An Overview of Service Placement Problem in Fog and Edge Computing

FARAH AÏT SALAHT and FRÉDÉRIC DESPREZ, Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, France
ADRIEN LEBRE, STACK Research Group - IMT-Atlantique, Inria, LS2N, France

To support the large and various applications generated by the Internet of Things (IoT), Fog Computing was introduced to complement the Cloud Computing and offer Cloud-like services at the edge of the network with low latency and real-time responses. Large-scale, geographical distribution, and heterogeneity of edge computational nodes make service placement in such infrastructure a challenging issue. Diversity of user expectations and IoT devices characteristics also complicate the deployment problem. This article presents a survey of current research conducted on Service Placement Problem (SPP) in the Fog/Edge Computing. Based on a new classification scheme, a categorization of current proposals is given and identified issues and challenges are discussed.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Computing methodologies** • **Networks** → Cloud computing; • **Theory of computation** → *Theory and algorithms for application domains*;

Additional Key Words and Phrases: Fog computing, edge computing, service placement, deployment taxonomy, optimization, classification

65

ACM Reference format:
Farah Aït Salaht, Frédéric Desprez, and Adrien Lebre. 2020. An Overview of Service Placement Problem in Fog and Edge Computing. *ACM Comput. Surv.* 53, 3, Article 65 (June 2020), 35 pages.
https://doi.org/10.1145/3391196

1 INTRODUCTION

In recent years, the Internet of Things (IoT) has become ingrained in our society by transforming objects of everyday life such as wearables, transportation, augmented reality, and so on, in communicating devices, and thus introduces new challenges and opportunities. With more than 50B devices connected to the network by 2020, according to Cisco [1], it is clear that the current infrastructures will not be able to support all the data that will be generated. Indeed, the current Cloud infrastructure alone can not support a large number of the current IoT applications for essentially three main reasons: First, the huge amount of generated data makes their transfer from where they are created (end-devices), to where they are processed (Cloud servers), impractical due to bandwidth limitations, processing overhead, and transmission costs. Second, the significant end-to-end

Authors' addresses: F. A. Salaht (corresponding author) and F. Desprez, Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, France; emails: {farah.ait-salaht, frederic.desprez}@inria.fr; A. Lebre, STACK Research Group - IMT-Atlantique, Inria, LS2N, France, Nantes, France; email: adrien.lebre@inria.fr.

0360-0300/2020/06-ART65 $15.00
https://doi.org/10.1145/3391196

delay from end-devices to the Cloud servers that are often too far from end-users can deter the performance of applications that require real-time analysis, such as online gaming, video applications, and so on. Finally, some data may have implications in terms of privacy and security, and it is advisable or even forbidden for this data to cross the entire Internet [2].

To cope with these issues, a promising paradigm able to avoid network bottlenecks, overcome communication overheads, and reduce the delay of data transfer has been identified [3]. This new conceptual approach that combines the benefits of Cloud and the decentralized processing of services on edge devices is known as Fog or Edge Computing [3]. The community has not yet converged against crisp definitions of these terms [4–7]. In the following, we use the term Fog Computing for simplicity.

Fog Computing [8] extends Cloud Computing and services to the edge of the network, bringing processing, analysis, and storage closer to where requests are created and used. The objective is to reduce the amount of data sent to the Cloud and reduce latency and computation costs. As a new paradigm, Fog Computing poses old and new challenges, and one of the main open issues is service management and more precisely the service placement problem. Indeed, one of the major barriers to the adoption of Fog is "how to efficiently deploy services on available Fog nodes." Unlike Cloud data centers, Fog devices are geographically distributed, resource-constrained, and highly dynamic, which makes the problem quite challenging. Therefore, the definition of an efficient, effective, and fair provisioning for the IoT applications will be important to provide and hence ensure end-to-end guaranteed services to end-users. Moreover, depending on the context and the interest, different aspects may come into focus: resource utilization [9], Quality of Service (QoS) [10–12], Quality of Experience (QoE) [13], and so on. These past few years, several works have been carried out and attempt to address this issue. Different assumptions, characteristics, and strategies have been considered to propose an efficient service placement. In this article, we review a wide range of works that studied this issue in Fog environments and explore the methodologies and the strategies proposed in the literature. We underscore that the survey goes beyond just describing the main approaches developed in the literature. It first identifies five main scenarios based on user expectations, problem descriptions, and deployment objectives. Second, it provides a new *classification* scheme where the different variants of SPP and the various solutions coming from the research community are classified. The following aspects are considered: *problem statement*, placement *characteristics*, technical *formulation*, problem *objectives*, *optimization strategies*, and *experimental tools*.

The article is organized as follows: Section 2 summarizes existing surveys and tutorials on Fog Computing and resource management in the related area and highlights the contributions of our article. Section 3 gives an overview of Fog systems: definition, architecture, main characteristics, and advantages. Section 4 introduces the service placement problem and summarizes the most common formulations, optimization strategies, major design objectives, and evaluation tools proposed in the literature to address this issue. The provided classification for the SPP approaches is presented in Section 5. Section 6 outlines the open challenges and highlights emerging research directions. Finally, Section 7 concludes this survey.

2 EXISTING SURVEYS

There are several surveys that address different aspects of Fog Computing and the related challenges [4, 7, 8, 14–25]. Indeed, different works have been proposed to discuss the concept and the role of Fog computing [4, 14, 16, 17, 26–29]. For instance, Bonomi et al. [14] investigate the role of Fog Computing in the IoTs domain, its characteristics, and its applications. Saharan and Vaquero et al. [26] give an overview of the concept in terms of enabling technologies and emerging trends. Chiang and Zhang [4] discuss, in the context of IoT, the need for a new architecture for computing and storage. In Reference [16], the authors give the definition of Fog Computing and closely related concepts and present three motivating applications: augmented reality, Content delivery, and Mobile Data Analytics. Many other papers discuss the Fog Computing characteristics, application domains, and related research challenges, such as References [27–29], however, they do not investigate and discuss the problem of service placement in such geo-distributed and large-scale environments. Indeed, these studies do not provide insights into how IoT applications are deployed over the network (what are the application requirements, infrastructure characteristics, domain constraints; mapping strategies; metrics to optimize, etc.).

Recent surveys attempt to fill the resource management and service placement problems. Among them, we quote the work of Yousefpour et al. [24], which provides a tutorial on Fog Computing, compares the Fog to the related computing paradigms, and discusses the resource management and service deployment (orchestration and migration) in Fog environment. In Reference [19], the authors present a taxonomy of Fog Computing, its related challenges and features, and discuss briefly the problem of service management. Nath et al. [30] focus on the architectures and features of Fog Computing systems, applications of Fog, security and privacy of Fog, and discuss the future scopes and open research. In their survey, the authors briefly discuss challenges related to resource management, orchestration between fog nodes and the cloud, and give some comparison of different QoS aspects of Fog computing. In Reference [31], Mouradian et al. present a detailed review on Fog architectures and algorithms and illustrate two application domains, namely, IoT and Content Delivery Networks (CDN). Li et al. [22, 23] provide a survey on Edge Computing architecture and system management. They propose to characterize Edge Computing by considering the following aspects: architecture characteristics, management approaches, and design objectives. Brogi et al. [25] propose also to review the existing SPP proposals. They pursue three main objectives: elaborate an overview of the current algorithms, available prototypes, and experiments; classify the works based on the application and Fog infrastructure characteristics; and identify and discuss some open challenges.

Although these surveys explore the resource management and service placement problem in Fog/Edge Computing, we note that these works are limited in at least one of the following: (1) limited review and discussion on SPP in Fog/Edge Computing; (2) lack of comprehensive descriptions of the problem; state-of-the-art efforts regarding problem taxonomy, resolution approaches, evaluation environments; concrete research directions; (3) do not provide an in-depth comparison, classification, or some useful insights regarding the existing works (e.g., how we can evaluate/compare the different proposals).

Our article is different in the content and research issues. Mainly dedicated to achieving an exhaustive and very clear overview of SPP in the Fog environment, our survey aims at simplifying the user's access to references and identifying a flavor of challenges. It is characterized by the following contributions.

2.1 Our Contributions

The main contributions of our article can be summarized as follows:

(1) Provide an exhaustive and very clear description and overview of the SPP in the Fog environment.
(2) Provide a taxonomy of the problem in the area of large-scale, geo-distributed, and heterogeneous systems.
(3) Propose a classification of surveyed works based on the identified scenarios, provided taxonomy and optimization strategies.
(4) Finally, highlight the open challenges and discuss future research directions.

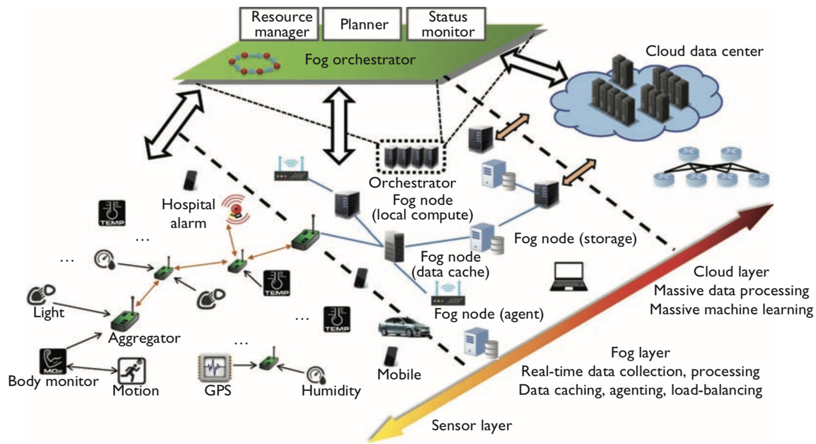

Fig. 1. Generic Fog computing architecture [32].

3 OVERVIEW OF FOG COMPUTING

In this section, we provide a brief overview of Fog Computing: definition, architecture, main characteristics, and advantages.

3.1 Definition

Fog Computing (FC) is a highly virtualized platform that offers computational resources, storage, and control between end-users and Cloud servers. Introduced by Cisco in 2012 [14], FC is a new paradigm in which centralized Cloud coexists with distributed edge nodes and where the local and global analyses are performed at the edge devices or forwarded to the Cloud.

3.2 Architecture

Several architectures have been provided for FC. Mostly derived from the fundamental three-layers structure (as depicted in Figure 1), a Fog infrastructure consists of IoT devices (End layer), one or more layers of Fog Nodes, and at least one Cloud Data Center (Cloud layer).

- *End layer:* Bottom-most layer and closest to the end-users. It is composed of various IoT devices (e.g., cameras, mobile phones, smart cars, smoke detectors). Widely geographically distributed, these devices enable sensing events and forward them to their immediate upper layer in the hierarchy for analysis and storage.
- *Fog layer:* Middle layer, consists of a set of devices that are able to process and store the received requests. Denoted by *Fog Nodes* (FNs), these devices that include access points, routers, gateways, switches, base stations, laptops, specific Fog servers, and so on, are connected to the Cloud servers and are able to send requests to data centers. Distributed between the end-users and DCs, these resources can be fixed devices (static) at some location or mobile (such as smartphones, vehicles, intelligent transportation systems, drones).
- *Cloud layer:* Upper-most layer in this architecture. It is composed of several servers and Data Centers (DCs) able to perform complex analyses and store a large amount of data.

3.3 Architectural Characteristics and Advantages

Considered as the future of Cloud systems and the Internet of Things, the Fog Computing involves a number of characteristics and advantages; the main ones are mentioned below:

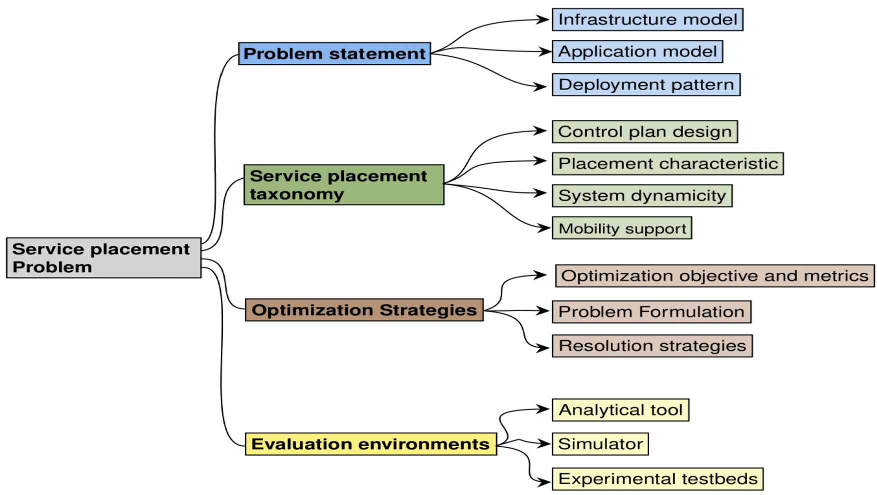

Fig. 2. The structure of the section.

(1) *Location awareness and low latency.* Most latency-sensitive applications, such as augmented reality, gaming, or video streaming, require sub-second processing time and do not necessarily need to be sent across long routes to data centers or Cloud services to be processed. With the Fog Computing, the support of these aspects is provided through the geo-distribution of the various Fog nodes in different locations and their proximity to end-users. Sarkar and Misra [33] proved by theoretical modeling that the service latency in FC is significantly lower than that with Cloud Computing.
(2) *Save bandwidth.* FC helps to unclog the network and speed up the processing of certain tasks by performing locally some computation tasks and sending only part of useful data or those that require significant analysis to the Cloud.
(3) *Scalability.* The number of connected devices grows rapidly, and the IoT data and application generated by these trillions of things [34] increases also exponentially. Given this large amount of data, processing the whole IoT applications in the Cloud is neither efficient nor feasible, so Fog intervenes as a complementary paradigm able to support all these requests and help the scalability of such systems.
(4) *Support for mobility.* Having widely distributed fog devices that provide computational and storage capabilities over the network, the Fog Computing is more suited to support the mobility of end-users than the traditional centralized Cloud servers and thus will allow to provide service guarantees for the mobile end-users without interruptions.

4 SERVICE PLACEMENT IN FOG COMPUTING

The service placement problem has been highly discussed in the literature and several proposals have emerged. Based on different application descriptions, network assumptions, and expected outcomes, these solutions are generally difficult to compare with each other. In this section, we propose to describe the methodology usually employed to address the SPP and give an overview of the following aspects: problem statement, placement taxonomy, optimization strategies, and evaluation tools (as depicted in Figure 2). These elements will allow us to clearly describe the problem and define the different aspects on which we will base our comparison and classification of the surveyed works.

4.1 Problem Statement

The statement of the SPP problem goes through the description of the following three parts: the infrastructure model, the application model, and the deployment pattern with its related constraints.

4.1.1 Infrastructure Model. The physical Fog infrastructure (see Figure 1) consists, respectively, of a set of devices with no computational capabilities (sensors, and actuators) and a set of resources that possess computational power and/or storage capacity (Fog nodes and Cloud data centers). Due to their physical structure [35], the fog nodes are resource-constrained and heterogeneous to each other. And any devices equipped with computational resources (in terms of CPU, memory, storage, bandwidth) can be considered as potential FNs, such as routers, small servers, access points, laptops, gateways, and so on.

The infrastructure network is generally abstracted as a connected graph where the vertices denote the set of IoT devices, Fog nodes, and Cloud servers, and the edges denote the links between the nodes. We mention hereafter the most common resources type and characteristics depicted in the literature to describe the Fog infrastructure.

- Resource type. *Computing:* servers, PCs, cloudlets [36, 37], and so on. *Networking:* gateways, routers, switches, base stations, and so on. *Storage:* every node that can provide storage. *Mobile:* vehicles, smartphones, and so on. *Endpoint abstraction:* sensors, actuators (e.g., GPS devices, wireless sensors, cameras, voice collector, radar).
- Characteristics. *Computing:* CPU power, number of cores, RAM, battery life, and so on. *Networking:* Type: wireless, wired; Capabilities: latency, bandwidth, error rate, and so on. *Storage:* Disk, and so on. *Virtualization:* VMs, containers, unikernel, and so on. *Hardware:* GPU, NUMA, FPGA, and so on.

4.1.2 Application(s) Model. Several abstractions and model definitions are used in the literature to characterize the applications generated by the IoT devices and treated at Fog resources and Cloud servers. According to the surveyed papers, we identify the following main descriptions: *(a)* a monolithic service, *(b)* a set of inter-dependent components, and *(c)* a connected graph.

(a) Monolithic service. The application sent by end-users or IoT devices is represented in the form of a single component (monolithic service). As an example, we can mention the case of an image processing application or data instance that needs to be proceeded or stored in a single physical node. The application can be defined in this case as a monolithic service.

(b) Set of inter-dependent services. This case assumes that the application is pre-partitioned into a set of components (services), and each performs some specific operation (functionality) in the application. In that case, dependencies between the application components are not considered.

(c) A connected graph. The application, in this case, is composed of a set of inter-dependent components represented as a connected graph. The vertices represent the processing/computational components of the application, and edges represent the inter-dependencies and communication demand between nodes [38].

Different topologies of a graph can be identified and among them, we have, respectively: line graph, tree application graph, and Directed Acyclic Graph (DAG). The DAG application topology is the most often used, because it models a large range of realistic IoT applications such as video processing [39–41], gaming [42], and healthcare [43] applications. Figure 3(a) illustrates an example of DAG application (cognitive assistance application).

Regarding the application requirements, we can summarize some of them in the following: *Computing:* CPU power, number of cores, RAM, and so on. *Network-oriented:* Bandwidth, Latency, Error-rate, Jitter (per link, end-to-end). *Task-oriented:* Deadline. *Location-oriented:* the application

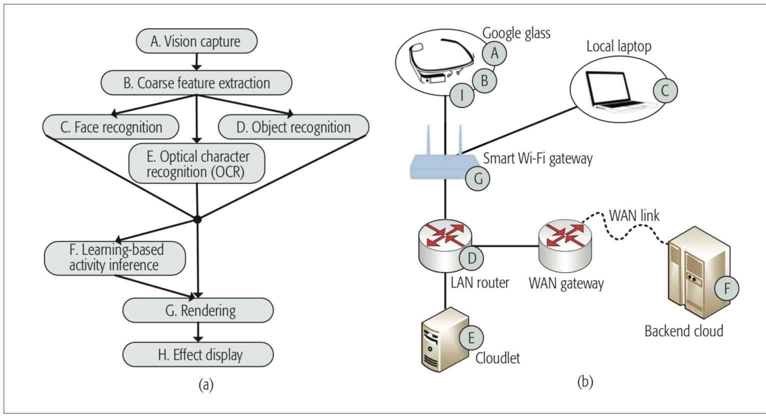

Fig. 3. Cognitive assistance application [44], shown in (a), and deployed onto Fog network, shown in (b).

must run in a specific geographical location (for instance, in Paris); the application can run only at some Fog node, and so on.

4.1.3 Deployment Pattern. The application placement problem defines a mapping pattern by which applications components and links are mapped onto an infrastructure graph (i.e., computing devices and physical edges). Figure 3 shows a mapping example of an application modeled as a DAG (Figure 3(a)) to available Fog nodes (Figure 3(b)).

Typically, application placement involves finding the available resources in the network (nodes and links) that satisfy the application(s) requirements, satisfy the constraints, and optimize the objective (if any). For instance, respect the applications (services) requirements, not exceed the resource capacities, satisfy the locality constraints, minimize the energy consumed, and so on. Service providers have to take into account these constraints to, first, limit the research space and, second, provide an optimum or near-optimum placement. We propose hereafter to depict some of the constraints mostly considered in the literature.

- ⋆ Resource constraints (C_R). An infrastructure node is limited by finite capabilities in terms of CPU, RAM, storage, bandwidth, and so on. Therefore, when placing application(s) (service components), we need to respect the resource requirements, i.e., ensure that the resources of the components deployed on the infrastructure nodes do not exceed their capabilities.
- ⋆ Network constraints (C_N). A network link can also be bounded by constraints such as latency, bandwidth, and so on, and these constraints need to be satisfied when deploying applications.
- ⋆ Application constraints: We highlight here two kinds of application constraints:
  - —Locality requirement (C_L). Locality requirement typically restricts certain services' executions to specific locations. Due to specific hardware, privacy requirements, or given policy, some components can be restricted to be deployed on specific areas (zone) or devices. Locality constraints can be based on: a set of Fog nodes [45, 46]; a specific geo-spatial location (using GPS for instance) [33], impose a co-localization of components [47], and so on.
  - —Delay sensitivity (C_D). Some applications can specify a deadline for processing operation or deploying the whole application in the network. This constraint is generally specified by defining a threshold to not exceed.

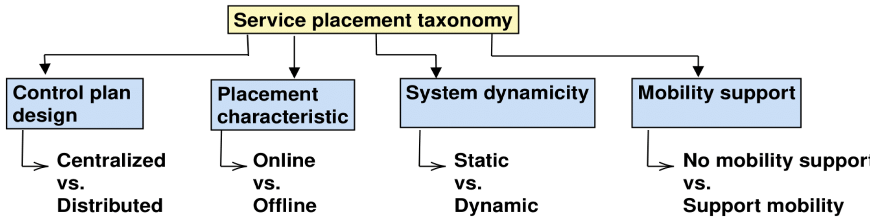

Fig. 4. Service placement taxonomy.

4.2 Service Placement Taxonomy

Addressing the SPP involves considering some specificities and criteria when designing the deployment strategies. Denoted as a service placement taxonomy, we propose in this article to pay attention to the following four main aspects as depicted in Figure 4:

As such, the first specificity considers whether the mapping coordination is done in a centralized or distributed manner. The second is based on whether the problem is tackled as an offline or online deployment. The third proposes to observe whether the dynamicity of the system is handled or not (i.e., handle the changes in the system, or not). Finally, the fourth characteristic describes whether the mobility of end-users and/or Fog devices is supported by the provided solution or not.

These eight characteristics described hereafter are used in Section 5.1 to classify the SPP proposals coming from the literature.

4.2.1 Control Plan Design: Centralized vs. Distributed. The development of a placement strategy and service management starts first by selecting the coordination strategy to adopt. Two common control plane models are presented in this article: *centralized* and *distributed* coordination. Relevant for multi-layered and geo-distributed systems, these approaches are relatively different, and each has its own advantages and inconveniences, as presented in the following:

(a) Centralized. A centralized control plane requires global information about application demands and infrastructure resources to take and disseminate global deployment decisions. The advantage of centralized placement algorithms is to potentially find a globally optimal solution; however, they are vulnerable regarding the scalability and the computational complexity issue.

When surveying papers, we observed that a large number of works considers a centralized control plane when addressing the SPP. Among these works, we mention the work of Hong et al. [48], which considers a central coordinator to make deployment decisions of IoT applications over Fog infrastructure.

(b) Distributed. Unlike centralized solution, a distributed approach considers multiple authority and orchestrator nodes to control the services mapping. Generally distributed in the network (as illustrated in Figure 5), the management elements compute placement decisions based on local resources and information. This control plane is more flexible and can be more efficient to handle the dynamic changes of infrastructure like Fog Computing without resorting to network-wide computations. The distributed approach helps to address the scalability and the locality awareness issues and allows providing services that best fit with the local context; however, no guarantees are provided regarding the global optimality of the computed solutions.

Among the works that considered a distributed control plane, we quote the work of Wang et al. [49] that considers a fog-based architecture composed of fog nodes and fog nodes coordination. The FNs sub-layer deals with the tasks that need real-time processing. The complex tasks that require more computational capabilities are transmitted to the FNs coordination sub-layer or forwarded to the Cloud.

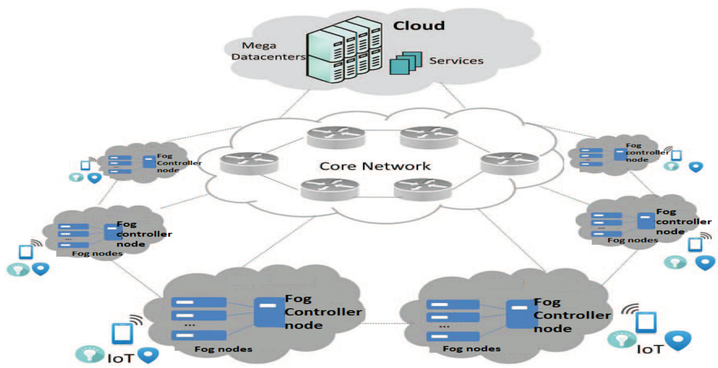

Fig. 5. Example of a distributed control plane.

4.2.2 *Offline vs. Online Placement.* The service placement problem can be tackled in an offline or online manner. More precisely, we say that the placement is offline if it takes a deployment decision at the compile-time, where all required information is available. It needs complete information about the system activities and provides solutions that satisfy the given requirements. For online placement, the deployment decisions are made during the run-time of the system. The decisions are based on both process characteristics and the current state of the system. In most real use-cases, the SPP has to be addressed as an online problem. That is, the related algorithms have to consider the services as they arrive, rather than computing the placement in advance before their execution. We notice that an online placement can accommodate dynamic behavior (changes) of the system, but it cannot make the best use of system resources (provide optimal placement decision).

As an example of offline placement algorithms provided in the literature, we mention References [12, 50], which assume full information knowledge for the Fog network. For the online placement, we have for instance the work of Lee et al. [51], which proposes an online placement strategy that minimizes the computational latency for Fog system under uncertainty of available FNs. The provided online optimization framework proposes to sequentially observe the information on the network.

4.2.3 *Static vs. Dynamic Placement.* This criterion tackles the fact that proposals handle or do not handle the dynamicity of the system. We can identify, respectively, two aspects: the dynamicity of Fog infrastructure and the dynamicity of applications. The Fog network is highly dynamic, where entities join and leave network due to instability of network links or failures. Resources capabilities can also vary over time. From an application point of view, the application graph structure can evolve over time in response to changes in real-life conditions. New sources or devices may appear or existing ones may disappear (adding new cameras, breakdown of certain components, users can decide at any time to start or stop sending their data for a service, etc.). In addition, changes in the application information can also be observed (e.g., on the amount of data, component requirements).

To deal with the dynamic nature of Fog infrastructure and/or applications, it is required to define reactive strategies able to determine when adaptation is required, provide a transparent mechanism, and deliver the satisfying QoS. Thus, an approach is said to be dynamic if the provided placement strategy is able to deploy new services and replace or release services already deployed to meet the QoS constraints and optimize a given objective (if any).

By surveying the literature, we identified a number of works that propose to manage the dynamic nature of Fog environment. Among them we mention the work of Yousefpour et al. [52],

Table 1. Classification Regarding the Optimization Objectives

Optimization objective	References
Mono-objective	[40, 53, 62–68], [69]*, [9, 10, 12, 13, 17, 35, 41, 45, 46, 48, 49, 51, 52, 56–58, 70–103], [104]*, [105]
Multi-objective	[106, 107], [69]*, [60, 108, 109], [104]*

which proposes a dynamic provisioning of services in Fog infrastructure that satisfies the QoS requirements and the Service Level Agreements (SLA) and minimizes the resources cost. To handle the dynamicity of IoT applications, Mahmud et al. [35] propose a policy that dynamically determines host nodes for the deployed components and handles sudden changes in frequencies of the received services.

4.2.4 *Mobility Support.* Managing mobility is a major challenge in Fog Computing. Providing a solution that supports the mobility of end-users and/or fog devices and ensures that the users always receive the associated services and desired performance without interruptions is a complex issue in Fog Computing. Frequent changes in locations for end nodes (or fog nodes, e.g., smartphone, smart car) can lead to excessive delays or to packet loss. In such a situation, the system manager (coordinator) should be able to move the service transparently from the previous devices to the new ones to ensure its continuity.

In References [49, 53–60], for instance, the authors attempt to address the problem of end-user mobility by providing dynamic placement approaches. In Reference [59], Saurez et al. propose to handle the mobility of end-users by providing strategy based on the following decisions: "When-to-Migrate," based on the latency parameter; and "Where-to-Migrate," based on the proximity of an FN to the mobile devices and the current processing node.

4.3 Optimization Strategies

Optimizing the service placement problem in a Fog infrastructure has been tackled from several different objectives, with different formulations and diverse algorithm proposals. This section discusses the possible objectives that may be pursued in such systems, the metrics considered to evaluate the provided deployment, the problem formulation used by existing proposals, the resolution strategies, and algorithms used to solve the SPP.

4.3.1 *Optimization Objective and Metrics.* We present first a global classification of optimization strategies proposed in the literature. On one hand, we distinguish the following two categories: mono-objective and multi-objective optimization. On the other hand, we present the metrics most often considered during optimization.

(a) Mono- vs. Multi-objective optimization. Mono-objective optimization proposes to optimize only one objective function, while multi-objective proposes to optimize simultaneously a collection of objective functions [61]. A first classification of SPP solutions regarding these two optimizations is given in Table 1. Works that have studied both aspects (mono-objective and multi-objective) are marked with an asterisk (*).

(b) Optimization metrics. We provide hereafter a list of optimization metrics usually considered in the literature in the context of the works provided so far (see Table 2). The optimization can be addressed to maximize or minimize the value of the following metrics:

• *Latency.* Low latency for delay-sensitive applications. Achieving lower latency has attracted attention in several surveyed papers. Indeed, several works aim at minimizing services latency

Table 2. Optimization Metrics

Metrics		References
Latency		[9, 12, 33, 45, 49–51, 58, 63, 66, 67, 71, 73, 77, 81, 83, 89–91, 96, 98, 101–104, 106, 108–115]
Resource utilization		[10, 17, 48, 60, 62, 64, 76, 80, 82, 87, 94, 95, 99, 106]
Cost		[40, 48, 52–54, 56, 57, 65, 70, 72, 74, 75, 79, 88, 93, 100, 106–109, 115–120]
Energy consumption		[21, 33, 50, 60, 67–69, 85, 85, 86, 97, 106, 108, 111, 121]
Others	Quality-of-experience	[13, 41, 92, 107, 110]
	Congestion ratio	[46, 84, 105]
	Blocking probability / Failed requests / Number of computationally active FNs	[78] / [64] / [35]

deployed on available resources while satisfying the set of requirements and constraints. For instance, in References [102, 103], the authors propose to minimize the service delay of deploying IoT applications on the IoT-Fog-Cloud framework.

• *Resource utilization.* An important issue in Fog Computing is how to optimize resource utilization while deploying the maximum number of service over appropriate fog nodes. Among the works found in the literature that investigate this goal, we can cite the work of Hong et al. [48], which provides deployment decisions while maximizing the number of satisfied IoT analytics requests. Skarlat et al. in Reference [94] propose also to maximize the number of satisfied application requests by prioritizing the applications with the closest deadline.

• *Cost.* Cost-related factors become very influential in Fog service management, from the service provider's point of view or from the users' point of view. We can identify two main types of costs: the networking cost for the data transmission charges and associated expenses; and execution cost related to the computational expenses of Fog nodes. Other expenses can also be identified: costs related to storage, deployment, security safeguards, migration, and so on.

In Reference [52], the authors propose to minimize a total cost that includes the cost of processing and storage in Cloud and Fog, the cost of communication between Fog and Cloud and between Fog nodes, and the communication cost of service deployment from the Fog service controller to FNs.

• *Energy consumption.* Energy efficiency is one of the main concerns in IoT systems and a significant performance metric that several works attempt to investigate within the Fog context. The energy consumption encompasses mostly two things: the type of service to process and the energy consumption at the service level that includes three main aspects: when the service is sent by the end-user to the fog device; when the service is processed by the FN; and when the Fog needs the Cloud. For instance, Sarkar et al. [21, 33] and Nishio et al. [50] investigate the energy consumption issue in the Fog environment by considering energy consumption in both the network and the device side.

• *Other metrics.* Other metrics can be considered when addressing the service placement problem. Some of these metrics are quality of experience, congestion ratio, blocking probability, failed requests, and so on. Accepted as the user-centric measurement, QoE encapsulates the user's requirements, perceptions, and intentions when deploying services [122]. As an example study, we mention the work done by Mahmud et al. [13] that provides a QoE-aware application placement strategy by which the deployment prioritizes the user expectations. Regarding the congestion

Table 3. Classification According to Technical Formulation

Technical formulation	Model	References
Integer programming	Integer Linear Programming (ILP)	[10, 12, 13, 41, 45, 45, 46, 48, 51, 58, 60, 65, 66, 72, 76, 78, 82, 86, 87, 89, 90, 94, 95, 97, 117]
	Integer Nonlinear Programming (INLP)	[51, 52]
	Mixed Integer Linear Programming (MILP) / Mixed-integer non-linear programming (MINLP)	[40, 53, 64, 104, 123, 124] / [40, 74, 101, 104]
	Mixed Integer Quadratic Programming (MIQP)	[46]
Constrained optimization		[9, 11, 23, 33, 38, 51, 54, 56, 63, 63, 67, 68, 73, 75, 77, 79, 93, 96, 99, 105, 108–111, 114, 115, 125–130]
Others	Matching game	[71]
	Machine learning	[57, 107]
	Stochastic optimization / Petri nets / Potential games / Quadratic Assignment Problem / General convex optimization / Linear programming	[91, 119] / [120] / [92] / [88] / [69]
No technical formulation is provided		[55, 110]

ratio, Yu et al. [46] propose to consider the minimum ratio between the flow and the capacity of link to address the service placement and data routing of real-time processing applications in IoT.

4.3.2 Technical Formulation. The SPP is generally formalized using one of these two main categories: integer programming or constrained optimization. Table 3 groups the surveyed works according to the identified problem formulation briefly described below.

(a) Integer programming. Class of problems that encompasses mathematical optimization problems in which some or all of the variables are integers. We have different variants of integer programming approaches. We list the main ones hereafter.

• *Integer Linear Programming (ILP).* This category of problems expresses the optimization of a linear function subject to a set of linear constraints over integer variables. Several works, such as References [10, 76, 90, 94, 117], formulate the placement problem with ILP.

• *Integer NonLinear Programming (INLP).* An integer nonlinear program is an ILP that presents nonlinear constraints. For instance, in Reference [52], Yousefpour et al. formulate the dynamic Fog service provisioning problem as an INLP.

• *Mixed Integer Linear Programming (MILP)/Mixed Integer NonLinear Programming (MINLP).* The class of a problem known as a Mixed Integer Programming Problem assumes that some decision variables are not discrete. In Reference [64], to study the latency-critical services management in an Edge-Cloud, the authors formulate the problem as a MILP that minimizes the number of failed requests. Mixed Integer NonLinear Programming (MINLP) considers continuous and discrete variables and nonlinear objective function and/or constraints. Due to the high computational complexity for solving this class of problems, most of the work proposes to linearize it into a MILP. As an example, to investigate the cost-efficient service deployment problem in the Fog architecture, Arkian et al. [40] first formulate the cost minimization problem as an MINLP and then linearize it into MILP to solve it more easily.

• *Mixed Integer Quadratic Programming (MIQP).* This problem refers to optimization problems with quadratic objective function in the integer and in the continuous variables, and linear constraints in the variables of both types. As an example, in Reference [46], the authors formulate the problem of real-time processing applications provisioning in IoT as an MIQP. To overcome the high complexity of solving such a problem, the authors propose to relax some of the constraints and present an approximation scheme.

(b) Constrained optimization. Constrained optimization is a set of methods designed to find out the best possible values of certain variables (i.e., optimizing process) in the presence of some restrictions (constraints). It uses a set of constraints that can easily be extended further to involve more aspects.

For instance, in Reference [125], Ait-Salaht et al. propose to handle the SPP problem by providing a generic and easy-to-upgrade constraint programming model. Brogi et al. [11, 126] propose a constrained model to determine the feasible deployments (if any) of an application in the Fog infrastructure.

(c) Other technical formulations. As other formulations found in the literature, we quote: Matching Game [131], Markov Decision Process (MDP) [132], stochastic optimization [133], petri nets [134], potential games [135], quadratic assignment problem [136], and general convex optimization [137].

For instance, in Reference [71], a matching game is employed to formulate the task placement problem in Mobile Edge Computing systems while minimizing the computation latency. In Reference [57], Urgaonkar et al. model the migration problem as an MDP (also known as reinforcement learning). In Reference [120], the authors use priced timed Petri nets (PTPNs) to study the service placement in Fog Computing while optimizing price and time costs for completing a task.

4.3.3 Resolution Strategies. Compute optimal application scheduling in Fog infrastructure is an NP-hard problem [138–140]. Indeed, several issues complicate the compute of effective services placement in such a context: first, the heterogeneous nature and the limited capacities of most Fog nodes (resource-constrained); second, the dynamicity of the environment, resources may appear and disappear, others are moving, infrastructure and application information may change over time (e.g., the variation of the workload); third, the geographical distribution of fog devices over a large-scale infrastructure. Several specificities and constraints make the SPP problem in Fog environment a challenging task. In the literature, we identify four main approaches used to solve such a problem: exact resolution, approximation strategies, and heuristic or meta-heuristic policies. We briefly describe these procedures in the following:

(a) Exact solution. The definition of an exact solution is often computed by using an ILP solver or by performing exhaustive research (by enumerating all solutions). Among the works that attempted to solve the SPP in an exact way, we mention References [58, 60, 87, 94, 97], which use ILP formulation and exact optimization solver to define an optimal solution, and Reference [105], which uses exhaustive placement research.

However, it is important to notice that performing an exact resolution requires long processing time before reaching the optimal solutions and can only be used for small problem instances. Indeed, finding an exact solution can be extremely time-consuming and not appropriate for large problems such as Fog environments. Thus, the main focus of works within the research community is based on providing an effective approximation, heuristic or meta-heuristic approaches where suboptimal solutions can be computed in a short time.

(b) Approximations. Approximation techniques are used to compute solutions with provable guarantees regarding their distance to the optimal one. Approximations are efficient algorithms

that allow to compute suboptimal solutions of NP-hard optimization problems. For example, in Reference [46], the authors propose to use a fully polynomial-time approximation algorithm to address the problem of IoT application provisioning. This approach allows computing a suboptimal solution and bounds for SPP in a relatively small time.

(c) Heuristics. Because of the scale and the dynamic and mobile aspects of Fog infrastructures that make exact analysis almost inapplicable, heuristics are often investigated. Designed to obtain a solution in a reasonable time frame, a heuristic is a set of rules and methods that aims at finding a feasible solution for a given problem. However, with heuristic-based solutions, no performance guarantees are provided. As heuristic approaches, we can find, for instance, fail-first/fail-last heuristics used by Brogi et al. in References [11, 126]. The authors in these works propose to adopt these two strategies to determine, in a relatively short time, a feasible deployment for an application in the FC. The fail-first heuristic is used in that case to select the undeployed component that has fewer compatible nodes. The fail-last policy sorts the candidate nodes by decreasing the number of end-devices required by a software component and their hardware capabilities. To guarantee that all requests are satisfied, these heuristics compute the best resource in terms of spatial proximity and the most powerful devices that can support them.

(d) Meta-heuristics. Typically inspired by nature, the meta-heuristic solutions aim at providing the best solution by iteratively improving the quality of the result and helping the search process to escape from local optima within a reasonable time (unlike heuristics that can be stuck in a local optimum). Several meta-heuristic algorithms are provided in the literature, such as Genetic Algorithms [141], Ant Colony Optimization [142], Particle Swarm Optimization [143], and Tabu Search [144]. These algorithms are based on population evolution where, at each evolution, the best-founded population (solution) is kept into the next evolution to define at the end the best solution regarding a given objective (metric). As an example, we mention the work of Skarlat et al. [94], which uses a Genetic Algorithm (GA) to solve the SPP in FC. A GA [145] is an evolutionary algorithm that mimics the process of a natural evolution of a chromosome. In their work, the authors assume that each gene in a chromosome denotes a service placement decision. The placement is iteratively improved according to a fitness function based on the principle of encouragement.

4.4 Evaluation Environments

To evaluate the performance of their proposals, the research community uses different programming tools to perform extensive experiments and test their solutions in relevant environments and preferably in realistic setups. We depict hereafter the most common tools used in the surveyed papers. Table 4 summarizes the programming environment adopted in the literature.

4.4.1 Analytical Tool. Analytical tool is one of the common approaches used to compute and evaluate the performance of the formulated strategies. The most frequently mentioned tools are: Java [11, 52], C++ [46, 59], and Matlab [79]. For instance, Brogi et al. [11] prototype a proof-of-concept Java tool named FogTorch[1] that implements and solves the SPP. These tools are sometimes associated with other tools or APIs such as ILP solvers. As solvers frequently mentioned in the literature, we have IBM CPLEX[2] [90, 94], the commercial solver Gorubi[3] [74], or Choco-Solver[4] [125].

[1] https://github.com/di-unipi-socc/FogTorch.
[2] https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.
[3] http://www.gurobi.com.
[4] http://www.choco-solver.org.

Table 4. A Summary of Evaluation Environments

Environment	Evaluation Environment	References
Analytical tool	Java Tool	[11, 52]
	C++	[46, 59, 78]
	Matlab	[67, 68, 75, 79, 86]
	Optimization engine (IBM CPLEX, Gurobi, Xpress-MP, etc.)	[10, 12, 46, 53, 60, 65, 74, 90, 94, 97, 104, 117, 125]
	Others	[67, 72, 88, 105, 106]
Simulator	CloudSim	[41, 95, 109]
	iFogSim	[9, 10, 13, 17, 38, 85, 87, 89, 90, 90, 93, 94, 104, 127, 146]
	SimGrid	[45]
	OMNeT++	[54, 63, 111]
	Others (FogTorchΠ, YAFS [147], etc.)	[51, 51, 57, 64, 77, 80, 82, 83, 88, 91, 100, 105, 108, 114, 120, 126, 148, 149]
Testbed	Grid'5000 & Fit IOT-Lab	[70]
	OpenStack	[150]
	Ad hoc testbed	[48, 56, 73, 76, 101, 110, 151]

4.4.2 Simulator. Another commonly used approach is performing *simulation.* Among the simulators cited in the literature, we find a simulator CloudSim [152], designed for regular cloud environments, most often used with some extensions; a SimGrid[5] framework, designed for large-scale distributed systems; a simulator iFogSim [38], designed for Fog Computing, which extends CloudSim. iFogSim is a simulation toolkit proposed by Gupta et al. [38] for evaluating application design and resource management techniques in Fog systems. The simulator performs discrete event simulation and allows users to run applications over Fog infrastructure and measure metrics such as latency, energy consumption, and network usage. Other simulators are also cited, such as network generic simulator OMNeT++[6] (used for instance in Reference [63]); FogTorchΠ, provided by Reference [126], which employs the Monte Carlo method [153] to estimate the QoS-assurance of output deployments; event-driven simulator based on SimPy,[7] performed by Borylo et al. [111], and so on.

4.4.3 Experimental Testbeds. Finally, we have physical *testbeds.* As realistic environments, we can cite FIT/IoT-LAB [154] and Grid5000 [155] large-scale experimentation environments. Designed for testing the Future Internet of Things, FIT IoT-LAB testbed is a French large-scale open-platform that provides access to a variety of fixed and mobile technologies and services (with more than 2,700 heterogeneous sensors spread in six different sites). More focused on parallel and distributed computing, Grid'5000 is a scientific instrument for experimental research on large future infrastructures: Clouds, datacenters, HPC Exascale, Big Data infrastructures, networks, and so on. It contains a large variety of powerful resources, network connectivity, and storage access, grouped in homogeneous clusters spread in 10 sites in France. We have Software stacks "OpenStack" [156], which is a set of open-source software tools for deploying Cloud Computing infrastructures (resource reservation, disk image deployment, monitoring tools, data collection, and storage). We can identify also Cumulus [157] platform for computational offloading at the Edge or platforms such as FED4Fire [158], the largest federation of Next Generation Internet testbeds in Europe, and PlanetLab [159], a testbed for computer networking and distributed systems research, and so on.

[5] https://github.com/simgrid/simgrid.
[6] https://omnetpp.org/.
[7] https://simpy.readthedocs.io/en/latest/.

Table 5. Classification of Surveyed Works According to Identified Scenarios

Service model		Papers
Scenario 1	Scenario 1.1	[12, 13, 41, 75, 80, 91, 93, 95–97, 107, 129, 160, 161]
	Scenario 1.2	[40, 46, 49, 51, 52, 56, 57, 62, 64, 67–69, 71, 78, 81, 82, 86, 89, 90, 92, 98, 102, 103, 110, 111, 114, 131, 146]
	Scenario 1.3	[10, 58–60, 76, 84, 87, 94, 99, 120]
	Scenario 1.4	[9, 11, 35, 45, 48, 53, 54, 56, 63, 65, 66, 70, 72, 73, 77, 79, 83, 85, 88, 101, 106, 108, 117, 125–127, 162]
Scenario 2		[104, 105, 115, 121, 123, 130, 150]

4.5 Summary

This section describes the SPP and summarizes the most common formulations, resolution strategies, and evaluation tools used in the literature to address this issue. Next, we propose to categorize the surveyed works based on a new classification scheme that allows to simplify access to references in the category of interest and identify more easily the challenges and emerging research directions.

5 A CLASSIFICATION OF SERVICE PLACEMENT APPROACHES

In this section, we use the taxonomy developed in Section 4.2 to provide a new classification scheme and categorize the works provided on the SPP by the research community. First, we propose to categorize the problem in two main scenarios according to the problem description (see Table 5). Then, we provide the classification we perform. The idea is to group the works addressing the same issues to better understand the needs of each problem and facilitate the user's access to references.

5.1 Identified Scenarios

When surveying the literature, we found that, depending on the problem's features, we can identify two main scenarios: Scenario 1, which aims at deploying services in Fog while satisfying the QoS requirements; and Scenario 2, slightly different from the first one, which, to ensure minimum latency and satisfactory quality of service, must disseminate and deploy services (replicate some services and place them) over Fog infrastructure. The description of these scenarios is given hereafter.

(i) Scenario 1: Assigning services while satisfying requirements and optimizing a given objective (if any). Most of the surveyed works are part of this first scenario and try to provide answers and address the following question: *"Where should the service be deployed and executed to best fit with the objectives?"*

In this scenario, we remark that, according to how we characterize the services, we can identify the following sub-scenarios: assignment of set of monolithic applications, assignment of continuous request sent from IoT node to the Fog/Cloud layer, assignment of set of applications each composed by a set of inter-dependant components, assignment of set of applications each having a DAG topology.

- *Scenario 1.1: Deploy a set of monolithic applications.* This scenario addresses the problem of defining the best storage/process location for set of services that are assumed to be monolithic.
- *Scenario 1.2: Deploy applications that receive continuous requests from a data source.*
This scenario deals with the assignment of service flows between service producer (i.e., sensors) and service consumers (i.e., Fog resources). Application placement here involves both

determining the host devices and routing path that satisfies requirements and optimizes objective (if any).

- *Scenario 1.3: Deploy applications each abstracted as a set of interdependent services.* This scenario assumes that each application is composed of a set of independent components, i.e., without networking dependencies (requirements in terms of latency, bandwidth, and so on, between services are not considered).
- *Scenario 1.4: Deploy applications each abstracted as a Directed Acyclic Graph (DAG).* The application here is defined through a DAG topology, where each node represents an operational service of the application and the links describe the networking requirements between components (refer to Section 4.1.2(b) for more details).

(ii) Scenario 2: Ensure minimum latency and a satisfactory QoS when deploying services. Fog Computing allows bringing computational power to the edge of the network and reduce latency and overheads. However, when applications need access to data that are centrally stored (which is the case of many services, such as video streaming, video on demand, gaming, etc.), the benefits of the Fog can be quickly affected (deterioration of latency, presence of bottlenecks). To avoid these situations, one solution consists of disseminating data in a Fog environment. Offloading data to the Edge involves the use of data replication and placing the replicas on critical network nodes to provide a more cost-effective solution.

This scenario tackles the service placement problem in a slightly different context compared to Scenario 1. Indeed, where to place the service replicas involves satisfying additional and specific requirements and constraints, such as: do not place an identical service in the same place or region, which service replica to select, and so on. Moreover, addressing the SPP here is connect to others issues such as: "Which application components to replicate?," "How many replicas for each service should we create?," "When to create and destroy a copy?," and so on. So many factors make this problem and scenario a very challenging task.

5.2 A Classification of SPP According to Service Placement Taxonomy

Based on the identified scenarios and service placement taxonomy presented in Section 4.2, we propose to categorize some approaches elaborated in the literature. The provided classifications are depicted in Table 6.

A more detailed classification is depicted in Tables 7, 8, 9, 10, and 11. In these tables, we propose to outline the placement requirements considered (based on those mentioned in Section 4.1.3) and describe the individual contributions of each work. Table 7 (respectively, Table 8, Table 9, Table 10, and Table 11) is dedicated to approaches dealing with Scenario 1.1 (respectively, Scenario 1.2, Scenario 1.3, Scenario 1.4, and Scenario 2). The following syntax is introduced to associate each work to the related taxonomy:

[C|Di] / [Off|On] / [S|Dy] / [nM|M].

The first parameter denotes whether the considered control plan is **C**entralized or **Di**stributed. The second parameter denotes whether the scheduling is performed in **Off**line or **On**line manner. The third one denotes whether the placement is **S**tatic (i.e., considers unchanging infrastructure and applications topologies and information) or **Dy**namic (i.e., handles the dynamicity of the system). Finally, the fourth parameter denotes whether the provided strategy supports the mobility (**M**) of end-users and/or Fog devices or not (**nM**). So, an approach denoted as C/On/Dy/nM will be a centralized, online, dynamic, and not support mobility. This description allows categorizing quickly any given proposals and properly comparing with similar approaches. We note that each of these categories is mutually independent.

Table 6. A Classification of SPP According to Service Placement Taxonomy

Scenario	Reference	Service placement taxonomy			
		Control plane	Online	Dynamic	Mobility
Scenario 1.1	[75, 97]	C			
	[13, 41, 80, 96]	C	√		
	[129, 161]	C	√	√	
	[91, 107]	C	√	√	√
	[12]	Di			
	[93, 95, 160]	Di	√		
Scenario 1.2	[40, 46, 67–69, 71, 78, 82, 86, 89, 90, 92, 110, 111, 131]	C	√		
	[52, 64]	C	√	√	
	[56, 57]	C	√	√	√
	[51, 62, 81, 98, 102, 103, 114, 146]	Di	√		
	[49, 55]	Di	√	√	√
Scenario 1.3	[76]	C			
	[84, 120]	C	√		
	[58–60]	C	√	√	√
	[10, 87, 94, 99]	Di	√		
Scenario 1.4	[9, 11, 45, 48, 66, 72, 77, 79, 88, 108, 109, 125, 126]	C			
	[63, 65, 70, 83, 85, 100, 101, 117, 162]	C	√		
	[35, 73, 106]	C	√	√	
	[53, 54]	C	√	√	√
	[127]	Di	√		
Scenario 2	[105, 121]	C			
	[104, 123, 150]	C	√		
	[115, 130]	Di	√	√	

The √ mark means that the criterion is met; otherwise, the criterion is not met or not considered.

5.3 A Classification of SPP According to Resolution Approaches

We propose now to group similar works based on the used resolution approach. Some solutions along with their objectives are detailed in Tables 12, 13, 14, 15, and 16. Each table refers to the aforementioned scenarios (see Section 5.1).

Based on the provided tables, we propose in the next section to discuss the open research directions and the challenges related to SPP in Fog Computing.

6 EMERGING RESEARCH DIRECTIONS

This section discusses the current limitations and highlights the challenges related to service placement problem in the Fog Computing. Three main directions that need attention in the near future are identified: challenges related to the problem statement, optimization strategies, and evaluation environment.

6.1 Challenges Related to Problem Statement

Despite the recent research efforts outlined on SPP, many challenges still remain open, and one of them concerns the "problem statement," i.e., which problem we try to address (fits which scenario), which important information needs to be considered (infrastructure information, application description, mapping requirements), and under which aspects to address it (taxonomy). Here, we present some of the identified open problems.

Table 7. Taxonomy Dedicated to Scenario 1.1

Category	Reference	Requirements	Contributions
C/Off/S/nM	[75]	C_R	Provides data placement policy to help mobile applications by leveraging the edge resources offered by Fog Computing.
	[97]	C_R, C_N	Proposes a framework based on network functions virtualization to deploy services provided by the Cloud onto the FNs.
C/On/S/nM	[80]	C_R, C_D	Provides task scheduling algorithm in Fog-based system.
	[13]	C_R, C_N, C_D	Proposes a latency-aware application management policy to achieve improvements in network conditions and service QoS.
	[41]	C_R, C_N	Designs a score-based scheduling framework for latency-sensitive applications that maximize the end-users service quality experienced.
	[96]	C_R, C_N	Introduces an online optimization scheme for the task distribution under uncertainties in Fog network.
C/On/Dy/nM	[129]	C_R, C_N, C_D	Provides an online placement approach that attempts to fairly satisfy all web applications.
	[161]	C_R, C_N, C_L	Analyzes service placement strategies in dynamic IoT scenarios considering cloud computing, Fog Computing, and their combination F2C Computing.
C/On/Dy/M	[91]	C_R, C_N, C_L	Designs a dynamic and mobility-aware service placement framework to provide a desirable performance-cost trade-off in Mobile Edge Computing.
	[107]	C_R, C_N	Defines a mobility-aware dynamic service-migration mechanism based on the behavioral cognition of a mobile user in Edge Cognitive Computing.
Di/Off/S/nM	[12]	C_R, C_N, C_D	Provides a new formulation for the QoS-aware service allocation problem for Combined Fog-Cloud architectures.
Di/On/S/nM	[160]	C_R, C_N	Describes an efficient placement of Fog application modules either on the Edge or in the Cloud.
	[95]	C_R, C_N	Presents a conceptual framework for Fog resource provisioning.
	[93]	C_R, C_D	Proposes a security- and deadline-aware task scheduling algorithm.

6.1.1 Scenario That Considers Dependencies at the Data Level. Based on the surveyed papers, we identified two main scenarios considered by the research community (see Section 5.1), however, we noticed that for most works that fit into the aforementioned scenarios, an important use-case was not really (and marginally) considered in SPP. It corresponds to "computation placement with data dependencies," which represents one of the problems that motivate the Fog, since the goal of the Fog Computing is to keep the services close to the devices that produce and act on the data. Thus, when deploying services, the mapping must consider the dependencies at the data level, either in terms of locality (e.g., if a service is deployed in a particular zone, the related components must also be deployed in the same area for security reasons, for instance) or in terms of minimizing the flow of data exchanged, and so on. Marginally addressed in the literature (to the best of our knowledge), the application placement with data dependencies represents real challenges that need to be more considered and studied.

Table 8. Taxonomy Dedicated to Scenario 1.2

Category	Reference	Requirements	Contributions
C/On/S/nM	[40]	C_R, C_N, C_D	Proposes a cost-efficient resource provisioning strategy in Fog environment.
	[111]	C_R, C_N, C_D	Provides latency-aware deployment policy combined with anycast strategies for Fog and Cloud service provisioning.
	[67]	$C_R, C_N\ C_D$	Minimizes power costs when distributing workload in Cloud/Fog systems.
	[68]	$C_R, C_N\ C_D$	Defines a workload allocation policy for the Fog-Cloud Computing services and investigates the trade-off between power consumption and delay.
	[69]	C_R	Investigates the problem of service deployment while minimizing carbon footprint for video streaming service in Fog architecture.
	[71]	C_R, C_N, C_L, C_D	Provides a proactive tasks placement in Fog networks under latency and reliability constraints.
	[110]	C_R, C_N, C_D	Provides data placement policy in context of incidental disasters.
	[74]	C_R, C_N, C_D	Proposes a heuristic algorithm to address the task distribution and virtual machine placement problem toward cost-efficient Fog computation and medical cyber-physical systems.
	[78]	C_R, C_D	Provides a mathematical service placement model in Fog architecture.
	[82]	C_R	Elaborates a data placement policy in Fog architectures that aims to reduce network usage.
	[86]	C_R, C_N, C_D	Proposes a balanced energy-delay solution for IoT applications placement and energy consumption problem in Fog Computing.
	[89, 90]	C_R, C_N	Provides a framework called iFogStor for IoT data placement in a Fog infrastructure.
	[92]	C_R	Provides a placement mechanism that models the competition between IoT users and the efficient service deployment over a hierarchical Fog-Cloud computing system.
	[46]	C_R, C_N, C_D	Proposes provisioning schemes for real-time processing applications in Fog Computing.
C/On/Dy/nM	[64]	C_R, C_N, C_D	Proposes a set of strategies for service placement in Edge-Cloud environment.
	[52]	C_R, C_D	Introduces a dynamic Fog service provisioning policy that meets QoS constraints.
C/On/Dy/M	[57]	C_R, C_N	Designs an online control algorithm that provides where and when services should be migrated according to demand variation and user mobility in Edge-Cloud networks.
	[56]	C_R, C_L	Proposes a strategy that dynamically routes data to proper fog nodes in the context of real-time surveillance applications.
Di/On/S/nM	[114]	C_R, C_N, C_L	Proposes an algorithm to distribute workload in Fog Computing environment that minimize the response time and costs.
	[62]	C_R, C_N, C_D	Elaborates an approach for service mapping and service delegation between Fog and Cloud Computing.
	[146]	C_R, C_D	Proposes a QoS-aware service deployment technique in Fog Computing to reduce latency and network congestion.
	[81]	C_R, C_N	Proposes an online optimization framework to perform efficient task distribution over hybrid Fog-Cloud architecture.
	[51]	C_R, C_N	Provides an online dispatching and scheduling mechanism of tasks to distributed Edge-Cloud systems.
	[98]	C_R	Develops workload placement algorithms to efficiently execute mobile programs in the Edge-Cloud network.
	[103]	C_R, C_D	Proposes a delay-minimizing policy for IoT applications placement over IoT-Fog-Cloud network.
	[102]	C_R	Proposes a delay-minimizing task offloading scheme for IoT applications in Fog Computing.
Di/On/Dy/M	[49]	C_R, C_N, C_D	Proposes a new model to coordinate service deployment and migration that includes latency, location awareness, and mobility support in the smart grid.

Table 9. Taxonomy Dedicated to Scenario 1.3

Category	Reference	Requirements	Contributions
C/Off/S/nM	[76]	C_R	Proposes a heuristic algorithm to solve service deployment problem in Fog Computing.
C/On/S/nM	[84]	C_R, C_N	Designs an algorithm for traffic-aware VM placement and determines the maximum number of accepted VMs in the cloudlet mesh.
	[120]	C_R	Proposes a dynamic service mapping strategy that optimizes resource utilization and satisfies the users' QoS requirements in Fog Computing.
C/On/Dy/M	[60]	C_R, C_N, C_D	Evaluates resource provisioning in Smart City scenarios by providing an IoT application service placement mechanism.
	[59]	C_R, C_N, C_L, C_D	Proposes an application deployment and migration mechanism for geo-distributed systems.
	[58]	C_R	Proposes a service placement architecture for the IoT that continuously adapts (migrate) services according to the network, changing conditions and users' status.
Di/On/S/nM	[87]	C_R, C_N, C_D	Provides a service placement policy that leverages context-aware information (location, time, and QoS) in Fog landscapes.
	[10, 94]	C_R, C_N, C_D	Proposes provisioning and service placement approach to enable the exploitation of Fog-based computational resources.
	[99]	C_R, C_D	Provides a mapping strategy of IoT applications that optimizes resource utilization and satisfies QoS requirements.

6.1.2 Distributed Service Deployment. As mentioned earlier in the article, distributing the decision-making to multiple substrate nodes instead of relying the mapping on a single central node helps to spread the load and possibly increase scalability. With infrastructure such as Fog Computing, there is a tendency to believe that most works are based on this process. However, according to Tables 6, 7, 8, 9, 10, and 11, we observe that there is a lack of solutions proposed in the literature that address the SPP in a distributed way. This is mainly due to the fact that distributed algorithms are difficult to construct, and their implementations are often non-trivial to achieve in a real environment due to the complexity of the inter-process communication and synchronization. Moreover, the lack of knowledge of the global state makes it difficult to compute optimal solutions and even to reach near-optimal placement. We remark also that only a few approaches perform SPP in a distributed and dynamic way [49, 55, 115, 130] and still less that consider distributed solutions that handle dynamicity of the system and support mobility of end-users/FNs (References [49, 55] for Scenario 1.2). Thus, research in these directions is still open.

6.2 Challenges Related to Optimization Strategies

We detail hereafter some of the challenges and opportunities worth further research in terms of optimization strategy point of view.

6.2.1 Optimized Objective. Depending on the studied problem and considered use-case, different metrics can be observed: Latency/bandwidth between end-user, robustness in case of failures, energy consumption/battery life cycle of the sensors, cost, application-specific metric (frame/packet loss, end-to-end delay in processing some data), and so on. According to Table 1, we can easily observe that the performance metrics are usually taken as individual objectives

Table 10. Taxonomy Dedicated to Scenario 1.4

Category	Reference	Requirements	Contributions
C/Off/S/nM	[66]	C_R, C_N	Provides a latency-aware service placement heuristic in Cloud-Fog environment.
	[11, 126]	C_R, C_N, C_L	Proposes a QoS-aware deployment strategy for multi-component IoT applications in Fog infrastructure.
	[108]	C_R, C_N, C_L	Designs a task deployment framework for Mobile Edge Cloud Offloading that achieves a trade-off between applications' runtime, mobile device battery lifetime, and cost for the user.
	[72]	C_R, C_N	Provides a set of heuristics to address the service placement problem in Cloud-Fog network.
	[48]	C_R, C_N, C_L	Designs, implements, and evaluates a Fog Computing platform that runs analytics on multiple devices.
	[77]	C_R, C_N	Evaluates a set of heuristic algorithms for solving the service placement problem in the context of next-generation network architectures.
	[79]	C_R, C_N	Addresses the task assignment problem for the virtual Fog access point.
	[88]	C_R, C_N	Proposes a task deployment policy that assigns the processing tasks to nodes that provide the optimal processing time and near-optimal networking costs in Edge-Fog Cloud system.
	[109]	C_R, C_N	Provides a scheduling algorithm that guarantees application performance and reduces the cost of using Cloud resources.
	[9]	C_R	Proposes a service mapping solution for efficient resources utilization in the Fog infrastructure.
	[45]	C_R, C_N, C_L	Proposes a mechanism for placing distributed IoT applications in Fog infrastructure while minimizing the services response time.
C/On/S/nM	[63]	C_R, C_N, C_L	Provides an optimization placement framework of data stream processing applications that minimize end-to-end latency in Edge-Cloud Computing.
	[65]	C_R, C_N, C_D	Designs an IoT service placement strategy for IoT-Cloud infrastructure that satisfies end-user demands and minimizes overall operational cost.
	[117]	C_R, C_N	Proposes a solution for the distributed data stream application placement in a geographically distributed environment.
	[70]	C_R, C_N	Proposes orchestration mechanisms for service provisioning in Fog network that minimizes the provisioning cost of IoT applications.
	[38]	C_R, C_N	Proposes a transfer-time aware service scheduling policy for IoT-Fog-Cloud Computing environments.
	[83]	C_R, C_N, C_D	Proposes a policy for service placement in Fog devices based on communities and transitive closures notions.
	[85]	C_R	Proposes an energy-aware algorithm for mapping application components on Fog devices.
	[100]	C_R, C_N, C_L	Addresses the multi-component application placement problem in Edge environments and develops algorithms with provable performance bounds.
	[101]	C_R, C_N	Builds the Latency-aware Video Edge Analytics system to investigate task placement schemes.
C/On/Dy/nM	[106]	C_R, C_L	Proposes a decision-making strategy for virtual machine placement considering multiple optimization objectives.
	[73]	C_R	Proposes adaptive scheduling strategies for dynamic event analytic dataflows placement in Edge-Cloud devices that support Smart City applications' emerging needs.
	[35]	C_R, C_D	Refines the service provisioning algorithm to guarantee the application deadlines and optimizes Edge resource exploitation.
C/On/Dy/M	[53]	C_R, C_N	Proposes an online mobile application placement algorithm that minimizes the services computational cost.
	[54]	C_R, C_N, C_D	Proposes a service placement and migration policy for mobile event processing applications in Cloud and Fog resources.
Di/On/S/nM	[127]	C_R, C_N	Introduces a collaborative approach of executing application components in a distributed manner between fog devices.

Table 11. Taxonomy Dedicated to Scenario 2

Category	Reference	Requirements	Contributions
C/Off/S/nM	[121]	C_R, C_N	Proposes an efficient cache placement strategy based on content popularity to reduce energy consumption in Fog networks.
	[105]	C_R, C_N	Proposes a placement algorithm for virtual machine replica in Mobile Edge Computing that minimizes the average data traffic in the network.
C/On/S/nM	[123]	C_R, C_N, C_L	Provides two request-routing strategies to tackle the problem of energy-efficient and latency-aware data placement in geo-distributed Cloud data centers.
	[150]	C_R	Provides a Fog-aware replica placement algorithm based on the definition of failure groups.
	[104]	C_R, C_N	Designs a task scheduling strategy that minimizes task completion time for promoting the user experience.
Di/On/Dy/nM	[115]	C_R, C_N, C_L	Elaborates a dynamic placement (creation/replacement/removal) of data replicas across IaaS providers.
	[130]	C_R, C_N, C_L	Provides a bandwidth and availability-aware policy for service deployment on Micro-Cloud infrastructures that maximizes user QoS and QoE.

(minimize latency, maximize the number of satisfied applications, minimize energy consumption, etc.). However, to improve QoS and QoE, these metrics should be simultaneously considered in the objective function, instead of integrating them into the model as constraint functions. Considering multiple objectives at the same time has not received significant attention so far. Due to the complexity of such problems, only a few works have attempted to address such issues [60, 69, 104, 106–109]. Indeed, conducting multi-objective optimization and deriving effective solutions require a huge computational effort [165]. Multi-objective metaheuristics can be explored to solve such a problem, such as MOPSO [166] and NSGA-II [167], or scheduling approaches like Reference [168].

6.2.2 Solutions That Support Mobility. Due to the high mobility of some end-users and Fog devices, the elaborated solutions must ensure the continuity of the services and that the users always receive the desired requests. For this, the services must follow this mobility and perform some migrations across different Fog instances. When reviewing the literature, we observed that only a few works propose to provide a solution that supports the mobility of end-users/FNs (see Table 6). Moreover, most of the dynamic application migration approaches elaborated consider a restricted application topology (monolithic, line, or tree graph), support only the mobility of end-users (not those of FNs), and developed dynamic solutions based on complex techniques and algorithms. In view of that, it is clear that the definition of standard algorithmic techniques appropriate for implementation in real Fog systems is a real need for the community. Moreover, we have observed that, currently, most of the placement techniques that support mobility are reactive. Thus, addressing the problem from a proactive point of view by predicting the mobility pattern of users and devices could be more suitable in the Fog context. Indeed, understanding the movement behavior of end-devices (or FN) may be helpful for an efficient service placement and service management in Fog Computing.

6.2.3 Energy-efficiency. Regarding the energy consumption in Fog service management, we found that research is less prevalent in some aspects of energy consumption in the Fog

Table 12. Classification According to Resolution Approaches Dedicated to Scenario 1.1

Category	Solutions	References	Objective and a brief description of the resolution technique.
C/Off/S/nM	Exact	[97]	*Minimizes the number of Fog nodes.* Exact resolution of the ILP using CPLEX solver.
		[75]	*Minimizes the overall communication cost.* Uses linear programming solver.
C/On/S/nM	Approximation	[96]	*Minimizes the total response time over all the deployed tasks.* Uses a dual-fitting algorithm to find a feasible solution.
	Heuristic	[80]	*Maximizes the utilization of residual computing capabilities of terminals.* Provides two heuristics: (1) Prioritize the task with the earliest deadline. (2) Choose the device with the minimum remaining computation capacity.
		[41]	*Maximizes the QoE.* Computes a quality score that combines connectivity, bandwidth, and resource scores to deploy a service on the most suitable VM.
	Meta-heuristic	[13]	*Maximizes QoE-gain of the user.* Uses a fuzzy-logic-based reasoning.
C/On/Dy/nM	Heuristic	[129]	*Equally satisfies the applications.* Uses utility-driven application placement policy.
		[161]	*Optimizes the network usage.* Provides customized versions of strategies such as first-fit and best-fit algorithms.
C/On/Dy/M	Heuristic	[91]	*Minimizes the average service latency under cost budget constraints.* Uses Lyapunov optimization to decompose the problem into a set of problems that do not require *a priori* knowledge of user mobility.
	Machine Learning	[107]	*Minimizes the service costs, meantime, and improves the QoE.* Uses Q-learning method [163] to determine for each service request the host (node) that provides the optimal migration.
Di/Off/S/nM	Exact	[12]	*Minimizes the service latencies in a Fog while satisfying the QoS requirements.* Uses Gurobi optimizer [164] to solve the ILP provided model.
Di/On/S/nM	Heuristic	[93]	*Minimizes the cost of the user job.* Sorts the jobs in increasing order of deadlines.
		[95]	*Maximizes the utilization of Fog resources.* Uses a fog colonies notion and simulation approach to perform provisioning plan of requested services.
		[160]	*Maximizes the application deployment revenue.* Operates an iterative application deployment by region.

environment. We identify the following factors: a model that considers the carbon footprint, uses renewable energy sources like wind turbines or solar panels, and so on, take into account the energy-constrained of end-devices (sensors) in terms of residual battery lifetime, the energy of communication links, and so on. The use of Follow the Sun and Follow the Moon strategies in these cases could be helpful to manage efficiently the energy consumption; for instance, by controlling the devices in terms of intelligent lighting, ventilation, air conditioning, and so on.

6.2.4 Generic and Effective Mapping. It is clearly seen in Section 4.3 that many formulations and solutions are developed to address the SPP for a specific application. These placement policies given in the literature (mainly heuristics) consider different assumptions (infrastructure information, application topology, QoS attributes and metrics, etc.) and different objectives that make them not easily comparable. Indeed, given the diversity of criteria, handling all these parameters is practically impossible. So, the questions arising today are: *Which criteria are most significant and should be considered to develop an effective solution? According to these criteria, can we compare the existing*

Table 13. Resolution Approaches Dedicated to Scenario 1.2

Category	Solutions	References	Objective and a brief description of the resolution technique.
C/Off/S/nM	Approximation	[46]	*Finds the minimum congestion ratio.* Uses *fully polynomial-time approximation* that, to reduce the resolution complexity of IoT application provisioning, proposes to decompose the initial problem into two sub-problems to be solved separately.
		[92]	*Each user maximizes its own QoE.* Uses *ϵ-Nash equilibrium* and offloading game to model the competition between end-users and provides near-optimal service mapping.
	Heuristic	[67, 68]	*Minimizes the power consumption in the Cloud-Fog Computing.* Decomposes the initial problem into three sub-problems that can be independently solved and uses convex optimization techniques, Generalized Benders' Decomposition, and Hungarian method to find feasible solution.
		[110]	*Minimizes service delay and expensive resource over provisioning.* Computes the shortest path that satisfies application QoS constraints.
		[78]	*Minimizes the blocking probability* (ratio between a number of rejected workloads and the total number of workloads). Proposes three resolution approaches: Random, Lowest latency, and Maximum available capacity policies.
		[82]	*Optimizes the network usage.* Determines the Fog device with the closest and most even distance to the data sources. The placement decision considers the FN with the highest centrality value.
		[89, 90]	*Minimizes the overall latency of storing and retrieving data in a Fog.* Provides a geographical partition to decrease the problem-solving time.
		[40, 69, 71, 74, 111]	
	Meta-heuristic	[86]	*Minimizes the total energy consumption of mobile applications.* Uses a modified genetic algorithm.
C/On/Dy/nM	Heuristic	[64]	*Minimizes failed requests.* Proposes two heuristics: strictest deadline first, and first-in/first-out policies.
		[52]	*Minimizes overall cost (processing, storage, and communication).* Proposes two heuristics: (1) Min-Viol: aims at minimizing the deadline violations. (2) Min-Cost: aims at minimizing the total cost.
C/On/Dy/M	Heuristic	[57]	*Minimizes the cost of execution, delays, and location constraints.* Decouples the initial Markov Decision Process (MDP) into two independent MDPs and solves the problems using a Lyapunov optimization.
		[56]	*Minimizes the average cost over time.* When emergency event happens, computes reconfiguration and reallocation only on the impacted zone.
Di/On/S/nM	Heuristic	[114]	*Minimizes the response time and maximizes the throughput.* Schedules jobs on VMs based on service-level agreement.
		[62]	*Meets SLA and QoS.* Prioritizes the mapping based on a linearized decision composed by services size, completion time, and VMs capacity. Components with higher priority are mapped first and the ones with lower priority are deployed last.
		[98]	*Minimizes the cumulative delay of executing mobile services.* Solves the MNIP problem as follows: first transforms the mixed nonlinear integer program to a convex optimization problem and then solves the problem that only contains the integer variables.
		[103]	*Reduces the service delay for IoT applications.* Compares the estimated waiting time of task at a given FN with their deadline. If it is smaller, accept the task; if not, the FN offloads the service to one of its neighbors or to the Cloud.
		[51, 81, 102, 146]	
Di/On/Dy/M	Heuristic	[49]	*Reduces the application delay of IoT applications in the Smart Grid.* Chooses the best nodes from the set of nodes that satisfy QoS of end-nodes and having the nearest deadline.

Table 14. Classification According to Resolution Approaches Dedicated to Scenario 1.3

Category	Solutions	References	Objective and a brief description of the resolution technique.
C/Off/S/nM	Heuristic	[76]	*Maximizes the number of satisfied services.* Considers the applications with fewer components first (if the same, takes those that have the least feasible resources).
C/On/S/nM	Heuristic	[84]	*Minimizes the total inter-cloudlet communication traffic in cloudlet mesh.* Iterates on jobs and provides placement that minimizes inter-cloudlet communication and satisfies the communication demands and resource constraints.
C/On/Dy/M	Heuristic	[60]	*Optimizes multiple objectives: maximize number of accepted IoT application requests, maximize service bandwidth, minimize service migrations between iterations, minimize number of active computational nodes, minimize the number of active gateways, minimize hop count between computational nodes and end -devices, and minimize path loss.* Computes iteratively the solution so every iteration refines the previous obtained solution by improving the model with an additional optimization objective.
		[59]	*Optimizes tasks mapping by saving bandwidth and reducing latency.* Determines incrementally the fog nodes that match the capacity constraints and handles the application dynamism.
		[58]	*Minimizes: the hop count between end-nodes and hosting nodes, the hop count between communication nodes, and the number of service migrations.* Deploys the services in random locations and later adapts to the network and application constraints and requirements.
Di/On/S/nM	Exact	[87]	*Maximizes the number of deployed applications.* Uses ILP solver.
		[10]	*Maximizes the number of services deployed on Fog landscape.* Uses ILP solver.
	Heuristic	[10]	*Maximizes the number of deployed services to Fog infrastructure.* Prioritizes the applications having the minimum value between their deadline and their deployment time.
	Meta-heuristic	[94]	*Maximizes the number of deployed services to Fog devices rather than to Cloud ones.* Uses a genetic algorithm.

proposals and identify the relevant approaches? Or should we make a clean sweep and propose a new generic and easy-to-upgrade methodology? Many open issues that deserve to be deepened.

6.3 Challenges Related to Evaluation Environments

We propose to describe here one of the challenges that in our opinion is the most important regarding the evaluation environments' point of view.

6.3.1 Uniform Environment. Through Table 4, we can easily observe that different tools are used by the research community to test and perform their experiments. Each tool has its own specificities and is adapted to a given problem (use-case), as described in Section 4.4. While there are a number of works that have addressed the SPP challenge, we notice that there is not yet a generic development environment that handles a large range of standardized IoT applications and allows considering various network topologies. So, there is a requirement of making a uniform platform involving most of the concepts that is easy to take in hand and that favors the realization of extensive experiences. Based on the classification scheme and the scenarios provided in Section 5.1, our contribution throughout this article consists to bring some basics on which we can rely to design a generic and extensible model that will take over the different Fog Computing use cases.

Table 15. Resolution Approaches Dedicated to Scenario 1.4

Category	Solutions	References	Objective and a brief description of the resolution techniques.
C/Off/S/nM	Exact	[125]	*Provides feasible (respectively, optimal) service placement solutions in Fog environment.* Uses the constraint programming Choco-solver.
		[66]	*Minimizes the overall latency and ensures the QoS requirements.* Uses ILP-solver CPLEX.
	Heuristic	[11]	*Determines eligible deployments of composite applications.* Performs pre-processing plus backtracking to determine an eligible deployment.
		[126]	*Determines eligible deployments of composite applications.* Exploits Monte Carlo simulations [153] to handle the communication links variations and performs pre-processing plus backtracking to determine the final eligible deployment.
		[108]	*Optimizes the following objectives: minimize runtime and user cost, and maximize battery lifetime.* Computes the local optimum solution for each objective and then selects the one with the lowest score (calculated according to a given equation).
		[72]	*Minimizes the overall cost (placement and link costs).* Provides six heuristics: (1) Limits the deployment of an application component to a subset of nodes; (2) Restricts the placement of a component to one particular node; (3) Applies the co-location of some components to one node; Accelerates the previous heuristics by (4) Combining heuristics (2) and (1); 5) Combines heuristics (2) and (3); and (6) Combines heuristics (1) and (3).
		[48]	*Maximizes the number of satisfied IoT analytics.* Prioritizes the scarcest resource first and the closer to source device next.
		[77]	*Minimizes end-to-end delay.* Elaborates a layered graph placement algorithm that proposes to find a lowest cost path that includes communication cost and processing cost.
		[79]	*Minimizes maximum cost service node.* Selects the mapping with the minimum total cost by iterating on the set of all possible mappings.
		[88]	*Minimizes network cost.* Minimizes processing cost first and then optimizes the network cost.
		[9]	*Optimizes utilization of network resources.* Prioritizes the components placement based on the resource expectation.
		[45]	*Minimizes the average response time.* Proposes three heuristics based on backtracking solution and the notion of anchor.
C/On/S/nM	Exact	[65]	*Minimizes overall operational cost.* Uses the linear programming solver Xpress-MP.
		[117]	*Minimizes the application end-to-end latency.* Uses the ILP solver.
	Appro.	[100]	*Minimizes the maximum weighted cost on network nodes and links.* Provides polynomial-logarithmic worst-case optimality bound. Splits the application graph into simple branches and solves the problem recursively.
	Heuristic	[63]	*Minimizes end-to-end latency.* Breaks down the latency calculation into computational latency and network transfer latency, and minimizes the sum.
		[70]	*Minimizes the provisioning cost.* Adopts a divide-and-conquer approach and incrementally computes the best solution.
		[162]	*Determines an eligible application placement.* Prioritizes inter-dependent components based on the computation cost and communication time.
		[83]	*Minimizes the network delays between interrelated services while optimizing the QoS and the service availability for the users.* Uses a first fit decreasing approach to place applications in device communities and then prioritizes the applications with the shortest deadlines.
		[85]	*Minimizes energy consumption.* Deploys the incoming application components to Fog resources based on the remaining CPU, energy consumption, and related deadline.
		[101]	*Minimizes the response time.* Performs sequential quadratic programming and divides the problem into two sub-problems to minimize computational complexity.
C/On/Dy/nM	Heuristic	[35]	*Minimizes the network cost during the task assignment.* Proposes to minimize first the processing cost and then optimizes the network cost.
	Meta-heuristic	[106]	*Optimizes a multi-objectives function: Minimize cost, maximize user support, minimize latency, and maximize user footprint.* Converts the multi-objective optimization problem to a single objective problem by using a scalarization method. And provides Pareto optimal solution.
		[73]	*Minimizes the total makespan while meeting energy and QoS constraints.* Proposes two prior GA meta-heuristics (GA-Incremental and GA-Global) to support dynamically the multiple dataflows arriving and departing.
C/On/Dy/M	Heuristic	[53]	*Minimizes the cost to run the application.* Performs an iterative matching process and local search phase to compute the best solution.
		[54]	*Minimizes the costs of migration and placement of a single component.* Creates time-graph model and considers the shortest path from data source to identify possible migration nodes.
Di/On/S/nM	Heuristic	[127]	*Minimizes the cost to run the application.* Prioritizes the placement according to dependencies between the components and computational power of edge devices.

Table 16. Classification According to Resolution Approaches Dedicated to Scenario 2

Category	**Solutions**	**References**	**Objective and a brief description of the resolution technique.**
C/Off/S/nM	Exact	[105]	*Minimizes the average data traffic in the Edge environment.* Exhaustive research is performed, i.e., enumerates all placements of service replica and selects the solution that minimizes the objective among all computed placement solutions.
	Heuristic	[105]	*Minimizes the overall latency of storing and retrieving data in a Fog.* Splits the original service placement problem into set of subproblems each performing an optimal placement solution for one component.
		[121]	*Maximizes the energy efficiency while maintaining the successful delivery.* The provided algorithm proposes to categorize the services into three popularity levels and strategically cache them in Fog resources.
C/On/S/nM	Heuristic	[104]	*Minimizes the maximum average task completion time.* First, partition the problem on two sub-problems and optimize each of them. Then, re-couple the two solutions to optimize the main objective.
		[150]	*Achieves minimal latency in between the replicas and between the replicas and the data sources and sink.* Exploits end-users and service locality in Fog network when deploying.
Di/On/Dy/nM	Heuristic	[115]	*Defines a trade-off between cost and latency.* Evaluates cost of storing replicas and expected latency improvement to make a migration or duplication decision.
		[130]	*Maximizes the end-to-end performance.* Provides bandwidth and availability-aware service placement policy.

7 CONCLUSION

This article focuses on the Service Placement Problem (SPP) in a Fog environment, which is currently an open issue that calls for extensive discussions and solutions. This article gives a survey of current works. A description of the SPP was provided. Five scenarios related to this issue were identified. A categorization of solutions along four distinct dimensions (centralized vs. distributed control plan, offline vs. online scheduling, static vs. dynamic system, not support vs. support mobility of end-users/fog nodes) was elaborated. A number of algorithmic proposals to the SPP were discussed. Finally, these aspects were used to create a classification of SPP solutions elaborated in the literature based on placement taxonomy. More precisely and compared to existing surveys, our work highlights a new classification scheme that aims to simplify the user's access to references in a specific context and identify more easily the placement-related challenges.

REFERENCES

[1] Cisco. 2018. Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are. White Paper. 2016. Retrieved on 8 April, 2018 from http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf.

[2] S. Yi, Z. Qin, and Q. Li. 2015. Security and privacy issues of fog computing: A survey. In *Wireless Algorithms, Systems, and Applications*, Kuai Xu and Haojin Zhu (Eds.). Springer International Publishing, Cham, 685–695.

[3] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu. 2014. *Fog Computing: A Platform for Internet of Things and Analytics.* Springer International Publishing, Cham, 169–186.

[4] M. Chiang and T. Zhang. 2016. Fog and IoT: An overview of research opportunities. *IEEE Internet Things J.* 3, 6 (Dec. 2016), 854–864. DOI: http://dx.doi.org/10.1109/JIOT.2016.2584538

[5] E. Elmroth, P. Leitner, S. Schulte, and S. Venugopal. 2017. Connecting fog and cloud computing. *IEEE Cloud Comput.* 4, 2 (Mar. 2017), 22–25. DOI: http://dx.doi.org/10.1109/MCC.2017.29

[6] Kashif Bilal, Osman Khalid, Aiman Erbad, and Samee U. Khan. 2018. Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers. *Comput. Netw.* 130 (2018), 94–120. DOI: http://dx.doi.org/10.1016/j.comnet.2017.10.002

[7] P. Hu, S. Dhelim, H. Ning, and T. Qiu. 2017. Survey on fog computing: Architecture, key technologies, applications and open issues. *J. Netw. Comput. Applic.* 98 (2017), 27–42. DOI: http://dx.doi.org/10.1016/j.jnca.2017.09.002

[8] A. Vahid Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya. 2016. Fog computing: Principles, architectures, and applications. *CoRR* abs/1601.02752 (2016).

[9] M. Taneja and A. Davy. 2017. Resource aware placement of IoT application modules in fog-cloud computing paradigm. In *Proceedings of the IFIP/IEEE IM.* 1222–1228.

[10] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar. 2017. Towards QoS-aware fog service placement. In *Proceedings of the ICFEC.* IEEE Computer Society, 89–96.

[11] A. Brogi and S. Forti. 2017. QoS-aware deployment of IoT applications through the fog. *IEEE Internet Things J.* 4, 5 (Oct. 2017), 1185–1192.

[12] V. B. C. Souza, W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren, and G. Tashakor. 2016. Handling service allocation in combined fog-cloud scenarios. In *Proceedings of the IEEE ICC.* 1–5.

[13] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya. 2018. Quality of experience (QoE)-aware placement of applications in fog computing environments. *Journal of Parallel and Distributed Computing* 132 (2019), 190–203. DOI: https://doi.org/10.1016/j.jpdc.2018.03.004

[14] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. 2012. Fog computing and its role in the Internet of Things. In *Proceedings of the MCC.* ACM, New York, NY, 13–16.

[15] K. P. Saharan and Anuj Kumar. 2015. Fog in comparison to cloud: A survey. *Int. J. Comput. Applic.* 122, 3 (July 2015), 10–12.

[16] S. Yi, C. Li, and Q. Li. 2015. A survey of fog computing: Concepts, applications, and issues. In *Proceedings of the Mobidata.* ACM, New York, NY, 37–42. DOI: http://doi.acm.org/10.1145/2757384.2757397

[17] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, and A. V. Vasilakos. 2017. Fog computing for sustainable smart cities: A survey. *CoRR* abs/1703.07079 (2017).

[18] C. Puliafito, E. Mingozzi, and G. Anastasi. 2017. Fog computing for the internet of mobile things: Issues and challenges. In *Proceedings of the IEEE SMARTCOMP.* 1–6. DOI: http://dx.doi.org/10.1109/SMARTCOMP.2017.7947010

[19] R. Mahmud, R. Kotagiri, and R. Buyya. 2018. *Fog Computing: A Taxonomy, Survey and Future Directions.* Springer Singapore, 103–130. DOI: https://doi.org/10.1007/978-981-10-5861-5_5

[20] M. Mukherjee, L. Shu, and D. Wang. 2018. Survey of fog computing: Fundamental, network applications, and research challenges. *IEEE Commun. Surv. Tutor.* (2018), 1–1. DOI: http://dx.doi.org/10.1109/COMST.2018.2814571

[21] S. Sarkar, S. Chatterjee, and S. Misra. 2018. Assessment of the suitability of fog computing in the context of Internet of Things. *IEEE Trans. Cloud Comput.* 6, 1 (Jan. 2018), 46–59.

[22] K. Toczé and S. Nadjm-Tehrani. 2018. A taxonomy for management and optimization of multiple resources in edge computing. *CoRR* abs/1801.05610 (2018).

[23] C. Li, Y. Xue, J. Wang, W. Zhang, and T. Li. 2018. Edge-oriented computing paradigms: A survey on architecture design and system management. *ACM Comput. Surv.* 51, 2, Article 39 (Apr. 2018), 34 pages. DOI: http://dx.doi.org/10.1145/3154815

[24] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue. 2018. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *CoRR* abs/1808.05283 (2018).

[25] A. Brogi, S. Forti, C. Guerrero, and I. Lera. 2019. How to place your apps in the fog—State of the art and open challenges. *CoRR* abs/1901.05717 (2019).

[26] L. M. Vaquero and L. Rodero-Merino. 2014. Finding your way in the fog: Towards a comprehensive definition of fog computing. *SIGCOMM Comput. Commun. Rev.* 44, 5 (Oct. 2014), 27–32. DOI: http://dx.doi.org/10.1145/2677046.2677052

[27] M. Yannuzzi, R. Milito, R. Serral-Gracia, D. Montero, and M. Nemirovsky. 2014. Key ingredients in an IoT recipe: Fog computing, cloud computing, and more fog computing. In *Proceedings of the IEEE CAMAD.* 325–329. DOI: http://dx.doi.org/10.1109/CAMAD.2014.7033259

[28] I. Stojmenovic. 2014. Fog computing: A cloud to the ground support for smart things and machine-to-machine networks. In *Proceedings of the ATNAC.* 117–122.

[29] A. V. Dastjerdi and R. Buyya. 2016. Fog computing: Helping the Internet of Things realize its potential. *Comput.* 49, 8 (Aug. 2016), 112–116. DOI: http://dx.doi.org/10.1109/MC.2016.245

[30] Shubha Brata Nath, Harshit Gupta, Sandip Chakraborty, and Soumya K. Ghosh. 2018. A survey of fog computing and communication: Current researches and future directions. *CoRR* abs/1804.04365 (2018).
[31] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos. 2018. A comprehensive survey on fog computing: State-of-the-art and research challenges. *IEEE Commun. Surv. Tutor.* 20, 1 (2018), 416–464.
[32] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos. 2017. Fog orchestration for Internet of Things services. *IEEE Internet Comput.* 21, 2 (Mar. 2017), 16–24. DOI: http://dx.doi.org/10.1109/MIC.2017.36
[33] S. Sarkar and S. Misra. 2016. Theoretical modelling of fog computing: A green computing paradigm to support IoT applications. *IET Netw.* 5, 2 (2016), 23–29.
[34] F. Li, M. Voegler, M. Claessens, and S. Dustdar. 2013. Efficient and scalable IoT service delivery on cloud. In *Proceedings of the IEEE CLOUD*. 740–747. DOI: http://dx.doi.org/10.1109/CLOUD.2013.64
[35] R. Mahmud, K. Ramamohanarao, and R. Buyya. 2018. Latency-aware application module management for fog computing environments. *ACM Trans. Internet Technol.* 19, 1 (2019), 21 pages.
[36] Y. Ai, M. Peng, and K. Zhang. 2017. Edge computing technologies for Internet of Things: A primer. *Dig. Commun. Netw.* Retrieved from http://www.sciencedirect.com/science/article/pii/S2352864817301335.
[37] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. 2009. The case for VM-based cloudlets in mobile computing. *IEEE Pervas. Comput.* 8, 4 (Oct. 2009), 14–23. DOI: http://dx.doi.org/10.1109/MPRV.2009.82
[38] H. Gupta, A. Vahid Dastjerdi, S.-K. Ghosh, and R. Buyya. 2016. iFogSim: A toolkit for modeling and simulation of resource management techniques in Internet of Things, edge, and fog computing environments. *CoRR* abs/1606.02007 (2016).
[39] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar. 2017. Mobility-aware application scheduling in fog computing. *IEEE Cloud Comput.* 4, 2 (Mar. 2017), 26–35. DOI: http://dx.doi.org/10.1109/MCC.2017.27
[40] H. Reza Arkian, A. Diyanat, and A. Pourkhalili. 2017. MIST: Fog-based data analytics scheme with cost-efficient resource provisioning for IoT crowdsensing applications. *J. Netw. Comput. Applic.* 82 (2017), 152–165. DOI: http://dx.doi.org/10.1016/j.jnca.2017.01.012
[41] V. Scoca, A. Aral, I. Brandic, R. De Nicola, and R. Brundo Uriarte. 2018. Scheduling latency-sensitive applications in edge computing. In *Proceedings of the CLOSER*.
[42] J. K. Zao, T. T. Gan, C. K. You, S. J. R. Méndez, C. E. Chung, Y. T. Wang, T. Mullen, and T. P. Jung. 2014. Augmented brain computer interaction based on fog computing and linked data. In *Proceedings of the IE*. 374–377. DOI: http://dx.doi.org/10.1109/IE.2014.54
[43] T. N. Gia, M. Jiang, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen. 2015. Fog computing in healthcare Internet of Things: A case study on ECG feature extraction. In *Proceedings of the IEEE CIT/IUCC/DASC/PICom*. 356–363. DOI: http://dx.doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.51
[44] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan. 2014. Towards wearable cognitive assistance. In *Proceedings of the MobiSys*. ACM, New York, NY, 68–81. DOI: http://dx.doi.org/10.1145/2594368.2594383
[45] Y. Xia, X. Etchevers, L. Letondeur, T. Coupaye, and F. Desprez. 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 SAC*. ACM, New York, NY, 751–760. DOI: http://dx.doi.org/10.1145/3167132.3167215
[46] R. Yu, G. Xue, and X. Zhang. 2018. Application provisioning in fog computing-enabled Internet-of-Things: A network perspective. In *Proceedings of the IEEE INFOCOM*.
[47] M. Abderrahim, M. Ouzzif, K. Guillouard, J. François, A. Lebre, C. Prud'homme, and X. Lorca. 2019. Efficient resource allocation for multi-tenant monitoring of edge infrastructures. In *Proceedings of the PDP*. 158–165. DOI: http://dx.doi.org/10.1109/EMPDP.2019.8671621
[48] H. J. Hong, P. H. Tsai, A. C. Cheng, M. Y. S. Uddin, N. Venkatasubramanian, and C. H. Hsu. 2017. Supporting Internet-of-Things analytics in a fog computing platform. In *Proceedings of the IEEE CloudCom*. 138–145. DOI: http://dx.doi.org/10.1109/CloudCom.2017.45
[49] P. Wang, S. Liu, F. Ye, and X. Chen. 2018. A fog-based architecture and programming model for IoT applications in the smart grid. *CoRR* abs/1804.01239 (2018).
[50] T. Nishio, R. Shinkuma, T. Takahashi, and N. B. Mandayam. 2013. Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud. In *Proceedings of the MobileCloud*. ACM, New York, NY, 19–26. http://doi.acm.org/10.1145/2492348.2492354.
[51] G. Lee, W. Saad, and M. Bennis. 2017. An online secretary framework for fog network formation with minimal latency. In *Proceedings of the IEEE ICC*. 1–6.
[52] A. Yousefpour, A. Patil, G. Ishigaki, J. P. Jue, I. Kim, X. Wang, H. C. Cankaya, Q. Zhang, and W. Xie. 2017. QoS-aware dynamic fog service provisioning. *CoRR* abs/1802.00800.
[53] T. Bahreini and D. Grosu. 2017. Efficient placement of multi-component applications in edge computing systems. In *Proceedings of the SEC*. ACM, New York, NY, Article 5, 11 pages. DOI: http://dx.doi.org/10.1145/3132211.3134454

[54] B. Ottenwälder, B. Koldehofe, K. Rothermel, and U. Ramachandran. 2013. MigCEP: Operator migration for mobility driven distributed complex event processing. In *Proceedings of the ACM DEBS*. ACM, New York, NY, 183–194. DOI: http://dx.doi.org/10.1145/2488222.2488265

[55] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe. 2013. Mobile fog: A programming model for large-scale applications on the Internet of Things. In *Proceedings of the ACM MCC*. ACM, New York, NY, 15–20.

[56] J. Wang, J. Pan, and F. Esposito. 2017. Elastic urban video surveillance system using edge computing. In *Proceedings of the SmartIoT*. ACM, New York, NY, Article 7, 6 pages. DOI: http://doi.acm.org/10.1145/3132479.3132490

[57] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung. 2015. Dynamic service migration and workload scheduling in edge-clouds. *Perform. Eval.* 91 (2015), 205–228.

[58] K. Velasquez, D. P. Abreu, M. Curado, and E. Monteiro. 2017. Service placement for latency reduction in the Internet of Things. *Ann. Telecommun.* 72, 1 (01 Feb. 2017), 105–115. DOI: https://doi.org/10.1007/s12243-016-0524-9

[59] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, and B. Ottenwälder. 2016. Incremental deployment and migration of geo-distributed situation awareness applications in the fog. In *Proceedings of the ACM DEBS*. ACM, New York, NY, 258–269.

[60] J. Santos, T. Wauters, B. Volckaert, and F. De Turck. 2017. Resource provisioning for IoT application services in smart cities. In *Proceedings of the CNSM*. 1–9. DOI: http://dx.doi.org/10.23919/CNSM.2017.8255974

[61] R. T. Marler and J. S. Arora. 2004. Survey of multi-objective optimization methods for engineering. *Struct. Multidisc. Optim.* 26, 6 (01 Apr. 2004), 369–395. DOI: http://dx.doi.org/10.1007/s00158-003-0368-6

[62] A. Abdullah Alsaffar, P. Phuoc Hung, C. Seon Hong, E.-N. Huh, and M. Aazam. 2016. An architecture of IoT service delegation and resource allocation based on collaboration between fog and cloud computing. *Mob. Inf. Syst.* 2016 (2016), 1–15. DOI: http://dx.doi.org/10.1155/2016/6123234

[63] G. Amarasinghe, M. D. de Assuncao, A. Harwood, and S. Karunasekera. 2018. A data stream processing optimisation framework for edge computing applications. In *Proceedings of the IEEE ISORC*, Vol. 00. 91–98. DOI: http://dx.doi.org/10.1109/ISORC.2018.00020

[64] O. Ascigil, T. K. Phan, A. G. Tasiopoulos, V. Sourlas, I. Psaras, and G. Pavlou. 2017. On uncoordinated service placement in edge-clouds. In *Proceedings of the IEEE CloudCom*. 41–48.

[65] M. Barcelo, A. Correa, J. Llorca, A. M. Tulino, J. L. Vicario, and A. Morell. 2016. IoT-cloud service optimization in next generation smart environments. *IEEE J. Select. Areas Commun.* 34, 12 (Dec. 2016), 4077–4090. DOI: http://dx.doi.org/10.1109/JSAC.2016.2621398

[66] A. R. Benamer, H. Teyeb, and N. Ben Hadj-Alouane. 2018. Latency-aware placement heuristic in fog computing environment. In *Proceedings of the OTM*. Springer International Publishing, Cham, 241–257.

[67] R. Deng, R. Lu, C. Lai, and T. H. Luan. 2015. Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing. In *Proceedings of the IEEE ICC*. 3909–3914. DOI: http://dx.doi.org/10.1109/ICC.2015.7248934

[68] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang. 2016. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. *IEEE Internet Things J.* 3, 6 (Dec. 2016), 1171–1181. DOI: http://dx.doi.org/10.1109/JIOT.2016.2565516

[69] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong. 2015. Approximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing. In *Proceedings of the ICOIN*. 324–329. DOI: http://dx.doi.org/10.1109/ICOIN.2015.7057905

[70] B. Donassolo, I. Fajjari, A. Legrand, and P. Mertikopoulos. 2019. Fog based framework for IoT service provisioning. In *Proceedings of the IEEE CCNC*.

[71] M. Saad El Bamby, M. Bennis, and W. Saad. 2017. Proactive edge computing in latency-constrained fog networks. *CoRR* abs/1704.06749 (2017).

[72] J. Gedeon, M. Stein, L. Wang, and M. Mühlhäuser. 2018. On scalable in-network operator placement for edge computing. In *Proceedings of the ICCCN*. 1–9. DOI: http://dx.doi.org/10.1109/ICCCN.2018.8487419

[73] R. Ghosh, S. Prakash Reddy Komma, and Y. Simmhan. 2018. Adaptive energy-aware scheduling of dynamic event analytics across edge and cloud resources. *CoRR* abs/1801.01087 (2018).

[74] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang. 2017. Cost efficient resource management in fog computing supported medical cyber-physical system. *IEEE Trans. Emerg. Topics Comput.* 5, 1 (Jan. 2017), 108–119.

[75] M. A. Hassan, M. Xiao, Q. Wei, and S. Chen. 2015. Help your mobile applications with fog computing. In *Proceedings of the IEEE SECON Workshops*. 1–6.

[76] H. J. Hong, P. H. Tsai, and C. H. Hsu. 2016. Dynamic module deployment in a fog computing platform. In *Proceedings of the APNOMS*. 1–6.

[77] X. Huang, S. Ganapathy, and T. Wolf. 2009. Evaluating algorithms for composable service placement in computer networks. In *Proceedings of the IEEE ICC*. 1–6. DOI: http://dx.doi.org/10.1109/ICC.2009.5199007

[78] K. Intharawijitr, K. Iida, and H. Koga. 2016. Analysis of fog model considering computing and communication latency in 5G cellular networks. In *Proceedings of the IEEE PerCom Workshops*. 1–4. DOI: http://dx.doi.org/10.1109/PERCOMW.2016.7457059

[79] J. Jijin, B. C. Seet, P. H. J. Chong, and H. Jarrah. 2017. Service load balancing in fog-based 5G radio access networks. In *Proceedings of the IEEE PIMRC*. 1–5.

[80] V. Kochar and A. Sarkar. 2016. Real time resource allocation on a dynamic two level symbiotic fog architecture. In *Proceedings of the ISED*. 49–55. DOI: http://dx.doi.org/10.1109/ISED.2016.7977053

[81] G. Lee, W. Saad, and M. Bennis. 2017. An online optimization framework for distributed fog network formation with minimal latency. *CoRR* abs/1710.05239 (2017).

[82] I. Lera, C. Guerrero, and C. Juiz. 2018. Comparing centrality indices for network usage optimization of data placement policies in fog devices. In *Proceedings of the FMEC*. 115–122. DOI: http://dx.doi.org/10.1109/FMEC.2018.8364053

[83] I. Lera, C. Guerrero, and C. Juiz. 2019. Availability-aware service placement policy in fog computing based on graph partitions. *IEEE Internet Things J.* 6, 2 (Apr. 2019), 3641–3651. DOI: http://dx.doi.org/10.1109/JIOT.2018.2889511

[84] K. Li and J. Nabrzyski. 2017. Traffic-aware virtual machine placement in cloudlet mesh with adaptive bandwidth. In *Proceedings of the IEEE CloudCom*. 49–56. DOI: http://dx.doi.org/10.1109/CloudCom.2017.47

[85] M. M. E. Mahmoud, J. J. P. C. Rodrigues, K. Saleem, J. Al-Muhtadi, N. Kumar, and V. Korotaev. 2018. Towards energy-aware fog-enabled cloud of things for healthcare. *Comput. Elect. Eng.* 67 (2018), 58–69. DOI: http://dx.doi.org/10.1016/j.compeleceng.2018.02.047

[86] A. Mebrek, L. Merghem-Boulahia, and M. Esseghir. 2017. Efficient green solution for a balanced energy consumption and delay in the IoT-fog-cloud computing. In *Proceedings of the IEEE NCA*. 1–4.

[87] S. Q. T. Minh, D. T. Nguyen, A. Van Le, H. D. Nguyen, and A. Truong. 2017. Toward service placement on fog computing landscape. In *Proceedings of the NICS*. 291–296.

[88] N. Mohan, P. Zhou, K. Govindaraj, and J. Kangasharju. 2017. Managing data in computational edge clouds. In *Proceedings of the MECOMM*. ACM, New York, NY, 19–24. DOI: http://dx.doi.org/10.1145/3098208.3098212

[89] M. I. Naas, L. Lemarchand, J. Boukhobza, and P. Raipin. 2018. A graph partitioning-based heuristic for runtime IoT data placement strategies in a fog infrastructure. In *Proceedings of the SAC*. ACM, New York, NY, 767–774. DOI: http://dx.doi.org/10.1145/3167132.3167217

[90] M. I. Naas, P. Raipin Parvedy, J. Boukhobza, and L. Lemarchand. 2017. iFogStor: An IoT data placement strategy for fog infrastructure. In *Proceedings of the IEEE ICFEC*. 97–104.

[91] T. Ouyang, Z. Zhou, and X. Chen. 2018. Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing. *IEEE J. Select. Areas Commun.* (2018), 1–1. DOI: http://dx.doi.org/10.1109/JSAC.2018.2869954

[92] H. Shah-Mansouri and V. W. S. Wong. 2017. Hierarchical fog-cloud computing for IoT systems: A computation offloading game. *CoRR* abs/1710.06089 (2017).

[93] A. Singh, N. Auluck, O. Rana, A. Jones, and S. Nepal. 2017. RT-SANE: Real time security aware scheduling on the network edge. In *Proceedings of the UCC*. ACM, New York, NY, 131–140.

[94] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner. 2017. Optimized IoT service placement in the fog. *Serv. Orient. Comput. Applic.* 11, 4 (01 Dec. 2017), 427–443.

[95] O. Skarlat, S. Schulte, M. Borkowski, and P. Leitner. 2016. Resource provisioning for IoT services in the fog. In *Proceedings of the IEEE SOCA*. 32–39.

[96] H. Tan, Z. Han, X. Y. Li, and F. C. M. Lau. 2017. Online job dispatching and scheduling in edge-clouds. In *Proceedings of the IEEE INFOCOM*. 1–9. DOI: http://dx.doi.org/10.1109/INFOCOM.2017.8057116

[97] R. I. Tinini, L. C. M. Reis, D. M. Batista, G. B. Figueiredo, M. Tornatore, and B. Mukherjee. 2017. Optimal placement of virtualized BBU processing in hybrid cloud-fog RAN over TWDM-PON. In *Proceedings of the GLOBECOM*. 1–6.

[98] L. Tong, Y. Li, and W. Gao. 2016. A hierarchical edge cloud architecture for mobile computing. In *Proceedings of the IEEE INFOCOM*. 1–9. DOI: http://dx.doi.org/10.1109/INFOCOM.2016.7524340

[99] S. Venticinque and A. Amato. 2018. A methodology for deployment of IoT application in fog. *J. Amb. Intell. Hum. Comput.* (06 Apr. 2018). DOI: http://dx.doi.org/10.1007/s12652-018-0785-4

[100] S. Wang, M. Zafer, and K. K. Leung. 2017. Online placement of multi-component applications in edge computing environments. *IEEE Access* 5 (2017), 2514–2533. DOI: http://dx.doi.org/10.1109/ACCESS.2017.2665971

[101] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li. 2017. LAVEA: Latency-aware video analytics on edge computing platform. In *Proceedings of the ACM/IEEE SEC*. ACM, New York, NY, Article 15, 13 pages. DOI: http://dx.doi.org/10.1145/3132211.3134459

[102] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue. 2018. On reducing IoT service delay via fog offloading. *IEEE Internet Things J.* 5, 2 (Apr. 2018), 998–1010. DOI: http://dx.doi.org/10.1109/JIOT.2017.2788802

[103] A. Yousefpour, G. Ishigaki, and J. P. Jue. 2017. Fog computing: Towards minimizing delay in the Internet of Things. In *Proceedings of the IEEE EDGE*. 17–24. DOI: http://dx.doi.org/10.1109/IEEE.EDGE.2017.12

[104] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu. 2016. Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. *IEEE Trans. Comput.* 65, 12 (Dec. 2016), 3702–3712.

[105] L. Zhao, J. Liu, Y. Shi, W. Sun, and H. Guo. 2017. Optimal placement of virtual machines in mobile edge computing. In *Proceedings of the GLOBECOM*. 1–6. DOI: http://dx.doi.org/10.1109/GLOCOM.2017.8254084

[106] R. G. Aryal and J. Altmann. 2018. Dynamic application deployment in federations of clouds and edge resources using a multiobjective optimization AI algorithm. In *Proceedings of the FMEC*. 147–154. DOI: http://dx.doi.org/10.1109/FMEC.2018.8364057

[107] M. Chen, W. Li, G. Fortino, Y. Hao, L. Hu, and I. Humar. 2018. A dynamic service-migration mechanism in edge cognitive computing. *CoRR* abs/1808.07198 (2018).

[108] V. De Maio and I. Brandic. 2018. First hop mobile offloading of DAG computations. In *Proceedings of the IEEE/ACM CCGRID*. 83–92. DOI: http://dx.doi.org/10.1109/CCGRID.2018.00023

[109] X.-Q. Pham and E.-N. Huh. 2016. Towards task scheduling in a cloud-fog computing system. In *Proceedings of the APNOMS*. 1–4. DOI: http://dx.doi.org/10.1109/APNOMS.2016.7737240

[110] R. Gargees, B. Morago, R. Pelapur, D. Chemodanov, P. Calyam, Z. Oraibi, Y. Duan, G. Seetharaman, and K. Palaniappan. 2017. Incident-supporting visual cloud computing utilizing software-defined networking. *IEEE Trans. Circ. Syst. Vid. Technol.* 27, 1 (Jan. 2017), 182–197. DOI: http://dx.doi.org/10.1109/TCSVT.2016.2564898

[111] P. Borylo, A. Lason, J. Rzasa, A. Szymanski, and A. Jajszczyk. 2016. Energy-aware fog and cloud interplay supported by wide area software defined networking. In *Proceedings of the IEEE ICC*. 1–7.

[112] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer. 2006. Network-aware operator placement for stream-processing systems. In *Proceedings of the ICDE*. 49–49. DOI: http://dx.doi.org/10.1109/ICDE.2006.105

[113] G. T. Lakshmanan, Y. Li, and R. Strom. 2008. Placement strategies for internet-scale data stream systems. *IEEE Internet Comput.* 12, 6 (Nov. 2008), 50–60.

[114] Swati Agarwal, Shashank Yadav, and Arun Yadav. 2016. An efficient architecture and algorithm for resource provisioning in fog computing. *Int.J.Inf. Eng. Electron. Bus.* 8 (01 2016), 48–61. DOI: http://dx.doi.org/10.5815/ijieeb.2016.01.06

[115] A. Aral and T. Ovatman. 2018. A decentralized replica placement algorithm for edge computing. *IEEE Trans. Netw. Serv. Manag.* 15, 2 (June 2018), 516–529. DOI: http://dx.doi.org/10.1109/TNSM.2017.2788945

[116] D. Breitgand and A. Epstein. 2011. SLA-aware placement of multi-virtual machine elastic services in compute clouds. In *Proceedings of the IFIP/IEEE IM*. 161–168.

[117] V. Cardellini, V. Grassi, F. Lo Presti, and M. Nardelli. 2016. Optimal operator placement for distributed stream processing applications. In *Proceedings of the ACM DEBS*. ACM, New York, NY, 69–80.

[118] D. Ye, M. Wu, S. Tang, and R. Yu. 2016. Scalable fog computing with service offloading in bus networks. In *Proceedings of the IEEE CSCloud*. 247–251.

[119] H. Zhu and C. Huang. 2017. Availability-aware mobile edge application placement in 5G networks. In *Proceedings of the GLOBECOM*. 1–6. DOI: http://dx.doi.org/10.1109/GLOCOM.2017.8254591

[120] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu. 2017. Resource allocation strategy in fog computing based on priced timed petri nets. *IEEE Internet Things J.* 4, 5 (Oct. 2017), 1216–1228. DOI: http://dx.doi.org/10.1109/JIOT.2017.2709814

[121] I. Althamary, C.-W. Huang, P. Lin, S.-R. Yang, and C.-W. Cheng. 2018. Popularity-based cache placement for fog networks. In *Proceedings of the IWCMC*. 800–804. DOI: http://dx.doi.org/10.1109/IWCMC.2018.8450495

[122] K. U. R. Laghari, N. Crespi, B. Molina, and C. E. Palau. 2011. QoE aware service delivery in distributed environment. In *Proceedings of the IEEE AINA*. 837–842. DOI: http://dx.doi.org/10.1109/WAINA.2011.58

[123] Y. Fan, J. Chen, L. Wang, and Z. Cao. 2018. Energy-efficient and latency-aware data placement for geo-distributed cloud data centers. In *Communications and Networking*. Springer International Publishing, Cham, 465–474.

[124] A. Nazari, D. Thiruvady, A. Aleti, and I. Moser. 2016. A mixed integer linear programming model for reliability optimisation in the component deployment problem. *J. Open Res. Softw.* 67, 8 (01 Aug. 2016), 1050–1060. DOI: https://doi.org/10.1057/jors.2015.119

[125] Farah Aït-Salaht, Frédéric Desprez, Adrien Lebre, Charles Prud'homme, and Mohamed Abderrahim. 2019. Service placement in fog computing using constraint programming. In *Proceedings of the IEEE SCC*. 19–27. DOI: http://dx.doi.org/10.1109/SCC.2019.00017

[126] A. Brogi, S. Forti, and A. Ibrahim. 2017. How to best deploy your fog applications, probably. In *Proceedings of the IEEE ICFEC*. 105–114.

[127] M. M. Shurman and M. K. Aljarah. 2017. Collaborative execution of distributed mobile and IoT applications running at the edge. In *Proceedings of the ICECTA*. 1–5. DOI: http://dx.doi.org/10.1109/ICECTA.2017.8252057

[128] S. S. N. Perala, I. Galanis, and I. Anagnostopoulos. 2018. Fog computing and efficient resource management in the era of Internet-of-Video Things (IoVT). In *Proceedings of the IEEE ISCAS*. 1–5. DOI: http://dx.doi.org/10.1109/ISCAS.2018.8351341

[129] D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguade. 2008. Utility-based placement of dynamic web applications with fairness goals. In *Proceedings of the IEEE NOMS*. 9–16. DOI: http://dx.doi.org/10.1109/NOMS.2008.4575111

[130] M. Selimi, D. Vega, F. Freitag, and L. Veiga. 2016. Towards network-aware service placement in community network micro-clouds. In *Euro-Par - Volume 9833*. Springer-Verlag New York, Inc., 376–388. DOI: http://dx.doi.org/10.1007/978-3-319-43659-3_28

[131] Y. Gu, W. Saad, M. Bennis, M. Debbah, and Z. Han. 2015. Matching theory for future wireless networks: Fundamentals and applications. *IEEE Commun. Mag.* 53, 5 (May 2015), 52–59. DOI: http://dx.doi.org/10.1109/MCOM.2015.7105641

[132] M. L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (1st ed.). John Wiley & Sons, Inc., New York, NY.

[133] M. J. Neely. 2010. *Stochastic Network Optimization with Application to Communication and Queueing Systems.* Morgan and Claypool Publishers.

[134] P. A. Abdulla and R. Mayr. 2013. Priced timed petri nets. *Log. Meth. Comput. Sci.* 9, 4 (2013). DOI: http://dx.doi.org/10.2168/LMCS-9(4:10)2013

[135] D. Monderer and L. Shapley. 1996. Potential games. *Games Econ. Behav.* 14, 1 (1996), 124–143. Retrieved from https://EconPapers.repec.org/RePEc:eee:gamebe:v:14:y:1996:i:1:p:124-143.

[136] R. E. Burkard, E. Çela, P. M. Pardalos, and L. S. Pitsoulis. 1998. The quadratic assignment problem. In *Handbook of Combinatorial Optimization.* Springer.

[137] S. Boyd and L. Vandenberghe. 2004. *Convex Optimization.* Cambridge University Press, New York, NY.

[138] S. H. Bokhari. 1981. On the mapping problem. *IEEE Trans. Comput.* C-30, 3 (Mar. 1981), 207–214. DOI: http://dx.doi.org/10.1109/TC.1981.1675756

[139] R. Eidenbenz and T. Locher. 2016. Task allocation for distributed stream processing. *CoRR* abs/1601.06060 (2016).

[140] Y.-K. Kwok and I. Ahmad. 1999. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Comput. Surv.* 31, 4 (Dec. 1999), 406–471. DOI: http://dx.doi.org/10.1145/344588.344618

[141] J. H. Holland. 1992. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* The MIT Press, Cambridge, MA.

[142] M. Dorigo. 1992. *Optimization, Learning and Natural Algorithms.* Ph.D. Dissertation. Politecnico di Milano.

[143] J. Kennedy and R. C. Eberhart. 1995. Particle swarm optimization. In *Proceedings of the IEEE IJCNN.* 1942–1948.

[144] F. Glover. 1986. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* 13, 5 (May 1986), 533–549. DOI: http://dx.doi.org/10.1016/0305-0548(86)90048-1

[145] D. Whitley. 1994. A genetic algorithm tutorial. *Stat. Comput.* 4, 2 (01 June 1994), 65–85. DOI: http://dx.doi.org/10.1007/BF00175354

[146] G. C. Jana and S. Banerjee. 2017. Enhancement of QoS for fog computing model aspect of robust resource management. In *Proceedings of the ICICICT.* 1462–1466. DOI: http://dx.doi.org/10.1109/ICICICT1.2017.8342785

[147] I. Lera, C. Guerrero, and C. Juiz. 2019. YAFS: A simulator for IoT scenarios in fog computing. *CoRR* abs/1902.01091 (2019).

[148] W. Tärneberg, A. Mehta, E. Wadbro, J. Tordsson, J. Eker, M. Kihl, and E. Elmroth. 2017. Dynamic application placement in the mobile cloud network. *Fut. Gen. Comput. Syst.* 70 (2017), 163–177.

[149] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, and A. Wolman. 2010. Volley: Automated data placement for geo-distributed cloud services. In *Proceedings of the USENIX NSDI.* 17–32.

[150] R. Mayer, H. Gupta, E. Saurez, and U. Ramachandran. 2017. FogStore: Toward a distributed data store for fog computing. *CoRR* abs/1709.07558 (2017).

[151] N. K. Giang, M. Blackstock, R. Lea, and V. C. M. Leung. 2015. Developing IoT applications in the fog: A distributed dataflow approach. In *Proceedings of the IOT.* 155–162.

[152] R.-N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya. 2011. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.* 41, 1 (Jan. 2011), 23–50.

[153] W. L. Dunn and J. K. Shultis. 2011. *Exploring Monte Carlo Methods.* Elsevier Science & Technology.

[154] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne. 2015. FIT IoT-LAB: A large-scale open experimental IoT testbed. In *Proceedings of the IEEE WF-IoT.* 459–464.

[155] D. Balouek, A. Carpen-Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lebre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Pérez, F. Quesnel, C. Rohr, and L. Sarzyniec. 2013. Adding virtualization capabilities to the Grid'5000 testbed. In *Proceedings of the CLOSER.* Communications in Computer and Information Science, Vol. 367. Springer International Publishing, 3–20. DOI: http://dx.doi.org/10.1007/978-3-319-04519-1_1

[156] A. Shrivastwa, S. Sarat, K. Jackson, C. Bunch, E. Sigler, and T. Campbell. 2016. *OpenStack: Building a Cloud Environment.* Packt Publishing.

[157] H. Gedawy, S. Tariq, A. Mtibaa, and K. Harras. 2016. Cumulus: A distributed and flexible computing testbed for edge cloud computational offloading. In *Proceedings of the CIoT.* 1–6.

[158] 2014. Fed4fire project. (2014). Retrieved from http://www.fed4fire.eu/.

[159] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. 2003. PlanetLab: An overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.* 33, 3 (2003), 3–12. DOI: 10.1145/956993.956995

[160] F. Faticanti, F. De Pellegrini, D. Siracusa, D. Santoro, and S. Cretti. 2018. Cutting throughput on the edge: App-aware placement in fog computing. *CoRR* abs/1810.04442 (2018).

[161] V. B. Souza, X. Masip-Bruin, E. López, J. Garcia, G. J. Ren, A. Jukan, and A. Juan Ferrer. 2018. Towards a proper service placement in combined Fog-to-Cloud (F2C) architectures. *Fut. Gen. Comput. Syst.* 87 (2018), 1–15. DOI: http://dx.doi.org/10.1016/j.future.2018.04.042

[162] H. Gupta, S. Brata Nath, S. Chakraborty, and S. K. Ghosh. 2016. SDFog: A software defined computing architecture for QoS aware service orchestration over edge devices. *CoRR* abs/1609.01190 (2016).

[163] K. Hwang and M. Chen. 2017. *Big-Data Analytics for Cloud, IoT and Cognitive Computing* (1st ed.). Wiley Publishing.

[164] LLC Gurobi Optimization. 2018. Gurobi Optimizer Reference Manual. (2018). Retrieved from http://www.gurobi.com.

[165] G. Chiandussi, M. Codegone, S. Ferrero, and F. E. Varesio. 2012. Comparison of multi-objective optimization methodologies for engineering applications. *Comput. Math. Applic.* 63, 5 (2012), 912–942. DOI: http://dx.doi.org/10.1016/j.camwa.2011.11.057

[166] J. Hao, Z. Jin-hua, and C. liang jun. 2019. Multi-objective particle swarm optimization algorithm based on enhanced ?-dominance. 1–5. DOI: http://dx.doi.org/10.1109/ICEIS.2006.1703200

[167] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolut. Comput.* 6, 2 (Apr. 2002), 182–197. DOI: http://dx.doi.org/10.1109/4235.996017

[168] H. M. Fard, R. Prodan, and T. Fahringer. 2014. Multi-objective list scheduling of workflow applications in distributed computing infrastructures. *J. Parallel Distrib. Comput.* 74, 3 (2014), 2152–2165. DOI: http://dx.doi.org/10.1016/j.jpdc.2013.12.004

Received October 2019; revised February 2020; accepted March 2020