Designing an Efficient Clustering Strategy for Combined Fog-to-Cloud Scenarios

A. Asensio[1], X. Masip-Bruin[1], R.J. Durán[2], I. de Miguel[2], G. Ren[3], S. Daijavad[3], A. Jukan[4]

[1]*Advanced Network Architectures Lab, Universitat Politècnica de Catalunya, CRAAX-UPC, Spain.*
{aasensio, xmasip}@ac.upc.edu
[2]*Universidad de Valladolid, Valladolid, Spain.*
{rduran, ignacio.miguel}@tel.uva.es
[3]*IBM, Almaden Research Center, San Jose, USA.*
{gren, shahrokh}@us.ibm.com
[4]*Technische Universität Carolo-Wilhelmina zu Braunschweig, Germany.*
a.jukan@tu-bs.de

***Abstract*— The evolution of the Internet of Things (IoT) is imposing many distinct challenges, particularly to guarantee both wide and global systems communication, and to ensure an optimal execution of services. To that end, IoT services must make the most out of both cloud and fog computing (turning into combined fog-cloud scenarios), which indeed requires novel and efficient resource management procedures to handle such diversity of resources in a coordinated way. Most of the related works that can be found in the literature focus on resource mapping for service-specific execution in fog-cloud; however, those works assume a control and management architecture already deployed. Interestingly, few works propose algorithms to set that control architecture, necessary to execute the services and effectively implement services and resource mapping. This paper addresses that challenge by solving the problem of optimal clustering of devices located at the edge of the network offering their resources to support fog computing while defining the control and management role of each of them in the architecture, in order to ensure access to management functions in combined fog-cloud scenarios. In particular, we set out the Fog-Cloud Clustering (FCC) problem as an optimization problem, which is based on multi-objective optimization, including realistic, novel and stringent constraints; e.g., to improve the architecture's robustness by means of a device acting as a backup in the cluster. We model the FCC problem as a Mixed Integer Linear Programming (MILP) formulation, for which a lower and an upper bound on the number of required clusters is derived. We also propose a machine learning-based heuristic that provides scalable and near-optimal solutions in realistic scenarios in which, due to the high number of connected devices, solving the MILP formulation is not viable. By means of a simulation study, we demonstrate the effectiveness of the algorithms comparing its results with those of MILP formulation.**

***Index Terms*—Fog-to-Cloud systems, Fog-to-Cloud management, Clustering, Mixed Integer Linear Programming, Machine Learning**

I. Introduction

THE explosion of the number of edge devices along with the unstoppable growth of Machine-to-Machine (M2M) connections exemplify the current evolution of the Internet of Things (IoT). Indeed, Cisco forecasts that, by 2021, the number of M2M connections will reach up to 3.3 billion [1], thus recognizing that future IoT environments will rely on a huge number of interconnected devices located at the edge of the network. To satisfy these connectivity requirements, remarkable efforts towards exploring novel network technologies and architectures are intensively carried out by the global network community. A prime example of those efforts not only include the efforts in 5G (expected to become a reality by 2021 [1]), but also in network *softwarization*, mainly through Software Defined Networking (SDN) and Network Functions Virtualization (NFV) [2].

Moreover, the massive data volume and the huge number of IoT devices deployed at the edge, along with some service-specific requirements (such as mobility and low latency) in IoT scenarios, have revealed the need for a new platform to provide cloud-like services to users at the edge of the network. In fact, despite the maturity of *cloud computing* [3] and its proven efficiency, enabling large scale data storage and processing, cloud computing lacks of some of the required capabilities to efficiently support the IoT environment. Therefore, early work (e.g., [4]) proposed to extend the cloud computing paradigm (mainly, computation, storage and networking services) closer to the edge of the network; where, generally speaking, users are located. This idea is named *fog computing* [4]. However, services required at the edge of the network, impose stringent conditions to support, among others, mobility and/or real-time interactions.

Ideally, considering both cloud and fog computing, computing and data storage services should be able to select any resource or a group of resources from the edge of the network up to the cloud, which has been referred to as cloud-to-thing continuum [5], [6] or resources continuum [7]. Notwithstanding, effective strategies that can guarantee the desired QoS, while taking advantage of the whole stack of resources, are still an open challenge. To this end, significant efforts, most notably in the (former) OpenFog

Consortium [8], currently rebranded as the Industrial Internet Consortium, focus on standardizing the whole architecture [6] to manage fog resources (usually located at the edge of the network and close to the users) and cloud resources (usually hosted in datacenters and located far from the users); as well as in the initiative driven by the European H2020 mF2C project [9], focused on designing and implementing the hierarchical Fog-to-Cloud (F2C) management architecture proposed in [10]. A key challenge in such combined fog-cloud scenarios is the design of strategies intended to map services into resources while guaranteeing QoS. Thus, identifying sets of resources and their relationships, optimizing resource utilization and avoiding the potential degradation motivated by changes in the physical topology, are engineering problems of significant interest. In particular, an efficient management of the entire set of fog-cloud resources will require a proper clustering strategy that fulfills the constraints of that specific problem. Thus, in this paper, the Fog-Cloud Clustering (FCC) problem is defined as the decision about how to group fog resources (i.e., resources from edge devices, located at the edge of the network) ensuring access to management functions from fog computing to cloud computing. Indeed, fog-cloud constraints jointly with the large number of devices offering their resources to the system and the parameters that need to be considered in realistic scenarios, result in a non-trivial problem to solve by fog-cloud operators that need to set the architecture to facilitate services deployment.

In particular, since we face an optimization problem, we formally model the FCC problem as a Mixed Integer Linear Programming (MILP) formulation and propose a lower and a higher bound of the number of required clusters. Furthermore, since the proposed mathematical model scalability is compromised in scenarios involving large number of devices, we propose a machine learning heuristic based on the well-known *k-means* clustering method [11], complemented with a number of procedures to ensure the fulfillment of all the constraints of the FCC problem. The proposed MILP and the heuristic are both validated and analyzed, and their performance is compared in terms of the quality of the solutions and scalability. In short, the main contribution of this paper is threefold: *i*) a formal description of the FCC problem, modeled by an MILP formulation, to optimally solve the problem of identifying sets of resources and their role to set the F2C architecture presented in [10], *ii*) a lower and a higher bound of the number of clusters in the FCC problem, and *iii*) a scalable machine learning-based algorithm providing near-optimal solutions to solve the problem in realistic scenarios.

The remainder of this paper is organized as follows. Section II introduces the background and the FCC problem, and also describes related work. Then, Section III formally states and models the FCC problem as an MILP formulation, and analytically derives lower and upper bounds on the number of clusters and studies the impact of the ratio between fog and cloud costs on that number. Section IV presents the machine learning-based algorithm to solve the FCC problem in a scalable fashion. Section V presents numerical results. Finally, Section VI concludes the paper.

II. Background of the Fog-Cloud Clustering Problem and Related Work

In this section, the Fog-to-Cloud (F2C) management architecture is described and an introduction to the fog-cloud clustering problem is provided. Moreover, related work on the topic is also reviewed.

A. F2C Management Architecture and Introduction to the FCC Problem

The need to define a suitable management architecture in F2C systems has been already identified in [10]. Since we consider that architecture as the baseline to represent the F2C resources management problem, we will start by briefly describing some architectural aspects that we identify as mandatory in order to make reading as smooth as possible. The first relevant idea is that the F2C architecture is hierarchical (Fig. 1), and, unlike other resource-combined approaches, the proposed architecture introduced in [10] is not static. To set the F2C architecture, different layers are defined, grouping the whole set of resources and enabling services to decide, on the fly, the set of resources best suiting services demands, be it at fog, cloud, or both of them. Certainly, resource management becomes a key problem to efficiently manage resources and optimize service execution.

As described in Section I, a first attempt to implement the management architecture introduced in [10], has been done in the framework of the European H2020 mF2C project [9]. Indeed, [9] defines an entity referred to as *agent*, which is deployed in all elements participating in the F2C system, and which is responsible for enabling devices to execute the management functions required in the architecture. To perform control and management operations, control data is sent between agents at different layers through a module named *platform manager*. In this paper, we assume that an implementation of the aforementioned agent is deployed in edge devices and cloud, thus enabling the required control and management functions to set the architecture. Moreover, in that architecture, sets of devices are clustered under the control of a single agent, which is referred to as *leader*, and which receives aggregated information from the elements in the layer below. Additionally, another device of the cluster may take the role of *backup leader* in order to enhance the robustness of the system. The role of devices that are neither leaders nor backup leaders is referred to as *regular*.

Obviously, these roles (necessary to set the architecture) impose additional constraints to the clustering problem in fog-cloud. Some are: *i*) in the south bound, leaders must ensure communication (coverage range) with each device in its cluster; *ii*) in the north bound, leaders must guarantee its access to cloud; and *iii*) backup leader must ensure connectivity to all devices in the cluster. It is worth noting that, although most of the devices may have access to the Internet, there may be some devices having none or limited access, thus being not appropriate to act as leaders; e.g., a tablet or a smart watch with WiFi and/or Bluetooth but without cellular option. Interestingly, these devices can benefit from the shared resources in fog-cloud, joining to the system as regular. Nevertheless, clustering methods as well as leader and backup leader selection strategies are still work in progress.

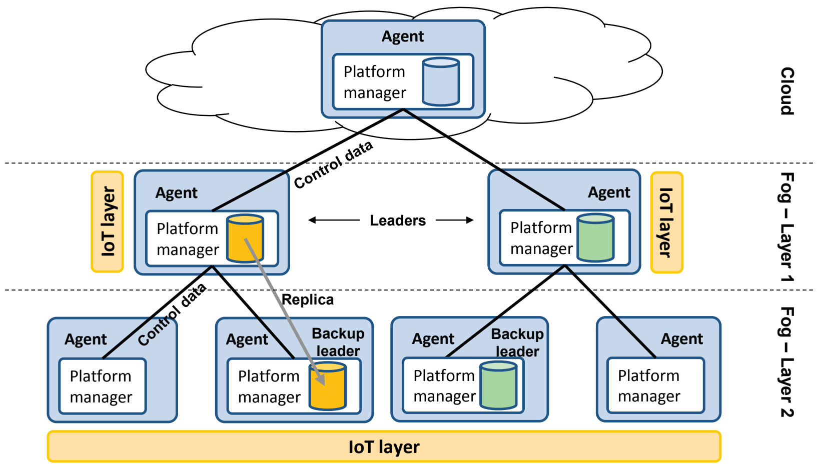

Fig. 1. Example of 3-layers architecture in F2C.

Fig. 1 reproduces the F2C layered architecture and control data flow proposed in [9], considering a particular example presenting three layers: Cloud (top), Fog layer 1 (middle layer hosting leader functionalities) and Fog layer 2 (bottom). It is worth highlighting that both Fog layer 1 and 2, interface the IoT layer. The IoT layer may consist, for instance, of Raspberry Pi boards running an implementation of the agent and having a set of sensors connected to them. Moreover, to illustrate the backup leader role, databases required to manage the architecture have been represented at each layer as well as the elements acting as backup leader, which host replicas from the corresponding leaders' databases.

In view of the described architecture, it is clear that devices offering their resources to support fog computing (i.e., devices at fog layers) need to be grouped into clusters, and that a role (leader, backup leader or regular) needs to be assigned to each of them to set the architecture. Such grouping or clustering strategy must ensure that: *i*) distance, and thus latency, to management functions is minimized, *ii)* transmission power is minimized not only to reduce energy consumption but also to avoid interference, *iii*) associations to groups are not frequently changed, and one device per group is selected to provide management functions (i.e., is selected to become the leader of that cluster), and *iv*) reliability is guaranteed in the case that the cluster leader becomes unavailable (by means of a backup leader, which could be promoted to leader if required).

To meet these high level requirements, the FCC problem needs to focus on optimally mapping devices from a physical topology into a logical one in the fog-cloud architecture. This, in turn, results in: *i*) identifying the role of each device in the architecture aimed at efficiently mapping it into the logical topology, i.e., an edge device may be leader, backup leader or regular, and *ii*) optimizing the logical topology to meet the fog-cloud operators' requirements to manage the networked system. Such optimization can be done through the minimization of an objective function with constraints related to latency, transmission power, and also to the stability of the connectivity of the leaders, as well as a number of other constraints imposed by the location and features of the different devices composing the fog-cloud network.

In order to clarify these issues, let us consider the following example. Let us assume a small size geographic area, with a radius of few tens to hundreds of meters, with a set of devices interfacing the IoT layer (sensors and actuators), along with cloud facilities located in a comparably more distant area (typically from hundreds to thousands of kilometers). This is illustrated in Fig. 2, which shows a fog-cloud network representing the physical and logical views for a set of (edge) devices and cloud resources, including cloud, fog and IoT layers. First, Fig. 2a represents the distribution of the set of devices interfacing the IoT layer. For illustrative purposes, let us depict each device by a node (a vertex in the topological graph), numbered from 1 to 9, in the physical topology representation, and the objective is to group them into clusters to participate in the fog-cloud system. The coverage area for each device is depicted by a dashed circle; except for node 8, where, for completeness, two different coverage levels are represented by two concentric circles depicted with solid lines. It is worth noting that for the red circle (low coverage), lower transmission power is required (details on discrete power levels and power adaptive control are shown in [12] and [13]). Moreover, in this example, we assume that each edge device (also referred to as node) can manage three nodes at most.

Fig. 2b represents three clusters (colored in green, blue and yellow in the figure) resulting after applying the two following cluster rules: *i*) a cluster consists of a set of edge devices, one of which provides management functions (the *leader*), and *ii*) the leader is directly managed by the cloud. Therefore, only one device in each cluster is directly connected to cloud (nodes {3, 4, 8}, i.e., the leaders), thus taking advantage of reduced latency when managing the underlying nodes {1, 2, 5, 7, 6, 9}, compared to a cloud computing approach where all nodes would be connected to cloud. Moreover, as it is represented in Fig. 2b, the low coverage level of node 8 would be enough to ensure connectivity to all elements in the cluster (thus saving transmission power). In addition, an edge device has been chosen in each cluster to guarantee service continuity in the event that the leader node in the cluster fails or suddenly disappears, i.e., it acts as a backup leader [9]. In this example, nodes 2 and 6 have been selected as backup leaders, since they ensure connectivity to all elements in their cluster and to cloud if required. Finally, Fig. 2c, represents the logical topology inferred from the clustering solution shown in Fig. 2b.

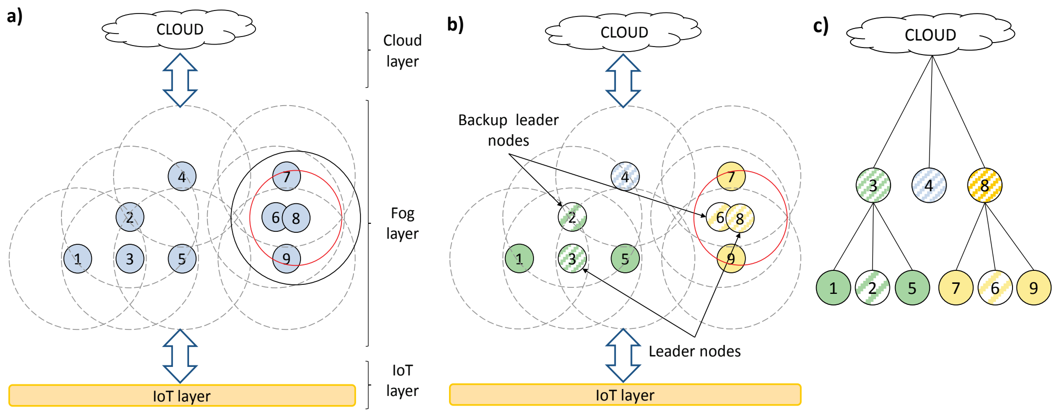

Fig. 2. Physical topology and coverage (a), clusters (b), and logical topology (c).

B. *Related Work*

Although the fog-cloud architecture presented in [9] proposes to group edge devices into clusters –and also selecting a role for each device (leader, backup leader or regular)–, to the best of our knowledge, there is no contribution formally describing this clustering problem in the particular fog-cloud architecture, which is the main contribution of this paper.

Nevertheless, clustering problems are very well known in a wide range of fields, and in particular in fields related to edge scenarios, but fulfilling different constraints than those of FCC problem. For instance, a hierarchical approach is adopted in Mobile Ad-hoc Networks (MANET) by dividing the network into clusters with certain nodes in each cluster being chosen as *clusterheads* (nodes providing connectivity among clusters). Such a problem is formally stated as an Integer Linear Programming (ILP) problem in [14], which minimizes a multi-objective function that includes the cost of connecting nodes as well as a weight associated to the clusterheads. However, ILP formulations do not scale properly when addressing large scale networks, so different algorithms have also been proposed –a survey of those methods for wireless sensor networks and for MANETs is provided, respectively, in [15] [16]. For instance, authors in [17] propose a distributed clustering algorithm for ad-hoc networks, relying on a weighted algorithm to identify clusters and clusterheads, determining the topology and its stability. In addition, authors in [18] perform a stochastic geometry analysis to formulate the optimal number of fog nodes in a cloud-fog-thing IoT-based scenario, enhancing signal-to-interference-plus-noise ratio. In the recent study published in [19], an energy-aware and delay constrained algorithm is proposed. In that work, authors assume that fog nodes can process requests locally in a Fog-IoT network, without requiring the cloud. Moreover, although the scenario described is based on a three-tier Fog-IoT network, their proposal focuses on the two bottom layers; that is, the Fog and the IoT layers.

A different example of the clustering problem in distributed computing paradigms can be found in [20]. That work proposes a framework in a 4-layers environment (i.e., from down to top: IoT, edge, fog and cloud), which, by analyzing spatio-temporal mobility data and other contextual information, and by means of a machine learning algorithm, the location of IoT and edge devices that are on the move can be predicted. The study highlights the importance of a hierarchical infrastructure for critical-time applications. Interestingly, to model users' movement patterns, the authors propose a User Movement Graph, which is a multi-layer graph. One of the layers of this graph stores frequent movement paths for the user; that information is utilized to perform spatio-temporal trajectory clustering aimed at identifying the paths frequently followed by a user. Another approach to group resources can be found in [21], where the authors propose a monitoring tool (FogMon) and set the monitoring network based on latency between follower nodes (similar to the idea of regular nodes) and leaders. However, differently from our manuscript, the role of each element in the architecture is already defined; whereas in our proposal, clustering is utilized not only to group resources but to identify the role of each element in the cluster; thus allowing the architecture to be dynamic.

In addition, several papers can be found in the recent literature focusing on providing an efficient use of resources to satisfy applications' needs in fog-cloud scenarios. Aligned with this topic, in [22], the authors assume a fog-cloud architecture and focus on the decision of offloading data received from end-user tasks, to be processed in different fog nodes or in cloud and taking into account delay constraints. The authors in [23] propose an algorithmic approach, referred to as Module Mapping Algorithm, to map application modules into fog-cloud resources. Focusing on time-critical IoT applications, CPU, RAM and bandwidth are the requested attributes considered by each applications' module. Modules are mapped into fog and cloud resources satisfying modules' requirements. A different approach to map resources to users' needs is presented in [24]. In that paper, based on a game theory approach, each user tries to maximize its Quality of Experience (QoE) by allocating fog resources efficiently. Similarly, as in the previously reviewed papers, in [24] authors pay special attention to delay-sensitive IoT applications. Interestingly, in addition to time-sensitive applications, authors in [25] also consider a more stringent scenario where user (application) requirements can change dynamically and resource allocation needs to consider additional constraints, such as a deadline. Moreover, a different approach for application placement in fog-cloud can be found in [26]. In [26], the proposed ILP model focuses on enhancing provider's profit while fulfilling applications' required QoS. In addition, a pricing model for fog instances and a compensation method satisfying both user's and provider's interest are presented. However, in these manuscripts, the focus is set into the services, while assuming an underlying control and management architecture that allows implementing their proposals.

As described above, a large number of papers found in the literature do not focus on setting the architecture, but on the service allocation. Nevertheless, aligned with proposals in the Industrial Internet Consortium or in the mF2C project, some recent manuscripts focusing on the architecture can also be found in the literature. The works in [27] and [28] propose a framework, named to as FogFrame, which defines communication mechanisms to instantiate and maintain service execution in the fog landscape, and following the notation of OpenFog, assuming: *i*) *fog cells* as the representation of virtual resources in IoT devices, *ii*) *fog nodes* as fog cells with additional functionalities and acting as access points for other fog cells, *iii*) *fog colonies,* formed by sensors and actuators connected to fog cells, which in turn are connected to a fog node, and *iv*) *fog controllers*, acting as a central component in charge of discovering new resources. However, the roles of the different elements are already defined; i.e., are known in advance, or if, according to the corresponding policy applied when a fog node joins the system, they need to be assigned; the decision of defining a new fog node requesting to join the fog landscape as the head of a new colony or as a fog node in the lower tier depends on the fact that the new fog node is or it is not on the range of a previously joined fog node. Differently, in our proposal, the role of each element is assigned at the clustering definition time and aimed at optimizing the whole architecture, without knowing a priori, the clustering elements neither their roles. Moreover, it is worth highlighting that in our proposal, two devices within their coverage range may act as leaders of different clusters.

Interestingly, the authors in [29] propose a strategy based on centrality indices to select fog colony controllers. However, cloud is not considered and the metric utilized is based on the distance: *i*) between the controller device and its subordinated devices, and *ii*) between the colonies' controllers, while parameters such as power or mobility are not considered. Moreover, no backup technique is studied. In the work in [30], it is highlighted the relevance of hierarchical and distributed architectures in novel and future IoT-based scenarios, since they improve scalability without compromising stability [30]. Although they consider a hierarchical and decentralized architecture, differently from our proposal their proposal is focused on solving the specific problem of Data Stream Processing, which is different from the FCC problem tackled in this paper, thus resulting in different problem definition, constraints and formulation. Interestingly, similarly as in our proposal, machine-learning is considered to set the architecture. Specifically, authors propose a reinforcement learning –based approach. The resource management framework proposed in [31] is based on Particle Swarm Optimization to simultaneously optimize performance parameters such as energy and latency, among others, and oriented to resource management for smart homes IoT devices. Again, this is different from our proposal, which proposes a MILP formulation aimed at identifying, from the entire set of devices, which ones act as leaders, and which ones are under its control, forming the cluster aimed at setting the control and management fog-cloud architecture proposed in [10] while satisfying the specific constraints imposed.

In addition, similarly as in our work in this paper, in [32] the authors assume the F2C architecture from [10]. In [32], the concept of *Area* refers to a group of fogs that are located at different layers and that facilitate resource coordination among layers. However, areas' definition and configuration is out of the scope of that manuscript and it is an open research issue. In fact, the work presented in this manuscript is aligned with those ideas of grouping resources and map them into different layers to facilitate coordination among them. The lack of algorithms or studies covering in a complete manner algorithmic aspects in three-tier IoT scenarios considering IoT, fog and cloud layers is remarked in the recent survey in [33], in the related work review. Moreover, among the different research areas and opportunities described in the state-of-the-art and future directions sections in the survey in [34], we highlight the following ideas because of their direct relationship with the problem that we solve in this paper:

i) *Management*: in a decentralized management architecture, different devices need to implement similar processes to handle management issues. This idea is aligned with the agent definition in [9] and we assume it in this paper as described in Section II A.

ii) *Fault tolerance*: fault tolerance issues in fog computing need to be investigated; interestingly, in our proposal we study two different approaches to set the architecture, with and without backup.

iii) *Standard architecture*: with the contribution in this manuscript, the authors believe that their proposal can be considered by fog-cloud operators to set the management architecture that facilitates the deployment of fog-cloud services and thus contributes to facilitate the opportunity "to explore and gain deeper insight into each layer with proper validation" [34]; specifically, our proposal solves the problem of implementing the architecture proposed in [10] to control and manage fog-cloud resources.

To summarize the reviewed literature, clustering problems have been widely studied in different fields; however, in fog-cloud scenarios, a large number of works focus on services deployment considering the problem of assigning resources to services/applications but assuming a generic fog-cloud control and management architecture. Indeed, these works could take advantage of our proposal, since they could benefit from deploying their proposed algorithms while delegating control and management issues to the agents to implement the solutions obtained. Moreover, proposals facing architectural aspects, focus on service-specific architectures (e.g., the works in [30] and [31]) and/or consider that roles of the devices in the different layers are known a priori or are assigned according to a less complex criteria than the one considered in our proposal; e.g., coverage range. Aligned with the trends described in the reviewed literature, our work proposes utilizing an optimization technique, i.e., Mixed Integer Linear Programming, to minimize a multi-objective function considering relevant parameters such as latency, energy and mobility; as well as a machine-learning method; i.e., by means of a k-means based algorithm. Additionally, to add robustness to the architecture, the role of the backup leader is also considered, which impose additional constraints in fog-cloud clustering.

Finally, our approach first formalizes the FCC problem by means of an MILP formulation, thus explicitly stating the objectives and the specific constraints of this particular problem, and then proposes a scalable algorithm providing near-optimal solutions. For the latter, we will rely on an unsupervised machine learning approach to solving the FCC problems. A survey of clustering algorithms can be found in [35]. Although there is a subset of methods targeted to solve constrained clustering problems (i.e., problems which incorporate domain knowledge in the form of constraints) [36], they cannot be directly applied here, as the type of constraints considered by those methods are usually related to imposing that two elements must be placed into the same cluster, that two elements must be placed into different clusters, or related to imposing limits on the number of elements per cluster. However, in general, in the FCC problem, that type of constraints cannot be explicitly posed *a priori*, since the problem does not only involve grouping nodes in a cluster but also selecting the cluster leader (which also has some restrictions). For instance, two distant nodes can belong to the same cluster even if they cannot communicate (due to their separation) as long as they can communicate with the cluster leader (e.g., nodes 1 and 5 in Fig. 2). Thus, it cannot be enforced *a priori* the need for two nodes to be in the same or in different clusters. Therefore, we use the *k*-means algorithm, which is a time efficient clustering algorithm based on partition [35], but we enrich it with mechanisms to enforce the constraints imposed by the FCC problem.

III. FCC Problem Formulation

In this section, the Fog-Cloud Clustering (FCC) problem is formally stated and a MILP formulation is proposed to model it. Then, upper and lower bounds in the number of clusters are obtained, and finally, the impact of the ratio between fog and cloud costs on the number of clusters is studied.

A. Problem Statement

The FCC problem can be formally stated as follows:

Given:

- A network topology represented by the graph $G(V, E)$; being V the set of nodes (representing devices offering resources at fog layer) and E the set of edges. Each edge e is characterized by its distance.
- The tuple $\{coordinates_v, \gamma_v, d_{vc}, K_v, M_v\}$ for each $v \in V$; where $coordinates_v$ defines the node's position, γ_v represents a binary parameter indicating if v can connect to cloud or not, d_{vc} represents the distance to cloud facilities, K_v represents v's capacity (i.e., the maximum number of underlying devices that v can manage; limited to guarantee certain QoS), and M_v represents a binary parameter indicating if v is considered as static or mobile.
- Parameter p_{ij}^{TX} representing the required transmission power to connect nodes $i, j \in V$.
- Binary parameter ρ representing whether backup is required or not.

Output: The definition of each cluster; that is, clusters' elements and their role.

Objective: First objective is to minimize average distance (or latency) to management functions. In addition, secondary objectives are to minimize required transmission power and the number of mobile nodes acting as leaders.

B. Definitions and Mathematical Model

The aim of the FCC problem is to determine how to group edge devices into clusters, allocating one leader in each cluster (with connection to cloud and enough capacity to support the rest of the nodes in the cluster), and also a backup leader (if required), intended at minimizing latencies and transmission power, prioritizing the selection of static nodes rather than mobile nodes to act as leaders, and fulfilling reachability constraints.

Thus, in order to model the FCC problem, the following parameters have been defined:

V	Set of nodes representing devices offering resources at fog layer.
d_{ic}	Parameter representing distance from $i \in V$ to cloud.
d_{ij}	Parameter representing distance between $i, j \in V$.
λ_{ij}	Binary parameter with value 1 if nodes i and j ($j \neq i$) have potential connectivity (are reachable if operating at full transmission power); 0 otherwise.
γ_i	Binary parameter with value 1 if node i can connect to cloud; 0 otherwise.
K_i	Constant value representing the maximum number of nodes that can be connected to node i; i.e., capacity.
p_{ij}^{TX}	Transmission power required for a connection from node i to node j (which depends of the location of the nodes)
M_i	Binary parameter, representing if node i is considered as static (value 0) or mobile (value 1).
ρ	Binary parameter to select if backup leaders need to be considered (value 1) or not (value 0).
c_{ij}^{F}	Parameter representing the cost (or the latency) of connecting nodes i and j; proportional to d_{ij}
c_{i}^{C}	Parameter representing the cost (or the latency) of connecting node i to cloud; proportional to d_{ic}.
$bigM$	Large constant value.
α	Weight for the connectivity cost at fog level.
β	Weight for the connectivity cost to cloud.
φ	Weight for the transmission power term.
σ	Weight for the mobility term.

In addition, the following variables have been defined, which will provide the solution to the FCC problem:

x_{ij}	Binary. Value 1 if node *i* is managed by node *j* (referred to as *leader*); 0 otherwise.
y_i	Binary. Value 1 if node *i* is managed by cloud; 0 otherwise.
z_{ij}	Binary. Value 1 if node *i* is the backup leader for node *j*; 0 otherwise.
v_i	Auxiliary continuous variable, accounting transmission power required for node *i*.
n_i	Auxiliary binary variable. Value 1 if node *i* is leader; 0 otherwise.

Then, the MILP model for the FCC problem is defined as:

$$Minimize\ FCC\_Cost \tag{1}$$

where the FCC cost, or objective value, is defined by:

$$FCC\_Cost = \frac{1}{|V|}\left[\alpha \sum_{i\in V}\sum_{j\in V, j\neq i} c_{ij}^{F} x_{ij} + \beta \sum_{i\in V} c_i^C y_i + \varphi \sum_{i\in V} v_i + \sigma \sum_{i\in V} M_i n_i\right] \tag{2}$$

and subject to:

$$\sum_{j\in V} x_{ij} + y_i = 1 \quad \forall i \in V \tag{3}$$

$$x_{ij} \leq \lambda_{ij} \quad \forall i,j \in V \tag{4}$$

$$\sum_{i\in V, i\neq j} x_{ij} \leq K_j \quad \forall j \in V \tag{5}$$

$$n_j \geq \frac{\sum_{i\in V, i\neq j} x_{ij}}{bigM} \quad \forall j \in V \tag{6}$$

$$n_j \leq \sum_{i\in V, i\neq j} x_{ij} \quad \forall j \in V \tag{7}$$

$$y_i \geq n_i \quad \forall i \in V \tag{8}$$

$$y_i \leq \gamma_i \quad \forall i \in V \tag{9}$$

$$v_i \geq p_{ij}^{TX} \cdot \left(x_{ij} + x_{ji}\right) \quad \forall i,j\ (j \neq i) \in V \tag{10}$$

$$\sum_{i\in V} z_{ij} = \rho \cdot n_j \quad \forall j \in V \tag{11}$$

$$\sum_{j\in V} z_{ij} \leq \rho \cdot (1 - n_i) \quad \forall i \in V \tag{12}$$

$$z_{ij} \leq \rho \cdot \gamma_i \cdot x_{ij} \quad \forall i,j \in V \tag{13}$$

$$\left(\sum_{i\in V, i\neq j, i\neq k} \lambda_{ik} \cdot x_{ij}\right) + 1 \geq \sum_{i\in V, i\neq j} x_{ij} - bigM \cdot \left(1 - \rho \cdot z_{kj}\right) \quad \forall j,k \in V \tag{14}$$

$$\sum_{i\in V, i\neq j, i\neq k} x_{ij} \leq K_k + bigM \cdot \left(1 - \rho \cdot z_{kj}\right) \quad \forall j,k \in V \tag{15}$$

The objective function (1)-(2) jointly minimizes distance (or latency) to management functions (main objective, first and second terms), and the transmission power required, as well as the number of mobile devices acting as leaders (secondary objectives, third and fourth terms). Similarly, as in [17], different weights can be considered for each term in the objective function by tuning α, β, φ, and σ. In this way, this is a generic formulation that can be tuned to match fog-cloud operators' requirements. For instance, setting σ to a high value enforces static nodes to act as leaders rather than mobile nodes. On the other hand, since connectivity costs at fog and cloud layers (c_{ij}^F and c_i^C) are related to distances between nodes and between nodes and the cloud, respectively, setting $\alpha = \beta = 1$ and $\varphi = \sigma = 0$ minimizes the distance (and thus latency) to management functions without any additional considerations. Later, in Section III.D, we analyze the impact of the connectivity costs at fog and cloud layers and their associated weights (α and β). It is worth noting that, although c_{ij}^F and c_i^C could be merged into one parameter we define them separately for the sake of clarity (similar comment applies to parameters d_{ic} and d_{ij}). Since the multi-objective cost function considers a variety of terms (related to distance, transmission power and mobility), the weighting parameters (α, β, φ, and σ) are set to have units as the inverse of the units of the term they multiply and, as a result, the FCC cost is dimensionless. It is worth noting that the FCC cost in equation (2) should just be interpreted as an optimization metric, but not as an economic cost. As previously mentioned, the MILP formulation aims to set a generic framework to solve the FCC problem, but not a techno-economic model. That type of model is out of the scope of this work.

Constraint (3) ensures that each node is managed either by one and only one leader or by the cloud. Constraints (4)-(8) deal with leader-based clusters constraints. Specifically, equation (4) guarantees that only nodes within the coverage range are grouped in the cluster; whereas equation (5) guarantees that the capacity of leader nodes is not exceeded. Constraints (6)-(8) account for leader nodes and ensure that if node *i* acts as leader, it is managed by the cloud. Constraint (9) guarantees that nodes managed by the cloud can, in fact, access cloud. Constraint (10) accounts for the transmission power required to guarantee connectivity. Finally, constraints (11)-(15) deal with backup settings when management service guarantees are required. Specifically, constraint (11) ensures that a backup leader is selected to support each leader, while constraint (12) ensures that leaders are not selected as backup leaders. Constraint (13) guarantees that the selected backup leader and leader belong to the same cluster and that the backup leader can access to cloud; whereas constraints (14) and (15) guarantee intra-clustering connectivity (i.e., the backup leader has potential connectivity to all nodes in the cluster) and that the backup leader capacity is not exceeded, respectively.

It is worth recalling that, although in our proposal the selection of mobile devices acting as leaders tends to be minimized, it is clear that mobile devices may be selected as leaders in benefit of reducing distance to management functions. However, operators may impose stringent constraints with respect to the selection of mobile devices acting as leaders; e.g., in the case that the cost of selecting a new leader becomes too high or in the case that the mobility of leaders impacts negatively on the performance. Although some specific easy rules may be considered to minimize the effects of topology changes (e.g., fog-cloud operators can take advantage of devices' historical data stored at cloud, and identify whether for a given period of time a device can be considered as a static device or not), the proposed mathematical model can be adapted to avoid selecting mobile nodes as leaders. In this case, the last term in the objective function (related to mobility), can be removed from the objective function and added as a new constraint. For completeness, these changes are formally described in equations (16) and (17). Equation (16) represents the new FCC cost to be considered; whereas the new constraint added is shown in equation (17).

$$FCC\_Cost = \frac{1}{|V|}\left[\alpha \sum_{i\in V}\sum_{j\in V, j\neq i} c_{ij}^{F} x_{ij} + \beta \sum_{i\in V} c_{i}^{C} y_i + \varphi \sum_{i\in V} v_i\right] \tag{16}$$

$$M_i \cdot n_i = 0 \quad \forall i \in V \tag{17}$$

Moreover, although it is clear that three layers can be identified (from top to down, each layer mapping: cloud, leaders, rest of the nodes, respectively) the model can be easily extended to a different number of layers by executing it iteratively.

C. *Upper and Lower Bound of the Number of Clusters*

A well-known constraint of MILP formulations is the scalability with the size of the problem, which drives the need for algorithms that can solve the problem in a feasible time (although possibly suboptimally) when facing scenarios with many nodes. Knowing good bounds on some performance parameters is extremely helpful to design and/or evaluate these methods. Therefore, we propose an upper and lower bound of the number of clusters required to solve the FCC problem. Indeed, an upper bound can be easily computed by equation (18), which just assumes that all nodes with connection to cloud become cluster leaders. As we will show later, this is a scenario coming up if connections to cloud have a low cost.

$$num\_clusters\_upper\_bound = \sum_{i\in V} \gamma_i \tag{18}$$

On the other hand, regarding the computation of the lower bound, the key idea is as follows. A node *i* having connectivity to cloud (γ_i=1) is a potential leader, which could serve up to K_i devices (capacity constraint) or up to $\sum_{j\in V}\lambda_{ij}$ devices (i.e., the number of reachable devices from it). Then, potential leaders are sorted in descending order according to the maximum number of devices they can serve (the minimum of K_i and $\sum_{j\in V}\lambda_{ij}$), and by adding those values until the total number of nodes is reached, a lower bound of the number of required clusters is determined. Note that since many constraints are omitted in that procedure, the value obtained is a lower bound. Nevertheless, as we will show later, in many realistic scenarios the lower bound is very tight (in fact, exact). The computation of the lower bound can be done using Algorithm 1 (which follows the key idea previously mentioned together with required subtleties for a right computation).

Algorithm 1: Lower bound in the number of clusters

```
1:  procedure num_clusters_lower_bound($V$, $\lambda_{ij}$, $\gamma_i$, $K_i$)
2:    num_isolated_leaders ← 0
3:    min_number_of_clusters ← 0
4:    for $i$ in 1 to $|V|$ do
5:      num_reachable_devices_from[$i$] ← $\sum_{j\in V}\lambda_{ij}$
6:      if num_reachable_devices_from[$i$] == 0 then
7:        if $\gamma_i$ == 0 then
            # A node is isolated (cannot reach any node) and has no connection to cloud
8:          set problem as UNFEASIBLE
9:          go to end procedure
10:       else
```

```
                # A node is isolated, but it has connection to cloud,
                # so it must constitute a cluster of a single node (itself)
11:             num_isolated_leaders ← num_isolated_leaders + 1
12:             min_number_of_clusters ← min_number_of_clusters + 1
13:             max_served_devices_by_node[i] ← 0
14:           end if
15:         else
16:           if $\gamma_i$ == 0 then
                # A node not connected to cloud cannot serve any node; thus, it cannot be a leader
17:             max_served_devices_by_node[i] ← 0
18:           else
                # A node connected to cloud can serve other nodes (limited by capacity and reachability);
                # thus, it can be a leader
19:             max_served_devices_by_node[i] ← min($K_i$, num_reachable_devices_from[i]) +1
                # The +1 is added in the previous line because a leader also serves to itself
20:         end if
21:       end for
22:       num_V_reduced ← |V| - num_isolated_leaders
23:       num_potential_served_devices ← num_isolated_leaders
24:       sorted_max_served_devices_by_node ← sort(max_served_devices_by_node, order = descending)
25:       for i in 1 to num_V_reduced do
26:         # For the bound, assume that the node which can serve more devices (and is not a leader yet)
            # is selected as the leader of a new cluster,
            # and assume that those devices that it can serve were not served by another leader
          num_potential_served_devices ←num_potential_served_devices +sorted_max_served_devices_by_node[i]
27:         min_number_of_clusters ← min_number_of_clusters + 1
28:         if num_potential_served_devices ≥ |V| then
              # All devices are already leaders or assumed to be served by a leader; so the lower bound is returned
29:           return min_number_of_clusters
30:         end if
31:       end for
32:       set problem as UNFEASIBLE      # Not all nodes can be served
33:       end procedure
```

D. *Influence of Fog and Cloud Costs*

In this subsection, the influence of the weights of cloud and fog costs in the number of clusters created is analyzed, and those numbers are compared with the upper and lower bounds defined in the subsection above.

Focusing on the main objective of the FCC problem (i.e., reducing the average distance to management functions), in view of equation (2), it is clear that, depending on the distance to cloud (d_{ic}) and between edge devices (d_{ij}), which in turn determine the cost of cloud connections (c_i^C) and fog connections (c_{ij}^F), respectively, as well as on the weights α and β, the number of clusters (which is not known in advance) will vary. To evaluate the impact of these parameters, equation (19) shows the average weighted cost of a connection from a node to cloud, $\overline{cloudCost}$, while equation (20) shows the average weighted cost of a (fog) connection between two nodes $\overline{fogCost}$ as:

$$\overline{cloudCost} = \beta \cdot \frac{\sum_{i \in V} c_i^C \cdot \gamma_i}{\sum_{i \in V} \gamma_i} \tag{19}$$

$$\overline{fogCost} = \alpha \cdot \frac{\sum_{i \in V} \sum_{j \in V, j \neq i} c_{ij}^F \cdot \lambda_{ij}}{\sum_{i \in V} \sum_{j \in V, j \neq i} \lambda_{ij}} \tag{20}$$

Then, we define the Cloud/Fog Cost Ratio (CFR) as:

$$CFR = \frac{\overline{cloudCost}}{\overline{fogCost}} \tag{21}$$

Fig. 3 shows the average number of clusters obtained, as a function of the CFR, when using the MILP formulation to solve the FCC problem (therefore, the solutions are the optimal ones). Each point has been obtained by simulating 10 scenarios (with different nodes positions, connectivity, values of K_i and values of α resulting in CFR values ranging from 10^{-4} to 10^{6} for a fixed value of $\beta = 1$, and setting $\varphi = \sigma = 0$) with 20, 40 and 60 nodes randomly distributed in a 100 m × 100 m area, and assuming a distance to cloud of approximately 1000 km (the exact distance depends of the location of each node in the 100 m × 100 m area).

As expected, Fig. 3 shows that if connections to cloud have a higher weighted cost than connections at fog layer (high CFR), the best solutions are those that deploy the lowest number of clusters as possible, close or equal to the lower bound (area identified as *Fog-Cloud* in the figure). In contrast, if connections to cloud are not so costly (low CFR), a very high number of clusters, close or equal to the maximum bound, will be used (area identified as *Cloud* in the figure). In these simulations, the transition between those two areas (depicted in grey in the figure) takes place when the CFR ranges from 0.05 to 10. Therefore,

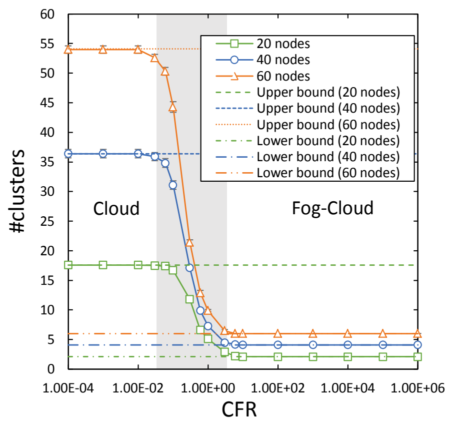

Fig. 3. Number of clusters against Cloud/Fog Cost Ratio, CFR.

if the weighted cost of connections to cloud is at least around 10 times higher than that of connections at fog layer, which is reasonable, the solutions to the FCC problem will use a number of clusters equal (or very close) to the lower bound. Note that by setting $\alpha = \beta = 1$, which leads to finding solutions which minimize the distance (and thus latency) to management functions without any additional considerations, corresponds to an average CFR of $1.92 \cdot 10^4$ in this scenario (since the distance to cloud is ~1000 km, and the average distance between two nodes in a 100 m × 100 m area is 52.14 m). Therefore, that setting implies working into the Fog-Cloud region.

IV. Proposed Heuristic

Aimed at solving the scalability issues of the mathematical model proposed in Section III, in this section we propose a heuristic to solve the FCC problem. It is based on the unsupervised machine learning *k*-means clustering method [11], [35] which organizes a set of elements into *k* clusters. This is typically done by means of a repetitive procedure: each cluster is characterized by a centroid, the elements are assigned to the closest centroid (and thus to that cluster), and then the centroids are recomputed by averaging the elements assigned to it. This procedure is repeated until the assignments no longer change. The *k*-means method may converge to a local optimum rather than to a global one. Therefore, the whole procedure is typically executed several times (from now on, *iterations*), each time starting with a different random initial position of the centroids. As previously explained at the end of Section II, *k*-means is the core or our approach, but it has to be enriched with additional mechanisms to enforce the constraints imposed by the FCC problem. Algorithm 2 shows the pseudocode of the proposed heuristic.

The proposed heuristic receives the input parameters of the FCC problem (as defined in the previous section) and a user-defined parameter setting the maximum number of iterations for the underlying *k*-means heuristic (lines 1 and 14 of Algorithm 2). Then, since the original *k*-means technique does not take into account the constraints due to the lack of connectivity between nodes, the proposed heuristic builds a reachability graph (according to λ_{ij} values) and finds connected components (subnets), in order to run the procedure to create clusters within each of those subnets independently (lines 4-5).

On the other hand, the *k*-means method requires, as an input, the number of clusters to be created (*k*), but, unfortunately, this is not known *a priori*. Nevertheless, as shown in Section III.D, pragmatic scenarios are of high CFR type, and in these scenarios the optimal number of clusters is generally equal to the lower bound. Therefore, for each subnet, *k* is initially set to the lower bound for that subnet (line 9), which is computed using Algorithm 1, and then the original *k*-means method is applied (line 16). In this way, the nodes of the subnet are split into *k* clusters according to their coordinates (the position of the devices). The next step consists in executing a procedure to select the leader of each of those clusters (line 17). It consists of three phases. First of all, for each of those clusters, it selects as leader to the node with cloud connectivity (γ_i=1) that can serve a maximum number of devices (taking into account reachability, λ_{ij}, and capacity, K_i, constraints). Those nodes of the cluster which cannot be really served by that leader (due to those constraints) are temporally marked as *unassigned*. Secondly, for those clusters where none of the nodes have cloud connectivity, the status of all nodes of that cluster are set to *unassigned*, the cluster is deleted, and a non-leader node with cloud connectivity which was initially part of another cluster, is elevated to the role of leader of a new to-be-formed cluster. Thirdly, all nodes marked as *unassigned* in the previous two steps are assigned to the closest cluster leader which is reachable and has enough capacity to serve it. If the whole procedure is successful (i.e., all nodes are connected to cloud directly or via a cluster leader), the procedure returns the set of leaders, otherwise it returns an empty set (lines 17-20).

Then, three optimization stages (line 21) are applied to feasible solutions:

1. In each cluster, the leader role will be reassigned to a different node of the cluster if it has connectivity with the rest of cluster nodes and reduces the solution cost.
2. For each non-leader node, it will be assigned to another cluster, if the cost decreases with that change and the cluster leader of that cluster has idle capacity to serve that node.
3. For each pair of non-leader nodes, the clusters assigned to them will be exchanged, if those nodes have connectivity to the new leader and the cost is reduced.

When the backup flag, ρ, is activated (lines 22-27), the heuristic searches for a backup leader in each cluster (which must have cloud connectivity and be able to serve the rest of cluster nodes). The solution is discarded if no backup leader is found for each cluster.

Finally, the cost associated to the subnet is computed using equation (2), line 28, and the lowest cost solution found in all iterations of the *k*-means procedure (line 14) is considered as the solution for that subnet (lines 29-31).

As previously stated, *k* (the number of clusters to be created) is initially set to the lower bound for the subnet. Although in fog-cloud scenarios that number of clusters usually provides the minimum cost, that is not always the case. Then, unless a stopping criterion is met (line 34), *k* will be increased in one unit and thus the algorithm will search for a solution with an additional cluster in another execution of the loop (lines 35-36 and 12). There are three stopping criteria: *i*) *k*+1 is higher than the higher bound; *ii*) it is mathematically impossible to reduce the FCC cost with an additional cluster (this condition can be checked very quickly); *iii*) if the cost obtained for *k*-1 clusters is lower than that of *k* clusters. In cases *i*) and *ii*) the solution with *k* clusters will be selected as the final solution for that subnet. In case *iii*) the solution with *k*-1 number of clusters is selected as the final solution for that subnet. The final clustering solution is obtained by aggregating the solutions obtained for each of the subnets (lines 37-44).

In the next section, the proposed model and algorithm are validated and their performance compared.

Algorithm 2: *k*-means-Based Heuristic

```
1:  procedure kmeans_based_heuristic(problem_input_parameters, number_of_iterations)
2:    solution ← ∅
3:    solution_cost ← 0
4:    subnets ← find_all_connected_components([$\lambda_{ij}$])
5:    for each subnet S in subnets do
6:      best_subnet_solution ← ∅
7:      best_subnet_cost ← ∞
8:      num_clusters_upper_bound ← $\sum_{i \in S} \gamma_i$
9:      num_clusters_lower_bound ← num_clusters_lower_bound(S, [$\lambda_{ij}$], [$\gamma_i$], [$K_i$])
10:     num_clusters ← num_clusters_lower_bound
11:     finish ← False
12:     while finish = False and num_clusters ≤ num_clusters_upper_bound do
13:       iteration ← 0
14:       while iteration < number_of_iterations do
15:         iteration ← iteration + 1
16:         clusters ← k_means_algorithm(nodes_positions, num_clusters)
17:         clusters_leaders ← select_clusters_leaders(clusters, S, [$\lambda_{ij}$], [$\gamma_i$], [$K_i$]))
18:         if clusters_leaders = ∅ then
19:           continue   # Not feasible solution (jump to line 14)
20:         end if
21:         clusters ← solution_optimization(clusters, clusters_leaders)
22:         if $\rho$ = 1 then
23:           clusters_backup_leaders ← select_clusters_backup_leaders(clusters, clusters_leaders, S, [$\lambda_{ij}$], [$\gamma_i$], [$K_i$]))
24:           if clusters_leaders = ∅ then
25:             continue   # Not feasible solution (jump to line 14)
26:           end if
27:         end if
28:         subnet_cost ← cost_estimation(clusters, clusters_leaders)
29:         if subnet_cost < best_subnet_cost then
30:           best_subnet_cost ← subnet_cost
31:           best_subnet_solution ← clusters
32:         end if
33:       end while
34:       finish ← check_finish_criteria(best_subnet_solution, best_subnet_cost)
35:       num_clusters ← num_clusters + 1
36:     end while
37:     if best_subnet_solution ≠ ∅ then
38:       solution ← solution ∪ best_subnet_solution
39:       solution_cost ← solution_cost + best_subnet_cost
40:     else
41:       solution ← ∅
42:       solution_cost ← ∞
43:       go to end procedure
44:     end if
45:   end for
46:   return solution, solution_cost
47: end procedure
```

V. Validation And Performance Comparison

In this section, we apply the proposed model and heuristic to pragmatic scenarios to validate their use, and to analyze their performance in terms of the quality of the solutions and scalability.

A. Scenarios Description

Different scenarios based on a 100 m × 100 m area are considered to represent different density of devices having high connectivity among them. Nodes representing edge devices are randomly placed along the area according to a uniform distribution. Moreover, 90% of the nodes (randomly selected) can access to cloud and 75% (again, randomly selected) are considered as static. Distance to cloud for those nodes with access to cloud is approximately 1000 km (the exact value depends on the location of the node in the 100 m × 100 m area). The number of nodes that can be supported by each node is randomly selected from 0 to 10. Regarding the transmission power, we assume that each node supports up to three distinct levels of power to manage coverage (see [12], [13]). The maximum coverage of a node is set to 100 m and the maximum transmission power to 100 mW. Regarding weights α, β, φ, σ in the objective function, values for φ (which controls the importance of minimizing the required transmission power of the nodes) and σ (which penalizes the selection of mobile nodes as leaders) are set to 1 and 10, respectively, aimed not to impact the main objective of minimizing distance, but prioritizing static nodes to serve as leaders instead of minimizing transmission power. Regarding, α and β, setting both of them to 1 would lead to an average value of CFR = $1.92 \cdot 10^4$, well into the high CFR area of Fig. 3, as previously mentioned in Section III.D. However, we have set α = 1000 and β = 1, so that the average value of CFR is 19.2, which still corresponds to the area labeled "fog-cloud" in Fig. 3, but close to the grey area in the picture. Therefore, this setting turns into scenarios where the number of clusters is typically minimized (pragmatic scenarios), but some of scenarios require more clusters than those determined by the lower bound (which represent a good setting to test the performance of the heuristic, as that situation deviates from the optimal context for the use of the proposed heuristic).

The MILP model for each problem instance has been solved using CPLEX [37], while the algorithm has been implemented in Python using the scikit-learn [38] implementation of *k*-means. A 64-bits computer built upon 12 Intel core ES-2620 v2 at 2.1GHz and 128 GB RAM has been used in both cases, and the computing time required with each technique to solve the FCC problem has been measured. Specifically, for fairness, the computing time for the MILP model accounts the time to solve the formulation, while the time to generate the problem is not accounted; in fact, it is much lower and not relevant compared to the time required to solve the problem, in the evaluated scenarios. The solving time together with the FCC cost of the solutions obtained are the main performance metrics that will be analyzed, although the number of clusters created when using both techniques will also be studied.

B. Mathematical Model and Algorithm Validation

Before comparing performance of the proposed mathematical model and machine learning-based heuristic for large sets of problem instances, let us focus on the results obtained for very small scenarios (i.e., having 10 nodes), aimed at: *i*) validating the mathematical model and the algorithm, and; *ii*) showing, graphically, illustrative solutions obtained for the sake of a comprehensive understanding of the problem and the subsequent results.

We solved 10 randomly generated problem instances by means of the MILP and the heuristic (20 iterations), assuming both a backup leader is and is not required. In nine cases (either requiring or not requiring backup), the solutions obtained when the problem instances were solved utilizing the proposed heuristic were optimal (i.e., same solutions as the ones obtained when considering the mathematical model). Regarding the instance that resulted in a different solution, Fig. 4 depicts both the optimal solution (Fig. 4a) and the pseudo-optimal solution obtained by the heuristic (Fig. 4b). For each figure, the two clusters obtained are represented: cluster 1 is represented by triangles, whereas cluster 2 is represented by dots. Moreover, the leader and backup leader (selected in scenarios where backup is required) of each cluster are identified. Although two clusters were obtained in both cases and the same backup leaders were selected, there is a slight difference in the leader selection in one of the clusters (cluster 2) and the assignment of an element to cluster 2 instead of cluster 1. Interestingly, despite of this slight difference, the objective value for the pseudo-optimal solution is very close to the optimal one; i.e., a gap around 0.3% is shown in the FCC costs.

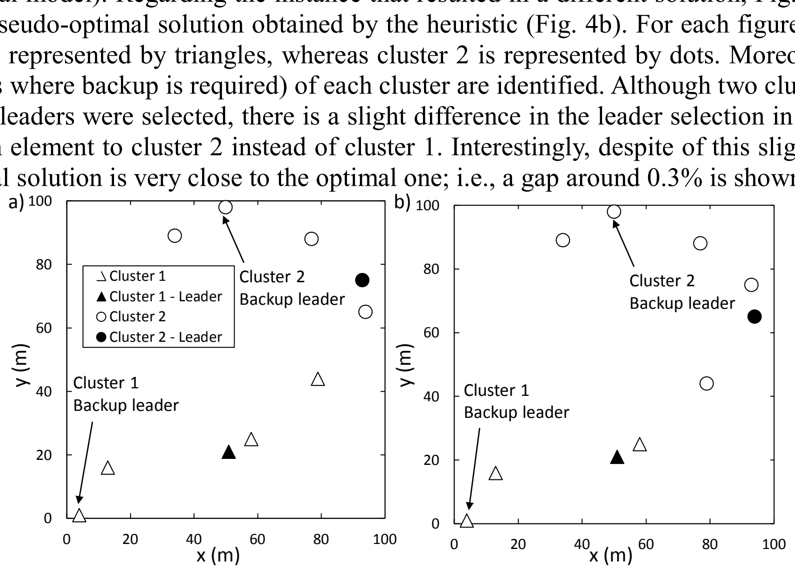

Fig. 4. Optimal (a) and pseudo-optimal (b) solutions.

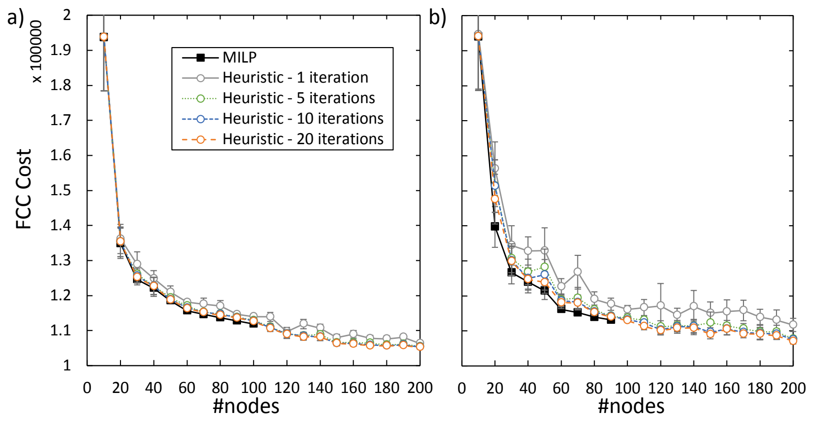

Fig. 5. Average FCC cost against number of nodes, without backup (a) and with backup (b).

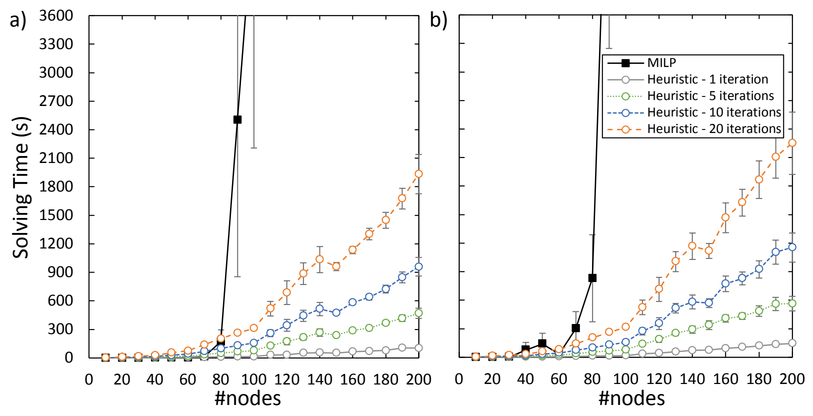

Fig. 6. Average solving time against number of nodes, without backup (a) and with backup (b).

Next, we compare the performance of the mathematical model and the algorithm for larger problem instances.

C. *Performance Comparison*

The performance of the MILP model and the heuristic has been compared in the scenarios described in subsection V.A, varying the number of nodes from 10 to 200. For each number of nodes, 10 scenarios have been randomly generated and all results are presented in average with the 95% confidence interval. Two different cases have been analyzed in each scenario: *i*) when no backup is required (ρ=0), and *ii*) when backup is required (ρ=1).

Fig. 5 shows the FCC cost of the solutions provided by MILP and the heuristic depending on the number of nodes. The corresponding values of solving time are shown in Fig. 6. In both figures, subfigure (a) shows results without backup and (b) with backup.

When comparing the results in terms of FCC cost (Fig. 5) it can be seen that increasing the number of nodes results in lower values of the objective function. The reasoning behind this is that, the higher the number of nodes, the larger are the chances to find nodes with higher capacity, thus minimizing the number of clusters. Therefore, the contribution from the cloud cost is significantly reduced. According to the results obtained, for instances where 10 nodes are considered, the solutions obtained by the algorithm are optimal in terms of cost in most of the cases. For larger number of nodes, and setting the number of iterations of the heuristic to 5, the gap against the optimal value is below 1% when backup is not required (Fig. 5a), and around or below 3% when backup is considered (Fig. 5b). Increasing the number of iterations from 5 to 20 reduces the cost of the solutions obtained, but not very significantly.

Regarding the time required to solve problem instances, Fig. 6 depicts the solving time against the number of nodes when using the 64-bit computer described in Section V.A. Although the MILP formulation provides optimal solutions, it can be seen that the computing time dramatically increases when the number of nodes grows beyond 90 nodes (80 nodes in case of using backup). For instance, it solves the FCC problem in less than 3 minutes (in average) for scenarios with 80 nodes without backup, but requires around 80 minutes in average for scenarios with 100 nodes. Moreover, some instances could not be solved to optimality after 5 hours. In contrast, the computing time of the heuristic does not increase so drastically when facing scenarios with many devices. For instance, when using the configuration which just run one iteration of the *k*-means method, it solves scenarios with 80 nodes in just 10 seconds (in average) and scenarios with 200 nodes in less than 2 minutes, making it suitable for networking scenarios with many devices, in contrast with the MILP formulation.

In addition, Fig. 7 shows the number of clusters obtained in each case, when backup is not required and when it is required (Fig. 7a and Fig. 7b, respectively). In view of these figures, it is clear that the number of clusters obtained by the heuristic is very

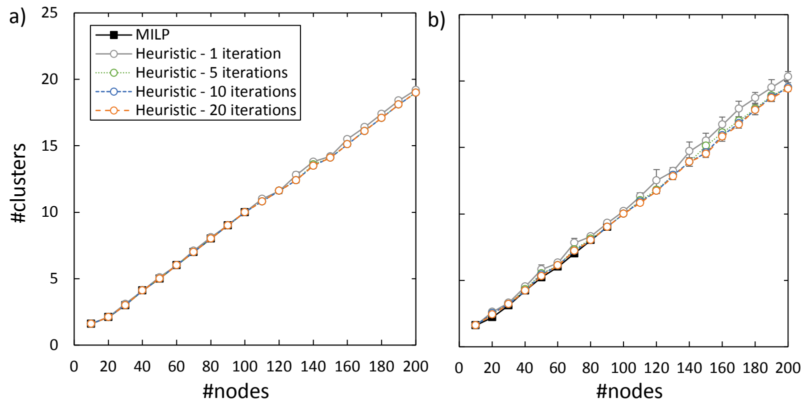

Fig. 7. Average number of clusters against number of nodes, without backup (a) and with backup (b).

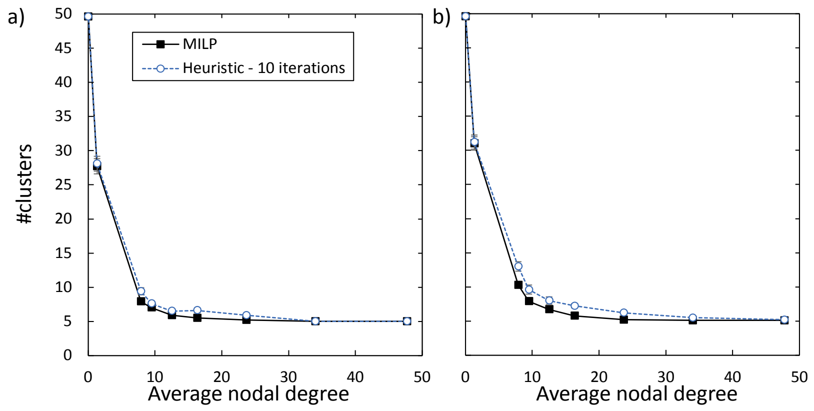

Fig. 8. Average number of clusters against nodal degree for scenarios without backup (a) and with backup (b).

similar to that obtained when the MILP formulation is used. Similarly, it can be seen that there is little or none difference in the number of clusters obtained by the heuristic when 5 to 20 iterations are executed; therefore, the main difference between considering 5 to 20 iterations is in the elements forming each cluster and their role.

Finally, for completeness, we analyze the solutions obtained for scenarios with a fixed number of nodes (50 nodes) and varying connectivity (by increasing the area in which the nodes are distributed). Fig. 8 illustrates the number of clusters against the average nodal degree for scenarios without and with backup. It can be observed that, when almost all nodes have connectivity among them or when most of the nodes are isolated, the number of clusters obtained by the heuristic is the same or very close to the number of clusters in the optimal solutions. For medium-low connectivity scenarios, the average number of clusters is slightly increased when the heuristic is considered (as these scenarios have more stringent constraints).

In summary, the MILP formulation provides the optimal solutions of the FCC problem but suffers from scalability problems when the number of nodes is higher than 80 (not a huge number considering the prediction of future IoT scenarios). However, the machine learning-based heuristic obtains nearly the same performance in terms of both the number of clusters required and FCC cost, but it employs much less computing time than the MILP formulation to solve the FCC problem when the number of nodes increases.

VI. Conclusions

The advent of novel fog-cloud architectures (F2C) and the softwarization of the telecommunication industry become crucial to facilitate future IoT environments. However, it requires novel and efficient resource management procedures to handle the diversity of resources in a coordinated way.

In this paper, we have formally presented the Fog-Cloud Clustering problem, referred to as the FCC problem, considering a hierarchical and distributed F2C management architecture. The FCC problem has been modeled as an MILP formulation, and we have also obtained lower and higher bounds to the number of clusters required. Since the proposed mathematical model does not scale well with the number of network nodes (i.e., the computing time dramatically increases), a machine learning-based algorithm for FCC has also been proposed.

By means of a simulation study, both the MILP model and the algorithm were validated and their performances were compared in different realistic scenarios in terms of quality of solutions and solving time. The results from those studies show that the solutions obtained by the machine learning-based algorithm are very close to the optimal ones (the ones provided by MILP formulation) while the solving time of the algorithm is much lower than that of the MILP formulation when the number of nodes increases.

In view of the results obtained, the proposed machine learning-based algorithm could be considered by fog-cloud operators to efficiently identify groups of resources and their role (i.e., to map resources) to set the F2C architecture while improving scalability, compared to the MILP formulation.

In our opinion, this work opens at least three research lines. First of all, the MILP formulation proposed in the paper sets a generic framework to solve the FCC problem, enabling the operator to establish trade-offs between the importance of minimizing latencies between edge devices, latencies between nodes and the cloud, the required transmission power, and the number of mobile devices acting as leaders. However, the formulation would greatly benefit from the incorporation of a techno-economic model accounting for the previous issues and being used as the driving optimization function. Moreover, a techno-economic model would open the opportunity to integrate, in the proposed formulation, the problem of selecting the best cloud provider/facilities. It should be noted that there are several works focused on economic models for fog-cloud environments (a review of some of these works can be found in [26]). However, they are mainly focused on task scheduling or workload/service allocation, so that their mapping to the MILP formulation presented here is not straightforward. Precisely, analyzing the operation and performance improvements of task scheduling and service allocation when using an architecture designed with the methods proposed in this paper, would be a second research line, and a third research line consists in developing a small testbed scenario to demonstrate the operation of the complete system.

ACKNOWLEDGMENT

This work is supported by the Spanish Thematic Networks under contract RED2018-102585-T and TEC2015-71932-REDT, by the Spanish Ministry of Economy and Competitiveness and by the European Regional Development Fund under contracts RTI2018-094532-B-I00 (MINECO/FEDER) and TEC2017-84423-C3-1-P (MINECO/FEDER), as well as by the H2020 mF2C project (730929) and the European Regional Development Fund (ERDF) through the project 0667_DISRUPTIVE_2_E of the cooperation programme Interreg V-A Spain-Portugal (POCTEP) 2014-2020.

REFERENCES

[1] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021", White paper, Feb. 2017.

[2] H. Freeman and R. Boutaba, "Network Industry Transformation Through Softwarization", IEEE Communications Magazine, vol. 54, no. 8, pp. 4-6, 2016.

[3] M. Armbrust, et al., "A View of Cloud Computing", Communications of the ACM, vol. 53 no. 4, pp. 50-58, 2010. doi: 10.1145/1721654.1721672.

[4] F. Bonomi, et al., "Fog Computing and Its Role in the Internet of Things", Proc. of MCC workshop on Mobile cloud computing, pp. 13-16, August, 2012.

[5] T.Coughlin, "Convergence through the Cloud-to-Thing Consortium", IEEE Consumer Electronics Magazine, vol. 6, no. 3, pp. 14-17, 2017. doi: 10.1109/MCE.2017.2684914.

[6] IEEE 1934-2018 - IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing https://standards.ieee.org/standard/1934-2018.html [Accessed: Feb. 2020], 2018.

[7] H. Gupta, et al., "SDFog: A Software Defined Computing Architecture for QoS Aware Service Orchestration over Edge Devices", https://arxiv.org/pdf/1609.01190.pdf , [Accessed: Oct.. 2019]

[8] Open Fog Consortium Working Group, "OpenFog Reference Architecture for Fog Computing", White paper, Feb 2017

[9] mF2C project, http://www.mf2c-project.eu, [Accessed: Oct. 2019]

[10] X. Masip-Bruin, et al., "Foggy Clouds and Cloudy Fogs: a Real Need for Coordinated Management of Fog-to-Cloud (F2C) Computing Systems", IEEE Wireless Communications Magazine, vol. 23, no. 5, pp. 120-128 2016. doi: 10.1109/MWC.2016.7721750.

[11] J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations", 5th Berkeley Symp. on Mathematical Statistics and Probability. UCAL Press. pp. 281–297, 1967.

[12] J.E. Wieselthier, G. D. Nguyen, A. Ephremides, "The Effect of Discrete Power Levels on Energy-Efficient Wireless Broadcast in Ad Hoc Networks", 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, vol.4, pp. 1655-1659, 2002. doi: 10.1109/PIMRC.2002.1045460.

[13] N. Botezatu and R. Dhaou, "Adaptive Power Control in 802.11 Wireless Mesh Networks", World Congress on Engineering 2011 vol. II, 2011.

[14] S. Z. H. Zahidi, et al., "Optimizing Complex Cluster Formation in MANETs Using SAT/ILP Techniques", IEEE Sensors Journal, vol. 13, no. 6, pp. 2400-2412, 2013. doi: 10.1109/JSEN.2013.2254234.

[15] A.A. Abassi, M. Younis, "A Survey on Clustering Algorithms for Wireless Sensor Networks," Computer Communications, vol. 3, no. 14-15, pp. 2826-2841, 2007. doi: 10.1016/j.comcom.2007.05.024.

[16] R. Agarwal, M. Motwani, "Survey of Clustering Algorithms for MANET," International Journal on Computer Science and Engineering, vol.1, no. 2, pp. 98-104, 2009.

[17] M. Chatterjee, S. K. Das, D. Turgut, "WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks", Cluster Computing, vol. 5, no. 2, pp. 193-204, 2002. doi: 10.1023/A:1013941929408.

[18] E. Balevi, R.D.Gitlin, "Optimizing the Number of Fog Nodes for Cloud-Fog-Thing Networks", IEEE Access, vol. 6, pp. 11173-11183, 2018. doi: 10.1109/ACCESS.2018.2808598.

[19] F. S. Abkenar and, A. Jamalipour, "EBA: Energy Balancing Algorithm for Fog-IoT Networks," in IEEE Internet of Things Journal, vol. 6, no. 4, pp. 6843-6849, Aug. 2019. doi: 10.1109/JIOT.2019.2911953.

[20] S. Ghosh, et al., "Mobi-IoST: Mobility-aware Cloud-Fog-Edge-IoT Collaborative Framework for Time-Critical Applications," IEEE Transactions on Network Science and Engineering, Sept. 2019. doi: 10.1109/TNSE.2019.2941754.

[21] A. Brogi, S. Forti, and M. Gaglianese, “Measuring the Fog, Gently.,” In Yangui S., Bouassida Rodriguez I., Drira K., Tari Z. (eds) Service-Oriented Computing. ICSOC 2019. Lecture Notes in Computer Science, vol. 11895, pp. 523-538. Springer, Cham. doi: 10.1007/978-3-030-33702-5_40.

[22] M. Mukherjee et al., “Task Data Offloading and Resource Allocation in Fog Computing With Multi-Task Delay Guarantee,” IEEE Access, vol. 7, pp. 152911-152918, 2019. doi: 10.1109/ACCESS.2019.2941741.

[23] M. Taneja and A. Davy, “Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm,” Proc. IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, 2017, pp. 1222-1228. doi: 10.23919/INM.2017.7987464.

[24] H. Shah-Mansouri and V. W. S. Wong, “Hierarchical Fog-Cloud Computing for IoT Systems: A Computation Offloading Game,” IEEE Internet of Things Journal, vol. 5, no. 4, pp. 3246-3257, Aug. 2018. doi: 10.1109/JIOT.2018.2838022.

[25] R. K. Naha, et al., “Deadline-based dynamic resource allocation and provisioning algorithms in Fog-Cloud environment,” Future Generation Computer Systems, vol. 104, pp. 131-141, March 2020. doi: 10.1016/j.future.2019.10.018

[26] R. Mahmud, S. Narayana Srirama, K. Ramamohanarao, and R. Buyya, “Profit-aware application placement for integrated Fog–Cloud computing environments,” Journal of Parallel and Distributed Computing, vol. 135, pp 177-190, Jan. 2020. doi: 10.1016/j.jpdc.2019.10.001.

[27] O. Skarlat, et al., “A framework for optimization, service placement, and runtime operation in the fog,” IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC). IEEE, 2018.

[28] O. Skarlat, K. Bachmann, and S. Schulte, “Fogframe: Iot service deployment and execution in the fog,” KuVS-Fachgespräch Fog Computing 2018, pp. 5-8, 2018.

[29] C. Guerrero, I. Lera, and C. Juiz, “On the influence of fog colonies partitioning in fog application makespan,” IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), 2018. DOI 10.1109/FiCloud.2018.00061

[30] M. Nardelli, et al., “A Multi-level Elasticity Framework for Distributed Data Stream Processing,” European Conference on Parallel Processing. Springer, Cham, 2018. doi:10.3390/a11090134.

[31] S. Singh Gill, P. Garraghan, and R. Buyya, “ROUTER: Fog enabled cloud based intelligent resource management approach for smart home IoT devices,” Journal of Systems and Software, vol. 154, pp. 125-138, Aug. 2019. doi: 10.1016/j.jss.2019.04.058.

[32] X. Masip-Bruin et al., “Managing resources continuity from the edge to the cloud: Architecture and performance,” Future Generation Computer Systems, vol. 79, pp. 777–785, 2018. doi: 10.1016/j.future.2017.09.036.

[33] M. Ghobaei-Arani, A. Souri and A.A. Rahmanian, “Resource Management Approaches in Fog Computing: a Comprehensive Review,” Journal of Grid Computing, pp. 1-42, 2019. doi: 10.1007/s10723-019-09491-1.

[34] R. K. Naha et al., “Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions,” IEEE Access, vol. 6, pp. 47980-48009, 2018. doi: 10.1109/ACCESS.2018.2866491.

[35] D. Xu and Y. Tian, “A Comprehensive Survey of Clustering Algorithms”, Annals of Data Science. vol. 2, no. 2, pp. 165-193, 2015. doi: 10.1007/s40745-015-0040-1.

[36] K. L. Wagstaff, “Constrained Clustering”, in *Encyclopedia of Machine Learning* (C. Sammut, G.I. Webb, eds.), Springer, Boston, MA, 2011. doi: 10.1007/978-0-387-30164-8_163.

[37] IBM CPLEX Optimizer, https://www.ibm.com/analytics/cplex-optimizer [Accessed: Oct. 2019]

[38] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python”, Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.