The Journal of Supercomputing (2022) 78:1983–2014
https://doi.org/10.1007/s11227-021-03941-y

A survey on computation offloading and service placement in fog computing-based IoT

Kaouther Gasmi[1] · Selma Dilek[2] · Suleyman Tosun[3] · Suat Ozdemir[3]

Accepted: 8 June 2021 / Published online: 24 June 2021

Abstract
In recent years, fog computing has emerged as a computing paradigm to support the computationally intensive and latency-critical applications for resource limited Internet of Things (IoT) devices. The main feature of fog computing is to push computation, networking, and storage facilities closer to the network edge. This enables IoT user equipment (UE) to profit from the fog computing paradigm by mainly offloading their intensive computation tasks to fog resources. Thus, computation offloading and service placement mechanisms can overcome the resource constraints of IoT devices, and improve the system performance in terms of increasing battery lifetime of UE and reducing the total delay. In this paper, we survey the current research conducted on computation offloading and service placement in fog computing-based IoT in a comparative manner.

Keywords Fog computing · Computation offloading · Internet of Things (IoT) · Service placement · Optimization algorithms

✉ Suleyman Tosun
stosun@cs.hacettepe.edu.tr

Kaouther Gasmi
kaouther.gasmi@enit.rnu.tn

Selma Dilek
selmadilek@hacettepe.edu.tr

Suat Ozdemir
ozdemir@cs.hacettepe.edu.tr

1 Higher Institute of Applied Languages and Computer Sciences in Beja, Tunis, Tunisia

2 Graduate School of Science and Engineering, Hacettepe University, Beytepe, Ankara, Turkey

3 Department of Computer Engineering, Hacettepe University, Ankara, Turkey

1 Introduction

It has been estimated that the number of deployed smart devices connected through the Internet of Things (IoT) will be around 75 billion by 2025 [1], while the amount of data generated by these devices is expected to be 73.1 zettabytes per year [2], as opposed to 1.1 zettabytes of data per year estimated in 2016 [3]. Most of this massive amount of data requires real-time processing to make efficient decisions [4]. However, the majority of the end devices in IoT are battery powered and have limited processing and storage capabilities. Therefore, they are unable to satisfy the requirements of the applications that need massive data processing. Hence, it is necessary to utilize a resourceful computing paradigm that enables processing the data generated from the end IoT devices. This paradigm is often called cloud computing [5]. The cloud computing paradigm provides extensive processing power and unlimited storage through powerful virtual servers, which enables fast processing and unbounded storage. It can be useful for applications that are not delay sensitive and do not require higher responsiveness. However, it may not be an attractive solution for applications that require real-time processing and high responsiveness, due to the high network bandwidth usage and large end-to-end delay for continuously pushing large bulks of raw data. For this purpose, fog computing [6] has been designed as a promising computing paradigm for tackling the aforementioned issues related to limited device resources, limited bandwidth, and large end-to-end latency, by bringing computation, storage, and networking services directly to the network edge. Here, we describe "Edge" as any intelligent computing resources along the path between data sources and cloud data centers. These computing resources are referred to as edge devices. Some examples of edge devices include routers, switches, and smart gateways.

As fog-based systems ensure additional computing capabilities at the edge of a network, users can profit from the advantages of fog computing mainly by means of computation offloading[1], which is a mechanism that can overcome the problem of resource constraints at the edge devices. Edge devices are connected to devices that have computation, storage, and networking capabilities such as routers and switches. Specifically, it can help to improve the performance of computation-intensive applications and battery life. However, selection of the tasks to be offloaded is a challenging problem. Generally, the purpose of decision making on task offloading is to ascertain whether the offloading is cost-effective for the user equipment in respect of energy consumption and execution delay [7].

Another challenge in computation offloading is making a decision about which node to assign to the offloaded task (i.e., service placement[2] problem), while considering the selected metrics such as energy consumption and/or latency. Since the fog nodes are geographically distributed, resource-constrained, and highly dynamic,

[1] We use computation offloading and task offloading interchangeably throughout this paper, since both terms are used by different studies.

[2] We use service placement, application placement, and task placement interchangeably throughout this paper, since all expressions are used by the previous studies with the same meaning.

service placement problem becomes extremely difficult to solve. Several studies have addressed the aforementioned problem. Different strategies and techniques have been considered with a goal of designing efficient computing offloading and service placement schemes in fog environment. In this survey, we review a broad range of the recent existing studies that focus on these issues, and explore the strategies and techniques proposed in the literature. First, we present the concept of fog computing and application domains. Second, we consider application models for offloading, computation offloading decisions, and provide a classification of the various algorithms proposed for computation offloading. Finally, we discuss optimization methodologies and performance metrics for service placement, and present a classification scheme for service placement methodologies and techniques.

We can list the main contributions of this survey as follows:

1. We provide an exhaustive overview of the computing offloading problem, by identifying several factors that can affect the offloading decision, presenting the key application models for offloading to fog environment, and classifying the algorithms and techniques that have been proposed recently for the computing offloading problem. The proposed classification is based on the offloading type and objective parameters.
2. We propose a novel classification for the optimization strategies that have been proposed to solve the service placement problem for IoT applications over fog nodes. This classification is based on the used methods and the optimization objectives.
3. Finally, we highlight the open challenges and discuss the future research directions in fog-based systems.

The rest of the paper is organized as follows. Section 2 summarizes existing surveys on fog computing. Section 3 presents an overview of fog computing paradigm: definition, architecture, main characteristics, advantages, application domains, as well as the differences and similarities with related computing paradigms. Section 4 introduces the concepts of computation offloading: application models and the proposed methodologies for designing computation offloading schemes. Section 5 introduces the service placement problem, performance metrics, and proposed optimization methods. Finally, Sect. 7 concludes the paper.

2 Related work

Different aspects and challenges of fog computing have been addressed by several surveys. Mahmud et al. [8] focus on key services that fog computing provide. Further, they present the major factors that have been considered for efficient resource and service provisioning in fog. Varshney and Simmhan [9] state the characteristics and requirements of applications that push the need for using the fog environment. Hong and Varghese [10] classify the architectures, infrastructure, and underlying algorithms for managing resources in fog/edge computing.

Ren et al. [11] discuss the state-of-the-art research in terms of computation offloading, caching, security, and privacy in edge computing and related computing technologies. Yi et al. [12] mainly discuss the issues related to fog networking. Mouradian et al. [13] review the proposed architectures and algorithms for building fog. The authors in [14, 15] summarize the proposed approaches for resource management such as resource allocation, task scheduling, resource provisioning, task offloading, and application placement, with a focus on the relevant resource management issues. In [16], the authors provide a comprehensive review of existing literature that use stochastic offloading mechanisms in computation paradigms. Furthermore, they provide a comprehensive comparison of offloading mechanisms based on Markov chain. The authors of [17] present an exhaustive overview of IoT–Fog–Cloud ecosystems and mainly discuss the standards, tools, and applications that have been used. In [18], the authors provide a comprehensive overview of various aspects of application management. They present an extensive exploration in IoT applications architecture distributed over fog nodes. They discuss programming models, service types, workload types, interaction methods, and functional layouts. Moreover, they overview the relevant elements associated with application placement such as mapping techniques and placement strategies. Ranesh Kumarnaha et al. [19] present a comprehensive review of existing literature with a focus on the requirements perspective related to infrastructure, platform, and application in fog computing. They further summarize some existing research studies in resource allocation, scheduling, and fault tolerance in fog with some application examples. The survey in [20] summarizes the characteristics of fog architecture and presents the main similarities with and differences from cloud. The paper also presents the key technologies applied in fog. In [21], the authors give a comparative discussion about fog computing and its related computing paradigms. Moreover, they summarize the software and tools for fog computing, as well as the applications that are exploited for the fog technology. Additionally, they investigate the problem of resource management, security, and privacy. The authors of [22] take a closer look at networking, latency, and energy consumption models in fog computing, and discuss the issues related to service allocation and resource management in a fog infrastructure. Several others surveys discussed similarities and differences of fog computing compared to the related computing paradigms, fog architectures, application domains, as well as the challenges and open issues [22–25].

Although some of the mentioned surveys explore the computation offloading and service placement problem in fog computing, we notice that these studies are limited in the following perspectives:

- They omit the comprehensive summary of the relevant elements of the offloading operation such as application model definitions, resolution strategies, offloading decision, and offloading type.
- Additionally, the aforementioned surveys do not provide an exhaustive review regarding the optimization strategies and their technical formulations for application placement. In particular, the machine learning-based intelligent solution strategies are neglected.

- They do not provide an insight into how to choose the best optimization strategy for the application placement under different objectives, as one optimization method may be better than the other for certain types of applications/scenarios.

The aforementioned limitations of the existing surveys motivated us to initiate this survey which is mainly dedicated to achieving an exhaustive and comprehensible overview of relevant computing offloading aspects and the service placement problem in fog computing.

3 Overview of fog computing

Fog computing, proposed first by Cisco in 2012 [26], is defined as a distributed computing infrastructure that extends cloud-like services to the network edge by delivering computation and storage resources closer to users [27]. According to the OpenFog Consortium [28],

> Fog Computing is a horizontal, system-level architecture that distributes computing, storage, control, and networking functions closer to the users along a Cloud-to-Thing continuum.

As opposed to cloud computing, fog computing introduces fog servers and fog nodes, which are devices physically closer to the users than their cloud counterparts, with a goal of handling some of the application workload closer to the network edge. Any device such as a controller, smart gateway, switch, router, or embedded server, which possesses capabilities for processing, networking and storage, may be employed as a fog node. The advantage of these devices is wider deployment opportunities, as they can be placed at any location where network connection is available, such as inside factory buildings, alongside railways, inside of vehicles, and even on power poles. The main idea behind this approach is to optimize the transmission time it takes for data to reach the intended processing nodes, since deploying nodes closer to the edge of a network shortens the data transmission time to a negligible delay [27]. According to OpenFog, fog nodes are equipped with intelligent algorithms that facilitate processing and storing data, as well as forwarding data from edge devices to fog, and from fog to cloud via smart networking.

3.1 Fog architecture

A number of architectures for fog computing have been proposed, most of which have a three-layer structure as illustrated in Fig. 1. A typical fog architecture comprises an IoT layer (also known as end layer) that consists of IoT devices, a fog layer that consists of fog nodes, and a cloud layer with a cloud data center and services [29], which are discussed in more detail below.

- *IoT layer* It is the lowest layer located at the nearest proximity to the end users. It incorporates various sensor nodes (temperature and humidity sensors, cam-

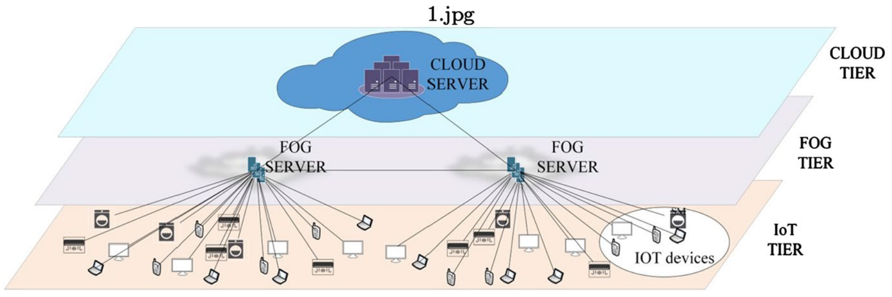

Fig. 1 Illustration of a fog computing architecture

eras, etc.) and actuators that may be distributed over a wide geographical area. Devices placed at this layer are responsible for collecting data about their environment, and dispatching the gathered data to the upper layer for further processing and/or storage.

- *Fog layer* It is an intermediary layer between a cloud and end devices, which consists of heterogeneous fog devices (e.g., access points, switches, routers, gateways, base stations, fog servers, etc.) that have relatively limited processing, storage, and communication capabilities. These fog nodes maintain a connection to cloud servers and can forward requests to cloud data centers. Fog nodes that have very limited processing, storage, and other capabilities are considered to be low-level nodes, while nodes with more abundant resources are classified as high-level nodes. Fog nodes may also be categorized as either stationary (if fixed at a certain location), or mobile, such as smartphones, vehicles, and drones [29, 30].
- *Cloud layer* It is the top layer in a fog computing architecture. Cloud layer incorporates a number of servers and data centers, which are capable of executing complex processing and analysis, storing huge amounts of data, and providing feedback and results back to fog nodes [29, 30].

Considering this layered fog architecture, it is evident that fog services are located in a much closer proximity to the end users, allowing a denser geographical distribution, and offering a better mobility support, as opposed to a cloud environment in which services may reside in much farther locations necessitating wider bandwidths. Furthermore, a fog ecosystem also contributes to reduced latency and provides context awareness due to fog node localization, resulting in pervasive, scalable, and united network connectivity [3].

3.2 Key characteristics and advantages of fog computing

The distinct characteristics and advantages of fog computing can be listed as follows.

- *Low latency*: The key motivation behind this emerging paradigm is to decrease the data transmission latency while increasing the data transmission rate [31].

Some novel applications such as intelligent vehicle-to-vehicle communication networks, virtual reality, and online gaming have an extremely high requirement for low delay. For instance, in a virtual reality application, a small delay in range of milliseconds can damage the user experience [23]. Due to the close proximity of fog nodes to end users, fog computing is capable of providing support for low latency and time-sensitive applications [11].

- *Support for mobility*: In applications such as vehicular networks the movement of the nodes can considerably affect the system performance, especially in scenarios that require handling fast channel changes [11]. Fog computing can facilitate the mobility of the end users through providing computational and storage resources over the entire network, as opposed to a traditional centralized cloud ecosystem [29].
- *Bandwidth*: Preprocessing of data before transferring it to a cloud for further analysis or storage plays an important role in reducing the network traffic. Fog computing enables data filtering and aggregation to be performed locally in order to speed up the execution of certain tasks that would otherwise take too long under a limited network bandwidth [29].
- *Scalability*: There are IoT scenarios in which a huge number of end users needs to be managed along with enormous amounts of data produced by billions of heterogeneous IoT objects with different performances and costs. In such scenarios, having fog nodes widely distributed and deployed in close proximity to the users plays an important role in making IoT applications scalable and adaptable to the network changes [11].

3.3 Fog computing issues

Fog computing paradigm promises to offer effective solutions for a number of problems that afflict IoT and cloud computing applications. Nevertheless, it also comes with its own challenges that need to be grasped and tackled in order for it to become an effective computing solution. Some of the fog-specific issues are discussed below.

- *Resource management* Fog environment contains heterogeneous fog nodes with limited computing and storage performance capabilities, which makes it harder to manage resource allocation, scheduling, and sharing. Efficient management solutions are needed for determining appropriate task placement strategies that satisfy applications' requirements (e.g., solutions based on priority and migration) [32]. Furthermore, in order to provide mobility support, particularly in the case of connected objects, resources may need to be pre-allocated using effective methods such as probabilistic ones based on user history [33]. Therefore, a framework that enables the performance evaluation of resource management policies in IoT or fog computing infrastructures is a necessity [3].
- *Privacy and security* Some of the proposed solutions for security issues in fog computing focus on intrusion detection, access control, authentication, and detection of various other malicious activities including denial of ser-

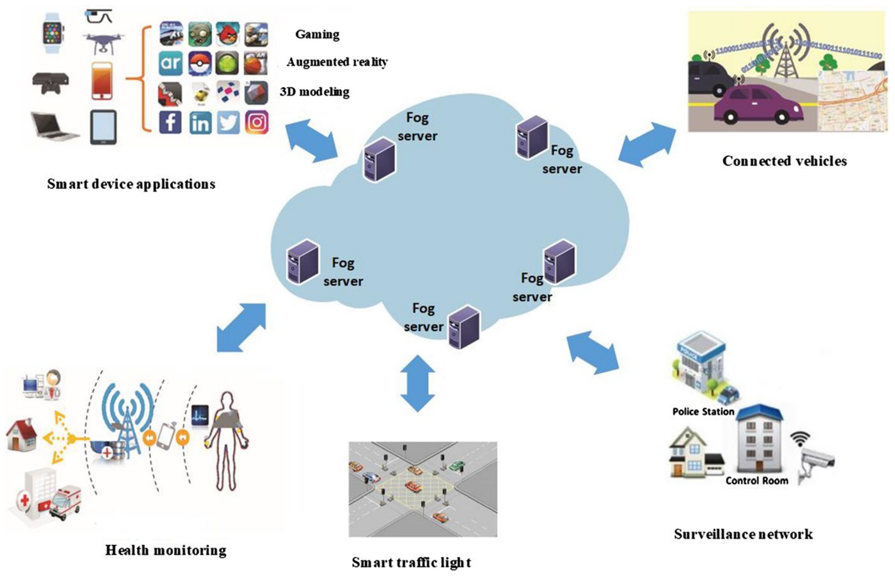

Fig. 2 Applications benefiting from fog computing paradigm

vice (DoS) attacks and port scanning. Introducing security mechanisms in fog computing is necessary for each level of the fog architecture [12].

- *Energy management* A fog infrastructure incorporates a large number of geographically distributed nodes. As a result, energy consumption in a fog ecosystem is higher in comparison with that of a cloud [23, 33]. Significant research efforts are needed to develop effective solutions for energy management. For example, data processing and communication protocols that are less costly in terms of energy consumption need to be developed [33].
- *Quality of service (QoS)* One of the most important measures of network service quality in general is QoS, and this holds for fog computing as well. In [34], the authors consider QoS in fog computing based on four metrics: connectivity, reliability, capacity, and latency. Other metrics such as energy efficiency, bandwidth, and security may be taken into consideration as well, depending on the application requirements. The challenge with QoS provisioning often entails making trade-offs between different QoS metrics, which is further complicated by a varying and often dynamic nature of different applications.

3.4 Applications benefiting from fog computing

Figure 2 illustrates a scope of applications that stand to gain from the fog computing paradigm. In this subsection, we discuss some of those applications that can benefit from fog computing to a great extent.

3.4.1 Healthcare

Healthcare applications and services, especially applications for remote monitoring of critical patients whose physiological states may rapidly deteriorate necessitating agile response and decision making, are time-critical applications that demand real-time processing. In unpredictable network conditions, delivery of health data may become subject to high latency, making the data deficient, unreliable, and in some cases even useless. This issue may incur even worse outcomes for data that requires cascade-based analysis such as electrocardiogram (ECG) or electroencephalogram (EEG) signals [35, 36]. This type of problematic scenarios may be prevented through employing solutions based on fog computing.

Cao et al. [37] proposed a fog-based system called FAST, which enables detection, prediction, and avoidance of falls for patients who suffered a stroke. Their fall detection learning algorithm was distributed across both edge and cloud resources, enabling a shorter response time in comparison with other proposed solutions that are solely based on cloud implementation.

3.4.2 Augmented reality

Fog computing may also play a major role in enhancing the performance of augmented reality applications, which are highly delay-sensitive, as the user experience depends mainly on real-time response [33]

An example is the Augmented Brain Computer Interface developed by Zao et al. [38], which is capable of detecting the states of a user's brain in real-life situations using data collected through EEG headsets. Since both fog and cloud servers are utilized, the system is capable of perpetual real-time classification of a user's brain state performed at the fog layer, while employing cloud servers for regular tuning of classification models.

3.4.3 Traffic management system

The efficiency of a traffic signaling management system relies heavily on achieving real-time and location-aware system response. Fog computing can improve intercommunication between vehicles, access points, and traffic signals for the overall enhancement of such systems [39].

3.4.4 Caching and processing for improved networking

Fog computing may also facilitate caching and processing operations for websites that work with large databases and huge amounts of data that needs to be processed, such as social networking, library, or online shopping websites, with the goal of improving their performance by reducing overall time and space complexity [40]. Zhu et al. [41] proposed one such solution in which users connect to the Internet through devices placed in the fog layer, which act as relays for all HTTP requests. The fog devices improve the response time through employing various optimization tasks depending on the scenarios and network conditions. For instance, if congestion

Table 1 Comparison of fog, MEC, and cloudlet [13]

Paradigm	Location for computing	Virtualization	Operation mode	Target applications
Fog	Devices along the routing path	VM, container	Connected to cloud	Mobile offloading applications, any application better provisioned at the edge
MEC	Base stations and nearby devices	VM, container	Standalone	Mobile offloading applications, any application better provisioned at the edge
Cloudlet	Nearby devices	VM	Standalone or connected to cloud	Mobile offloading applications

occurs in the network, users may be provided lower-resolution graphics by a fog device in order to maintain acceptable response time. Another example of optimization would be choosing appropriate resolution for graphics based on the browser's rendering capabilities on a client machine [33].

3.5 Related technologies

Other middleware technologies such as mobile edge computing (MEC) and Cloudlets that have been proposed in the literature also fall within the scope of fog computing [3]:

- *Cloudlets* refer to servers or clusters of servers that have plenty of resources and are deployed in a single-hop proximity of mobiles users. They employ virtual machines (VMs) for enabling mobile devices to offload tasks in applications that require intensive computation [42]. The key motivation for employing cloudlets is to enable mobile devices that have scarce computational and storage resources to offload their intensive computation to the cloudlets, especially in applications that have stringent end-to-end latency requirements, with the goal of guaranteeing real-time interactive responses, which is something a distant cloud environment cannot guarantee. Another advantage of cloudlets is the fact that they can exist as a standalone environment acting as a full cloud at the edge of a network without cloud interaction, even though they actually reside in the middle of a three-tier hierarchy (mobile device - cloudlet - cloud) [13].
- *Mobile edge computing (MEC)* was put forward by the European Telecommunication Standards Institute (ETSI) with the goal of enabling cloud computing capabilities closer to the mobile subscribers and within the radio access network (RAN) [13]. MEC services can be set up at different platforms such as LTE base stations (eNodeB) and 3G radio network controllers (RNC) [13]. The main goal of MEC is to improve application performance and consequently user experience by enabling the processing of user requests at the network edge, and as a result, reducing network congestion [3]. Some of the target applications include augmented reality and video processing.

A brief comparison between Cloudlets, MEC, and fog computing is given in Table 1.

4 Computation offloading for fog computing paradigm

With the emergence and rapid proliferation of IoT applications, the necessity to efficiently manage the execution of increasingly complex tasks based on the varying requirements of different applications has also risen. Some of these differences stem from the varying capabilities of user devices. For instance, some devices such as smartphones may require faster response necessitating additional computational resources, whereas some devices may need to be able to accumulate and process data before dispatching it to the cloud, thus, requiring creation of local services [43]. In order to free mobile devices with limited resources from performing complex tasks, computation offloading has been proposed by both industry and academia as a promising solution for effective integration of resources in computing ecosystems that are designed based on edge-fog-cloud paradigm. Offloading of computation-intensive tasks from constrained end devices to fog or cloud servers can significantly save energy and reduce response time [11].

4.1 Computing offloading decision

Offloading decision could be affected by several factors. This subsection discusses principal determinants and criteria used in making decisions whether proceeding with offloading or avoiding it would be more cost-effective.

4.1.1 Latency requirements

IoT applications that have strict latency requirements in the order of milliseconds can be considerably affected by the distance between nodes. In order to reduce end-to-end latency in such applications, offloading tasks such as data analytics at the network edge can prove to be very effective and lead to much faster response [44]. An example would be offloading a part of or the entire content from a multimedia services cloud to be cached at an edge or fog node in order to bring the content to a closer proximity to users for faster access [45].

4.1.2 Load balancing

Load balancing within a service provider's ecosystem plays an important role with regards to optimizing processor throughput, response time, and resource utilization, as well as prevention of node overload, since a device that has reached its processing capacity is not capable of performing any additional tasks. In such scenarios, tasks could be appropriately distributed among multiple nodes/servers in a fog data center with the goal of balancing the load of incoming requests [45].

4.1.3 Intensive computation and other resource constraints

With the resource-constrained IoT devices, it is often the case that computation or storage requirements of an application exceed the capabilities of the device. In such cases, it is necessary to offload demanding tasks to other nodes that have more available resources (e.g., utilizing a cloud for updating map according to satellite data in a geographical map service application on a smartphone) [44, 45].

4.1.4 Privacy and security

In scenarios in which privacy and security concerns are raised, offloading may be preferred based on the susceptibility and privacy of data or tasks. For instance, data handled in a hospital or company may be transferred from a local storage to a private cloud. Likewise, a user may prefer to store private data in a personal mobile edge cloud instead of a smartphone [46].

4.1.5 Long-term storage

Depending upon the service type, long-term storage may take a lot of space. Thus, it may not be practical or sometimes even possible to satisfy storage requirements on small end devices such as smartphones. Offloading storage tasks in such cases to a cloud server that has abundant storage space for allocation is a practical solution [45].

4.1.6 Network bandwidth

In IoT applications, end users generate enormous amounts of data; however, these systems usually have limited bandwidth. Therefore, enabling data preprocessing and analytics at the network edge/fog layers can considerably minimize the network traffic load, and improve the overall network performance [44].

4.1.7 Energy efficiency

Energy is another important indicator that must be considered when making offloading decisions. The existing studies show that users usually prefer longer battery life over other features. A distributed nature of a fog environment that integrates many fog nodes enables distributed and more energy-efficient computation models as opposed to a centralized cloud-based model of computation [33]. Figure 3 illustrates how energy consumption affects decision making in fog. In this figure, we assume that the tasks of IoT devices (i.e., tablet, smart phone, and VR glasses) are partitioned into six sub computations (C1-C6). Then, it is decided which sub-computations will be offloaded based on the energy concern. Basically, a computation offloading decision may prevail on the following:

- *Local execution* The entire computation is executed on a local device instead of performing offloading to fog servers. This situation occurs either when off-

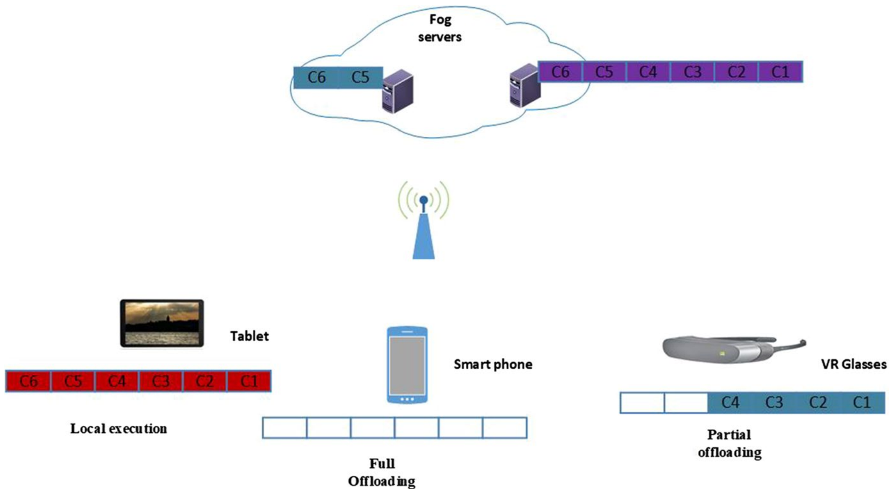

Fig. 3 Possible outputs of computing offloading decision for different application components at mobile devices

loading is not cost-effective or when fog resources are unavailable. The tablet in Fig. 3 executes all computations locally.

- *Partial offloading* Offloading only a part of the whole computation to fog nodes, while having the rest of the computation tasks executed locally. The VR glasses in Fig. 3 execute four computations locally and offload the remaining two to the fog server.
- *Full offloading* Offloading the entire computation to be processed by fog nodes. The smart phone in Fig. 3 offloads all computations to the fog server.

4.2 Application models for computation offloading

Generally, mobile applications can be divided into *N* components, which makes it possible to implement partial/full offloading. Each component may differ in the amount of computation and latency requirements. Therefore, it is necessary to determine which components should be offloaded. As illustrated in Fig. 3, the first, second, third, and fourth components of the application at VR glasses are locally executed, while the fifth and sixth components are offloaded to the fog network. Likewise, all components can be fully offloaded in case of insufficient resources on mobile devices. In many applications, there are also some components that cannot be offloaded in any case (e.g., data I/O, camera, etc., which must be processed locally). Furthermore, some components are dependent on each other, and this dependency cannot be neglected in the offloading process, especially in cases when the outputs of some components act as inputs to other components, otherwise the execution performance could suffer greatly. Three prominent models for partitioning of applications in the process of offloading are task graph, inter-dependent modules, and deep neural network [10].

Task graph The simplest model for application partitioning is the task graph, usually represented as a directed acyclic graph (DAG), which is a widely used construction in cloud and fog computing for describing dependencies between components in complex distributed applications. In [47–50], the task-graph-based models were applied to model applications, which denote the dependency between different sub-tasks and the automated partitioning strategies for generating an optimal offloading. Furthermore, DAG can contain other relevant information in its vertices such as the number of necessary CPU cycles and the amount of required memory, as well as in its edges such as representing the amount of I/O data as edge weights [51].

Inter-dependent modules In this model, a large-scale application is partitioned into a set of modules that are deployed over the distributed nodes. Each component has different size and computation complexity. The dependencies between modules can be expressed as a unidirectional data flow. After the deployment of all modules, each module is expected to finish its computation within a fixed time. For example, [52] employs an inter-dependent module-based model to deploy a set of applications over fog nodes. The set of components represents the functionalities of applications, and each output of a module is served as an input to another module. The module management problem is tackled using linear programming and a heuristic approach, while considering both strict deadline and resource optimization.

Deep neural network (DNN) model DNN model has become increasingly popular as the core machine learning technique. It can be deployed as multilayers, distributed over different nodes, where each node is a processing element called *neuron* that applies a function to its inputs and generates an output. Different neurons are directly connected, and each connection defines a flow of data between two neurons. The output of a layer serves as an input to the next layer. The depth of a DNN is determined by the number of layers. An example application is a fog computing based industrial manufacture inspection application, which detects possible defects regarding the product, and sends back recommendations and production status [53]. This application is implemented as a 6-layer DNN model. The application computations are distributed among mobile devices, fog nodes, and a cloud server based on the computation costs. The main idea is to enable processing immensely large amount of data in real-time, which is achieved through adapting a deep learning-based classification model the fog computing paradigm with the goal of alleviating the computational burden of the central servers with the help of fog nodes.

Likewise, another research works focus on the distribution of the DNN across user devices, fog and cloud nodes, particularly to achieve low end-to-end latency and high energy efficiency [54, 55]. In [54], the authors designed a DNN-based Neurosurgeon framework to automatically partition the computation between mobile devices and fog nodes. The proposed Neurosurgeon framework models the partitioning between layers in order to achieve low end-to-end latency and energy efficiency. Their idea was to consider different partitioning points in the network and predict the energy consumption at each of those points, so that a partitioning which optimizes data transfer delay and minimizes energy consumption can be chosen. Similarly, their study was extended by employing distributed DNN across hierarchical fog nodes in [55].

A mobile application can be considered as a sequence of tasks. A task can be represented as $T\langle I, D, W\rangle$ where I stands for the size of the input in bits, D for the completion deadline of the task in seconds, and W for the computation workload expressed in CPU cycles per bit. Tools such as task profilers can be used to estimate these parameters, which are essential parts of applications in terms of capturing the fundamental properties of mobile applications including computation and communication requirements. Furthermore, they can also assist with estimation of execution latency and energy consumption [51].

4.3 Proposed offloading algorithms and techniques

In this subsection, we present several offloading algorithms and techniques for fog computing recently proposed in the literature. Similar to [11], we classify the existing studies into three categories according to their optimization objectives: (1) studies that focus solely on minimizing energy consumption; (2) studies whose primary aim is to minimize delay; and (iii) studies that consider both energy and delay as metrics for optimization. Table 2 summarizes the surveyed studies.

4.3.1 Minimizing energy consumption

Minimizing the energy consumption in the computation offloading process is an optimization problem that has caught a lot of attention in research. Zhao et al. [56] proposed a full offloading algorithm that minimizes the energy consumption of mobile devices in a fog/cloud system while meeting the application's latency and maximum transmission power requirements. The offloading decision to either fog or cloud is made dynamically based on the computed energy consumption of both models. Offloading to fog entails data transfer from the device to a fog node, the device idle time energy consumption, and the energy consumption that results from the actual computation at the fog node. Offloading to a cloud node, on the other hand, involves additional data transfer from the fog node to the cloud, idle time energy consumption of both device and fog node, as well as the energy consumed during the execution of the computation at the cloud. The experimental testing done through simulation showed that the proposed algorithm improves energy consumption of the system over a model solely based on cloud computing.

Chang et al. [57] designed a distributed algorithm based on the alternating direction method of multipliers (ADMM) technique to offload tasks in multi-user fog systems with the main purpose of minimizing energy consumption while satisfying the latency constraint. A queuing model is used to efficiently model the energy consumption at a mobile device and fog nodes. The simulation results have shown that the energy consumption is decreased when the offloading probability is increased while using a lower transmission power.

Table 2 Overview of the papers that focus on fog-based computation offloading

Design objective	Offloading type	Ref.	Proposed algorithm
Energy	Full	[56]	A minimum energy consumption oriented algorithm that enables making offloading decisions in fog-cloud or only in cloud
		[57]	A distributed algorithm based on alternating direction method of multipliers (ADMM) technique, which decides on fog offloading while satisfying the delay constraint
Delay	Full	[58]	A real-time signal processing algorithm for fog offloading
		[59]	A Latency Aware Workload Offloading algorithm in the Cloudlet Network
Energy and delay	Full	[60]	A machine learning-based algorithm for secure offloading in fog/cloud systems
		[61]	A game theory-based distributed algorithm admitting Nash equilibrium for offloading in fog systems
		[62]	A nonlinear programming method for joint optimization of energy, delay, and the offloading cost
		[63]	A game theory-based offloading algorithm and the price of anarchy technique to quantify the distance between the proposed scheme and the optimal performance
	Partial	[64]	An algorithm for partial task offloading based on DVS technique
		[65]	A mixed-integer nonlinear programming-based algorithm for task offloading
		[66]	A game theory-based distributed offloading algorithm in fog

4.3.2 Minimizing delay

Minimizing the latency in healthcare applications is of a crucial importance, especially in emergency response scenarios. Craciunescu et al. [58] investigated offloading cloud tasks such as storage and processing of healthcare data closer to the network edge (to fog nodes) in order to decrease the application latency and guarantee a quick response for patients in case of an emergency. They tested the proposed method in an e-Health laboratory and achieved a decrease in latency of two to four seconds in comparison with the cloud-only implementation.

Another challenging problem associated with making offloading decision for end-to-end latency optimization is selection of optimal fog site that will take the workload. If a number of users aggregate at a single fog node (e.g., a cloudlet), while leaving other nodes unused, this may lead to overloading some nodes while wasting the capacity of the others [11]. Sun and Ansari [59] investigated this problem and employed software defined networking (SDN) technology to propose a latency-aware offloading solution for mobile devices in an ecosystem of geographically distributed cloudlets. Their objective was to minimize the average response time which entails both network and cloudlet delays by selecting optimal cloudlets to be allocated for computational workload.

Harvesting renewable energy was explored by Mao et al. [67]. They developed a low-complexity dynamic computation offloading algorithm based on Lyapunov optimization, designed for mobile devices that are powered with renewable energy. The algorithm adjusts the CPU clock frequency, transmission power, and computation offloading decision with the goal of minimizing energy cost, execution delay, and task failure. They showed that the proposed algorithm is asymptotically optimal and provides improved performance in regard to energy conservation and decreasing offloading failure.

4.3.3 Jointly minimizing energy consumption and delay

A number of studies have investigated fog-based computation offloading taking both energy consumption and delay into account. Machine learning techniques have been employed in studies that propose offloading solutions that minimize delay and energy consumption in fog environments. In [60], Alli and Alam (2019) proposed a secure offloading solution for a fog-cloud environment. A machine learning-based particle swarm optimization (PSO) algorithm is designed to choose an optimal fog node to which offloading will incur the minimum processing delay between a mobile device and the fog. A fog node which has a high available computing capacity and maintaining energy is selected as an optimal fog node. In case the selected fog node is not capable of handling the incoming workload, a dynamic offloading to cloud servers based on Q-learning mechanism comes into play. During the offloading process, first a classification of the task is employed based on the task size, complexity, and latency. Then, the tasks are offloaded to either a private or public cloud accordingly. Finally, the authors applied a neural-fuzzy model to evaluate the security of the incoming tasks before offloading. The simulation results confirmed that the proposed approach reduces the delay and energy consumption compared to other existing solutions.

The joint minimization of energy consumption and delay in fog environment is also investigated in [62], where the authors formulated a multi-objective optimization problem using the nonlinear programming method to simultaneously optimize the energy consumption, latency, and the offloading cost. They applied queuing models to different elements of the fog network in order to deeply study the system cost. The simulation results showed that by adjusting the offloading probability and transmission power, the trade-off between energy, delay, and cost can be optimized.

Ma et al. [66] proposed a computing offloading strategy that enables multiple homogeneous and heterogeneous mobile devices to offload their computation to multiple wireless access points (APs) with the main purpose of minimizing energy consumption and delay. The total energy consumed during the offloading process consists of the transmission energy, the scanning energy of the APs, and the maintaining energy consumed to uphold the interface during transmission. They designed distributed offloading algorithms based on game theory in the context of homogeneous and heterogeneous mobile devices. The performance of the proposed algorithms is compared with local computation algorithm and random selection (the mobile user chooses one AP). The simulation results revealed that in the context of heterogeneous devices the proposed offloading policy can optimize the energy and delay compared to other algorithms, while increasing the number of mobile devices; however, in the context of homogeneous devices, it can improve the system cost for only up to ten mobile devices by 20% compared to others algorithms.

With the same goal of the aforementioned study, the problem of minimizing energy consumption and performance delay was similarly formulated using the game theory approach in [61, 63]. In [63], the authors designed an offloading scheme to optimize the system cost in terms of energy consumption and delay of the offloading process. Furthermore, the authors used the price of anarchy of the proposed scheme to quantify the distance between the proposed scheme and the optimal performance. In [61], the authors designed a dynamic offloading computation algorithm for a fog system with energy harvesting mobiles devices. They used queuing theory in their formalization to deeply derive the uplink-transmission energy consumption and response time models during the offloading process. Compared to the other existing algorithms, their proposed algorithm can achieve better results when the requests arrival rate is minimal.

Another idea that targets minimizing both energy consumption and delay was presented in [65]. Du et al. [65] formulated an optimization problem using mixed-integer nonlinear programming with the main goal to minimize the maximal cost in terms of energy consumption and delay for each offloading decision: local execution, fog processing, and cloud computing in multi-user fog/cloud computing systems. Furthermore, they designed an algorithm that makes an offloading decision with the minimum cost value of latency and energy, which performs computation resource allocation based on the final solution. The simulation results showed that the number of beneficial UE from the computation offloading is close to the total number of UEs compared to the method based only on local execution.

Furthermore, apart from exploitation of renewable energy, a technique known as dynamic voltage scaling (DVS) has also been utilized as an energy conservation method. Wang et al. [64] proposed an algorithm for partial task offloading in mobile edge computing based on DVS technique, which has bi-objective optimization function of minimizing both energy consumption and latency. The DVS is employed on mobile devices for the purpose of adjusting the processing speed in order to conserve energy.

5 Service placement in fog computing

Fog computing is usually deployed in a heterogeneous and constrained environment, which leads to a complex and sophisticated resource management. Thereby, enabling an efficient deployment for offloaded applications is a critical challenge. A major question that arises is how to manage task execution. More precisely, to which nodes a particular task should be assigned, and what metrics should be considered to evaluate deployment performance. Several studies have addressed these questions and many strategies have been proposed accordingly. We review these proposed strategies in the following subsections. First, we present the existing challenges according the application placement. Then, we discuss the optimization metrics that have been considered. Finally, we cover application placement algorithms that have been proposed so far.

Before reviewing the recent studies that focus on service placement, let us define this problem formally. Let A be a multi-service (multi-component) application with a set of requirements R, and let I be a distributed fog infrastructure. Service placement into fog infrastructure is a mapping of each service in A to a computation fog/cloud node in I, while meeting all requirements in R, and optimizing a set of objectives metrics O used to evaluate the placement performance.

5.1 Service placement challenges

A number of challenges regarding the problem of service placement in fog computing arise [49]:

1. *Device heterogeneity*: A suitable placement of services in a fog environment is impeded by heterogeneity of devices that are present at any of the network layers and locations.
2. *Constraint diversity*: Different IoT applications have diverse constraints and requirements that need to be satisfied in order for proper performance to be provided. These constraints/requirements may involve either consumable or non-consumable properties such as computing and bandwidth resources as the former, and latency or privacy as the latter. Meeting the diverse application requirements necessitates an intelligent approach in service selection.
3. *Multi-tenancy*: A fog infrastructure needs to be able to support the placement of multiple applications that require simultaneous management.

4. *Scalability*: The complexity of placement problem dramatically increases with a growing infrastructure size (number of devices) and application size (number of components), which makes it even harder to deal with this large-scale problem.

5.2 Optimization metrics

The problem of resource allocation and service placement in a fog system entails optimization of one or more specified metrics, the values of which need to be either minimized or maximized depending on a metric's contribution to the system performance [49]. This subsection discusses the most widely considered optimization metrics in fog systems.

Latency: Reducing end-to-end latency of cloud-based delay-sensitive applications is one of the most important goals of fog computing, since it allows efficient monitoring and faster response of applications to their environment. There has been a lot of research that focuses on minimizing the latency of services placed on fog resources while meeting a number of constraints [62, 68, 69].

Energy consumption: Minimizing energy consumption in IoT in general and fog computing in particular is yet another crucial issue, especially due to the fact that IoT and fog devices have limited resources; hence, making energy consumption one of the most important metrics for performance optimization. It involves minimizing energy needed for transferring services from end users to fog nodes, energy consumed by the nodes for processing, and energy needed to transfer a service from a fog node to a cloud if necessary, all of which reduces the lifespan of a fog network.

Resource utilization: Optimizing resource utilization is another important metric in fog computing with a goal to maximize the number of services deployed over suitable fog nodes. The proposed approaches generally make deployment decisions that maximize the number of satisfied application requests (e.g., via prioritization of applications that have the closest deadline) [70, 71].

Cost: There are two main types of costs depending on the viewpoint (providers vs. users): data transmission cost (i.e., networking cost) and service execution cost. Other types of costs include expenses related to storage, deployment, etc.

QoS assurance: QoS provisioning may be associated with minimizing delays, but also many other metrics as well, some of which may not result in delay reduction. For example, if QoS is measured as the percentage of requests executed before a deadline, then guaranteeing QoS would involve keeping the execution times below a threshold, not necessarily resulting in delay minimization.

5.3 Proposed service placement strategies

In the literature, several studies have addressed the service placement problem in the fog environment. These studies were motivated by optimization objectives such as minimizing latency, reducing energy consumption, or improving the quality of service. In Table 3, we give a classification of the state-of-the-art studies according to the employed optimization strategies: (i) machine learning (ii) mathematical programming, (iii) heuristics, (iv) meta-heuristics, and (v) other methods. In the second

Table 3 Summary of the case studies

References	Application	Optimization method	Objective metrics	Advantages	Limitations
[72]	General	ILP	Latency	Considering the processing cost	High system model complexity
[73]	General	ILP	Deployment cost	Low computational complexity and considering the mobility of the fog nodes	Omitting important parameters (e.g., delay) in the problem formulation
[74]	Image processing	ILP	Latency	Considering the complexity increase and the mobility of fog nodes	Unrealistic assumptions of service rate model
[75]	Smart city	ILP	Latency	Improvement of service placement architecture	High computational time due to large space exploration
[49]	Smart home	Heuristic	Latency	Reducing execution times and scalable	Data privacy is not considered
[76]	General	Heuristic	Energy	Considering the wireless channel conditions	It is not evaluated in real case scenarios
[77]	General	Heuristic	Delay	Improvement of service delay	Resource mobility is not considered
[34]	Healthcare	Heuristic	Latency	Considering different network topologies	Energy is not evaluated
[47]	Patient monitoring	Heuristic	Energy	Positive impact on network usage	Patient's mobility is not considered
[48]	Random scenario	Heuristic	QoS	Improving IoT services availability	It is not evaluated in a real case scenario
[68]	General	Meta-heuristic	Energy, QoS	Multi-objective optimization	It is not evaluated in a real case scenario
[78]	Random	Meta-heuristic	User experience	Improving user satisfaction	Resource mobility is not considered
[69]	General	Meta-heuristic	Latency	Minimizing the CPU execution time	Transmission delay is not evaluated
[52]	Healthcare, Smart home	Heuristic	Latency	Scalable	It is not evaluated in a real case scenario
[79]	Tracking	Deep learning	Latency, Energy, Cost	User mobility is considered, multi-objective optimization, and scalable	High computational complexity
[80]	Mobile crowd sensing	Deep learning	QoS	Minimum computing is pushed to the cloud	High computational complexity
[81]	General	Deep learning	QoS	Improving QoS	High computational complexity

column of this table, we list the application types each method focuses on. We give the objective metrics in column three. In the last two columns of this table, we note the advantages and limitations of each method.

5.3.1 Mathematical programming

Mathematical programming is often used to solve optimization problems by formulating them as a mathematical model with constraints and an objective function. Then, the solution domain of the objective function is explored with the main purpose to either maximize or minimize its value, while guaranteeing to return the optimum solution. However, it is important to note that checking the whole solution space suffers from extremely high execution time, and is only feasible for smaller problems, unlike the complex service placement problem in a fog environment [29]. Different types of mathematical programming models such linear, nonlinear, and mixed-linear are studied in fog computing.

In [72], the authors proposed a framework for resource allocation in a fog computing environment. They defined a model based on integer linear programming (ILP) to optimize the placement of tasks in clusters of fog nodes. Each cluster is controlled by an efficient fog node, which plays the role of a broker and task orchestrator: it receives tasks to be executed from different IoT devices and places them in the cluster. Tasks that cannot be placed (e.g., due to a lack of resources) are redirected to the cloud. The objective function of their task placement model is to (i) maximize the number of tasks executed within each cluster, and therefore reduce the response time, and (ii) choose for each task a fog node that conforms to the processing resource requirements (i.e., CPU, RAM, storage, and bandwidth), and the node closest to the controller in terms of latency.

Furthermore, Daneshfar et al. [73], in addition to proposing a fog infrastructure, formulated the service placement problem as an ILP model with the goal to minimize the total cost of providing services when services are deployed onto fog nodes. Their formulation allows for a user to multi-cast their tasks to multiple fog servers in order to guarantee resource availability. They associated an availability value for each fog node, a cost of execution for each task (identical for all fog nodes), a maximum budget of time for each user to perform their tasks, and a maximum server number for each user to submit their tasks. The objective function of their model makes it possible to find a placement solution and multi-cast all tasks at a lower cost while respecting the desired availability for each user.

Similarly, Velasquez et al. [75] proposed an ILP-based approach for optimizing the placement of IoT services in fog with the following objectives: (i) minimize the number of jumps between the user and the requested service location in fog to minimize the latency, (ii) minimize the number of existing jumps between cooperative services, and (iii) minimize the total number of offloaded services compared to the previous offloading to improve the system stability.

Zeng et al. [74] considered service placement problem with task scheduling and workload balancing in software-defined embedded systems with a support of fog computing. They formulated the problem as a mixed-integer nonlinear programming (MINLP) problem and reduced it to a mixed-integer linear programming (MILP) problem to solve it with a commercial tool Gurobi. They compared the proposed method to server-greedy (i.e., cloud-ward) and client-greedy (i.e., edge-ward) placement strategies, and demonstrated through extensive simulations that their solution is computationally efficient even for such a high-complexity problem, while achieving improvements in performance and response time.

5.3.2 Heuristics

A dynamic and mobile nature of a fog infrastructure makes the service placement problem extremely complex in terms of computation, and an exact analysis of the entire solution space is practically inapplicable. Hence, heuristic approaches are often explored as they provide means to obtain solutions in a reasonable amount of time. Heuristics are a set of rules and techniques that facilitate getting feasible solutions for computationally complex problems. Nevertheless, they do not provide any performance guarantees [29].

Here, we present some of the most recent studies that employ heuristics for this problem. Xia et al. [49] proposed an objective function, which aims to minimize average response time of an application deployed onto a set of fog nodes. They developed two backtrack search-based algorithms, namely exhaustive search and native search, to find placement solutions. Exhaustive search tries to visit all existing solutions and returns the optimal one that minimizes the average response time, whereas native search returns the first found solution. Based on native search, they proposed two heuristics: the first one aims at minimizing the response time returned by the naive search and the second one accelerates the search process. The simulation results showed that the combination of both heuristics makes the placement decision-making process more scalable, which leads to a lower average response time.

Gu et al. [76] proposed a binary linear programming-based model and heuristics for task assignment in a MEC environment. Their model optimizes overall energy consumption induced by the execution and transmission of tasks, while ensuring that the delay constraint required by all tasks is satisfied. In their heuristics, server nodes (which process tasks) publish their performance among themselves (the frequency of the CPU). Then, each task is sent (from its source) to a potential server node for execution. The choice of a node is made by a function based on energy consumption. Upon receiving tasks, there are two possible cases: (i) the receiving node meets the required delay, and as a result, the task is handled by that node; and (ii) the node does not meet the required delay, in which case the task is forwarded to one of the nodes based on shared information about the nodes' performance. This is done without any task administrators.

In [77], Yousefpour et al. proposed an algorithm that adopts the concept of load sharing to place tasks in a fog infrastructure, in which fog nodes collaborate with each other to execute the received request. Their algorithm aims to optimize the end-to-end service delay. A fog node accepts to execute the received request based on its estimated waiting time; otherwise, it forwards the request to one of its neighbors or to the cloud. In addition, probabilistic models are built to estimate the time it takes to complete a task in a fog node.

In [34], Taneja et al. proposed heuristics to place IoT service requests in a fog infrastructure. Their idea is to sort in ascending order: (i) fog nodes according to their processing performance, and (ii) service requests according to the amount of resources required. Then, for each service request, their heuristics search for a fog node that meets the processing performance required by that request. Unplaced service requests are redirected to the cloud. Although the proposed approach is simple and easy to deploy, it has several drawbacks: it does not consider constraints on the execution time of services, nor the balancing of the workload between fog nodes.

In [47], Mahmoud et al. proposed an energy-aware allocation heuristic for placing application tasks on fog devices with objective to minimize energy consumption. Their idea is to use the dynamic voltage and frequency scaling technology to adjust the CPU frequency of fog devices in a way to ensure a minimum increment of the energy consumption.

In [48], Lera et al. proposed a heuristic based method which aims to minimize the fog network delays that occur between interlinked services, while optimizing the QoS and the service availability for the users. Their heuristic involves prioritizing the applications with the shortest deadlines in the application placement process.

Furthermore, Mahmud et al. [52] proposed a first -fit-based service placement approach that aims at (i) minimizing the total service delay. They aim to guarantee applications' QoS in satisfying service delivery deadlines.

5.3.3 Metaheuristics

Metaheuristic methods are generally inspired by nature. The main idea of these approaches is to try and improve the result in a reasonable time through an iterative process of searching for better solutions while trying to avoid getting stuck in local optima, unlike heuristic approaches that are prone to this problem. A number of metaheuristic techniques have been proposed in the literature, such as genetic algorithms (GAs) [82], ant colony optimization (ACO) [83], and particle swarm optimization (PSO) [84]. These algorithms are typically based on the idea of population (solution) evolution, in which the best solutions for a given objective are usually preserved for the next evolutionary step of obtaining a new generation of solutions with a hope of getting a fitter population [29].

Mebrek et al. [68] proposed a GA for optimizing task placement in a fog infrastructure. Their algorithm optimizes energy consumption incurred by the transfer and execution of tasks, while satisfying the delay constraints required for each task. In their optimization model, a task's response time includes the time it takes to deliver the task, the time it takes to complete it, and possibly the time it takes to redirect the query to the cloud (in the case of insufficient resources).

Li et al. [78] combined the fuzzy clustering algorithm with PSO to propose a resource scheduling method for fog computing. Their algorithm divides users' requirements into different classes: computing requirements, bandwidth requirements, and storage requirements. In addition, fog resources are classified according to their capabilities: storage, computing, and bandwidth resources, in order to match them to tasks.

In [69], Bitam et al. (2018) proposed a novel bio-inspired method, namely bees life algorithm (BLA), to address the task allocation problem in a fog computing environment. The aim of the study is to optimize the distribution of tasks among fog nodes by achieving optimal trade-off between CPU execution time and the memory used by fog nodes. The empirical evaluation of the performance in terms of response time and memory cost of the proposed method showed that it outperformed the existing PSO and GA approaches.

5.3.4 Machine learning

Machine learning techniques have also been employed by researchers to solve the service placement problem in fog systems. Deep reinforcement learning is the most used technique for service placement in fog computing. This technique enables learning policy in unknown environment through a trade-off between exploration and exploitation. In [79], Tang et al. exploited deep reinforcement learning technique to solve the service placement problem in fog computing. After modeling the problem as a multidimensional Markov decision process that aims to minimize communication delay, power consumption, and migration costs, they proposed a Q-learning algorithm to determine the optimal learning policy for placement decision making. The proposal took into account user mobility and was evaluated over a medium-sized infrastructure using real data.

In addition, Li at al. [80] proposed a deep reinforcement learning–based framework to decide the task scheduling strategy in hierarchical fog computing. They developed a four-layer neural network that incorporates two convolution layers and two fully connected layers, used to solve the scheduling problem after extensive training. Their main objective is guarantying QoS to the user by providing a minimum computing in cloud and bandwidth cost. The proposed framework is composed of a task scheduler and a fog node manager, and it operates as follows. The task scheduler collects the state information about all fog nodes and task requests, and forwards it to a learning network to generate several scheduling decisions. After that, the task scheduler chooses and forwards the best-valued decision to the fog node manager who allocates the fog nodes to the task requests based on that decision. Finally, a mobile device dispatches its data to the assigned fog node for processing.

5.3.5 Other strategies

A number of other strategies have also been investigated in the literature. In [62], Liu et al. formulated a multi-objective convex optimization problem with a set of constraints, which involves minimizing energy consumption, performance delay, and payment cost in fog-based mobile cloud computing (MCC) systems. First, they transformed the formulated multi-objective problem into a single objective problem using the scalarization method by applying a set of weight factors to reflect the importance of each objective including energy cost, execution time, and payment cost. Then, they addressed the problem by proposing an Interior point method-based algorithm, in which they focus on increasing the offloading probability for each service request in order to optimize the energy consumed by the mobile device, and strengthening the mobile device's transmission power in order to reduce the execution time for request processing.

In [85], Wang et al. proposed a Hungarian algorithm based approach to task assignment in a mobile edge infrastructure, in which user tasks are divided into sub-tasks and assigned to effectively selected neighboring edge servers considering their characteristics including location, computing capacity, and the estimated waiting time. Their main purpose is to reduce the energy consumption with respect to a task's delay constraint. The proposed approach is compared with greedy assignment and non-assignment methods.

5.4 Discussions

Based on our extensive review, we identified four main classes of commonly used approaches to deal with IoT application placement problem in fog computing: (1) mathematical programming such as integer linear programming (ILP), (2) heuristics such as best-fit, worst-fit, and backtracking search strategies, (3) meta-heuristics such as GA and PSO algorithms, and 4) machine learning-based approaches such as deep reinforcement learning.

From the scalability perspective, ILP-based service placement approaches suffer from a high computational CPU times as the problem size increases. The ILP solvers cannot handle the large number of variables in a reasonable time frame. ILP-based methods are not an efficient choice for a fog environment with a large number of resources and optimization parameters. On the other hand, heuristic and meta-heuristic approaches have much shorter execution times when compared to ILP-based methods; however, they do not guarantee the optimal solutions. Therefore, ILP-based methods are good candidates for small-sized problems, while heuristic and meta-heuristic methods can be preferred for large-sized problems. Furthermore, solutions returned by ILP-based methods can be used for evaluating heuristics and meta-heuristics. Finally, machine learning-based approaches such as deep reinforcement learning strategies suffer from a long training time in problems with large solution space for decision making.

When we consider the capabilities of aforementioned methods for the service placement problem, ILP is used to express the problem with mathematical formulations under the given constrains and objective function for systematically identifying the best candidate solution, with the aim of maximizing/minimizing the objective function. Meta-heuristic approaches like GA algorithms start by improving an initial valid placement iteratively. However, determining an initial valid placement is not an easy task due to the heterogeneous fog nodes and strict constraints of emergent IoT services. Heuristic approaches try to find a global optimal placement through an iterative optimization process; however, the initial solution may force the final solution to be a local optimal one. Therefore, both heuristics and meta-heuristics need good design decisions in terms of finding the global optimum. Deep reinforcement learning enables learning policy through an online approach under certain criteria. It can converge to the optimal decision for service placement problem. Nevertheless, the large solution space leads to long training times. Machine learning-based approaches are also suitable for fast changing environments of the fog systems. Most of the reviewed studies in this work apply Q-learning algorithm to search for optimal learning policy based on a trade-off between exploration and exploitation without any prior environment knowledge. Hence, Q-learning may be a suitable reinforcement learning method for decision making with limited information and a dynamic environment like the fog problems discussed in this survey.

6 Future research directions

After rigorously reviewing the previous studies, we observed some open issues in fog computing that can lead the future research on this topic to excel. We can list them as follows:

1. As we presented in Table 3, one of the most common disadvantages of the previous studies is not considering the mobility of the edge devices. They also do not take the user mobility into account. Therefore, a future work can focus on optimization methods that consider QoS along with the user mobility [86].
2. Most of the previous studies do not focus on handling multiple applications from numerous users. In such cases, real-time processing and online scheduling can be difficult problems to solve. Therefore, the new real-time and online scheduling methods for multiple applications coming from a number of different users can be an interesting problem to tackle [87].
3. Most of the existing service placement methods focus on energy, performance, cost, and QoS metrics. However, meeting QoS is very difficult in the presence of faulty fog nodes. Therefore, incorporating fault tolerance in service placement and computation offloading is a crucial issue. Thus, a new fault-tolerant offloading algorithms can be a good addition to the literature [87].
4. The prior work generally does not consider straggler nodes. However, a straggler can be present at any time in a network. This may pose a risk for the operation and QoS of the overall system. Therefore, identifying the straggler nodes in a network and mitigating their danger to the system are also very important problems to study [88].
5. Most of the previous task scheduling and offloading methods assume independent tasks. Therefore, there is still a huge need for considering dependent task scheduling under several factors such as communication overhead among edge devices and also fog devices. This problem becomes a very difficult optimization problem to solve when the number of metrics is increased [88].
6. Finally, applying machine learning algorithms on different network problems is still an open research area. Particularly, using these algorithms to predict the future network behavior is a very interesting research topic.

7 Conclusion

The fog computing concept brings computation resources closer to the network edge. This development has paved the way for the end users to offload their computation-intensive and delay-sensitive applications to fog nodes, in order to meet the strict application requirements of latency and to reduce energy consumption at end users. In this paper, we surveyed recent research related to computation offloading and service placement in fog. In particular, we have identified and classified the different approaches and strategies that have been developed according to their objectives. In this regard, the fog paradigm incurs critical challenges that need to be addressed to satisfy both users and service providers. Moreover, recent research validates the results of the proposed methods mostly in simplistic scenarios, while the tests under real assumptions have not been taken into account.

As the second contribution of this work, we reviewed the existing application placement strategies in fog computing from the perspectives of mapping techniques and placement objectives. We proposed separate taxonomies for each of the aspects of application placement and discussed their associated research issues. We also highlighted a perspective model for offloading applications in fog environments and presented several research directions for further exploration of the fog computing-related problems.

References

1. Statista: Internet of things (IoT) connected devices installed base worldwide from 2015 to 2025 (2016). https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/
2. IDC: Iot growth demands rethink of long-term storage strategies (2020). https://www.idc.com/getdoc.jsp?containerId=prAP46737220
3. Mahmood Z, Ramachandran M (2018) Fog computing: concepts, principles and related paradigms. In: Mahmood Z (ed.) Fog computing: concepts, frameworks and technologies, chap. 1. Springer, Berlin, pp. 3–21
4. Daniel A, Subburathinam K, Paul A, Rajkumar N, Rho S (2017) Big autonomous vehicular data classifications: towards procuring intelligence. Veh Commun 9:306–312
5. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I et al. (2010) A view of cloud computing. Commun ACM 53(4), 50–58
6. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role in the internet of things. In: Proceedings of the first edition of the MCC workshop on mobile cloud computing, pp 13–16
7. Mach P, Becvar Z (2017) Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun Surv Tutor 19(3):1628–1656
8. Mahmud R, Kotagiri R, Buyya R (2018) Fog computing: a taxonomy, survey and future directions. In: Internet of everything. Springer, pp 103–130
9. Varshney P, Simmhan Y (2017) Demystifying fog computing: characterizing architectures, applications and abstractions. In: 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC). IEEE, pp 115–124
10. Hong CH, Varghese B (2019) Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms. ACM Comput Surv. https://doi.org/10.1145/3326066
11. Ren J, Zhang D, He S, Zhang Y, Li T (2019) A survey on end-edge-cloud orchestrated network computing paradigms: transparent computing, mobile edge computing, fog computing, and cloudlet. ACM Comput Surv. https://doi.org/10.1145/3362031

12. Yi S, Li C, Li Q (2015) A survey of fog computing: concepts, applications and issues. In: Proceedings of the 2015 Workshop on Mobile Big Data, pp 37–42
13. Mouradian C, Naboulsi D, Yangui S, Glitho RH, Morrow MJ, Polakos PA (2018) A comprehensive survey on fog computing: state-of-the-art and research challenges. IEEE Commun Surv Tutor 20(1):416–464. https://doi.org/10.1109/COMST.2017.2771153
14. Ghobaei-Arani M, Souri A, Rahmanian A (2019) Resource management approaches in fog computing: a comprehensive review. J Grid Comput 18:1–42
15. Phan LA, Nguyen DT, Lee M, Park DH, Kim T (2021) Dynamic fog-to-fog offloading in sdn-based fog computing systems. Futur Gener Comput Syst 117:486–497
16. Shakarami A, Ghobaei-Arani M, Shahidinejad A (2020) A survey on the computation offloading approaches in mobile edge computing: a machine learning-based perspective. Comput Netw 182:107496
17. Alli AA, Alam MM (2020) The fog cloud of things: a survey on concepts, architecture, standards, tools, and applications. Internet Things 9:100177
18. Mahmud R, Ramamohanarao K, Buyya R (2020) Application management in fog computing environments: a taxonomy, review and future directions. ACM Comput Surv. https://doi.org/10.1145/3403955
19. Naha RK, Garg S, Georgakopoulos D, Jayaraman PP, Gao L, Xiang Y, Ranjan R (2018) Fog computing: survey of trends, architectures, requirements, and research directions. IEEE Access 6:47980–48009. https://doi.org/10.1109/ACCESS.2018.2866491
20. Hu P, Dhelim S, Ning H, Qiu T (2017) Survey on fog computing: architecture, key technologies, applications and open issues. J Netw Comput Appl 98:27–42. https://doi.org/10.1016/j.jnca.2017.09.002
21. Yousefpour A, Fung C, Nguyen T, Kadiyala K, Jalali F, Niakanlahiji A, Kong J, Jue JP (2019) All one needs to know about fog computing and related edge computing paradigms: a complete survey. J Syst Architect 98:289–330
22. Mukherjee M, Shu L, Wang D (2018) Survey of fog computing: fundamental, network applications, and research challenges. IEEE Commun Surv Tutor 20(3):1826–1857. https://doi.org/10.1109/COMST.2018.2814571
23. Hu P, Dhelim S, Ning H, Qiu T (2017) Survey on fog computing: architecture, key technologies, applications and open issues. J Netw Comput Appl. https://doi.org/10.1016/j.jnca.2017.09.002
24. Bellavista P, Berrocal J, Corradi A, Das SK, Foschini L, Zanni A (2019) A survey on fog computing for the internet of things. Pervasive Mobile Comput 52:71–99
25. Nath SB, Gupta H, Chakraborty S, Ghosh SK (2018) A survey of fog computing and communication: current researches and future directions. arXiv preprint arXiv:1804.04365
26. Nisha P (2015) Fog computing and its real time applications. Int J Emerg Technol Adv Eng 5(6):266–269
27. Binh HTT, Anh TT, Son DB, Duc PA, Nguyen BM (2018) An evolutionary algorithm for solving task scheduling problem in cloud-fog computing environment. In: Proceedings of the Ninth International Symposium on Information and Communication Technology, SoICT 2018, p 397–404. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3287921.3287984
28. Hardesty L (2017) Fog computing group publishes reference architecture. https://www.sdxcentral.com/articles/news/Fog-computing-group-publishes-reference-architecture/2017/02/. Accessed 20 April 2020
29. Salaht FA, Desprez F, Lebre A (2020) An overview of service placement problem in fog and edge computing. ACM Comput Surv (CSUR) 53(3):1–35
30. Jamil B, Shojafar M, Ahmed I, Ullah A, Munir K, Ijaz H (2020) A job scheduling algorithm for delay and performance optimization in fog computing. Concurr Comput Pract Exp 32(7). https://doi.org/10.1002/cpe.5581. https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5581. E5581 cpe.5581
31. Ding H, Fang Y (2018) Virtual infrastructure at traffic lights: vehicular temporary storage assisted data transportation at signalized intersections. IEEE Trans Veh Technol 67(12):12452–12456. https://doi.org/10.1109/TVT.2018.2871414
32. Yi S, Hao Z, Qin Z, Li Q (2015) Fog computing: platform and applications. In: 2015 third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb). IEEE, pp 73–78
33. Dastjerdi AV, Gupta H, Calheiros RN, Ghosh SK, Buyya R (2016) Fog computing: principles, architectures, and applications. In: Internet of things. Elsevier, pp 61–75

34. Taneja M, Davy A (2017) Resource aware placement of IoT application modules in fog-cloud computing paradigm. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, pp 1222–1228
35. Gia TN, Jiang M, Rahmani AM, Westerlund T, Liljeberg P, Tenhunen H (2015) Fog computing in healthcare internet of things: a case study on ECG feature extraction. In: 2015 IEEE International Conference on Computer and Information Technology. IEEE, pp 356–363
36. Shukla S, Hassan MF, Khan MK, Jung LT, Awang A (2019) An analytical model to minimize the latency in healthcare internet-of-things in fog computing environment. PLoS ONE 14(11):e0224934
37. Cao Yu, Chen Songqing, Hou Peng, Brown D (2015) Fast: a fog computing assisted distributed analytics system to monitor fall for stroke mitigation. In: 2015 IEEE International Conference on Networking, Architecture and Storage (NAS), pp 2–11. https://doi.org/10.1109/NAS.2015.7255196
38. Zao JK, Gan TT, You CK, Méndez SJR, Chung CE, Te Wang Y, Mullen T, Jung TP (2014) Augmented brain computer interaction based on fog computing and linked data. In: 2014 International Conference on Intelligent Environments. IEEE, pp 374–377
39. Ning Z, Huang J, Wang X (2019) Vehicular fog computing: enabling real-time traffic management for smart cities. IEEE Wirel Commun 26(1):87–93
40. Paul A, Pinjari H, Hong WH, Seo HC, Rho S (2018) Fog computing-based IoT for health monitoring system. J Sens. https://doi.org/10.1155/2018/1386470
41. Zhu J, Chan DS, Prabhu MS, Natarajan P, Hu H, Bonomi F (2013) Improving web sites performance using edge servers in fog computing architecture. In: 2013 IEEE Seventh International Symposium on Service-oriented System Engineering, pp 320–323 . https://doi.org/10.1109/SOSE.2013.73
42. Satyanarayanan M, Bahl P, Caceres R, Davies N (2009) The case for vm-based cloudlets in mobile computing. IEEE Pervasive Comput 8(4):14–23. https://doi.org/10.1109/MPRV.2009.82
43. Aazam M, St-Hilaire M, Lung CH, Lambadaris I, Huh EN (2018) IoT resource estimation challenges and modeling in fog. Springer, Cham. https://doi.org/10.1007/978-3-319-57639-8-2
44. La QD, Ngo MV, Dinh TQ, Quek TQ, Shin H (2019) Enabling intelligence in fog computing to achieve energy and latency reduction. Digital Commun Netw 5(1):3–9. https://doi.org/10.1016/j.dcan.2018.10.008, http://www.sciencedirect.com/science/article/pii/S2352864818301081. Artificial intelligence for future wireless communications and networking
45. Aazam M, Zeadally S, Harras KA (2018) Offloading in fog computing for IoT: review, enabling technologies, and research opportunities. Future Gener Comput Syst 87:278–289
46. Aazam M, Huh EN, St-Hilaire M (2018) Towards media inter-cloud standardization-evaluating impact of cloud storage heterogeneity. J Grid Comput 16(3):425–443
47. Mahmoud MM, Rodrigues JJ, Saleem K, Al-Muhtadi J, Kumar N, Korotaev V (2018) Towards energy-aware fog-enabled cloud of things for healthcare. Comput Electr Eng 67:58–69
48. Lera I, Guerrero C, Juiz C (2018) Availability-aware service placement policy in fog computing based on graph partitions. IEEE Internet Things J 6(2):3641–3651
49. Xia Y, Etchevers X, Letondeur L, Coupaye T, Desprez F (2018) Combining hardware nodes and software components ordering-based heuristics for optimizing the placement of distributed IoT applications in the fog. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC '18, p 751–760. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3167132.3167215
50. Mahmoodi SE, Uma RN, Subbalakshmi KP (2016) Optimal joint scheduling and cloud offloading for mobile applications. IEEE Trans Cloud Comput 7(2):301–313
51. Mao Y, You C, Zhang J, Huang K, Letaief KB (2017) A survey on mobile edge computing: the communication perspective. IEEE Commun Surv Tutor 19(4):2322–2358. https://doi.org/10.1109/COMST.2017.2745201
52. Mahmud R, Ramamohanarao K, Buyya R (2018) Latency-aware application module management for fog computing environments. ACM Trans Internet Technol (TOIT) 19(1):1–21
53. Li L, Ota K, Dong M (2018) Deep learning for smart industry: efficient manufacture inspection system with fog computing. IEEE Trans Ind Inf 14(10):4665–4673. https://doi.org/10.1109/TII.2018.2842821
54. Kang Y, Hauswald J, Gao C, Rovinski A, Mudge T, Mars J, Tang L (2017) Neurosurgeon: collaborative intelligence between the cloud and mobile edge. SIGPLAN Not. 52(4):615–629. https://doi.org/10.1145/3093336.3037698

55. Teerapittayanon S, McDanel B, Kung HT (2017) Distributed deep neural networks over the cloud, the edge and end devices. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp 328–339. https://doi.org/10.1109/ICDCS.2017.226
56. Zhao X, Zhao L, Liang K (2017) An energy consumption oriented offloading algorithm for fog computing. In: Lecture notes of the institute for computer sciences, social informatics and telecommunications engineering, pp 293–301. Springer. https://doi.org/10.1007/978-3-319-60717-7-29
57. Chang Z, Zhou Z, Ristaniemi T, Niu Z (2017) Energy efficient optimization for computation offloading in fog computing system. In: GLOBECOM 2017–2017 IEEE Global Communications Conference. IEEE, pp 1–6
58. Craciunescu R, Mihovska A, Mihaylov M, Kyriazakos S, Prasad R, Halunga S (2015) Implementation of fog computing for reliable e-health applications. In: 2015 49th Asilomar Conference on Signals, Systems and Computers. IEEE, pp 459–463
59. Sun X, Ansari N (2017) Latency aware workload offloading in the cloudlet network. IEEE Commun Lett 21(7), 1481–1484. https://doi.org/10.1109/LCOMM.2017.2690678
60. Alli AA, Alam MM (2019) Secoff-fciot: machine learning based secure offloading in fog-cloud of things for smart city applications. Internet Things 7:100070
61. Shah-Mansouri H, Wong VW (2018) Hierarchical fog-cloud computing for IoT systems: a computation offloading game. IEEE Internet Things J 5(4):3246–3257
62. Liu L, Chang Z, Guo X, Mao S, Ristaniemi T (2018) Multiobjective optimization for computation offloading in fog computing. IEEE Internet Things J 5(1):283–294. https://doi.org/10.1109/JIOT.2017.2780236
63. Chen L, Zhou S, Xu J (2018) Computation peer offloading for energy-constrained mobile edge computing in small-cell networks. IEEE/ACM Trans Netw 26(4):1619–1632. https://doi.org/10.1109/TNET.2018.2841758
64. Wang Y, Sheng M, Wang X, Wang L, Li J (2016) Mobile-edge computing: partial computation offloading using dynamic voltage scaling. IEEE Trans Commun 64(10):4268–4282
65. Du J, Zhao L, Feng J, Chu X (2018) Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. IEEE Trans Commun 66(4), 1594–1608
66. Ma X, Lin C, Xiang X, Chen C (2015) Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing. In: Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp 271–278
67. Mao Y, Zhang J, Letaief KB (2016) Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE J Sel Areas Commun 34(12), 3590–3605
68. Mebrek A, Merghem-Boulahia L, Esseghir M (2017) Efficient green solution for a balanced energy consumption and delay in the IoT-fog-cloud computing. In: 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA), pp 1–4 . https://doi.org/10.1109/NCA.2017.8171359
69. Bitam S, Zeadally S, Mellouk A (2018) Fog computing job scheduling optimization based on bees swarm. Enterprise Inf Syst 12(4):373–397
70. Hong H, Tsai P, Cheng A, Uddin MYS, Venkatasubramanian N, Hsu C (2017) Supporting internet-of-things analytics in a fog computing platform. In: 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp 138–145. https://doi.org/10.1109/CloudCom.2017.45
71. Skarlat O, Nardelli M, Schulte S, Borkowski M, Leitner P (2017) Optimized IoT service placement in the fog. SOCA 11:427–443
72. Skarlat O, Schulte S, Borkowski M, Leitner P (2016) Resource provisioning for IoT services in the fog. In: 2016 IEEE 9th International Conference on Service-oriented Computing and Applications (SOCA), pp 32–39 . https://doi.org/10.1109/SOCA.2016.10
73. Daneshfar N, Pappas N, Polishchuk V, Angelakis V (2018) Service allocation in a mobile fog infrastructure under availability and QOS constraints. In: 2018 IEEE Global Communications Conference (GLOBECOM). IEEE, pp 1–6
74. Zeng D, Gu L, Guo S, Cheng Z, Yu S (2016) Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. IEEE Trans Comput 65(12), 3702–3712. https://doi.org/10.1109/TC.2016.2536019
75. Velasquez K, Abreu DP, Curado M, Monteiro E (2017) Service placement for latency reduction in the internet of things. Ann Telecommun 72(1–2), 105–115
76. Gu B, Chen Y, Liao H, Zhou Z, Zhang D (2018) A distributed and context-aware task assignment mechanism for collaborative mobile edge computing. Sensors 18(8):2423

77. Yousefpour A, Ishigaki G, Jue JP (2017) Fog computing: towards minimizing delay in the internet of things. In: 2017 IEEE International Conference on Edge Computing (EDGE). IEEE, pp 17–24
78. Li G, Liu Y, Wu J, Lin D, Zhao S (2019) Methods of resource scheduling based on optimized fuzzy clustering in fog computing. Sensors (Basel, Switzerland) **19**(9). https://doi.org/10.3390/s19092122. https://europepmc.org/articles/PMC6539192
79. Tang Z, Zhou X, Zhang F, Jia W, Zhao W (2018) Migration modeling and learning algorithms for containers in fog computing. IEEE Trans Serv Comput 12(5), 712–725
80. Li H, Ota K, Dong M (2019) Deep reinforcement scheduling for mobile crowdsensing in fog computing. ACM Trans Internet Technol. https://doi.org/10.1145/3234463
81. Filippo Poltronieri Mauro Tortonesi CS, Sur N (2021) Reinforcement learning for value-based placement of fog services
82. Holland JH et al. (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press, Cambridge
83. Yaseen SG, Al-Slamy N (2008) Ant colony optimization. IJCSNS 8(6):351
84. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks. IEEE, vol 4, pp 1942–1948
85. Wang J, Wu W, Liao Z, Sangaiah AK, Simon Sherratt R (2019) An energy-efficient off-loading scheme for low latency in collaborative edge computing. IEEE Access 7:149182–149190. https://doi.org/10.1109/ACCESS.2019.2946683
86. Badri H (2019) Stochastic optimization methods for resource management in edge computing systems. Wayne State University, Detroit
87. Wright KL (2019) High-performance distributed computing techniques for wireless IoT and connected vehicle systems. Ph.D. thesis, University of Southern California
88. Sundar S (2019) Optimization algorithms for task offloading and scheduling in cloud computing. Ph.D. thesis