Mobile Edge Computing Resources Optimization: a Geo-clustering Approach

Mathieu Bouet, Vania Conan

Mobile Edge Computing Resources Optimization: a Geo-clustering Approach

Mathieu Bouet, Vania Conan

***Abstract*—Mobile Edge Computing (MEC) is an emerging technology that aims at pushing applications and content close to the users (e.g., at base stations, access points, aggregation networks) to reduce latency, improve quality of experience, and ensure highly efficient network operation and service delivery. It principally relies on virtualization-enabled MEC servers with limited capacity at the edge of the network. One key issue is to dimension such systems in terms of server size, server number, and server operation area to meet MEC goals. In this paper, we formulate this problem as a mixed integer linear program. We then propose a graph-based algorithm that, taking into account a maximum MEC server capacity, provides a partition of MEC clusters, which consolidates as many communications as possible at the edge. We use a dataset of mobile communications to extensively evaluate them with real world spatio-temporal human dynamics. In addition to quantifying macroscopic MEC benefits, the evaluation shows that our algorithm provides MEC area partitions that largely offload the core, thus pushing the load at the edge (e.g., with 10 small MEC servers between 55% and 64% of the traffic stay at the edge), and that are well balanced through time.**

***Index Terms*—Mobile edge computing, Multi-access edge computing, fog computing, network virtualization, dimensioning, clustering.**

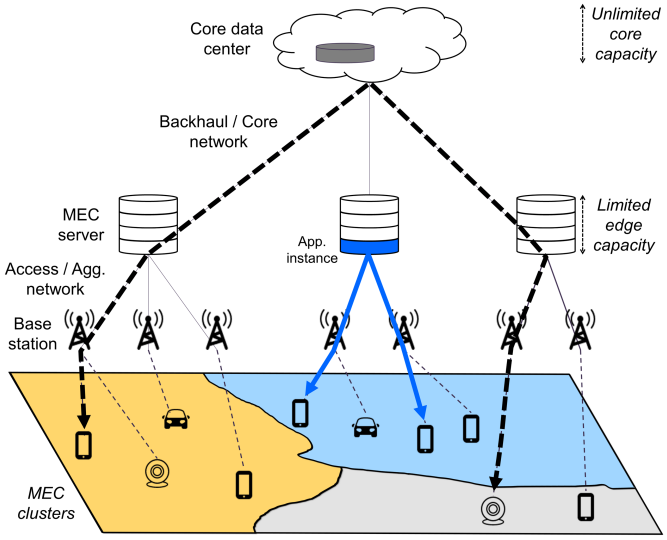


Fig. 1. MEC deployment: tasks and applications (e.g., video chat analytics, video chat customization) are mainly offloaded onto MEC servers at the edge of the network to reduce latency and offload the core network. The area is thus partitioned into MEC clusters, each cluster being served by a MEC server. Note that it can be an n-level architecture.

I. Introduction

MOBILE Edge Computing (MEC - also know as Multi-access Edge Computing [1], and similar to fog computing [2]) has emerged as a key enabling technology for realizing the IoT and 5G visions. It aims at reducing latency and ensuring efficient network operation and service delivery and pushing content and services close to the users. Numerous MEC applications are already envisioned and investigated, for example: chat/video analytics, video acceleration, augmented/virtual reality, location-based services, connected vehicles, and IoT gateways [3].

In a MEC deployment *MEC servers* are positioned in the infrastructure close to the edge of the network (see Fig. 1): they are small-scale datacenters with low to moderate resources collocated with the base stations, access points and/or placed in the access/aggregation network. They leverage virtualization to support MEC applications run as virtual machines, containers, microservices etc. [4]. The purpose of MEC servers is to host as many applications as possible at the edge to improve latency and alleviate congestion in the core. MEC thus performs application and network offloading from the *core data center* on to the edge [5], [6].

The central decision in a MEC system design is to decide which users, applications and share of traffic should be handled by the MEC servers. To address this key issue we name *MEC cluster* the area, and by extension the base stations and the users in the area, served by a MEC server. Indeed the efficiency of a MEC system heavily depends on such aspects as the distribution of communications and workloads in time and space. Its cost depends on server density, capacity, and interconnection. Imbalanced MEC clusters that handle highly different traffic volumes would lead to an inefficient use of resources and to unequal Quality of Experience (QoE). Put in terms of MEC clustering, the key question is thus: how to have an efficient partition of MEC areas? From this, a placement of MEC servers can be derived.

The problem is aggravated by the well documented fact that mobile traffic is very dependent on time and locality. Indeed mobile communications are spatially distributed according to the population density and activity, which vary in time. For instance, the mobile traffic in the business areas differ from the mobile traffic in the transport, residential and entertainment areas [7]–[9]. As it was shown by Qazi et al. [10], in a MEC perspective, such variations have a direct impact on the load of the potential MEC servers. In addition, it was shown by Tastevin et al. [9] that mobile communications in an urban environment have a high spatial locality - they tend to follow a power law, which motivates a local consolidation of applications at MEC servers. Such properties will be amplified with the realization of the IoT and 5G visions [3].

In this paper, we formally describe the MEC geo-clustering

problem and provides a Mixed Integer Linear Programming (MILP) formulation. However, as the large-scale dimension of MEC systems and mobile communications makes classic analytical and simulation-based approaches almost inapplicable, we then investigate a graph-based method. We propose an algorithm that, based on the spatial distribution of the communications, finds a MEC partition that favors application instantiation at the edge instead of at the core. The resulting clusters correspond to MEC areas. Our algorithm takes into account the maximum server capacity, that we express as the maximum number of served communications per unit of time, but that can be easily expressed in terms of resources (CPU, storage...) or application instances. We evaluate the MILP and the clustering algorithm using a dataset of mobile communications in a city provided by a mobile operator. We first show that the clustering takes into account the spatial distribution of the communications and enables to largely offload the core. In addition, the algorithm provides results that are close to the MILP results on small-scale problem instances. Then, we evaluate the clustering algorithm on larger problem sizes and outline the benefits of MEC even for very small MEC server sizes. The obtained MEC clusters have well balanced loads and enable to keep a large portion of the traffic at the edge. Finally, we evaluate the MEC partition over a week of communications and show that it largely supports temporal dynamics. There is almost no server saturation, i.e., traffic offloaded to the core, while the loads remain balanced.

In summary, this paper makes the following contributions:

1) We formulate the MEC clustering problem and provide a MILP formulation (Sec. III).
2) We design a MEC clustering algorithm (Sec. IV) that consolidates as many communications as possible at the edge.
3) We use a real-world dataset of spatially and temporally distributed mobile communications (Sec. V-A).
4) We evaluate our proposal and show that, despite the spatialtemporal dynamics of the traffic, our algorithm provides well-balanced MEC areas that are close to optimal on small problem instances (Sec. V-B) and serve a large part of the communications on real-world problem instances (Sec. V-C and Sec. V-D).

We discuss related work in Sec. II and conclude in Sec. VI.

In [11] we presented the geo-clustering algorithm and a first evaluation. In this paper, we introduce a mixed integer linear programming formulation of the problem and a formalization of the algorithm. We also evaluate both the MILP and the algorithm through extensive and detailed simulations.

II. Related Work

In the past few years, in parallel notably to the ETSI MEC ISG initiative [1] and to the OpenFog Consortium [2], MEC has emerged as a new promising research area. Very recently, first surveys have been published to present comprehensive panoramas of the use cases, architectures and challenges [12]–[14]. We present in this section challenges and related work that are linked with the problem of MEC resources dimensioning.

Dimensioning and MEC server placement. The MEC server placement problem was illustrated by Qazi et al. [10] who showed that the number and the locations of MEC servers have a direct impact on the QoE (imbalance loads and high latencies) and on the operational cost. However, they did not address the placement problem. They proposed an NFV-based orchestration for MEC. Note that the server placement problem is significantly different from the conventional base station site selection problem since, although both problems are constrained by the deployment budget, placing edge sites is coupled with the computational resource provisioning. Ceselli et al. [15] have proposed a mixed integer linear programming formulation of the joint problem of base stations allocation to MEC servers and routing to reduce infrastructure cost. Our proposal mainly differs on three important aspects. First, they assume the locations of the LTE 4G base stations are known. Their analytical formulation does not scale properly. Most of all, the clusters they obtained are not geo-consistent, meaning that the base stations associated to a MEC server can be completely scattered in space.

Control plane design. Recent proposals have addressed control plane design. They investigate how the current centralized LTE core architecture, where most of the traffic converge [16], can be decomposed and split to alleviate congestion and reduce latency [17]. Software-Defined Networking (SDN) is used to redirect peering traffic in-between the base stations and the Evolved Packet Core (EPC), thus offloading the core and improving latency [5]. SDN is also combined with NFV (Network Functions Virtualization) to propose a backwards-compatible orchestration architecture where virtual EPC functions are chained with SDN and instantiated in MEC servers to efficiently use resources [10].

System approaches. While NFV has gained momentum, recent proposals have focused on shortening network functions instantiation and reducing their system footprint with approaches based on unikernels [4]. In particular, it has been shown that an inexpensive commodity server is able to concurrently run up to 10,000 specialized virtual machines, instantiate a VM in as little as 10 milliseconds, and migrate it in under 100 milliseconds [18]. This technology is very promising in an MEC context where an application could be instantiated on the fly at MEC servers for a user or a group of users and shutdown once the communication is ended. Progress in this direction complements our deployment work as it would make it easier to instantiate locally applications at MEC servers.

Application/task and content offloading. Application offloading, both from the device to the edge and from the core to the edge, has been extensively studied. It notably includes task decomposition and packaging [19], assignment, and migration [6], server scheduling and selection, content caching and pre-fetching [20]. Some of the proposals are similar to those addressed in Mobile Cloud Computing (MCC), which addresses distributed clouds [21], [22].

III. MEC resources clustering

In this section, we formulate the MEC resource geo-clustering problem that we address and we present the cor-

responding mathematical optimization model.

A. Problem formulation

From a network system standpoint we consider a MEC deployment as presented in Fig. 1. All users belong to a *MEC cluster*, a geographic area whose traffic can be handled by a *MEC server*, that is a small-scale datacenter with low to moderate compute and storage resources. All user communications and applications, for instance ephemeral per-communication unikernel-based video analytics applications, are either handled by the local MEC server (e.g., the blue plain line in Fig. 1) or by a highly capacitated *core data center* (e.g., the black dotted line in Fig. 1), which can be farther in terms of latency.

We argue that one of the key design issues in a MEC system is to efficiently dimension MEC areas (or clusters). Such a MEC geo-partitioning must have the following properties:

1) MEC servers, as any compute, storage and network node, have a maximum capacity (e.g., in terms of CPU, storage resources or application hosting capabilities) that we considered as known a priori in our problem.
2) MEC server loads should be balanced both spatially and temporally to improve user experience and system expenditures.
3) The traffic between the MEC servers and the core should be minimized, in particular by consolidating applications at the MEC server level, such that the global latency is reduced.

This problem turns out to be both theoretically and computationally hard. It generalizes the graph cut based image segmentation problem with connectivity constraints, which is NP-hard [23], and introduces capacitated components.

In the following, we formulate the mathematical optimization formulation of the MEC geo-clustering problem and introduce alternative connectedness constraints that are restrictive but allow reducing the number of constraints.

Note that the following problem formulation focuses on edge-to-edge communications to be consolidated at the MEC servers. However, it could be easily extended to address edge-to-core communications.

B. Model

Input (problem data).

We assume that the considered area has been discretized in N cells. Let note G the set of the N cells. For example, if the discretization has been done according to a grid of length n, then $G = \{0, ..., n^2-1\}$. For a clearer constraint formulation, let note $Neigh(i)$ the cells that are spatially neighbors of the cell i. In a grid, a cell that is not on a boarder has 8 neighbor cells. $t_{i,j}$ corresponds to the amount of traffic or communications per unit of time from the cell $i \in G$ to the cell $j \in G$. The goal is to cluster the area cells. We note C the set of clusters. C is a partition of the set G. By default, C is equal to G, which means that the discretized parts are all in a unique cluster. The aim being to cluster cells, a solution of the program might result in a number of empty clusters. A cluster has a maximum capacity, in terms of traffic or communications per unit of time that can be processed at its MEC server, noted M.

Input (problem data)

N	Number of cells after area discretization
G	Set of N cells
$Neigh(i)$	The set of the cells that are spatially neighbors of the cell i
C	Set of clusters (clustered cells)
M	Maximum cluster capacity
$t_{i,j}$	Amount of traffic or communications per unit of time from the cell i and to the cell j
x_i, y_i	Integer variable, define the x and y coordinates of the cell i in a grid

Output (decision variables)

$a_{i,c}$	Binary variable, equal to 1 iff the cell i belongs to the cluster c
$b_{i,j,c}$	Binary variable, equal to 1 iff the cell i and the cell j are in the same cluster c
$f_{i,j,c}^{src,dst}$	Float variable in [0, 1], define the fraction of flow between the cell i and the j for a cluster c when the cell src and the cell dst are the source and destination of the flow respectively

TABLE I
PROBLEM DATA AND DECISION VARIABLES.

Output (decision variables).

We introduce two sets of binary variables. The first one corresponds to cell-cluster attachment variables: $a_{i,c}$ take value 1 if the cell $i \in G$ is in cluster $c \in C$. The second one is a set of intermediary variables: $b_{i,j,c}$ take value 1 if the cell $i \in G$ and the cell $j \in G$ are in the same cluster $c \in C$. Finally, since we want clusters that are geo-consistent, meaning that all their cells are connected, we introduce a set of float variables, noted F, for the commodity flow formulation that will ensure connectivity: $f_{i,j,c}^{src,dst} \in [0, 1]$ define the fraction of flow between $i \in G$ and $j \in G$ for cluster $c \in C$ when $src \in G$ and $dst \in G$ are the source and destination of the flow respectively.

Objective function.

$$\text{Maximize} \sum_{i \in G} \sum_{j \in G} \sum_{c \in C} t_{i,j}\ b_{i,j,c} \tag{1}$$

Constraints.

Cluster attachment unicity:

$$\sum_{c \in C} a_{i,c} = 1, \quad \forall i \in G \tag{2}$$

Intermediate variable: the cells i and j belong to the same cluster c

$$\begin{gathered} 0 \leq a_{i,c} + a_{j,c} - 2\ b_{i,j,c} \leq 1, \\ \forall i \in G, j \in G, c \in C \end{gathered} \tag{3}$$

Maximum cluster capacity (intra cluster communications):

$$\sum_{i \in G} \sum_{j \in G} t_{i,j}\ b_{i,j,c} \leq M, \quad \forall c \in C \tag{4}$$

Commodity flow constraints to ensure that a cluster c is connected (geo-consistent):

(i) For a cluster c, a flow between i and j exists if and only if i and j are in the same cluster c:

$$\begin{gathered} f_{i,j,c}^{src,dst} \leq b_{i,j,c}, \\ \forall src \in G, dst \in G, i \in G, j \in G, c \in C \end{gathered} \tag{5}$$

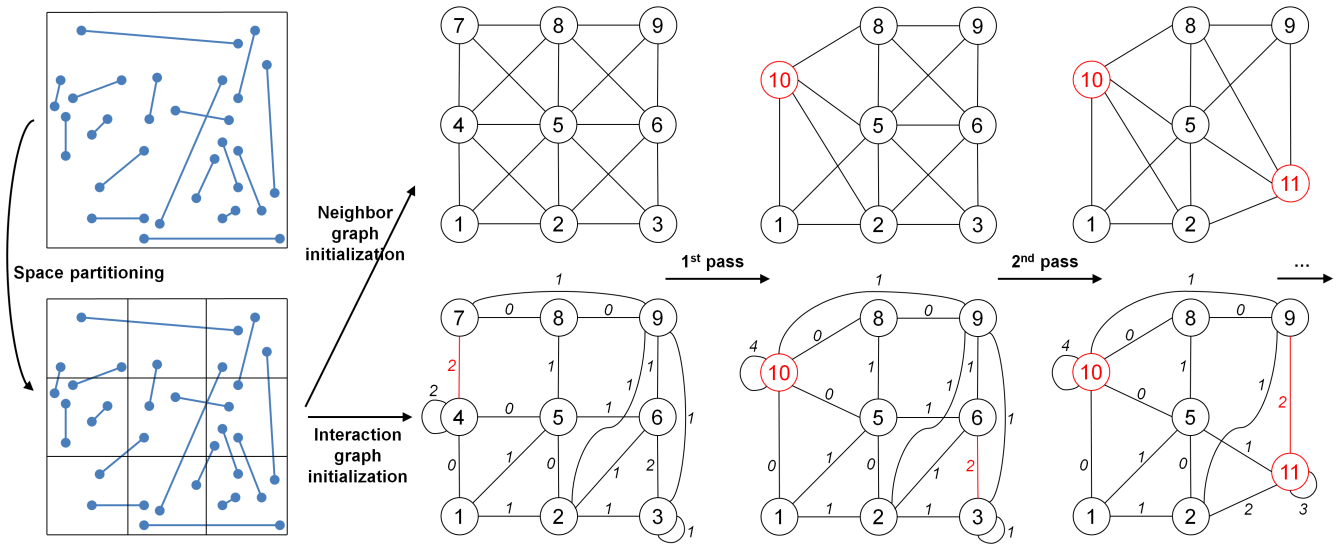


Fig. 2. Visualization of the steps of our graph-based algorithm. The area where are distributed the MEC communications is discretized into nodes which form MEC clusters. Each pass is made of two phases: one where the pair of neighbor nodes (i.e., clusters) that interact the most, while respecting the maximum cluster capacity, are selected; one where the two selected nodes (i.e., clusters) are merged to build a new/updated graph with an increased self-loop weight meaning that more communications or traffic are in the same cluster. The passes are repeated iteratively until no pair of neighbor nodes (i.e., clusters) can be merged because of the maximum MEC server capacity (self-loop weights). The result corresponds to a spatial partition of MEC clusters.

(ii) Flow conservation on transit cells:

$$\sum_{l \in Neigh(k)} f_{k,l,c}^{i,j} - \sum_{l \in Neigh(k)} f_{l,k,c}^{i,j} = 0, \quad \text{when } i = src, j = dst$$
$$\forall i \in G, j \in G, k \in G \setminus \{i, j\}, c \in C \tag{6}$$

(iii) Flow conservation at source and destination cells:

$$\sum_{l \in Neigh(i)} f_{i,l,c}^{src,dst} - \sum_{l \in Neigh(j)} f_{l,i,c}^{src,dst} = \begin{cases} b_{i,j,c} & \text{if } i = src \\ -b_{i,j,c} & \text{if } j = dst, i \neq j \end{cases} \tag{7}$$
$$\forall src \in G, dst \in G, i \in G, j \in G, c \in C$$

Alternative connectedness constraints.

The number of constraints explodes very rapidly with the number of cells generated by the discretization of the space. We thus define alternative connectedness constraints that can be substituted for the flow commodity formulation to reduce the number of constraints when the space discretization structure is a grid and thus manage larger problem instances.

On a grid, the cells i and j, whose coordinates are (x_i, y_i) and (x_j, y_j) respectively, are connected within the cluster c if it exists at least $|x_i - x_j| + |y_i - y_j| + 1$ cells of the same cluster in the rectangle they form on the grid:

$$(|x_i - x_j| + |y_i - y_j| + 1)\ b_{i,j,c} \leq \sum_{\substack{k \in [min(x_i,x_j),max(x_i,x_j)] \\ l \in [min(y_i,y_j),max(y_i,y_j)]}} b_{i,j,c} \quad \text{if } i \neq j$$
$$\forall i \in G, j \in G, c \in C \tag{8}$$

These constraints can be seen as working recursively. Two cells are connected in a cluster if and only if at least they each have one of their neighbor cells that are connected in the same cluster.

Note that these connectedness constraints are more restrictive than the commodity flow constraints since they impose that the path between two cells is strictly inside the rectangular they define on a grid.

IV. Graph-based Geo-clustering Algorithm

In this section, we present our graph-based algorithm for MEC area-geo-clustering. We first explain how it works and then present its formal description.

Given a maximum MEC server capacity, the algorithm finds MEC clusters (also referred to as MEC areas) which tend to maximize the traffic handled inside the clusters (i.e. at the edge by the MEC servers) and thus reduce the traffic that goes up to the core data center.

It is divided in two phases that are repeated iteratively. Assume that we start with two graphs that have the same set of nodes (see Fig. 2). These nodes correspond to the discretization of the area where the MEC communications demands are distributed into clusters. We note it C. The first graph $G_a = (C, E_a)$ represents the adjacencies of the nodes on the area. For instance, in a square grid, a node (a grid cell) has up to 8 adjacent nodes (grid cells). The second graph $G_{int} = (C, E_{int})$ represents the interactions (i.e., the communications or the traffic) between the nodes. The weight $w_{i,j} \in \mathbb{R}$ of the edge $e_{i,j} \in E_{int}$ represents the amount of interaction (e.g., the number of communications or traffic) between node i and node j. Note that a node i can have interaction with itself, leading to a self-loop $e_{i,i}$ - this is actually desired as it corresponds to communications that are inside the corresponding MEC cluster. So, in this initial partition there are as many MEC clusters as there are nodes.

Algorithm 1 Geo-clustering of MEC resources

Require: Graph of cluster adjacencies $G_a(C, E_a)$: undirected graph where C is the set of clusters
Graph of cluster interactions $G_{int}(C, E_{int})$: undirected edge-weighted graph where $E_a \subseteq E_{int}$ and $e_{i,j} \in E_{int},\ i, j \in C$ has a weight $w_{i,j} \in \mathbb{R}$
Maximum cluster capacity: M, meaning that $w_{i,j} \leq M$

Ensure: $G_a(C, E_a)$ and $G_{int}(C, E_{int})$

1: **repeat**
2: Select the two adjacent clusters that have the highest interaction weight:
$i, j \in C$ such that:
$\max_{\{e_{i,j} \in E_a\} \mid w_{i,i}+w_{i,j}+w_{j,j} \leq M} w_{i,j}$
3: Merge j with i in $G_a(C, E_a)$ {Update C and E_a}
4: Merge j with i in $G_{int}(C, E_{int})$: $w_{i,i} \leftarrow w_{i,i} + w_{i,j} + w_{j,j}$ {Update C and E_{int}}
5: **until** No adjacent clusters can be merged: $\forall e_{i,j} \in E_a, w_{i,i} + w_{i,j} + w_{j,j} \geq M$.

First, we consider all the edges in E_{int} and we select the edge $e_{i,j}$ that has the highest weight such that: i) node i and node j are neighbors in the area (i.e., $\exists\ e^a_{i,j} \in E_a$), ii) the amount of interaction between node i and node j is equal or lower than the maximum cluster capacity M (i.e., $w_{i,i} + w_{i,j} + w_{j,i} + w_{j,j} \leq M$). The selected edge corresponds to the best interaction reduction at this stage.

Secondly, we cluster node i and node j, updating the graphs G_a and G_{int} with a new node that represents their clustering. To do so, the neighbors of the new node in G_A and G_{int} are the former neighbors of node i and node j. The weights of the links between the new node and its neighbors in G_{int} are given by the sum of the weight of the links between the former node i and its neighbors and the former node j and its neighbors. Finally, the weight of the new self-loop corresponds to the sum of the two former self-loops plus the weights between node i and node j. Once this second phase (clustering) is completed, it is then possible to re-apply the first phase (selection) of the algorithm to the resulting graphs and to iterate (see Fig. 2).

By construction, the number of nodes (clusters) decreases at each pass. The passes are iterated until there are no more changes meaning that a local minimum of MEC cluster interaction is attained. The final result of our algorithm corresponds to a partition of the area into MEC clusters/areas whose load (self-loop weights) is inferior but close to the maximum MEC server capacity M. We present a formal description in Algorithm 1. Note that our algorithm can be used in an n-level MEC architecture, n designating the number of aggregation levels inbetween the base stations and the core (we consider only one level here: MEC servers), by applying it at each level.

The algorithm is reminiscent of the community detection algorithms in complex networks (e.g., Louvain algorithm [24]) in the way that it iteratively clusters nodes to increase a modularity (in our case function of the weight of the self-loops). However, it differs on several important points. First, it takes into account spatial properties between the nodes via the graph G_a, which constrains the clustering. Second, only two nodes are clustered at each iteration. Third, it considers a maximum clique/cluster capacity, that corresponds to the weight of the self-loops. Note that this threshold we introduce is adequate to our partitioning problem. However, it could be removed from our algorithm, making it a similar yet different hierarchical clustering algorithm than community detection algorithms. Finally, we purposely present a simple description of the algorithm, but heuristics may be introduced to improve its performance or introduce variants (e.g., order the edges in the first phase, consider more complex interactions such as group communications, perform a local search on the final result, consider different maximum cluster capacities etc.).

The time complexity of the algorithm described as above is $O(N.|E_{int}|)$, where $|E_{int}|$ is the number of interaction edges and N the number of vertices (i.e., the initial cardinality of the cluster set C) issued from the area discretization. Indeed, the first phase basically consists in going through all the $|E_{int}|$ edges of G_{int} and finding the non self-loop edge with the maximum weight. The second phase consists in adding and removing (i.e., clustering) nodes in graphs (G_a and G_{int}). With adjacency lists, you simply need to iterate over the edge list of the nodes to be clustered and update all those nodes. The algorithm stops when no pair of nodes can be merged anymore, which means maximum $N-1$ passes are done. The number of edges $|E_{int}|$ depends on the nature of the graph. For example, it averages $k.(N-1)/2$ in an Erdös-Rényi graph [25], where $k > 1$ is the mean vertex degree. The complexity of our algorithm would thus be $O(N^2)$. Note that it is a pessimistic upper bound since at each pass both the number of edges and vertices, and thus the number of operations, decrease.

V. Evaluation and analysis

In this section, we evaluate our MEC clustering algorithm with a real dataset of mobile communications. We first compare it to the model on small problem instances (i.e., coarse-grained area discretization) considering different day types and different periods of the day. Then, we evaluate it on large problem instances (i.e., fine-grained area discretization). Finally, we analyze its results through time.

A. Dataset

To evaluate our algorithm and show the benefits of the MEC approach compared to a classic centralized architecture, we use the dataset published as Open Data by Telecom Italia in 2014 [26].

This dataset contains geo-referenced Call Detail Records (CDRs) over the city of Milan from November 1st, 2013 to January 1st, 2014 [27]. During this period, every time a mobile user engaged a telecommunication interaction with another mobile user in the region, a CDR was created containing the date time of the call and the geographical locations of the mobile users (derived from the location of the base stations they used). The dataset was created combining all this anonymous information, with a temporal aggregation of time slots of ten minutes and a spatial aggregation of square grid calls of 235x235 meters (a grid of 100x100 cells to cover the city of Milan). The number of records in the dataset $S'_i(t)$

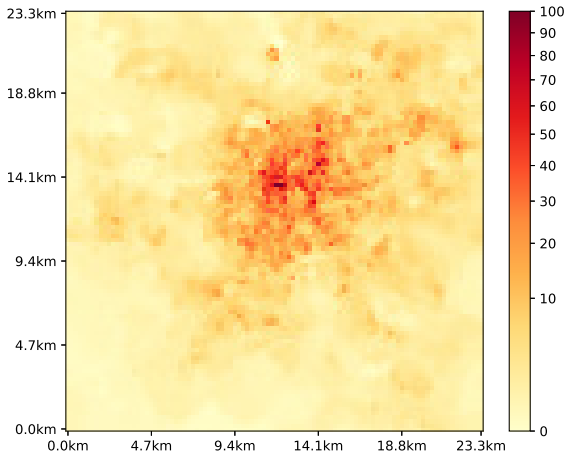


Fig. 3. Normalized mobile communication intensity in the city of Milan (5pm-6pm, 11/04/2013). The communications are concentrated in the city center.

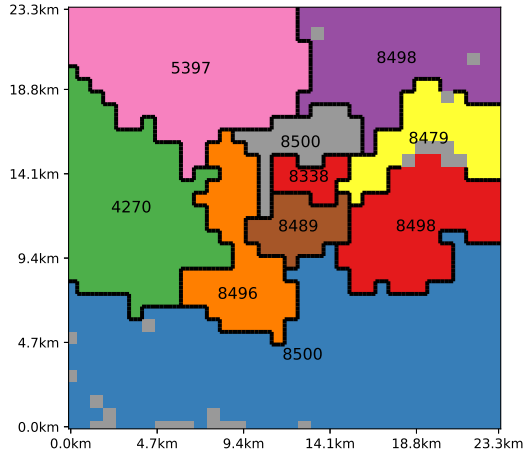


Fig. 4. Clustering result obtained with our algorithm for a maximum cluster capacity of 5% of the total communications, i.e., 8,500 communications (5pm-6pm, 11/04/2013). The numbers in the clusters correspond to their load.

in a grid square i at time t follows the rule: $S'_i(t) = k.S_i(t)$ where k is a constant defined by Telecom Italia. It aims at hiding the true number of calls, this information being business confidential. Since Telecom Italia only possesses the data of its own customers, the computed interactions are only between them. This means that (at most) 34% of population's data was collected, due to Telecom Italia's market share [27]. Around 1,3 million people live in the city of Milan.

To evaluate our algorithm, we generate from this anonymized dataset batches of communications aggregated by hour and spatially distributed over the city of Milan. We first consider a batch of 170,000 communications, which corresponds to roughly 13% of Milan's population, to study MEC resources partitioning (see Sec. V-B and Sec. V-C). Then, we use hourly batches for a whole week. The volume of the batches is proportional to what was measured by Telecom Italia, considering that at the peak hour of the week there is 170,000 communications (see Sec. V-D).

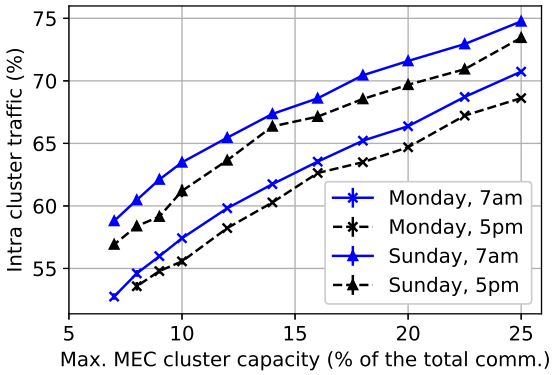
Fig. 3 shows the normalized mobile communication intensity distributed in the city of Milan between 5pm and 6pm during a working day (Monday). We can clearly distinguish the city center, which gathers most of the mobile calls. Fig. 4, further discussed in this section, illustrates a clustering result.

B. MEC resources partitioning: comparing our algorithm with the MILP

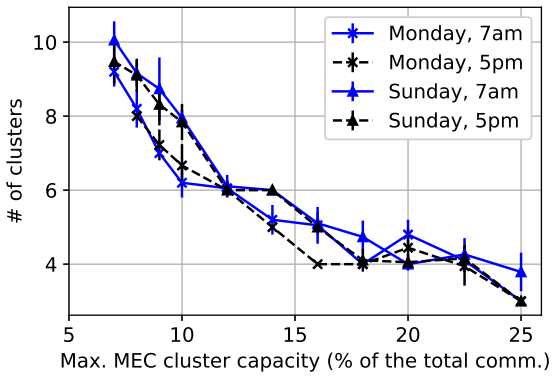
We evaluate the MILP with the alternative connectedness constraints. Since it is computationally hard, we use a coarse-grained area discretization (a 6 x 6 grid) to have small problem instances. We implemented the MILP in Python 3.5 using IBM ILOG CPLEX 12.7 [28]. Our experiments ran on an Intel Core i7-2620M CPU @ 2.70GHz x 4 workstation with 7.8 GiB of RAM and Ubuntu 14.04 64-bit.

Core offloading.

We first analyze the benefits of the MEC approach with respect to core offloading, that is we look at which portion of the communications is directly served at the edge. As the



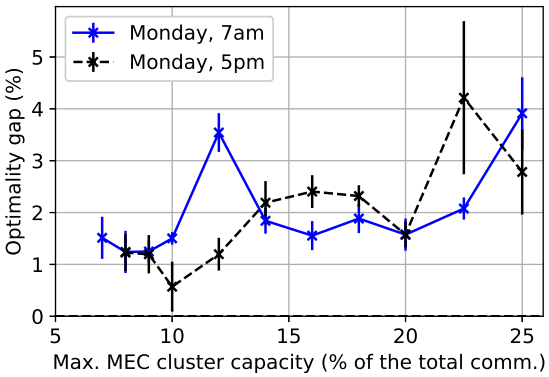
(a) Proportion of intra MEC cluster traffic.



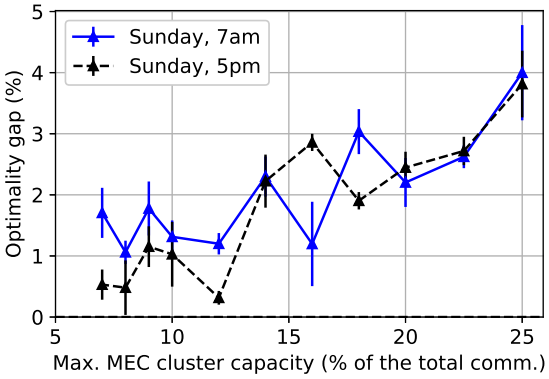
(b) Number of MEC clusters.

Fig. 5. MILP results: core offloading and number of clusters (Monday 11/04/2013 and Sunday 11/10/2013).

communication demands are spatially distributed in time and space according to human activity (residential, business, entertainment, transport etc.), we consider a working day (Monday 11/04/2013) and a weekend day (Sunday 11/10/2103) at two periods: beginning of the activities (7am to 8am) and communication peak (5pm to 6pm).

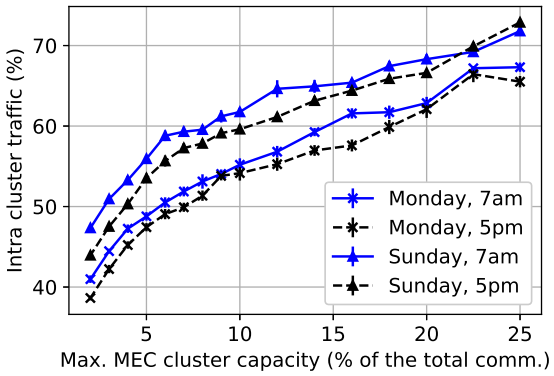


(a) Monday 11/04/2013.

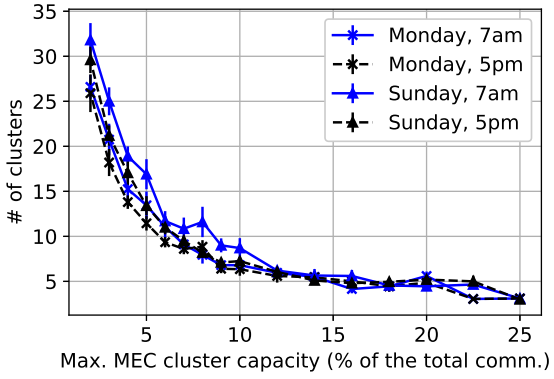


(b) Sunday 11/10/2013.

Fig. 6. Optimality gap (intra MEC cluster traffic ratio) for two different day types.



(a) Proportion of intra MEC cluster traffic.



(b) Number of MEC clusters.

Fig. 7. Geo-clustering algorithm results: core offloading and number of clusters (Monday 11/04/2013 and Sunday 11/10/2013).

Fig. 5a shows the the amount of communications directly handled at the MEC servers in function of the maximum MEC server capacity (expressed in percentage of the total communications to serve). 170,000 communications, derived from the dataset, were considered at each experiment; each point corresponds to 20 experiments. We can observe that an important part of the communications does not have to go up to the core. For example, with a maximum cluster capacity of 10%, which represents 17,000 communications, between 55% and 64% of the traffic is directly absorbed by the MEC servers. We can also observe that the gain varies according to the day and the time of the day. The lower gains are at the peak hour of the working day, while the upper gains are at the beginning of the weekend day. These observations can be explained by i) the spatial locality of the mobile communications and ii) the difference of human activities (mainly business and transportation on Monday at 5pm and residential on Sunday at 7am).

Fig. 5b presents the corresponding number of MEC servers. There is no major difference. Naturally, as the maximum cluster capacity increases the number of clusters diminishes to serve traffic at the edge. Note that with a spatial uniform distribution of the communications, we would have for instance 10 servers, instead of 6-8, for a maximum capacity of 10%.

We then compare our algorithm with the MILP under the same evaluation setup. Fig. 6a and 6b show the optimality gap, which is the percentage of difference between the intra cluster traffic ratio provided by the MILP and our geo-clustering algorithm, for a business day and a weekend day respectively. There is no observable impact of the day and daytime on the efficiency of the algorithm. However, when the maximum MEC cluster capacity is small, the results of the algorithm are close to optimal. The gap tends to increase with the maximum MEC cluster capacity as there are fewer clusters.

The difference in the number of clusters was on average +0.3. The mean execution time of the MILP was around two hours, while the algorithm was executed in approximately 0.26 seconds.

C. MEC resources partitioning: large instances

We then analyze MEC resources partitioning on large problem instances to have higher resolution and full MEC server capacity range. In the rest of this section, the experiments were conducted using the geo-clustering algorithm with a 33 x 33 discretization grid.

Geo-clustering.

We first illustrate the result of the algorithm. Fig. 4 presents the results of the geo-clustering algorithm on Monday 11/04/2013 between 5pm and 6pm with a maximum cluster capacity of 5% of the total communications, that represents maximum 8,500 communications. The algorithm started with

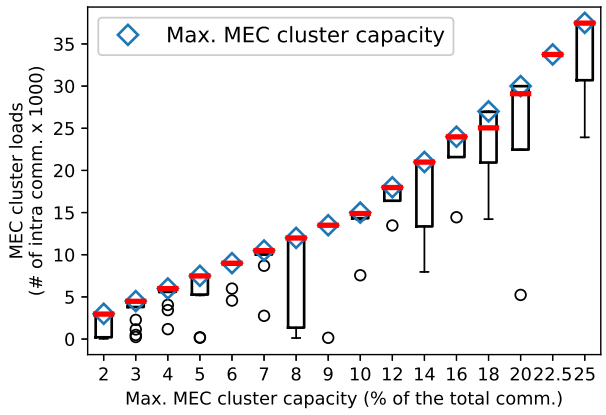


Fig. 8. MEC cluster loads (5pm-6pm, 11/04/2013).

1,089 clusters (33 x 33) and took, without any code optimization, 18.73 seconds to provide this result. The grey atomic squares correspond to grid cells with no or very low traffic. However, they could not be ultimately merged to a neighbor cluster because it would have increased their load above the maximum threshold. We thus consider that if their intra-cluster traffic is lower than 0.01% of the maximum capacity, they do not form a cluster and their communications are directly served in the core. We can see that the area of the clusters that cover the city center is smaller than the ones that serve low-density regions, the density of communications being higher there (see Fig. 3). Moreover, as seen in Fig. 8, most of the loads are close to the maximum server capacity. The MEC areas are thus well balanced.

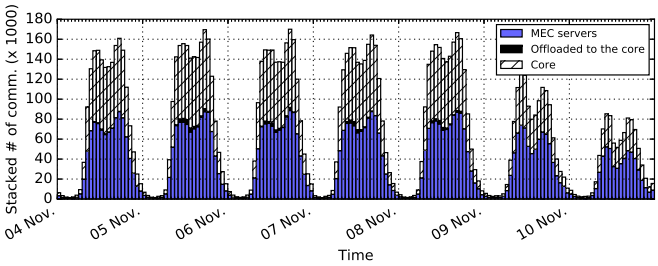
Core offloading.

Fig. 7a shows the benefits of the MEC approach with respect to core offloading. 170,000 communications, derived from the dataset, were considered at each experiment; each point corresponds to 20 experiments. The observations confirm that an important part of the communications can be handled at the edge, at the MEC servers, instead of going up to the core and that the ratio of traffic depends on the day type and daytime. The core offloading remains important even for very small cluster capacities.

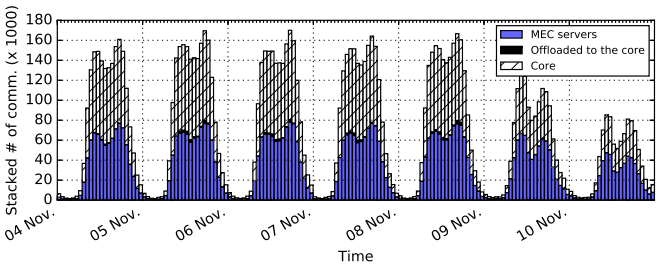
Fig. 7b presents the corresponding number of MEC clusters and hence MEC servers. We can see that naturally the number of clusters increases rapidly as the maximum cluster capacity diminishes. These results suggest that a trade-off has to be found between the ratio of traffic handled at the edge and the number of MEC servers to deploy.

Server load balancing.

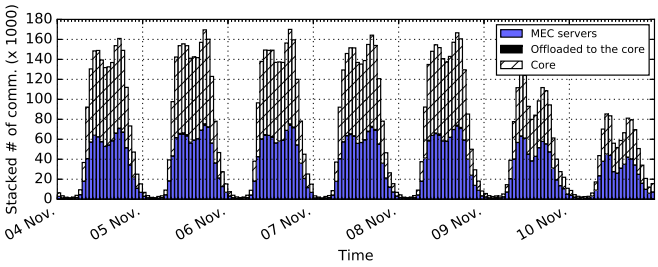
Fig. 8 shows the loads of the clusters at the peak hour on Monday. We can observe that the partition, and hence the load, is well balanced. Indeed, most of the clusters have a load close to the maximum cluster capacity. Moreover, in all cases except two (18% and 20%), the median values almost match the maximum values. We had the same observations for the other periods we evaluated.



(a) Partition done at 5pm-6pm on Monday 11/04/2013 with a maximum cluster capacity of 10% of the total communications.



(b) Partition done at 5pm-6pm on Monday 11/04/2013 with a maximum cluster capacity of 5% of the total communications.



(c) Partition done at 7am-8am on Monday 11/04/2013 with a maximum cluster capacity of 5% of the total communications.

Fig. 9. MEC servers and core traffic distributions over a week for different partitions.

D. Through time

Core offloading.

We finally evaluate the performance of our algorithm through time. To this aim, we first use the mobile data on a full week. We consider that at the peak hour of this period (Thursday, 5pm, 11/08/2013), there are 170,000 communications per hour. It represents more or less the volume of communications in the city of Milan for the market share of Telecom Italia in 2013. We retrieved from the dataset the proportion of communications hour by hour and their spatial distribution. We then considered three partitions obtained on Monday 11/04/2013 at different hours (7am-8am and 5pm-6pm) of the day and with different maximum cluster capacity (5% and 10% of the total communications at this period of the day).

In Fig. 9a, we can observe that around 53% of the communications are directly handled by the MEC servers during the working days. This share increases up to 61% during the week-end. Obviously, if we consider a maximum cluster capacity of 5% (Fig. 9b), the global load distribution through

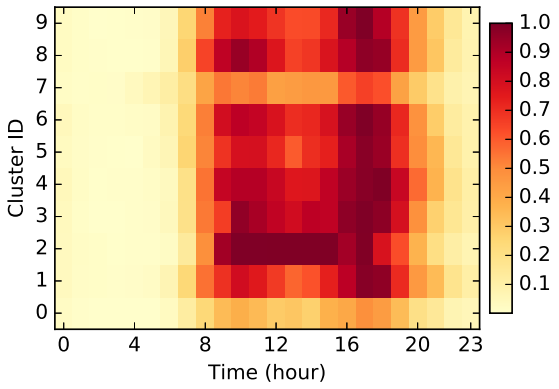


Fig. 10. Normalized MEC cluster loads over a day (11/04/2013) with a partition done at 5pm and a maximum cluster capacity of 5% of the total communications.

time remains the same, but the traffic share of the MEC servers drops to approximately 45%. In both cases, we can notice that the traffic offloaded to the core, that corresponds to the local traffic that could not be handled by MEC servers because they were saturated, is very small (maximum 3.1% on 11/05/2013 at 12am). Finally, we can notice that, if we consider a partition done in the morning (Fig. 9c) instead of the peak hour (Fig. 9b), the share of the MEC servers slightly decreases, while they remain almost unsaturated.

Server load balancing.

At last, Fig. 10 presents the distribution of the MEC server loads. The partition corresponds to the one shown on Fig. 4. It was done with the communications that occurred between 5pm and 6pm. We can distinguish human activities, that is low activity until 8am and after 9pm and two peaks around 10 am and 5pm. At each hour, the load is homogeneously distributed on the servers, which means that the MEC areas are well dimensioned and that all users experience good QoE.

VI. Conclusion and Perspectives

MEC is a key technology to support the ever-increasing growth of communication capability demands and realize the IoT and 5G visions. As operators are transforming their network architectures and are looking for deploying computation resources close to the users to improve QoE, it is necessary to adequately dimension MEC systems. In this paper, we formulated this problem as a mixed integer linear program and presented a graph-based algorithm that enable finding a partition of MEC areas that consolidates traffic at the edge, in MEC servers. We evaluated them using a real world dataset from a mobile operator. The evaluation results, beyond quantifying the benefits of the MEC approach, show that the core can be largely offloaded. They also show that the algorithm provides MEC areas that are well balanced in terms of load and close to optimal. Finally, we ran simulations over one week of communications and observed that there is almost no saturation of the MEC servers, while the traffic on the core is largely reduced. In future work, we expect to explore several aspects such as group communications, energy saving (in particular with respect to the temporal distribution of the demand) and latency. We also aim at combining our approach with online application offloading and migration to support micro-mobility.

References

[1] European Telecommunications Standards Institute (ETSI), *Mobile-Edge Computing (MEC); Service Scenarios (GS MEC-IEG 004)*, November 2015.

[2] OpenFog Consortium Architecture Working Group, *OpenFog Architecture Overview (OPFWP001.0216)*, February 2016.

[3] Y. C. Hu, M. Patel, D. Sabella, and et al., *Mobile Edge Computing A key technology towards 5G (ETSI White Paper No. 11)*, European Telecommunications Standards Institute (ETSI), September 2015.

[4] P. Ventre, C. Pisa, S. Salsano, and et al., "Performance evaluation and tuning of virtual infrastructure managers for (micro) virtual network functions," in *Proceedings of IEEE NFV-SDN Conference*, 2016.

[5] R. Saunders, J. Cho, A. Banerjee, and et al., "P2P offloading in mobile networks using SDN," in *Proceedings of ACM Symposium on SDN Research (SOSR)*, 2016.

[6] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," in *Proceedings of IEEE INFOCOM*. IEEE, 2016, pp. 1–9.

[7] H. Wang, F. Xu, Y. Li, P. Zhang, and D. Jin, "Understanding mobile traffic patterns of large scale cellular towers in urban environment," in *Proceedings of ACM Internet Measurement Conference (IMC)*, 2015.

[8] D. Naboulsi, M. Fiore, S. Ribot, and R. Stanica, "Large-scale mobile traffic analysis: a survey," *IEEE Communications Surveys and Tutorials*, 2015.

[9] N. Tastevin and M. Bouet, "Characterizing and modeling the distance of mobile calls: A metropolitan case study," in *Proceedings of IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Sept 2016.

[10] Z. A. Qazi, P. Krishna, V. Sekar, and et al., "KLEIN: A Minimally Disruptive Design for an Elastic Cellular Core," in *Proceedings of ACM Symposium on SDN Research (SOSR)*, 2016.

[11] M. Bouet and V. Conan, "Geo-partitioning of MEC resources," in *Proceedings of the ACM SIGCOMM Workshop on Mobile Edge Communications (MECOMM'2017)*, 2017, pp. 43–48.

[12] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[13] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[14] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys and Tutorials*, vol. PP, no. 19, 2017.

[15] A. Ceselli, M. Premoli, and S. Secci, "Mobile edge cloud network design optimization," *IEEE/ACM Transactions on Networking*, no. 99, pp. 1–14, 2017.

[16] Q. Xu, F. Qian, J. Huang, A. Gerber, and et al., "Cellular data network infrastructure characterization and implication on mobile content placement," in *Proceedings of ACM SIGMETRICS*, 2011.

[17] C. Chang, K. Alexandris, N. Nikaein, K. Katsalis, and T. Spyropoulos, "MEC architectural implications for LTE/LTE-A networks," in *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture (MobiArch)*, 2016, pp. 13–18.

[18] F. Manco, J. Martins, K. Yasukata, and et al., "The case for the superfluid cloud," in *Proceedings of USENIX HotCloud Workshop*, 2015.

[19] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM]-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[20] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 22–28, 2016.

[21] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2016, pp. 1–9.

[22] H. Tan, Z. Han, X. Li, and F. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2017, pp. 1–9.

[23] S. Vicente, V. Kolmogorov, and C. Rother, "Graph cut based image segmentation with connectivity priors," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[24] V. Blondel, J. Guillaume, R. Lambiotte, and E. Mech, "Fast unfolding of communities in large networks," *J. Stat. Mech*, 2008.
[25] P. Erdos and A. Rényi, "On the evolution of random graphs," *Bull. Inst. Internat. Statist*, vol. 38, no. 4, pp. 343–347, 1961.
[26] "Open Big Data," https://dandelion.eu/datamine/open-big-data/, 2014, accessed: 2017-03-23.
[27] G. Barlacchi, M. De Nadai, R. Larcher, and et al., "A multi-source dataset of urban life in the city of Milan and the Province of Trentino," *Scientific Data*, vol. 2, no. 150055, 2015.
[28] "IBM ILOG CPLEX Optimizer," http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, 2016.

Mathieu Bouet is a research expert in networking and communications at Thales, France. He received a Ph.D degree in computer science and an Habilitation degree (HDR) from Sorbonne University (previously UPMC - Paris VI) in 2009 and 2017 respectively. He currently manages research activities on network softwarization in the networking laboratory of the advanced studies department at Thales in Gennevilliers. His research interests are mainly: network virtualization and network optimization.

Vania Conan is a senior research expert in networking and communications at Thales, France. He received an Engineering degree (1990) and PhD in computer science (1996) from Mines ParisTech, and an Habilitation degree from Sorbonne University, Paris (2012). He is presently head of the networking laboratory in the Advanced Studies department at Thales in Gennevilliers. In the past years he has been conducting research in the fields of software defined communications and wireless networking. He has published over 100 international conference and journal papers and holds 10 patents in networking technologies. His current research topics include mobile network protocols and virtualized network design.