real-time response and low latency have emerged. Fog computing has become a viable platform for processing emerging IoT applications. However, fog computing devices tend to be highly distributed, dynamic, and resource-constrained, so deploying fog computing resources effectively for executing heterogeneous and delay-sensitive IoT tasks is a fundamental challenge. In this paper, we mathematically formulate the task scheduling problem to minimize the total energy consumption of fog nodes (FNs) while meeting the quality of service (QoS) requirements of IoT tasks. We also consider the minimization of the deadline violation time in our model. Next, we propose two semi-greedy based algorithms, namely priority-aware semi-greedy (PSG) and PSG with multistart procedure (PSG-M), to efficiently map IoT tasks to FNs. We evaluate the performance of the proposed task scheduling approaches with respect to the percentage of IoT tasks that meet their deadline requirement, total energy consumption, total deadline violation time, and the system's makespan. Compared with existing algorithms, the experiment results confirm that the proposed algorithms improve the percentage of tasks meeting their deadline requirement up to 1.35x and decrease the total deadline violation time up to 97.6% compared to the second-best results, respectively, while the energy consumption of fog resources and makespan of the system are optimized.

1. Introduction

The rapid development and widespread proliferation of Internet of Things (IoT) devices have enabled the emergency of many IoT applications such as e-health in healthcare domain, smart grid in energy management, livestock monitoring in agriculture, and smart traffic in road transportation management. These delay-sensitive IoT applications typically have strict quality of service (QoS) requirements and require processing capabilities beyond what could be offered by the IoT devices with severe resource constraints (Ghanavati et al., 2020b). To ameliorate this challenge, a fog computing platform (Mahmud et al., 2020; Baccarelli et al., 2017), that extends cloud computing infrastructure (Shojafar et al., 2019) to the edge of the network have recently gained significant attention for processing IoT applications (Ghanavati et al., 2020b; Mishra et al., 2018; Sun et al., 2018). The advantage of fog computing is that it offers computing resources within the proximity of IoT devices, which significantly reduces the time required for accessing computing resources and enables IoT requests to be processed quickly.

Although fog computing is suitable for processing IoT applications, it also possesses significant challenges with respect to provisioning and management of the fog resources. This is because fog computing resources are capacity limited, dynamic, heterogeneous, and distributed. Moreover, energy consumption is one of the critical issues to be optimized in fog computing utilization (Ghanavati et al., 2020b; Jiang et al., 2019; Abdel-Basset et al., 2020a; Gu et al., 2019; Vemireddy and Rout, 2021). Therefore, fog resource provisioning and management are critical to fully leverage the capabilities of the fog computing for efficiently executing IoT applications. However, the question of how heterogeneous and dynamic fog resources can be effectively allocated to IoT tasks with objectives of ensuring QoS of IoT tasks while minimizing energy consumption of the fog nodes (FNs) remains a fundamental problem that need to be addressed (Ghanavati et al., 2020b; Deng et al., 2016; Taami et al., 2019).

* Corresponding author.
E-mail addresses: s.azizi@uok.ac.ir (S. Azizi), m.shojafar@surrey.ac.uk (M. Shojafar), jemal.abawajy@deakin.edu.au (J. Abawajy), rbuyya@unimelb.edu.au (R. Buyya).

https://doi.org/10.1016/j.jnca.2022.103333
Received 12 August 2021; Received in revised form 21 October 2021; Accepted 7 January 2022
Available online 26 January 2022

Various approaches to efficiently schedule IoT tasks on heterogeneous and resource-limited fog networks based on different optimization techniques have been proposed (Mahmud et al., 2020; Alizadeh et al., 2020; Yang and Rahmani, 2020; Islam et al., 2021). Many previous works propose metaheuristic algorithms (Ghanavati et al., 2020b; Mishra et al., 2018; Sun et al., 2018; Abdel-Basset et al., 2020a; Aburukba et al., 2020; Hoseiny et al., 2021a) and machine learning techniques (Zhang et al., 2019b; Tang and Wong, 2020; Ale et al., 2021) to solve the IoT task scheduling problem. Although these approaches may produce reasonable solutions, their running time is challenging, especially for scheduling the IoT tasks with a firm and hard deadline. Also, many previous studies focus on minimizing the response time (Sun et al., 2018; Aburukba et al., 2020; Hoang and Dang, 2017; Liu et al., 2018; Auluck et al., 2019; Misra and Saha, 2019; Louail et al., 2020; Adhikari et al., 2020; Bu and Wang, 2021; Almutairi and Aldossary, 2021; Hoseiny et al., 2021b) or energy consumption of FNs (Mishra et al., 2018; Abdel-Basset et al., 2020a; Gu et al., 2019; Yang et al., 2018; Gai et al., 2020; Abdel-Basset et al., 2020b). However, solely focusing on one of the scheduling parameters is not sufficient to ensure both a high QoS for the IoT users and low energy cost for fog service providers. To address these issues, we pose and solve the following research question: How can we propose an efficient algorithm with reasonably low time complexity to schedule the IoT tasks on heterogeneous and resource-limited fog computing with the aims of (i) optimizing the energy consumption of FNs, and (ii) meet the deadline requirement of the IoT tasks?

To address the above research question, we propose two efficient IoT task scheduling algorithms that consider the deadline requirement of the tasks and the energy consumption of the FNs. We first model the task scheduling problem as a mixed integer nonlinear programming (MINLP) with the objective of energy consumption minimization and the constraint of meeting IoT task deadline. We also consider the minimization of the deadline violation time in our model. To provide an efficient solution for each instance of the problem, we propose two semi-greedy based algorithms called priority-aware semi-greedy (PSG) and PSG with multi-start procedure (PSG-M). We performed extensive simulation to verify the effectiveness of the proposed algorithms. Previously, we proposed two greedy heuristics algorithms for solving the task scheduling problem in volunteer computing systems (VCSs) with focus on minimizing the total cost in terms of computation, communication and deadline violation (Hoseiny et al., 2021b). Interestingly, in this paper, we improve our previous work by

1. Including the analysis for energy consumption of FNs in the system model;
2. Presenting a mathematical model for calculating the waiting time of the FN queue;
3. More importantly, proposing *two* semi-greedy based approaches to obtain more efficient results.

The major advantage of semi-greedy approaches over greedy heuristic algorithms is that they can avoid local optima due to the existence of randomness, thus the quality of the results is improved (Klincewicz, 1992).

Our main contributions are summarized as follows:

- We propose a MINLP model for the scheduling of IoT tasks in heterogeneous fog networks with the aim of optimizing the total energy consumption of the FNs while meeting the deadline of the tasks. Our model also includes the minimization of the deadline violation time.
- We propose two semi-greedy based algorithms called PSG and PSG-M to efficiently map the IoT tasks to available FNs in order to provide high QoS for IoT users in terms of response time and minimize the energy consumed by FNs.
- We conduct extensive experiments to evaluate the performance of the proposed approaches. The results demonstrate that using our proposed algorithms the deadline requirement of a very large percentage of the IoT tasks are met and the rest get their response with a little violation time. Meanwhile, the total energy consumption and makespan of the system is reduced so the profit of fog service providers is also maximized.

The rest of this paper is organized as follows. Section 2 reviews related studies on the task scheduling in fog computing. Section 3 presents the system model, including the architecture and problem formulation. Our proposed algorithms are described and analyzed in Section 4. Evaluation and experimental results are given in Section 5. Finally, Section 6 and Section 7 discusses and concludes the paper, respectively.

2. Related work

In recent years, extensive researches have been conducted on the scheduling of IoT tasks in the fog and edge environment (Mahmud et al., 2020; Alizadeh et al., 2020; Yang and Rahmani, 2020; Islam et al., 2021; Kaur et al., 2021). Each of them has focused on various aspects of the problem optimization. Generally speaking, they can be classified into three categories: (i) delay-aware, (ii) energy-efficient, and (iii) joint delay-aware and energy-efficient. In the following, we review some representative related works in each category.

2.1. Delay-aware algorithms

Due to the importance of the response time for IoT application requests, many researchers have considered this aspect in their optimization goals. In Hoang and Dang (2017), the authors investigate the task scheduling problem in a fog environment with multiple regions and formulate it as an integer program such that the task completion time is minimized. Then, they propose a priority-aware heuristic algorithm to solve the problem. Sun et al. (2018) propose an improved non-dominated sorting genetic algorithm II (NSGA-II) to reduce the completion time of a task and improve the overall stability of the task execution. In Liu et al. (2018), Liu et al. introduce dispersive stable task scheduling, named DATS, which is based on the stable matching theory to achieve minimal service delay in heterogeneous fog networks. DATS is a decentralized algorithm that considers the computation and communication delay in its model. The authors of Auluck et al. (2019) consider three types of tasks, named hard, firm and soft. Then, they try to schedule hard tasks on embedded devices, firm tasks on FNs and soft ones on cloud servers. The main goal of their task assignment problem is to minimize the total communication delay. Misra and Saha (2019) study the task offloading problem is software-defined networking (SDN)-enabled fog networks with focus on the minimization of IoT tasks delay and IoT devices' energy consumption. They model the problem as an ILP and solve it using a greedy-heuristic-based approach. Aburukba et al. (2020) formulate the scheduling of IoT service requests as an ILP and propose a customized genetic algorithm (GA) to minimize service time includes transmission, propagation, queueing delay and processing time.

Furthermore, a dynamic task scheduling approach for Industrial IoT (IIoT) with the goal of minimizing the number of rejected tasks is proposed in Louail et al. (2020). To achieve this goal, a new discipline based on the deadline and frequency of tasks is introduced. Adhikari et al. (2020) design a novel deadline and priority-aware task offloading strategy, named DPTO, for scheduling IoT tasks and put forward a multilevel feedback queueing model to minimize the overall latency of the offloaded tasks while meeting their deadline. In order to minimize the task processing delay, Bu and Wang (2021) present a three-schema mechanism that leverages SDN to assign each task to the most suitable edge computing server (ECS). In Almutairi and Aldossary (2021), the authors propose a fuzzy logic-based approach for scheduling the offloaded tasks in edge–cloud environments with the aim of minimizing the overall service time of latency-sensitive

applications. Recently, in our previous work (Hoseiny et al., 2021b), we propose two efficient heuristic algorithms to jointly minimize the cost of computation, communication and deadline violation in volunteer fog–cloud computing systems. The simulation results demonstrate that the proposed algorithms significantly outperform their counterparts in terms of the percentage of tasks that meet their deadline and the cost of violation. Although the aforementioned works make great contribution in their respective domains, they do not take into account energy consumption while this work does.

2.2. Energy-efficient algorithms

Energy efficiency is one of the major concerns in IoT ecosystem (Baccarelli et al., 2017; Jiang et al., 2019). Therefore, minimizing energy consumption of the FNs is considered to be one of the objectives in recent studies. Jalali et al. (2016) show that running IoT applications in fog computing is more energy-efficient than running the same applications in cloud data centers. In Yang et al. (2018), Yang et al. provide a comprehensive analytical model for evaluating the overall energy efficiency in fog networks with homogeneous FNs. Then, they propose a maximal energy-efficient task scheduling algorithm, named MEETS, to get an energy efficient solution for their model. Gu et al. (2019) study the task allocation and energy scheduling problem in green edge computing with the goal of minimization of brown energy consumption. They first formulate the problem as a MILP and then give a relaxation-based heuristic algorithm to solve the problem efficiently. Gai et al. (2020) address the task allocation problem in the context of high-performance fog computing. They have focused on the energy minimization under the constraint of total execution time length, i.e., makespan, and developed a novel heuristic algorithm to solve their optimization model.

To minimize the total system makespan and the consumed energy in the field of fog computing, several efforts have been carried out recently. For example, Mishra et al. (2018) propose different meta-heuristic algorithms for optimal allocation of a set of services to a set of heterogeneous virtual machines (VMs) in a fog computing domain. The approach in Abdel-Basset et al. (2020a) is based on a nature-inspired algorithm, named Harris hawks optimization (HHO) (Heidari et al., 2019), with adaptation of a local search strategy to improve the performance of the standard HHO algorithm. Similar to this work, Abdel-Basset et al. (2020b) use the marine predators algorithm (MPA) (Faramarzi et al., 2020) and their two improved versions to reduce the carbon dioxide emission rate by minimizing the energy consumption and makespan. Moreover, Ghanavati et al. (2020b) introduce ant mating optimization (AMO) and use it to solve the task scheduling problem in fog computing environment. In the above-mentioned studies, great efforts have been devoted to reduce the energy consumption in the context of fog computing environments. However, none of them considers the deadline requirement of IoT tasks.

2.3. Joint delay-aware and energy-efficient algorithms

Considering both of the delay requirement of IoT tasks and the energy consumption of FNs has become an active research area nowadays. For example, in order to achieve a trade-off between the service delay of artificial intelligence (AI) tasks and the energy consumed by the Cloudlet server, Zhang et al. (2019a) propose MASM, a multiple algorithm service model in which the computing VMs can adopt different AI algorithms for processing the same type task. Then, they develop a tide ebb algorithm (TEA) to derive robust solutions for their proposed model. Hassan et al. (2020) propose an efficient policy for the service scheduling problem in fog–cloud computing systems to provide low response time for IoT requests and energy efficiency for fog service providers. The authors classify the IoT services into critical and normal. For critical services, they propose MinRes whose goal is minimizing response time, and propose MinEng for normal ones to reduce the consumed energy of FNs. Recently in Ale et al. (2021), a deep reinforcement learning (DRL) approach to maximize the number of completed tasks before their deadline and minimize the consumed energy of edge servers is studied.

A summary of related work and their main features comparison is shown in Table 1. Although there are some studies on delay and energy optimization in fog/edge computing, the present work is essentially different from these studies in several significant aspects. First, our work focuses on the FNs energy consumption whereas the objective of the majority of studies such as Misra and Saha (2019), Shahryari et al. (2021), Li (2021) and Sun et al. (2019) is to minimize the energy consumption of mobile devices. Therefore, this work complements them. Second, unlike Zhang et al. (2019a) which make a trade-off between the service delay and energy consumption of a Cloudlet server, in our work the total energy consumption of FNs is optimized while the deadline requirement of the tasks is considered as a constraint. Third, Hassan et al. (2020) does not take into account the deadline requirement of each IoT service request, whereas in this work we consider it. Fourth, although DRL-based approaches like Zhang et al. (2019b) and Tang and Wong (2020) may provide a near-optimal solutions, unfortunately they need heavy computation cost (Ghanavati et al., 2020b). To cop with this, in this work, we present a semi-greedy-based approach in which the time complexity is very low compared to the most of AI techniques. Fifth, unlike heuristic-based approaches, the semi-greedy approach can avoid local optima and improve the quality of the results. Moreover, to our best knowledge, this work and our previous work (Hoseiny et al., 2021b) are the only studies that address the deadline violation time of tasks in fog computing environments.

3. System model

This section includes two subsections which together define our system model. In the first subsection, the general architecture of IoT-fog–cloud is presented. In the second subsection, the task scheduling problem is mathematically formulated.

3.1. Architecture

Fig. 1 illustrates the high-level system architecture of the IoT-fog–cloud environment, which consists of four parts, namely *IoT devices*, *gateways*, *fog environment* and *cloud environment*. In the following, we explain each part in detail.

- **IoT Devices:** This part includes a large number of IoT devices such as smart wearables, RFID tags, automobile sensors, security sensors, smart home appliances, thermostats, smart meters, industry devices, and so on. These devices are geographically distributed in various places and usually generate a lot of time sensitive data which requires (nearly) real-time processing. For example, data generated from a health monitoring system must be instantly processed and analyzed, where latency in these systems can lead to catastrophic consequences. However, most of IoT devices are resource constraint, i.e., they suffer from limited resources including processing capabilities, memory, and battery power. Therefore, they offload their tasks to the nearby gateways.
- **Smart Gateways:** The computation tasks offloaded from IoT devices are received by the smart gateways (or access points) present at the edge of network. Depending on the characteristics of tasks such as their priority and deadline (Adhikari et al., 2020), gateways decide where to submit a task for processing, distributed FNs or centralized cloud servers. This decision-making process can be done by applying different methods like machine learning techniques (Guevara et al., 2020) and data mining algorithms (Savaglio et al., 2019). Generally speaking, latency-sensitive tasks are submitted to fog environment while latency-tolerant ones are sent to cloud environment (Peng et al., 2018; Omer et al., 2021). It is worth mentioning that smart gateways

Table 1
A summary of related work and their attributes comparison.

Work	Approach	Delay-aware	Energy consumption of FNs	Violation time	Run time
Hoang and Dang (2017)	Heuristic	✓	×	×	Low
Sun et al. (2018)	NSGA-II	✓	×	×	High
Liu et al. (2018)	Stable matching theory	✓	×	×	Low
Auluck et al. (2019)	Heuristic	✓	×	×	Low
Misra and Saha (2019)	Greedy-heuristic	✓	×	×	Low
Aburukba et al. (2020)	GA	✓	×	×	High
Louail et al. (2020)	Heuristic	✓	×	×	Low
Adhikari et al. (2020)	Multilevel feedback queueing model	✓	×	×	Low
Bu and Wang (2021)	Heuristic	✓	×	×	Low
Almutairi and Aldossary (2021)	Fuzzy logic	✓	×	×	Low
Hoseiny et al. (2021b)	Heuristic	✓	×	✓	Low
Yang et al. (2018)	Analytical model	×	✓	×	Low
Gu et al. (2019)	Heuristic	×	✓	×	Low
Gai et al. (2020)	Heuristic	×	✓	×	Low
Mishra et al. (2018)	Metaheuristic	×	✓	×	High
Abdel-Basset et al. (2020a)	Harris hawks optimization	×	✓	×	High
Abdel-Basset et al. (2020b)	Marine predators algorithm	×	✓	×	High
Ghanavati et al. (2020b)	Ant mating optimization	×	✓	×	High
Zhang et al. (2019a)	Multiple algorithm service model	✓	✓	×	Low
Hassan et al. (2020)	Heuristic	✓	✓	×	Low
Ale et al. (2021)	Deep reinforcement learning	✓	✓	×	High
This work	Semi-greedy	✓	✓	✓	Low

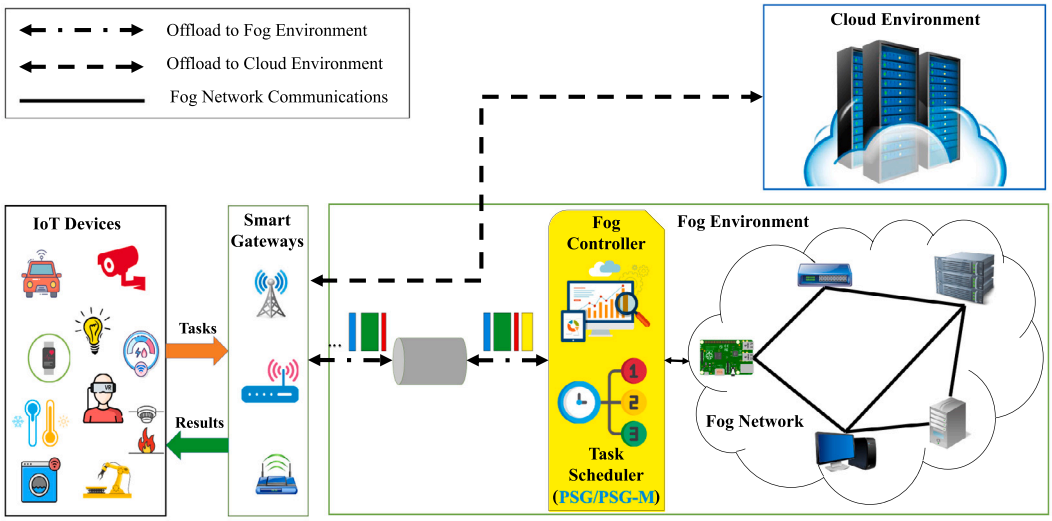

Fig. 1. Proposed System Architecture.

are as close as one hope to the IoT devices. Thus, they can be considered as the edge node for the IoT devices. Although edge nodes can be used to solve some IoT-related issues, their limited resources are not capable of executing large-scale IoT tasks (Mahmud et al., 2020).

- **Fog Environment:** A fog environment includes a fog controller (FC), also called broker, and a fog network, as it can be seen from Fig. 1. FC is a central component of a fog service provider where it manages the fog resources and schedules tasks according to the task scheduling algorithm. Fog network consists of a set of wide-spread and geographically distributed devices, e.g., high-end servers, set-top boxes, Raspberry Pis, routers, personal computers, and smart phones, known as FNs. The common characteristic of FNs is that they have the computing, storage, and networking capabilities in order to execute the IoT tasks (Marín-Tordera et al., 2017). Each FN contains a Foglet software agent which monitors the health and other state information of the FN, and sends the information to the FC using its API (Bonomi et al., 2014). Also, the submitted tasks from gateways to the fog environment are temporarily stored in a buffer. Then, FC periodically run the task scheduler algorithm in each time period based on the information of the FNs and requirements of the tasks.
- **Cloud Environment:** This part is mainly composed of a set of virtual machines (VMs) with high computing power and storage capacity (Calheiros et al., 2011; Azizi et al., 2020). Cloud VMs are usually more suitable for executing latency-tolerant and computation intensive tasks than FNs.

3.2. Problem formulation

In this section, we present a formal model for the task scheduling problem in a heterogeneous fog computing system. First, the basic elements and decision variables of the optimization problem are introduced. Then, the response time and energy models are described. Finally, the problem overview is given. Table 2 presents symbols and notations used in our problem formulation.

3.2.1. Basic elements

Fog nodes and submitted tasks from IoT devices are the basic elements of our model. In the following, we formally describe them.

- **Fog Nodes:** A fog network consists of several interconnected FNs which form a mesh topology. Thus, it can be modeled as follows.

Table 2
Main notation. D-variable: decision variables.

	Symbol	Description	Type - Unit	Appears in Eq.
Set	$\boldsymbol{F}$	Set of FNs, where $\mid \boldsymbol{F} \mid = m$	–	–
	$\boldsymbol{L}$	Set of communication links between FNs	–	–
	$\boldsymbol{T}$	Set of tasks, where $\mid \boldsymbol{T} \mid = n$	–	–
Index	i	Index of FNs, $i \in \boldsymbol{F}$	Integer - [units]	–
	j	Index of FNs, $j \in \boldsymbol{F}$	Integer - [units]	–
	k	Index of tasks, $k \in \boldsymbol{T}$	Integer - [units]	–
	l	Index of tasks, $k \in \boldsymbol{T}$	Integer - [units]	(18), (19)
Input Parameter	$\mathcal{F}_i^C$	CPU processing power of FN $\mathcal{F}_i$	Continuous - [W]	(7)
	$\mathcal{F}_i^{act}$	Power consumption of FN $\mathcal{F}_i$ in active state	Continuous - [W]	(17)
	$\mathcal{F}_i^{idle}$	Power consumption of FN $\mathcal{F}_i$ in idle state	Continuous - [W]	(17)
	e_{ij}^p	Propagation delay of link e_{ij}	Continuous - [s]	(5)
	e_{ij}^b	Bandwidth of link e_{ij}	Continuous - [MB]	(6)
	$\mathcal{T}_k^s$	Size of task $\mathcal{T}_k$	Continuous - [MI]	(7)
	$\mathcal{T}_k^d$	Deadline requirement of task $\mathcal{T}_k$	Continuous - [s]	(3), (7), (4), (12), (21),
	$\mathcal{T}_k^{in}$	Input file size of task $\mathcal{T}_k$	Continuous - [MI]	(6)
	$\mathcal{T}_k^{out}$	Output file size of task $\mathcal{T}_k$	Continuous - [MI]	(6)
Variable	d_k^{prp}	Propagation delay of task $\mathcal{T}_k$ from the FC to the destination FN	Continuous - [s]	(5), (9)
	d_k^{tx}	Transmission time of task $\mathcal{T}_k$ from the FC to the destination FN	Continuous - [s]	(5), (9)
	d_k^{exe}	Execution time of task $\mathcal{T}_k$ at the destination FN	Continuous - [s]	(7), (8),(9),(14)
	d_k^{wt}	Waiting time of task $\mathcal{T}_k$ at the destination FN	Continuous - [s]	(9)
	$\mathcal{R}_k$	Response time of task $\mathcal{T}_k$	Continuous - [s]	(4), (12), (21),
	$\mathcal{N}_S$	Number of tasks that meet their predefined deadline	Integer - [units]	(10), (11)
	$S^{\%}$	Percentage of tasks that meet their predefined deadline	Continuous - [units]	(11)
	ν_k	Deadline violation time of task $\mathcal{T}_k$	Continuous - [s]	(12),(13)
	$\mathcal{A}_i$	Active time of FN $\mathcal{F}_i$	Continuous - [s]	(14), (15), (16), (17)
	ι_i	Idle time of FN $\mathcal{F}_i$	Continuous - [s]	(16), (17)
	ϵ_i	Energy consumption of FN $\mathcal{F}_i$	Continuous - [KJ]	(17), (18)
	$\mathcal{M}$	Makespan of the system	Continuous - [s]	(15), (16)
	$\mathcal{V}^{tot}$	Total deadline violation time of the set of tasks	Continuous - [s]	(13), (23)
	$\mathcal{E}^{tot}$	Total Energy consumption of the system	Continuous - [KJ]	(18), (19)
D-Variable	x_{ik}	Value 1 if task $\mathcal{T}_k$ assigned to FN $\mathcal{F}_i$, otherwise zero	Binary - [units]	(1), (7), (8), (14), (20),(22)
	y_{ij}^k	Value 1 if link e_{ij} is chosen for routing task $\mathcal{T}_k$, otherwise zero	Binary - [units]	(2), (5), (6), (22)
	z_{kl}	Value 1 if task $\mathcal{T}_l$ has a higher priority that task $\mathcal{T}_k$, otherwise zero	Binary - [units]	(3), (8), (10), (22)
	s_k	Value 1 if deadline of task $\mathcal{T}_k$ is satisfied, otherwise zero	Binary - [units]	(4), (10)

Let us assume a heterogeneous fog network $G = (\boldsymbol{F}, \boldsymbol{L})$ where $\boldsymbol{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_m\}$ describes the set of m FNs and $\boldsymbol{L} = \{e_{ij} | i, j \in \boldsymbol{F}\}$ represents the set of communication links between FNs. Each FN $F_i \in \boldsymbol{F}$ has some specific characteristics that are (i) *CPU processing power* $\mathcal{F}_i^c$ in MIPS (Million Instruction Per Second), (ii) *power consumption in active state* $\mathcal{F}_i^{act}$ in W (Watt), as well as (iii) *power consumption in idle state* $\mathcal{F}_i^{idl}$ in W. Also, each link $e_{ij} \in \boldsymbol{L}$ is associated with two main characteristics that are (i) *propagation delay* e_{ij}^p in ms (milli second) and (ii) *bandwidth* e_{ij}^b in $Mbps$ (Mega bit per second).

- **Tasks:** Let us consider $\boldsymbol{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n\}$ is the set of n independent tasks offloaded from IoT devices to the FC during a specific time period. Each task $\mathcal{T}_k \in \boldsymbol{T}$ is defined by a four tuple: $\mathcal{T}_k = \langle \mathcal{T}_k^s, \mathcal{T}_k^d, \mathcal{T}_k^{in}, \mathcal{T}_k^{out} \rangle$, where $\mathcal{T}_k^s$ is the size of the task in MI (Million Instruction), $\mathcal{T}_k^d$ is the deadline requirement in ms, $\mathcal{T}_k^{in}$ is the input file size in KB (Kilo Byte), and $\mathcal{T}_k^{out}$ is the output file size in KB. It is assumed that the value of these parameters is known to the FC (Ghanavati et al., 2020b; Abdel-Basset et al., 2020a; Auluck et al., 2019). It can be estimated by the IoT devices or smart gateways. In order to take advantage of parallel and distributed computing power on a fog network and reduce the response time, IoT tasks usually are independent to each other (Ghanavati et al., 2020b; Nguyen et al., 2019; Konečnỳ et al., 2016). Hence, we assume this in our work, similar to Ghanavati et al. (2020b), Abdel-Basset et al. (2020a), Ale et al. (2021) and Adhikari et al. (2020).

3.2.2. Decision variables

Before the description of our response time and energy model for the considered system, we first introduce our decision variables. We denote $X_{m \times n}$ as the task assignment matrix, where its (i,k)th entry is represented by $x_{ik} \in \{0,1\}$ which is given by

$$x_{ik} = \begin{cases} 1 & \text{if task } \mathcal{T}_k \text{ is assigned to FN } \mathcal{F}_i \\ 0 & \text{otherwise} \end{cases}, \quad \forall i \in \boldsymbol{F}, \forall k \in \boldsymbol{T} \tag{1}$$

For each task $\mathcal{T}_k$, we define a binary variable $y_{ij}^k \in \{0,1\}$ to represent whether link $e_{ij} \in L$ is chosen for routing the task $\mathcal{T}_k$. So, we have

$$y_{ij}^k = \begin{cases} 1 & \text{if link } e_{ij} \text{ is chosen for routing } \mathcal{T}_k \\ 0 & \text{otherwise} \end{cases}, \ \forall i, j \in \boldsymbol{F}, \forall k \in \boldsymbol{T} \tag{2}$$

Also, we use a binary variable $z_{kl} \in \{0,1\}$ to compare the priority of two tasks $\mathcal{T}_k$ and $\mathcal{T}_l$. To this end, we set $z_{kl} = 1$, if task $\mathcal{T}_l$ has a higher priority than task $\mathcal{T}_k$; otherwise, we set its value as zero. In this work, the priority of tasks is determined by their deadline. Thus, we have

$$z_{kl} = \begin{cases} 1 & \text{if } \mathcal{T}_l^d < \mathcal{T}_k^d \\ 0 & \text{otherwise} \end{cases}, \quad \forall l, k \in \boldsymbol{T} \tag{3}$$

Finally, we use a binary variable $s_k \in \{0,1\}$ for showing whether the deadline of task T_k is satisfied. We can say

$$s_k = \begin{cases} 1 & \text{if } \mathcal{R}_k \leq \mathcal{T}_k^d \\ 0 & \text{otherwise} \end{cases}, \quad \forall l, k \in \boldsymbol{T} \tag{4}$$

where $\mathcal{R}_k$ denote the response time of task $\mathcal{T}_k$ and its value is obtained in the next subsection.

3.2.3. Response time

For a given task $\mathcal{T}_k$ submitted to the FC, its response time comprises: (a) propagation delay d_k^{prp} from the FC to FN $\mathcal{F}_i \in \boldsymbol{F}$ and vice versa, (b) transmission time d_k^{tx} which is included the time taken to transmit the input file size $\mathcal{T}_k^{in}$ from the FC to FN $\mathcal{F}_i \in \boldsymbol{F}$ and retransmit the output file size $\mathcal{T}_k^{out}$ from the FN $\mathcal{F}_i$ to the FC, (c) execution time d_k^{exe} and (d) time d_k^{wt} for waiting at the queue of FN $\mathcal{F}_i$.

The propagation delay experienced by the task $\mathcal{T}_k$ in the fog network can be represented as follows:

$$d_k^{prp} = \sum_{\forall e_{ij} \in \boldsymbol{L}} \left(2 \times e_{ij}^p \times y_{ij}^k\right), \quad \forall k \in \boldsymbol{T} \tag{5}$$

In our multi-hop fog network, the transmission time of the task $\mathcal{T}_k$ is derived from the following equation.

$$d_k^{tx} = \sum_{\forall e_{ij} \in \boldsymbol{L}} \frac{\mathcal{T}_k^{in} + \mathcal{T}_k^{out}}{e_{ij}^b} \times y_{ij}^k, \quad \forall k \in \boldsymbol{T} \tag{6}$$

CPU execution time is one the main factors in determining the response time of a given task. The execution time of the task $\mathcal{T}_k$ is obtained by dividing its size by the CPU processing power of the assigned FN. We have

$$d_k^{exe} = \sum_{\forall i \in \boldsymbol{F}} \frac{\mathcal{T}_k^s}{\mathcal{F}_i^c} \times x_{ik}, \quad \forall k \in \boldsymbol{T} \tag{7}$$

In this work, we assume that each FN can process only one task at a time. We also assume that tasks are non-preemptive, i.e., once a task is executed on a particular FN, it executes continuously till completion. Therefore, when a task arrives at a FN, it must wait in the FN queue until the completion of the high-priority tasks assigned to that FN. Based on this, the waiting time of the task $\mathcal{T}_k$ can be expressed as follows.

$$d_k^{wt} = \sum_{\forall l \in \boldsymbol{T}} \sum_{\forall i \in \boldsymbol{F}} \left(d_l^{exe} \times z_{kl} \times x_{ik} \times x_{il} \right), \quad \forall k \in \boldsymbol{T} \tag{8}$$

Therefore, the response time of the task $\mathcal{T}_k$ is calculated using the following equation.

$$\mathcal{R}_k = d_k^{prp} + d_k^{tx} + d_k^{exe} + d_k^{wt}, \quad \forall k \in \boldsymbol{T} \tag{9}$$

We can now calculate the percentage of IoT tasks that meet their deadline as well as the total deadline violation time for the set of n tasks submitted to the FC during a specific time period. For this end, let $\mathcal{N}_s$ represents the number of tasks that their predefined deadline is satisfied. Hence, we have

$$\mathcal{N}_s = \sum_{\forall k \in \boldsymbol{T}} s_k, \quad \forall k \in \boldsymbol{T} \tag{10}$$

We define $S^{\%}$ as the percentage of IoT tasks that meet the deadline requirement. $S^{\%}$ can be obtained as follows.

$$S^{\%} = \frac{\mathcal{N}_s}{n}. \tag{11}$$

For each task $\mathcal{T}_k$, the amount of its deadline violation time can be defined as follows.

$$v_k = \max\left(0, \mathcal{R}_k - \mathcal{T}_k^d\right), \quad \forall k \in \boldsymbol{T} \tag{12}$$

To measure the quality of a given task scheduler, we need to see what the total amount of the deadline violation time is. The following equation give us this value.

$$\mathcal{V}^{tot} = \sum_{\forall k \in \boldsymbol{T}} v_k \tag{13}$$

3.2.4. Energy

Here, we focus on the energy spent by the FNs to finish the execution of all the n tasks. During this time period, the energy consumed by a FN $\mathcal{F}_i \in \boldsymbol{F}$ is obtained from the sum of energy consumed in the active state and in the idle state. The active time of the FN $\mathcal{F}_i$ is equal to the time required to process all of the tasks assigned to it which can be calculated by the following formula.

$$\mathcal{A}_i = \sum_{\forall k \in \boldsymbol{T}} \left(d_k^{exe} \times x_{ik} \right), \quad \forall i \in \boldsymbol{F} \tag{14}$$

To obtain the idle time of each FN, we first need to have the makespan of the system, the maximum execution time of a FN among all FNs. The following equation gives us the value of makespan ($\mathcal{M}$).

$$\mathcal{M} = \max_{\forall i \in \boldsymbol{F}} (\mathcal{A}_i) \tag{15}$$

Now the idle time of the FN $\mathcal{F}_i$ can be easily achieved as follows.

$$\imath_i = \left(\mathcal{M} - \mathcal{A}_i \right), \quad \forall i \in \boldsymbol{F} \tag{16}$$

We can estimate the energy consumed by a FN based on its power consumption profile and the amount of time being in each state. Therefore, for a given FN $\mathcal{F}_i$, its energy consumption is computed as below (Mishra et al., 2018).

$$\epsilon_i = \left(\mathcal{A}_i \times \mathcal{F}_i^{act} + \imath_i \times \mathcal{F}_i^{idl} \right), \quad \forall i \in \boldsymbol{F} \tag{17}$$

The total energy consumption of the considered system is obtained by the summation of the energy consumed of each FN. That is:

$$\mathcal{E}^{tot} = \sum_{\forall i \in \boldsymbol{F}} \epsilon_i \tag{18}$$

3.2.5. Problem formulation

Let $\boldsymbol{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n\}$ is the set of n independent tasks offloaded from IoT devices to be scheduled on a set of m heterogeneous FNs $\boldsymbol{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_m\}$. The problem is to map these tasks to the FNs, $\Psi : \boldsymbol{T} \rightarrow \boldsymbol{F}$, such that the total energy consumption is optimized while the deadline of the tasks is satisfied. Thus, we formally model this problem as follows.

$$\min \ \mathcal{E}^{tot} \tag{19}$$

subject to

$$\sum_{\forall i \in \boldsymbol{F}} x_{ik} = 1, \quad \forall k \in \boldsymbol{T} \tag{20}$$

$$\mathcal{R}_k \leq \mathcal{T}_k^d, \quad \forall k \in \boldsymbol{T} \tag{21}$$

$$x_{ik}, y_{ij}^k, z_{kl}, s_k \in \{0, 1\}, \quad \forall i, j \in \boldsymbol{F}, \forall k, l \in \boldsymbol{T} \tag{22}$$

An equality constraint (20) states that each task can only be assigned to exactly one FN. An inequality constraint (21) requires that each task's deadline must be satisfied. The domain of our decision variables is determined by the constraint (22).

It is worth to mention that sometimes the deadline of some tasks may not be met. In this case, there are two possible choices: accept or reject them. In this work, we select the first one. Therefore, if the deadline of a given task is not satisfied, we should try to minimize its deadline violation time. As a result, we also add the following goal to our optimization problem.

$$\min \ \mathcal{V}^{tot} \tag{23}$$

To efficiently solve this optimization problem, we design a semi-greedy approach, which is described in the next section.

4. Proposed algorithms

In this section, we propose PSG, a Priority-aware Semi-Greedy algorithm for solving the task scheduling problem in a given fog environment. PSG is a semi-greedy algorithm because it adds randomization to a greedy algorithm (Resende and Ribeiro, 2016). Also, it considers the priority of tasks during the scheduling process. The algorithm tries to find an efficient solution for the optimization problem (see Section 3.2.5), and it will be periodically run by the FC (see Fig. 1) every τ milliseconds. The description of our proposed algorithm is presented in the following.

4.1. PSG algorithm

The main goal of the PSG algorithm is to schedule the IoT tasks based on their priority and assign them to the suitable FNs in order to minimize the energy consumption of the fog environment while meet the deadlines. However, if the deadline of a given task is not met, the algorithm tries to reduce its violation time as far as possible. It is worth mentioning that the primary aim of the PSG is to provide high

QoS for IoT users while optimizing the energy consumed by FNs is the secondary aim. Our proposed PSG consists of *four* main steps as follows.

Step 1: In first step, tasks are sorted in non-descending order of their predefined deadline. Note that the deadline of each task is determined by its smart gateway (see Fig. 1).

Step 2: In this step, the response time of a given task on all available FNs is computed using Eq. (9). Based on the result, the FNs are divided into two groups: *DSList* and *USList*. *DSList* is the deadline satisfied nodes, those FNs that meet the deadline of the task while *USList* is the list of FNs that do not satisfy the requested deadline of the task.

Step 3: If the *DSList* is not empty, i.e., there exist at least one FN that can meet the delay constraint of the task, for each FN in *DSList*, the energy consumption of the system is estimated by mapping of the given task to that FN. Then, a restricted candidate list (RCL) is created by the best FNs, i.e., those whose provide the least increase of energy consumption to the system. Finally, a FN is randomly chosen from RCL and the task is assigned to that.

Step 4: If the DSList is empty, i.e., none of the FNs can meet the task deadline, the task is mapped to the FNs in the *USList* with the minimum deadline violation time.

Algorithm 1 presents the proposed PSG algorithm for solving the task scheduling problem in a heterogeneous fog domain. First, the set of tasks are sorted in non-descending order based on their predefined order (line 1). Next, the algorithm gets the available time of all FNs from the FC (line 2). For each task, the algorithm tries to find a suitable FN for the selected task $\mathcal{T}_k$ (main loop, i.e., lines 3–19). To this end, first two empty sets are created for the task $\mathcal{T}_k$, *DSList* and *USList* (lines 4–5). Now, the second loop (lines 6–13) searches among all the available FNs and check if the FN $\mathcal{F}_i \in F$ can meet the deadline of the task $\mathcal{T}_k$ or not. If the response time of $\mathcal{F}_i$ is less than the deadline of $\mathcal{T}_k$, the algorithm adds it to the deadline satisfied list, i.e., *DSList*; otherwise, it is added to the unsatisfied list, i.e, *USList*. At this point, with regard to the size of *DSList*, the algorithm takes two different strategies (lines 14–18). If *DSList* is not empty, i.e., there exists at least one FN that can met the deadline requirement of the task $\mathcal{T}_k$, the algorithm calls *SEMI-GREEDY* strategy (Algorithm 2) as reported in line 15; otherwise, it calls *MIN-VIOL* strategy (Algorithm 3) as reported in line 17.

Algorithm 1 PSG: Priority-aware Semi-Greedy algorithm for task scheduling

Input: set of n independent tasks), $G = (\boldsymbol{F}, \boldsymbol{L})$ ($\boldsymbol{F}$: set of m FNs, $\boldsymbol{L}$: set of communication links between FNs), α (to control the amounts of greediness and randomness in the algorithm)
Output: $\Psi : \boldsymbol{T} \rightarrow \boldsymbol{F}$

1: **sort** $\boldsymbol{T}$ in non-descending order of deadline;
2: **get** the available time of FNs from the FC;
3: **for each** $\mathcal{T}_k \in \boldsymbol{T}$ **do**
4: $DSList \leftarrow \emptyset$;
5: $USList \leftarrow \emptyset$;
6: **for each** $\mathcal{F}_i \in \boldsymbol{F}$ **do**
7: **calculate** $\mathcal{R}_k$ using Eq. (9);
8: **if** $\mathcal{R}_k \leq \mathcal{T}_k^d$ **then**
9: $DSList \leftarrow DSList \cup \mathcal{F}_i$;
10: **else**
11: $USList \leftarrow USList \cup \mathcal{F}_i$;
12: **end if**
13: **end for**
14: **if** $DSList \neq \emptyset$ **then**
15: **call** $SEMI-GREEDY(\mathcal{T}_k, DSList, \alpha)$;
16: **else**
17: **call** $MIN-VIOL(\mathcal{T}_k, NSList)$;
18: **end if**
19: **end for**
20: **return** $DSList$ and $USList$

Algorithm 2 shows the pseudocode of our *SEMI-GREEDY* strategy. Assume the size of *DSList* is C (line 1). Lines 2 to 4 estimates the energy consumed of the system by mapping the task $\mathcal{T}_k$ to each of the FNs $\mathcal{F}_i \in$ *DSList*. After this process, the *DSList* is sorted in non-descending order of energy (line 5. Next, the first $(1+\lfloor \alpha \times C \rfloor)$ FNs from the *DSList* is selected (line 6), i.e., those FNs which provide the least increase of energy consumption to the system, and is placed in a restricted candidate list (RCL). Here, $\alpha \in [0, 1)$ is a constant parameter to control the amounts of greediness and randomness in the algorithm. It can be observed that setting $\alpha = 0$ corresponds to a pure greedy algorithm, i.e., the FN with the minimum energy consumed is selected. While setting $\alpha \cong 1$ leads to a random algorithm, i.e., each FN $\mathcal{F}_i \in$ *DSList* will have equal probability to be selected. In line 7, a FN is selected at random from the RCL. Then, the task $\mathcal{T}_k$ is assigned to that FN and its available time is updated (lines 8 and 9). If none of the FNs can meet the deadline of task $\mathcal{T}_k$, i.e., $DSList = \emptyset$, the *MIN-VIOL* strategy is run where its pseudocode is given in Algorithm 3. This strategy gets the task $\mathcal{T}_k$ and the *USList*. First, $v^{min} \rightarrow \infty$ is set (line 1). Next, lines 2 to 8 search among all the FNs $\mathcal{F}_i \in$ *USList* and finds a FN which provides the least deadline violation time for $\mathcal{T}_k$. Line 9 allocates $\mathcal{T}_k$ to the founded FN and update its available time in line 10.

Algorithm 2 SEMI-GREEDY

Input: $\mathcal{T}_k$, $DSList$, α, C
Output: Scheduling of task $\mathcal{T}_k$ to min $\mathcal{E}^{tot}$

1: **let** $|DSList| = C$the available time of FNs from the FC;
2: **for each** $\mathcal{F}_i \in DSList$ **do**
3: EstimateEnergy $(\mathcal{T}_k, \mathcal{F}_i)$ using Eq. (18);
4: **end for**
5: **sort** $DSList$ in non-descending order of energy;
6: $RCL \rightarrow$ pick the first $(1 + \lfloor \alpha \times C \rfloor)$ FNs from $DSList$;
7: **select** a FN from the RCL randomly;
save the selection on $\mathcal{F}_{index}$
8: **assign** $\mathcal{T}_k$ to $\mathcal{F}_{index}$;
9: **update** the available time of FN $\mathcal{F}_{index}$;
10: **return** scheduled $\mathcal{T}_k$

Algorithm 3 MIN-VIOL

Input: $\mathcal{T}_k$, $USList$
Output: Scheduling of task $\mathcal{T}_k$ to min $\mathcal{V}^{tot}$

1: $v^{min} \leftarrow \infty$;
2: **for each** $\mathcal{F}_i \in USList$ **do**
3: **calculate** v_k using Eq. (12);
4: **if** $v_k < v^{min}$ **then**
5: $v^{min} \leftarrow v_k$;
6: $index \leftarrow i$;
7: **end if**
8: **end for**
9: **assign** $\mathcal{T}_k$ to $\mathcal{F}_{index}$;
10: **update** the available time of FN $\mathcal{F}_{index}$;
11: **return** scheduled $\mathcal{T}_k$

4.2. PSG-M algorithm

Here, we improve the PSG algorithm by adding the multistart procedure to it. The improved algorithm, named PSG-M, conducts a series of the PSG algorithm and outputs the best solution found over all iterations. Note that each iteration or trial is independent of the others, therefore each of them produces an independent solution. There are different criteria for selecting the best solution. However, since the main focus of this study is providing high QoS for IoT users, the solution with the highest $S^{\%}$, i.e., the percentage of IoT tasks that meet the deadline requirement (see Eq. (11)), is reported as the best.

The pseudocode of the proposed PSG-M is shown in Algorithm 4. In first step, the algorithm set zero for $S^{\%}_{max}$ (line 1). In the next step, the PSG algorithm (Algorithm 1) is run N_{itr} times (lines 2–9) where each iteration generates a solution. Among these N_{itr} iterations, the solution which offer the maximum $S^{\%}$ is selected as the best solution and returned by the algorithm (line 10).

Algorithm 4 PSG-M: Priority-aware Semi-Greedy Algorithm with Multistart

Input: $\boldsymbol{T}$ (set of n independent tasks), $G = (\boldsymbol{F}, \boldsymbol{L})$ ($\boldsymbol{F}$: set of m FNs, $\boldsymbol{L}$: set of communication links between FNs), N_{itr} (number of iterations), α (to control the amounts of greediness and randomness in the algorithm)
Output: $\Psi : \boldsymbol{T} \rightarrow \boldsymbol{F}$

1: $S^{\%}_{max} \leftarrow 0$;
2: **for** $i = 1 : N_{itr}$ **do**
3: $S \leftarrow$ **call** $PSG(\boldsymbol{T}, G, \alpha)$;
4: **calculate** $S^{\%}$ for solution S using Eq. (11);
5: **if** $S^{\%} > S^{\%}_{max}$ **then**
6: $S^{*} \leftarrow S$;
7: $S^{\%}_{max} \leftarrow S^{\%}$;
8: **end if**
9: **end for**
10: **return** S^{*}

4.3. Complexity analysis

In the following, we give the time complexity analysis of our proposed algorithms.

(1) **PSG (Algorithm 1):** The complexity of line 1 is $O(n \log n)$ due to the sorting of tasks. To get and set the available time of FNs, the algorithm needs $O(m)$. For a given task $\mathcal{T}_k$ (line 3), grouping FNs require $O(m)$ times (lines 6–13). Thus, for a set of n tasks, the complexity of this phase is equal to $O(n \times m)$. To find a suitable FN for a given task $\mathcal{T}_k$, the *SEMI-GREEDY* strategy (Algorithm 2) or *MIN-VIOL* strategy (Algorithm 3) is run. The complexity of lines 2–4 of the *SEMI-GREEDY* strategy is $O(m)$, since the maximum cardinality of *DSList* is m, i.e., $|DSList| = m$. Also, sorting *DSList* in line 5, requires $O(m \log m)$. Other lines take constant time. Therefore, the worst-case time complexity of the *SEMI-GREEDY* strategy is $O(m \log m)$. On the other hand, the complexity of the *MIN-VIOL* strategy is in the order of $O(m)$ as it contains only one *for* loop. Note that the maximum cardinality of *USList* is m. Hence, finding a suitable FN for each task requires at most $O(m \log m)$, and for the set of n tasks need $O(n \times m \ log m)$. Therefore, the overall time complexity of PSG is $O(n \times m \log m)$.

(2) **PSG-M (Algorithm 4):** Since our PSG-M runs the PSG algorithm N_{itr} times, its overall time complexity is equal to $O(N_{itr} \times n \times m \log m)$.

5. Performance evaluation

In this section, we shed light on the performance and comparison of the proposed algorithms with other benchmarks in terms of: (1) the percentage of IoT tasks that meet their deadline requirement, i.e., $S^{\%}$ (see Eq. (11)); (2) the total energy consumption of the system, i.e., $\mathcal{E}^{tot}$ (see Eq. (19); (3) the total amount of the deadline violation time, i.e., $\mathcal{V}^{tot}$ (see Eq. (13); and (4) the makespan of the system, i.e., $\mathcal{M}$ (see eq. (15). The compared algorithms are described in Section 5.1. The simulation settings are given in Section 5.2. Finally, results of the experiments are presented in Section 5.3.

5.1. Compared algorithms

To show the effectiveness of our proposed algorithms, PSG and PSG-M, we compare them against the following baselines.

(1) **First Come First Serve (FCFS)** (Zhao and Stankovic, 1989): This is a simple task scheduling algorithm which aims to balance the load among the computing nodes of the environment. In the FCFS policy, the first IoT task that arrives at the FC is the first task to be scheduled. For each task, a FN is randomly selected to process the task.

(2) **Earliest Deadline First (EDF)** (Stankovic et al., 2012): EDF is a delay-aware scheduling algorithm where it gives higher priority to tasks with the lower deadline. Similar to FCFS, the FN selection phase is based on the random strategy.

(3) **Greedy for Energy (GfE)** (Xu et al., 2020): In this algorithm, the task selection is similar to the FCFS policy. However, its node selection phase is different. GfE assigns each task to the FN that offers the most energy-saving for the system.

(4) **Detour** (Misra and Saha, 2019): This scheme includes three aspects: (a) local decision-making process, (b) optimal FN selection, and (c) optimal path selection. The local decision-making process is done based on a utility function. The FN with the minimum waiting and execution time is selected for each offloaded task. Finally, the shortest path using the Dijkstra's algorithm is calculated to route the offloaded task to the destination FN. Since the main focus of this work is on the task scheduling process, we have considered the last two aspects in our implementation.

5.2. Simulation setting

For the simulation experiments carried out in this work, we consider a fog environment consisting of several heterogeneous, interconnected FNs with random mesh topology and a set of different tasks offloaded from IoT devices to be scheduled on the FNs. To conduct different experiments, the number of FNs is varied from 30 to 90 and the number of IoT tasks is changed from 100 to 500. To ensure the heterogeneity of FNs, the CPU processing power of each FN is assumed to uniformly distributed from 2000 to 6000 MIPS while its power consumption in active state is randomly generated in [80–200] W. The power consumption of a FN in idle state is considered to be about 60%–70% of its power consumed in active state (Greenberg, 2008; Hashimoto and Aida, 2012). The propagation delay between FNs is assumed to be [1–3] ms while the bandwidth of communication links is set to 1000 Mbps. For the IoT tasks, to take a meaningful relation between the size of the tasks and their deadline requirement into consideration, two different types of tasks are considered (Auluck et al., 2019): (1) hard real-time tasks and (2) firm real-time tasks. For the first type, the size of the tasks is generated randomly in [100–372] MI while their deadline is assumed to be uniformly distributed in the interval [100, 500] ms. For the second type, the values are set to [1028–4280] MI and [500–2500] ms, respectively. For both types, the input and output file sizes are randomly selected in [100–10000] KB and [1–1000] KB, respectively. The summary of parameter settings for FNs and IoT tasks are given in Table 3. Regarding of our proposed algorithms, a balance between greediness and randomness is required which can be controlled by setting the value of the parameter α. In the next subsection, we evaluate the performance of PSG and PSG-M using various experiments to specify the value of α. For the PSG-M algorithm, the number of iterations, i.e., N_{itr}, must be set. Although by increasing N_{itr} the value of $S^{\%}$ may be somewhat improved, the execution time of the algorithm is also increased. Since the running time of the algorithm is of great importance, the value of N_{itr} should be as low as possible. We did extensive testing and found that $N_{itr} = 100$ is a good value for this parameter.

All simulations are coded in C++ programming language on Dev-Cpp 5.11 IDE. The experiments were carried out on a laptop running on Intel® Core i7-6600U CPU, 2.6 GHz, 4 cores, 16 GB of RAM, and Windows 10 operating system. To provide results with high confidence, each experiment is repeated 30 times and reported an average of them. The source code of the paper is available in Azizi et al. (2021).

Table 3
Parameter settings for FNs and IoT tasks.

Parameter	Value	Parameter	Value
Number of FNs	{30, 45, 60, 75, 90}	Number of IoT tasks	{100, 200, 300, 400, 500}
FN CPU processing power	[2000–6000] MIPS	FN power consumption in active state	[80–200] W
Propagation delay between FNs	[1–3] ms	N_{itr}	100
IoT task size (type 1)	[100–372] MI	IoT task size (type 2)	[100–500] MI
FN power consumption in idle state	[60–70]% ×[80–200] W	Bandwidth of communication links	1000 Mbps
IoT task deadline (type 1)	[1028–4280] ms	IoT task deadline (type 2)	[500–2500] ms
Input file size for task	[100–10000] KB	Output file size for task	[1–1000] KB

Table 4
Performance of the PSG and PSG-M as a function of the parameter α, where the number of FNs is 60 and the number of tasks is 300. The bold cells represent the best values respect to the α ratio.

α	Algorithm	$S^{\%}$	$\mathcal{M}$	$\mathcal{E}^{tot}$	$\mathcal{V}^{tot}$
0	PSG	93.0	2.18	17,384	2.19
	PSG-M	93.1	2.19	17,424	2.16
0.2	PSG	94.3	2.15	17,214	2.13
	PSG-M	95.6	2.16	17,133	1.95
0.4	PSG	**95.2**	**2.12**	**17,100**	1.88
	PSG-M	**96.5**	**2.10**	**16,744**	0.95
0.6	PSG	92.4	2.25	17,631	**1.56**
	PSG-M	94.7	2.27	17,368	**0.89**
0.8	PSG	89.0	2.46	18,908	4.16
	PSG-M	92.4	2.34	17,837	2.00
1	PSG	90.6	2.44	19,207	1.97
	PSG-M	94.9	2.41	18,526	1.07

Table 5
Performance of the PSG and PSG-M as a function of the parameter α, where the number of FNs is 60 and the number of tasks is 500. The bold cells represent the best values respect to the α ratio.

α	Algorithm	$S^{\%}$	$\mathcal{M}$	$\mathcal{E}^{tot}$	$\mathcal{V}^{tot}$
0	PSG	52.6	3.54	27,456	103.55
	PSG-M	53.5	3.61	28,138	102.54
0.2	PSG	58.0	3.50	**26,222**	89.91
	PSG-M	60.6	3.53	**26,426**	86.88
0.4	PSG	**60.4**	**3.41**	26,527	**82.66**
	PSG-M	**62.0**	**3.49**	27,393	**71.00**
0.6	PSG	48.1	3.70	28,294	123.83
	PSG-M	55.1	3.70	29,395	111.53
0.8	PSG	56.8	3.60	28,526	92.68
	PSG-M	57.0	3.61	28,586	89.50
1	PSG	54.2	3.63	29,589	97.76
	PSG-M	58.4	3.61	28,818	99.22

5.3. Results

Here, we provide the results of our conducted experiments. We first investigate the effect of the α on the performance of the proposed algorithms. Then, we compare our algorithms with the existing algorithms by varying the number of tasks and fog nodes.

5.3.1. Setting the value of α for the proposed algorithms

To specify the value of α, we conducted two different experiments. For both experiments we fixed the number of FNs to 60. However, in the first experiment, the number of tasks is set to 300 while in the second one it is set to 500. The results are reported in Tables 4 and 5, respectively. As the results indicate, we notice that both of the proposed algorithms present better performance in the case of $\alpha = 0.4$. As the number of tasks increases from 300 to 500, the superiority of the proposed algorithms with $\alpha = 0.4$ is more significant than the other values. In particular, it can be observed that when the number of tasks is 300 and the value of α changes from 0 to 0.4, the improvement of PSG-M in terms of $S^{\%}$ is 3.65% while this improvement is reached 15.89% when the number of tasks is 500.

5.3.2. Impact of increasing number of tasks

Fig. 2 shows how the number of tasks affects the performance of the algorithms in different aspects. Here we fixed the number of FNs to 60. There are some important observations in the figure. First, in general, increasing the number of tasks imposes more load on the system. Consequently, more tasks miss their deadline which lead to increasing of the violation time. The energy consumption and makespan of the system is also increased. Second, from Fig. 2(a) it can be observed that in our proposed PSG and PSG-M algorithms the deadline for a significantly higher percentage of tasks is met as compared to the other strategies. This is expected, since our proposed algorithms consider the priority of the IoT tasks and the resource availability of FNs in their decision-making process. Particularly, the percentage of improvement for our PSG and PSG-M is up to 25.3% and 28.6% in compare with the second-best results, i.e., Detour, respectively. Third, the energy consumption and makespan of our algorithms is comparable with that of GfE, see Fig. 2(b) and Fig. 2(d). This is because our proposed algorithms are also taken into account the energy consumption of the system (see Algorithm 2). It is worth mentioning that the makespan of a system has a direct impact on the energy consumption (see Section 3.2.4). Hence, when the energy is minimized, the makespan is also expected to be minimized. Fourth, another interesting observation about our PSG

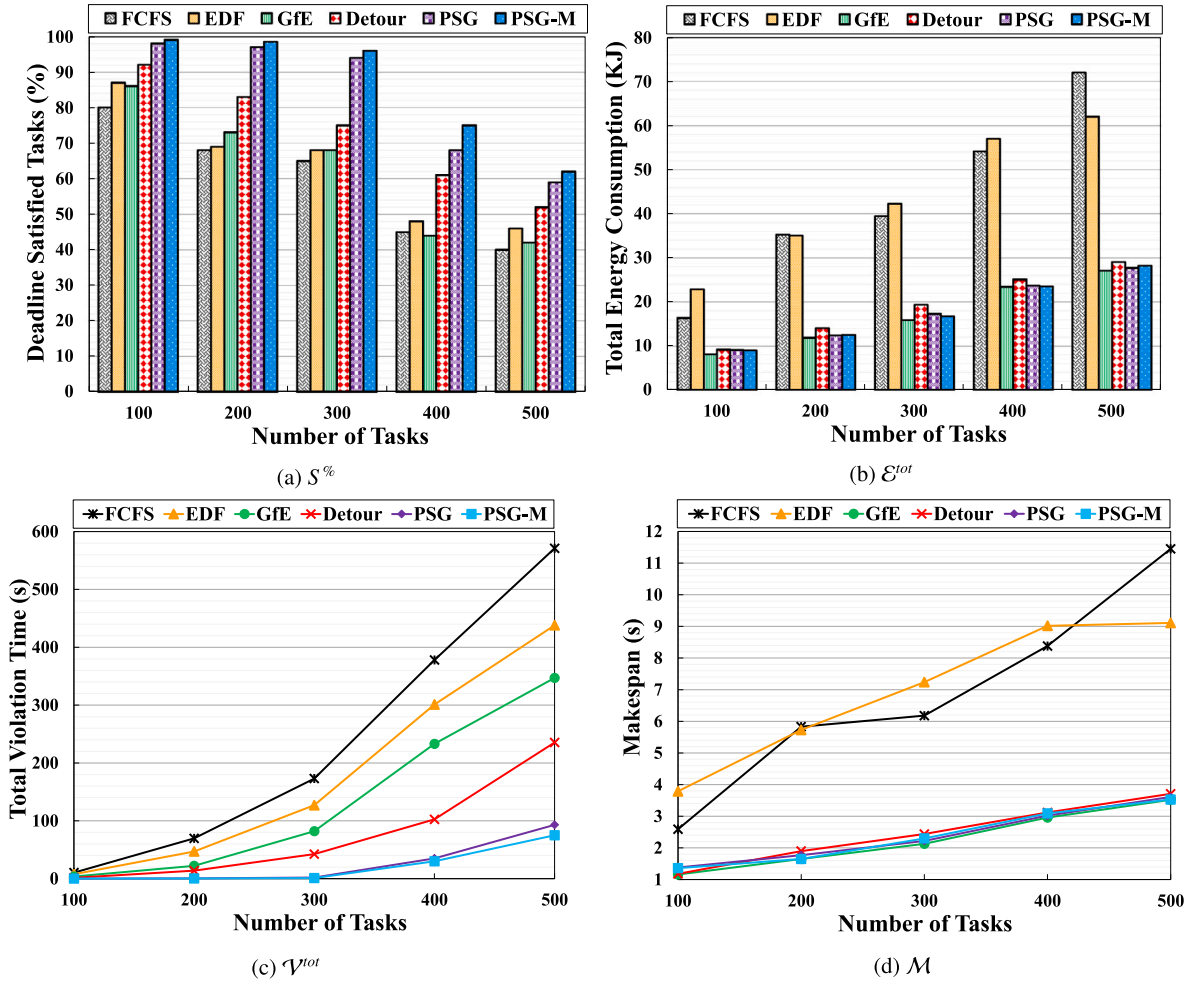

Fig. 2. Varying Number of Tasks — Simulation results for 60 FNs where 2(a) $S^{\%}$:= Deadline satisfaction, 2(b) $\mathcal{E}^{tot}$:= Total energy consumption, 2(c) $\mathcal{V}^{tot}$:= Total violation time, and 2(d) $\mathcal{M}$:= makespan,.

and PSG-M is that they provide a significant improvement over the benchmarks in terms of the total deadline violation time, as we can see from Fig. 2(c). To justify this, we should mention that when there exists no FN to satisfy the deadline of a given task, our algorithms try to find a FN which offers the minimum violation time for that task (see Algorithm 3). Finally, regarding our proposed algorithms, PSG-M somewhat performs better than PSG, especially in terms of the percentage of the deadline satisfied tasks (see Fig. 2(a)). This is due to the multistart procedure where it gives the chance to PSG-M to improve $S^{\%}$, as it selects the solution with the best $S^{\%}$.

5.3.3. Impact of increasing number of fog nodes

The goal of this experiment is to study the effect of number of FNs on the performance of the compared algorithms. In this experiment, the number of tasks is set to 300. Fig. 3(a) to Fig. 3(d) illustrate the results of simulations. Here are the most important observations. First, from Fig. 3(a) we observe that as the number of FNs increases, more tasks meet their deadline. This is expected, since adding more FNs to the system increases the available resources. However, due to the prioritizing of tasks and resource-awareness of the proposed algorithms, the percentage of the tasks that meet their deadline requirement is significantly higher than the other algorithms. In particular, when the number of FNs is 60, our PSG and PSG-M satisfy 95.2% and 96.5% of tasks, respectively, while this value is less than 75% for the other algorithms. Second, from the point of the energy consumed and makespan of the system (see Fig. 3(b) and Fig. 3(d)), GfE, PSG-M, PSG and Detour almost have the same performance while they are far better than FCFS and EDF. Third, it can be seen from Fig. 3(c), with an increase in the number of FNs, the total violation time of all algorithms is substantially decreased. Again, it is clear that the proposed PSG and PSG-M performs much better than the compared algorithms.

6. Discussion and limitations

Many of real-time IoT systems, including autonomous cars (Auluck et al., 2019), smart video surveillance system (Cob-Parro et al., 2021), industrial IoT (Aazam et al., 2018), health monitoring systems (Gia et al., 2015), and smart grids (Chen et al., 2019), have limited embedded resources which make them a performance bottleneck for executing various IoT tasks. With fog computing, there is no need to submit the time-sensitive tasks to remote cloud servers, leading to delayed response time. However, the running time of the fog task scheduler and the quality of the results are critical for such systems. Our time complexity analysis and performance evaluation verify that the proposed algorithms are promising solutions for scheduling the real-time IoT tasks in fog computing systems.

Although this study presents two efficient solutions for scheduling IoT tasks in a heterogeneous fog computing, there are still several aspects that should be addressed in future studies. First, in this work, we ignored the decision-making process of smart gateways for the computation tasks offloaded from IoT devices. However, how to classify the IoT tasks to be processed on the FNs or cloud servers is a challenging task. To address this issue, we plan to use methods such as deep reinforcement learning and k-means in our future work. Second, in the

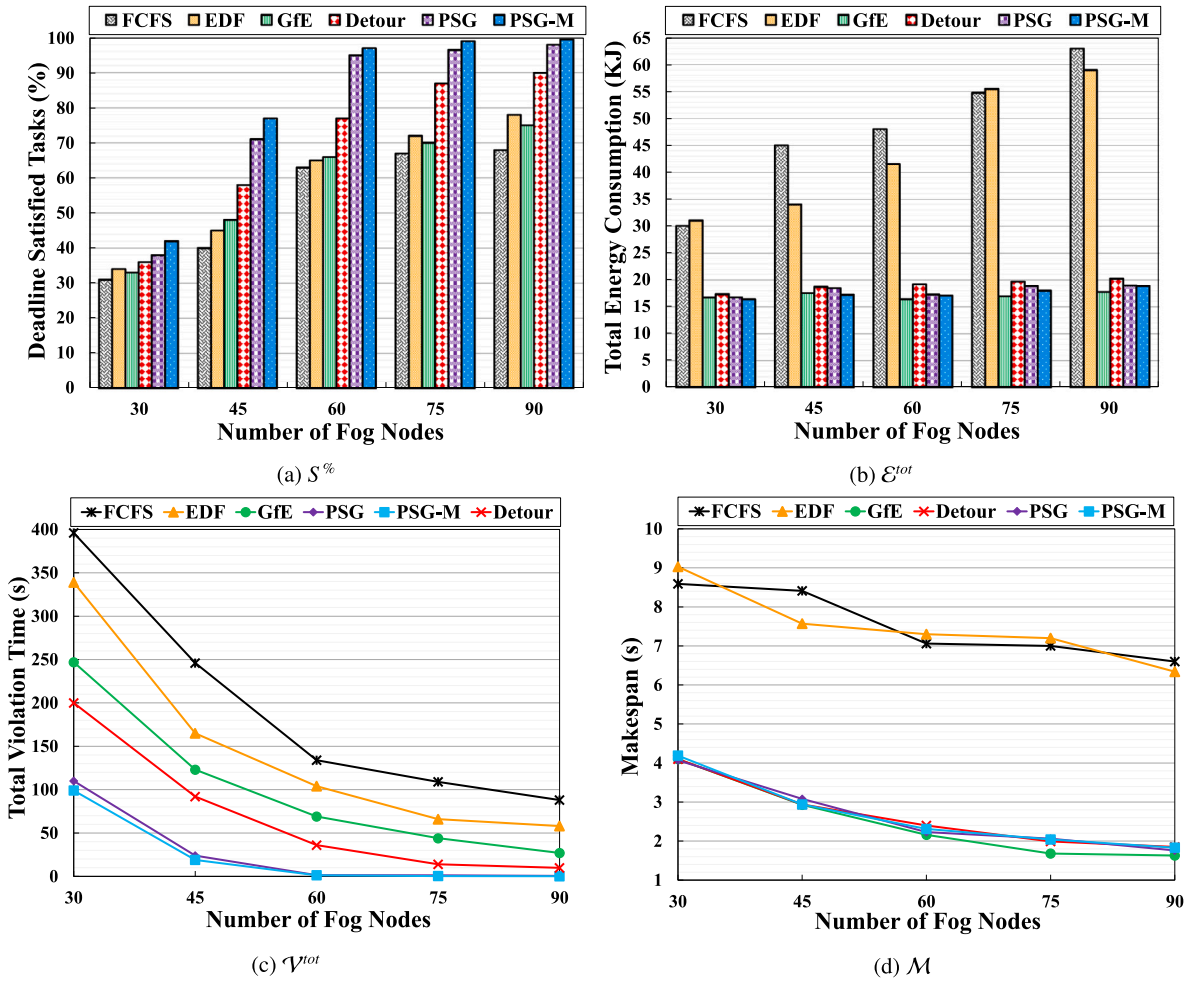

Fig. 3. Varying Number of Fog Nodes — simulation results for 300 of tasks where 3(a) $S^{\%}$:= Deadline satisfaction, 3(b) $\mathcal{E}^{tot}$:= Total energy consumption, 3(c) $\mathcal{V}^{tot}$:= Total violation time, and 3(d) $\mathcal{M}$:= makespan,.

real world, the type of tasks and FNs may be different. For examples, some tasks may need graphical processing unit (GPU). Thus, such tasks should be run on the GPU-enabled FNs. Therefore, it is necessary to modify the proposed algorithms in way that they can also be applied in such environments.

Third, in this work we assumed that each FN can only execute one task at any time. However, several lightweight and open-source tools based on containers have been recently developed such as KubeEdge[1], MicroK8s[2], and K3s[3], which allow us to run multiple tasks simultaneously in a single FN.

Fourth, PSG/PSG-M methods are unable to schedule the tasks on the fog resources when they face resource failures. To do this, we should real-time check the trade-off between task replication and task scheduling in the fog resources serving the heterogeneous IoT applications. In this way, we require to design a fault-tolerant task scheduling algorithm that aims to concurrently assign tasks and optimize the QoS. It helps to manage the uncertainty and dynamic nature of task execution in fog nodes (Ghanavati et al., 2020a; Gazori et al., 2020).

Finally, since the iterations of our PSG-M is independent to each other, its multistart procedure can be executed in a parallel mode. Therefore, the running time of the proposed PSG-M can be remarkably decreased compared with its sequence version.

[1] Kubeedge, https://kubeedge.io, accessed: 2021-05-16.
[2] Microk8s, https://microk8s.io, accessed: 2021-05-16.
[3] K3s, https://k3s.io, accessed: 2021-05-16.

7. Conclusion and future work

In this paper, we studied the scheduling of IoT tasks in a heterogeneous fog network. The key idea of this paper was to optimize the total energy consumption of the system while meeting the deadline of the tasks. If the deadline of a given task is not met, it is allocated to a fog node which provides the minimum violation from the deadline requirement of that task. To achieve these goals, two efficient priority-aware semi-greedy algorithms are proposed. The effectiveness of the proposed algorithms are evaluated through extensive experiments. The results demonstrated that the proposed algorithms significantly outperform existing algorithms in terms of the percentage of IoT tasks that meet their deadline requirement, the total energy consumption and makespan of the system, and the total amount of the deadline violation time. As future work, we plan to improve the proposed algorithms to schedule the dependent IoT tasks. We also aim to evaluate the performance of the proposed algorithms under different real-world datasets.

CRediT authorship contribution statement

Sadoon Azizi: Conceived of the presented idea, Writing – original draft, Made the simulations, Critical feedback, Shape the research, Analysis, Commented on the manuscript. **Mohammad Shojafar:** Involved in planning, Supervision, Contributed to the final version of the manuscript, Critical feedback, Shape the research, Analysis, Commented on the manuscript. **Jemal Abawajy:** Involved in planning,

Supervision, Contributed to the final version of the manuscript, Critical feedback, Shape the research, Analysis, Commented on the manuscript. **Rajkumar Buyya:** Involved in planning, Supervision, Contributed to the final version of the manuscript, Critical feedback, Shape the research, Analysis, Commented on the manuscript.

Declaration of competing interest

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

References

Aazam, M., Zeadally, S., Harras, K.A., 2018. Deploying fog computing in industrial internet of things and industry 4.0. IEEE Trans. Ind. Inform. 14 (10), 4674–4682.

Abdel-Basset, M., El-shahat, D., Elhoseny, M., Song, H., 2020a. Energy-aware metaheuristic algorithm for industrial Internet of Things task scheduling problems in fog computing applications. IEEE Internet Things J..

Abdel-Basset, M., Mohamed, R., Elhoseny, M., Bashir, A.K., Jolfaei, A., Kumar, N., 2020b. Energy-aware marine predators algorithm for task scheduling in IoT-based fog computing applications. IEEE Trans. Ind. Inform..

Aburukba, R.O., 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.

Adhikari, M., Mukherjee, M., Srirama, S.N., 2020. DPTO: A deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing. IEEE Internet Things J. 7 (7), 5773–5782.

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..

Alizadeh, M.R., Khajehvand, V., Rahmani, A.M., Akbari, E., 2020. Task scheduling approaches in fog computing: A systematic review. Int. J. Commun. Syst. 33 (16), e4583.

Almutairi, J., Aldossary, M., 2021. A novel approach for IoT tasks offloading in edge-cloud environments. J. Cloud Comput. 10 (1), 1–19.

Auluck, N., Azim, A., Fizza, K., 2019. Improving the schedulability of real-time tasks using fog computing. IEEE Trans. Serv. Comput..

Azizi, S., Shojafar, M., Abawajy, J., Buyya, R., 2021. PSG and PSG-M source code. https://github.com/S-Azizi/Sourcecodes/raw/main/Sadoon2021PSG_Sourcecode.zip.

Azizi, S., Zandsalimi, M., Li, D., 2020. An energy-efficient algorithm for virtual machine placement optimization in cloud data centers. Cluster Comput. 23 (4), 3421–3434.

Baccarelli, E., Naranjo, P.G.V., Scarpiniti, M., Shojafar, M., Abawajy, J.H., 2017. Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study. IEEE Access 5, 9882–9910.

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. Springer, pp. 169–186.

Bu, C., Wang, J., 2021. Computing tasks assignment optimization among edge computing servers via SDN. Peer-To-Peer Netw. Appl. 14 (3), 1190–1206.

Calheiros, R.N., Ranjan, R., Buyya, R., 2011. Virtual machine provisioning based on analytical performance and QoS in cloud computing environments. In: 2011 International Conference On Parallel Processing. IEEE, pp. 295–304.

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.

Cob-Parro, A.C., 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.

Deng, R., Lu, R., Lai, C., Luan, T.H., Liang, H., 2016. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. IEEE Internet Things J. 3 (6), 1171–1181.

Faramarzi, A., Heidarinejad, M., Mirjalili, S., Gandomi, A.H., 2020. Marine predators algorithm: A nature-inspired metaheuristic. Expert Syst. Appl. 152, 113377.

Gai, K., Qin, X., Zhu, L., 2020. An energy-aware high performance task allocation strategy in heterogeneous fog computing environments. IEEE Trans. Comput..

Gazori, P., Rahbari, D., Nickray, M., 2020. Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach. Future Gener. Comput. Syst. 110, 1098–1115.

Ghanavati, S., Abawajy, J., Izadi, D., 2020a. Automata-based dynamic fault tolerant task scheduling approach in fog computing. IEEE Trans. Emerg. Top. Comput..

Ghanavati, S., Abawajy, J.H., Izadi, D., 2020b. An energy aware task scheduling model using ant-mating optimization in fog computing environment. IEEE Trans. Serv. Comput..

Gia, T.N., Jiang, M., Rahmani, A.-M., 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.

Greenberg, A., 2008. The cost of a cloud: Research problems in data center networks. SIGCOMM Comput. Commun. Rev. 39, 68–73.

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.

Guevara, J.C., Torres, R.d.S., da Fonseca, N.L., 2020. On the classification of fog computing applications: A machine learning perspective. J. Netw. Comput. Appl. 159, 102596.

Hashimoto, Y., Aida, K., 2012. Evaluation of performance degradation in HPC applications with vm consolidation. In: 2012 Third International Conference On Networking And Computing. IEEE, pp. 273–277.

Hassan, H.O., 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.

Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H., 2019. Harris hawks optimization: Algorithm and applications. Future Gener. Comput. Syst. 97, 849–872.

Hoang, D., Dang, T.D., 2017. FBRC: Optimization of task scheduling in fog-based region and cloud. In: 2017 IEEE Trustcom/BigDataSE/ICESS. IEEE, pp. 1109–1114.

Hoseiny, F., Azizi, S., Shojafar, M., Ahmadiazar, F., Tafazolli, R., 2021a. 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.

Hoseiny, F., Azizi, S., Shojafar, M., Tafazolli, R., 2021b. Joint QoS-aware and cost-efficient task scheduling for fog-cloud resources in a volunteer computing system. ACM Trans. Internet Technol. 21 (4), 1–21.

Islam, M.S.U., Kumar, A., Hu, Y.-C., 2021. Context-aware scheduling in fog computing: A survey, taxonomy, challenges and future directions. J. Netw. Comput. Appl. 103008.

Jalali, F., Hinton, K., Ayre, R., Alpcan, T., Tucker, R.S., 2016. Fog computing may help to save energy in cloud computing. IEEE J. Sel. Areas Commun. 34 (5), 1728–1739.

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.

Kaur, N., Kumar, A., Kumar, R., 2021. A systematic review on task scheduling in fog computing: Taxonomy, tools, challenges, and future directions. Concurr. Comput: Prac. Exp. e6432.

Klincewicz, J.G., 1992. Avoiding local optima in thep-hub location problem using tabu search and grasp. Ann. Opera. Res. 40 (1), 283–302.

Konečnỳ, J., McMahan, H.B., Ramage, D., Richtárik, P., 2016. Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527.

Li, K., 2021. Heuristic computation offloading algorithms for mobile users in fog computing. ACM Trans. Embed. Comput. Syst. (TECS) 20 (2), 1–28.

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.

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.

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.

Marín-Tordera, E., Masip-Bruin, X., García-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.

Mishra, S.K., Puthal, D., Rodrigues, J.J., Sahoo, B., Dutkiewicz, E., 2018. Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications. IEEE Trans. Ind. Inform. 14 (10), 4497–4506.

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.

Nguyen, B.M., Thi Thanh Binh, H., Do Son, B., et al., 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.

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. Archit. 115, 101996.

Peng, L., Dhaini, A.R., Ho, P.-H., 2018. Toward integrated cloud–fog networks for efficient IoT provisioning: Key challenges and solutions. Future Gener. Comput. Syst. 88, 606–613.

Resende, M.G., Ribeiro, C.C., 2016. Optimization By GRASP. Springer.

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.

Shahryari, O.-K., Pedram, H., Khajehvand, V., TakhtFooladi, M.D., 2021. Energy and task completion time trade-off for task offloading in fog-enabled IoT networks. Pervasive Mob. Comput. 101395.

Shojafar, M., Cordeschi, N., Baccarelli, E., 2019. Energy-efficient adaptive resource management for real-time vehicular cloud services. IEEE Trans. Cloud Comput. 7 (1), 196–209.

Stankovic, J.A., Spuri, M., Ramamritham, K., Buttazzo, G.C., 2012. Deadline scheduling for real-time systems: EDF and related algorithms, Vol. 460. Springer Science & Business Media.
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.
Sun, H., Zhou, F., Hu, R.Q., 2019. Joint offloading and computation energy efficiency maximization in a mobile edge computing system. IEEE Trans. Veh. Technol. 68 (3), 3052–3056.
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.
Tang, M., Wong, V.W., 2020. Deep reinforcement learning for task offloading in mobile edge computing systems. IEEE Trans. Mob. Comput..
Vemireddy, S., Rout, R.R., 2021. Fuzzy reinforcement learning for energy efficient task offloading in vehicular fog computing. Comput. Netw. 108463.
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.
Yang, X., Rahmani, N., 2020. Task scheduling mechanisms in fog computing: review, trends, and perspectives. Kybernetes.
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.
Zhang, W., Zhang, Z., Zeadally, S., Chao, H.-C., Leung, V.C., 2019a. MASM: A multiple-algorithm service model for energy-delay optimization in edge artificial intelligence. IEEE Trans. Ind. Inform. 15 (7), 4216–4224.
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.
Zhao, W., Stankovic, J.A., 1989. Performance analysis of FCFS and improved FCFS scheduling algorithms for dynamic real-time computer systems. In: 1989 Real-Time Systems Symposium. IEEE Computer Society, pp. 156–157.

Sadoon Azizi received the M.Sc. degree in computer science, with focus on high-performance computing, and the Ph.D. degree in computer science, with focus on cloud data centers, from Amirkabir University of Technology, Tehran, Iran, in 2012 and 2016, respectively. He is currently an Assistant Professor with the Department of Computer Engineering and IT, University of Kurdistan, Sanandaj, Iran. He is also the director of the Internet of Things and Computing Systems Research Laboratory (IoTComSys Lab) and Head of the High-Performance Computing (HPC) center, University of Kurdistan. His current research interests include Cloud Computing, Fog/Edge Computing, Internet of Things, Green Computing, and Data Center Networks. For additional information: https://research.uok.ac.ir/~sazizi/en.

Mohammad Shojafar is a Senior Lecturer (Associate Professor) in the network security and an Intel Innovator, Fellow of Higher Education Academy, Professional ACM member, ACM Distinguished Speaker, and a Marie Curie Alumni, working in the 5G & 6G Innovation Centre (5GIC & 6GIC), Institute for Communication Systems (ICS), at the University of Surrey, UK. Before joining 5GIC & 6GIC, he was a Senior Researcher and a Marie Curie Fellow in the SPRITZ Security and Privacy Research group at the University of Padua, Italy. Also, he was CNIT Senior Researcher at the University of Rome Tor Vergata contributed to 5G PPP European H2020 "SUPERFLUIDITY" project. Dr. Mohammad is a PI of AutoTrust, a 750k euro 5G secure autonomous vehicular communication project supported by European Space Agency (ESA) in 2021 and was a PI of PRISENODE project, a 275k euro Horizon 2020 Marie Curie global fellowship project in the areas of Fog/Cloud security collaborating at the University of Padua, Italy. He received the Ph.D. degree from Sapienza University of Rome, Italy, in 2016 with an "Excellent" degree. His main research interest is in the area of Network and network security and privacy. In this area, he published more than 150 papers in topmost international peer-reviewed journals and conference, e.g., IEEE T-ITS, IEEE TCC, IEEE TNSM, IEEE TII, IEEE TGCN, and IEEE ICC/GLOBECOM (h-index=35, 4.9k+ citations). He is an Associate Editor in *IEEE Transactions on Network and Service Management*, *IEEE Transactions on Consumer Electronics*, *IEEE Systems Journal* and Computer Networks. He is a Senior Member of the IEEE. For additional information: http://mshojafar.com.

Jemal Abawajy is a full professor at school of Information Technology, Faculty of Science, Engineering and Built Environment, Deakin University, Australia. He is currently the Director of the Parallel and Distributing Computing Laboratory. He is a Senior Member of IEEE Computer Society; IEEE Technical Committee on Scalable Computing (TCSC); IEEE Technical Committee on Dependable Computing and Fault Tolerance and IEEE Communication Society. Professor Abawajy has served on the editorial-board of numerous international journals and currently serving as associate editor of the International Journal of Big Data Intelligence and International Journal of Parallel, Emergent and Distributed Systems. He has also guest edited many special issues. He is the author/co-author of five books, more than 250 papers in conferences, book chapters and journals such as IEEE Transactions on Computers and IEEE Transactions on Fuzzy Systems. He also edited 10 conference volumes. For additional information: https://www.deakin.edu.au/about-deakin/people/jemal-abawajy.

Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in Cloud Computing. He has authored over 850 publications and seven text books including "Mastering Cloud Computing" published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets respectively. Dr. Buyya is one of the highly cited authors in computer science and software engineering worldwide (h-index=152, g-index=332, and 120,900+ citations). Dr. Buyya is recognised as Web of Science "Highly Cited Researcher" for six consecutive years since 2016, IEEE Fellow, and Scopus Researcher of the Year 2017 with Excellence in Innovative Research Award by Elsevier. He has been recognised as the "Best of the World" twice for research fields (in Computing Systems in 2019 and Software Systems in 2021) as well as "Lifetime Achiever" and "Superstar of Research" in "Engineering and Computer Science" discipline twice (2019 and 2021) by the Australian Research Review. Recently, he received "Research Innovation Award" from IEEE Technical Committee on Services Computing and "Research Impact Award" from IEEE Technical Committee on Cloud Computing. Software technologies for Grid, Cloud, and Fog computing developed under Dr. Buyya's leadership have gained rapid acceptance and are in use at several academic institutions and commercial enterprises in 50+ countries around the world. Manjrasoft's Aneka Cloud technology developed under his leadership has received "Frost New Product Innovation Award". He served as founding Editor-in-Chief of the IEEE Transactions on Cloud Computing. He is currently serving as Editor-in-Chief of Software: Practice and Experience, a long standing journal in the field established 50+ years ago. For further information on Dr. Buyya, please visit his cyberhome: www.buyya.com.