Multi-Objective Mobile Edge Provisioning in Small Cell Clouds

Vincenzo De Maio
Vienna University of Technology
Vienna, Austria
vincenzo@ec.tuwien.ac.at

Ivona Brandic
Vienna University of Technology
Vienna, Austria
ivona@ec.tuwien.ac.at

ABSTRACT

In recent years, Mobile Cloud Computing (MCC) has been proposed as a solution to enhance the capabilities of user equipment (UE), such as smartphones, tablets and laptops. However, offloading to conventional Cloud introduces significant execution delays that are inconvenient in case of near real-time applications. Mobile Edge Computing (MEC) has been proposed as a solution to this problem. MEC brings computational and storage resources closer to the UE, enabling to offload near real-time applications from the UE while meeting strict latency requirements. However, it is very difficult for Edge providers to determine how many Edge nodes are required to provide MEC services, in order to guarantee a high QoS and to maximize their profit. In this paper, we investigate the static provisioning of Edge nodes in a area representing a cellular network in order to guarantee the required QoS to the user without affecting providers' profits. First, we design a model for MEC offloading considering user satisfaction and provider's costs. Then, we design a simulation framework based on this model. Finally, we design a multi-objective algorithm to identify a deployment solution that is a trade-off between user satisfaction and provider profit. Results show that our algorithm can guarantee a user satisfaction above 80%, with a profit for the provider of up 4 times their cost.

ACM Reference Format:
Vincenzo De Maio and Ivona Brandic. 2019. Multi-Objective Mobile Edge Provisioning in Small Cell Clouds. In *Tenth ACM/SPEC International Conference on Performance Engineering (ICPE '19), April 7–11, 2019, Mumbai, India.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3297663.3310301

1 INTRODUCTION

Mobile Cloud Computing (MCC) attracted significant interest in the scientific community as a technique to enhance mobile devices' capabilities. MCC works by offloading computation from the UE to remote Cloud resources. However, according to [7], MCC can significantly increase the response time of mobile applications, because of the geographical distance between UE and Cloud resources. Such latency can be a huge problem in case of applications with near real-time requirements.

To this end, Mobile Edge Computing (MEC) has been proposed as a solution. MEC relies on Edge infrastructures, that enhance Cloud capabilities by using Edge nodes, geographically closer to the mobile devices. By offloading computation to Edge nodes, UE is able to overcome high latency problems faced in MCC. This way, they achieve the advantages of computation offloading without affecting users' satisfaction.

MEC has been described in [5]. There are several architectural models for MEC described in literature, such as Small Cell Cloud (SCC) [29], Mobile Micro Cloud (MMC) [45], Follow Me Cloud (FMC) [44] and CONCERT [28]. However, in each one of the aforementioned architectural models, it is very difficult for Edge providers to plan (1) the geographical locations where Edge nodes should be deployed and (2) how many nodes are necessary to cover a given area in order to guarantee a high level of user satisfaction. These factors have a strong effect on providers' profits. Typical methods for Cloud provisioning [43] are not suitable for this task, as parameters such as geographical proximity to the user, that are of paramount importance for Edge, are not considered in the Cloud context. Therefore, it is necessary to provide a method to identify a deployment plan for Edge nodes, providing the required QoS to the users and maximize providers' profit at the same time.

In this paper, we design a static provisioning method to find the location and the number of Edge nodes in a MEC infrastructure. The proposed method aims at finding a trade-off solution between users' satisfaction and providers' profit. First, we define models for (1) user satisfaction, considering application response time and user cost as parameters; (2) providers' cost, considering electricity costs for running a MEC infrastructure composed of geographically distributed Cloud data centers and Edge nodes located in proximity of the users. Afterwards, we define a simulation framework, based on real-world traces of electricity prices and mobile workloads, to evaluate different provisioning strategies for MEC. Then, we design a multi-objective algorithm to identify a trade-off solution between user satisfaction and providers' cost. Finally, we evaluate our algorithm in comparison with other three provisioning strategies using the proposed simulation framework.

This work focuses on SCC architecture for MEC, that is used also in european projects like SESAME[1]. Mobile applications are simulated generating DAG application models and combined according to real-world usage data of mobile applications. Evaluation employs Monte-Carlo simulation, that allows to accurately model the variability of remote infrastructure.

Results show that our Multi-Objective algorithm can achieve (1) a user satisfaction around 80% in a MEC scenario for different pricing models and (2) a profit for the provider up to 4 times more the providers' energy costs, depending on (1) pricing model, (2) user requirements and (3) the number of UE.

The paper is organized as follows: first, we provide a background on MEC and SCC in Section 2. Then, we define the theoretical model

[1] https://5g-ppp.eu/sesame/

ICPE '19, April 7–11, 2019, Mumbai, India

ACM ISBN 978-1-4503-6239-9/19/04…$15.00
https://doi.org/10.1145/3297663.3310301

of our work in Section 3 and our offloading algorithms in Section 4. In Section 5 we define our experimental setup, while in Section 6 we present the results of our experiments. Related work is described in Section 7 and our work is concluded in Section 8.

2 BACKGROUND

2.1 Mobile workload

A mobile workload is composed of an unbounded number of mobile applications that are executed sequentially by mobile users. Mobile applications are composed of different interdependent tasks, each one with different requirements. Dependencies and task requirements can affect task offloading: for example, task could not be offloaded since it needs specific devices, not available on the remote infrastructure (e.g. camera or GPS), or because task execution depends on another task's results [13].

2.2 Small Cell Cloud (SCC)

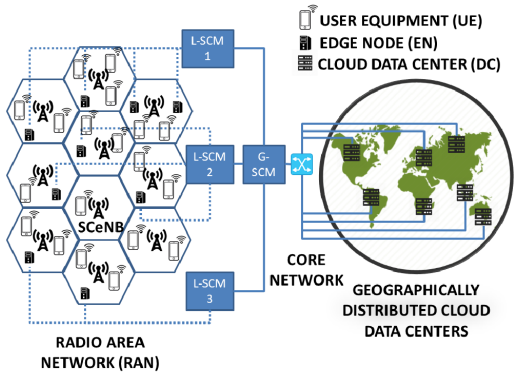


Figure 1: SCC overview

Among different proposals for integrating Cloud/Edge capabilities into mobile network architecture [28, 29, 44, 45], we select a Small Cell Cloud architecture (SCC), due to the higher maturity of the standard and the number of research projects involving this type of architectures, like SESAME[2]. The envisioned SCC architecture is summarized in Figure 1. SCC is based on the concept of Small Cell Radio Access Networks (RAN), used in many types of modern cellular networks such as LTE, featuring short-range (up to 2 kilometers) small cell base stations (SCeNBs). The main reason for employing small cells is to boost the spectral efficiency of modern cellular networks thanks to a shorter distance between the user equipment and the base station. In SCC architecture, the capabilities of SCeNBs are improved by adding Edge nodes. Such nodes will be used to deliver Edge services to end users. To facilitate integration of computational resources in this context, a new entity called Small Cell Manager (SCM) is introduced at each base station. The SCM is responsible of managing the resources of the SCeNBs, performing dynamic and elastic management within the SCC. There are two different ways of deploying SCM: *centralized* and *distributed/hierarchical*. Although the deployment of centralized SCM is easier, we also want to include into this module the management of Cloud resources, as we show in Figure 1. For this reason, having a single node managing both Edge and Cloud resources might result in a bottleneck for the whole architecture. For this reason, we focus on distributed/hierarchical architecture for this work. We consider then two types of SCM: the local SCM (L-SCM), responsible for managing resources at the level of SCeNBs or of clusters of SCeNBs, while the global SCM (G-SCM) is responsible for resource management between different SCeNBs clusters and different Cloud resources. In Figure 2 we present an overview of the functioning of a L-SCM: first, it collects data about the structure of the applications running on the UEs, as well as the requirements of each task. At the same time, it collects from the SCeNBs it is managing data concerning (1) the current load of each Edge node and (2) other environmental parameters, such as local electricity prices. It also collects data from the SCC to prepare offloading of tasks to other L-SCMs or to the Cloud, if necessary. From application structure and requirements a new data structure, named mobile workload, is generated. The mobile workload contains all the data relative to the mobile applications that are executed in the SCeNBs managed by the L-SCM. Using these data, the L-SCM computes (1) the mobile workload scheduling, containing the scheduling of single application tasks on UEs, Edge and Cloud nodes and (2) the local cells management plan for its SCeNBs, containing infrastructure management instructions. In this work, we focus on the static provisioning of Edge nodes. We determine the location of the nodes and how many of them are needed to cover a given urban area. We also simulate scheduling of mobile workload on the given MEC infrastructure to evaluate the quality of the proposed provisioning.

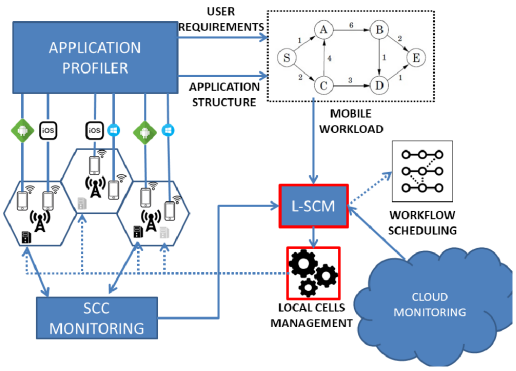


Figure 2: L-SCM Overview

3 MODEL

3.1 Application model

First of all, we define mobile applications. Mobile applications are software artifacts that can be executed by UE. An application $\mathcal{A}$ is defined as a set of interdependent task represented by a DAG (Directed Acyclic Graph), whose nodes are the tasks and edges model dependencies between them. Namely,

[2]https://5g-ppp.eu/sesame/

Definition 1. *$\mathcal{A} \stackrel{\text{def}}{=} (\mathcal{T}_{\mathcal{A}}, \mathcal{L}_{\mathcal{A}})$, where $\mathcal{T}_{\mathcal{A}}$ is the set of tasks of the application $\mathcal{A}$ and $\mathcal{L}_{\mathcal{A}}$ is the set of edges connecting tasks in $\mathcal{T}_{\mathcal{A}}$.*

For each task t_i and edge l_{ij} we define the vector of the demand

$$\vec{d}(t_i) \stackrel{\text{def}}{=} \langle \texttt{SIZE}(t_i), \texttt{CPU}(t_i), \texttt{DATA}_{in}(t_i), \texttt{DATA}_{out}(t_i), \texttt{OL}(t_i) \rangle, \quad (1)$$

respectively, the size of the task t_i in millions of instruction, the number of CPUs required by t_i, the size of input/output data and a boolean value defining whether the task can be offloaded or not. Concerning edges, we define

$$\vec{d}(l_{ij}) \stackrel{\text{def}}{=} \langle \texttt{l}(l_{ij}), \texttt{b}(l_{ij}) \rangle \quad (2)$$

as, respectively, the maximum latency demand between the two tasks t_i and t_j and the minimum bandwidth requirements between t_i and t_j.

3.2 Workload model

After defining application, we introduce the concept of mobile workload. A mobile workload is a set of mobile applications $\{\mathcal{A}_0, \ldots, \mathcal{A}_n\}$ that are executed sequentially. To execute sequentially two applications $\mathcal{A}_0$ and $\mathcal{A}_1$ we need to join the applications' DAGs. To do this, we employ the *join* operator $\bowtie$. The join $\mathcal{A}_0 \bowtie \mathcal{A}_1$ gives a new DAG $\mathcal{A}' = (\mathcal{T}_{A_0} \cup \mathcal{T}_{A_1}, \mathcal{L}_{A_0} \cup \mathcal{L}_{A_1} \cup \mathcal{T}_{A_0} \times \mathcal{T}_{A_1})$. The edges added by the join operation are used only to preserve the execution order and do not affect the normal application execution. Therefore, we assume without loss of generality that

$$\forall l_{ij} \in \mathcal{T}_{\mathcal{A}_0} \times \mathcal{T}_{\mathcal{A}_1}, \vec{d}(l_{ij}) \stackrel{\text{def}}{=} \langle \infty, 0 \rangle.$$

The join operator is *associative* and its neutral element is $\emptyset$.

Definition 2. *A mobile workload for a user equipment ue, $w(ue)$, is defined as (1) a single application $\mathcal{A}$ or (2) the join of more application, formally:*

- $w(ue) \stackrel{\text{def}}{=} \mathcal{A}$;
- $w(ue) \stackrel{\text{def}}{=} \mathcal{A}_0 \bowtie \mathcal{A}_1 \bowtie \ldots \bowtie \mathcal{A}_n$.

By the definition of $w(ue)$ and the properties of $\bowtie$ follows that the join of two workloads is also a workload.

3.3 Map model

In this section we define the map of the area where computational and mobile nodes are distributed. Concerning the urban area, where Edge nodes are distributed, we model it as a square hexagonal grid, as done by many works in mobile cellular networks [20]. The map $\mathcal{M}(n)$ is defined as a $m \times m, m, n \in \mathbb{N}$ hexagonal grid. We employ a doubled coordinates system to identify each cell in the grid. Therefore, we define X coordinates as $X = \{x \in 2\mathbb{N} : x < 2m\}$, and Y coordinates as $Y = \{y \in N : y < n\}$ as in Figure 3. This coordinate system allows to easily identify each cell using two (x, y) coordinates. Each hexagonal cell models a SCeNBs in the RAN. We assume for simplicity that there is at most one Edge node per SCeNBs. UEs can access any Edge node in a number of hops that is equal to the distance between two SCeNB. To calculate the

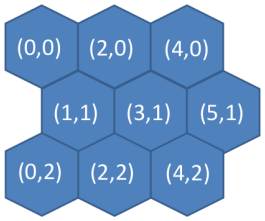


Figure 3: 3×3 hexagonal grid with doubled coordinates.

distance between a UE and Edge node, we employ Equation 3.

$$dist((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + \max(0, \frac{|x_1 - x_2| - |y_1 - y_2|}{2}). \quad (3)$$

Concerning Cloud data centers, they are geographically distributed around the globe and are reachable from the UEs by considering the latency of the Internet between the UE's location and the site of the Cloud data center, $\texttt{loc}(c_i)$. These values are defined in Section 5.

3.4 Infrastructure model

The type of infrastructure that we define in this work is a Mobile Edge Computing (MEC) infrastructure.

Definition 3. *We define a MEC infrastructure as a set $\mathcal{I} = \{\mathcal{M}_{\mathcal{I}}, \mathcal{N}_{\mathcal{I}}, \mathcal{L}_{\mathcal{I}}\}$, where $\mathcal{M}_{\mathcal{I}}$ is the map of the area where computational nodes are deployed, $\mathcal{N}_{\mathcal{I}}$ is the set of computational nodes and $\mathcal{L}_{\mathcal{I}}$ the network connections between nodes.*

$\mathcal{N}_{\mathcal{I}}$ is defined as $\{\mathcal{D}_{\mathcal{I}} \cup \mathcal{E}_{\mathcal{I}} \cup C_{\mathcal{I}}\}$, where $\mathcal{D}_{\mathcal{I}}$ is the set of User Equipments (UE)s, $\mathcal{E}_{\mathcal{I}}$ the set of Edge nodes and $C_{\mathcal{I}}$ the set of cloud nodes. We define each $ue_i \in \mathcal{D}_{\mathcal{I}}$ as ue_i and its resources vector $\vec{r}(ue_i) \stackrel{\text{def}}{=} \langle \texttt{CPU}(ue_i), \texttt{MIPS}(ue_i), \texttt{BATTERY}(ue_i), \texttt{coords}(ue_i), w(ue_i) \rangle$, where $\texttt{CPU}(ue_i)$ is the number of CPUs in the UE, $\texttt{MIPS}(ue_i)$ represents the millions of instruction per second that each CPU of ue can execute, $\texttt{BATTERY}(ue_i)$ the energy available in UE's battery, $\texttt{coords}(ue_i)$ the coordinates of the UE on map $\mathcal{M}(n)$ and $w(ue_i)$ is the mobile workload of ue_i. Concerning Edge nodes, their main characteristics are (1) a limited amount of resource compared to the cloud nodes, and (2) geographical proximity to the user, in this case, the UEs [7]. Therefore, we define the resource vector $\vec{r}(e_i) \stackrel{\text{def}}{=} \langle \texttt{CPU}(e_i), \texttt{MIPS}(e_i), \texttt{coords}(e_i), p_e(e_i, \ldots), c_e(e_i, \ldots) \rangle$ for each $e_i \in \mathcal{E}_{\mathcal{I}}$, where $\texttt{CPU}(e_i)$ is the number of CPUs available on the nodes, $\texttt{MIPS}(e_i)$ represents the millions of instruction per second that each core in e_i can execute, $\texttt{coords}(e_i)$ the coordinates of the Edge nodes on map $\mathcal{M}(m, n)$, $p_e(e_i)$ the price penalty for execution on edge node e_i (as in [31]) and c_e the cost of electricity source of Edge node. We define a Cloud node $c_i \in C_{\mathcal{I}}$ and its resource vector as $\vec{r}(c_i) \stackrel{\text{def}}{=} \langle \texttt{CPU}(c_i), \texttt{MIPS}(c_i), p_c(c_i, \ldots), c_c(c_i, \ldots), \texttt{loc}(c_i) \rangle$, where $\texttt{CPU}(c_i)$ is the number of cores available on the node, $\texttt{MIPS}(c_i)$ is the million of instruction per second that c_i can execute, $p(c_i)$ is the pricing function for using Cloud resources on node c_i (as defined in [31]) and c_c the cost of electricity source available on the cloud node. Regarding $\mathcal{L}_{\mathcal{I}}$, $\mathcal{L}_{\mathcal{I}} = \mathcal{N}_{\mathcal{I}} \times \mathcal{N}_{\mathcal{I}}$. For each link $l_{ij} \in \mathcal{L}_{\mathcal{I}}$ we define the latency between n_i and n_j as $\texttt{l}(n_i, n_j)$,

ICPE '19, April 7–11, 2019, Mumbai, India Vincenzo De Maio and Ivona Brandic

where $\mathtt{l}(n_i, n_i) = 0$, and the bandwidth of the link $\mathtt{b}(n_i, n_j)$, where $\mathtt{bw}(n_i, n_i) = \infty$.

3.5 UE Battery Lifetime Model

We define UE battery lifetime as the amount of energy left on *ue* at a given time instant. Each time a new task t_i is assigned to a ue, the battery value is updated as in:

$$BATTERY(ue, \tau) = \mathtt{BATTERY}(ue, \tau - 1) - E_{ue}(ue, t_i, \tau). \quad (4)$$

where E_d is the energy, defined as the integral of the instantaneous power over time. Therefore, we define energy consumption on the UE *ue* for task t_i.

$$E_{ue}(ue, t_i) = \int_0^{\tau_{end}(t_i)} \sum_{t_i \in \Delta(ue,\tau)} P_p(t_i, \tau) + \\ + \sum_{t_i : \Delta(t_i) \neq ue} P_{\mathtt{off}}(t_i, \Delta(t_i), \tau)\, d\tau. \quad (5)$$

Where $P_p(ue, \tau)$ is the instantaneous processing power draw on mobile device *ue* at the instant τ, $P_{\mathtt{off}}(t_i, \tau)$ is the instantaneous power draw for offloading task.

Mobile device energy consumption model. For energy consumption of mobile device, we employ the CPU model of by [3], described by Equation 6:

$$P_p(ue, \tau) = \sum_{i=0}^{i<\mathtt{CPU}(ue)} \beta_{\mathtt{freq}}(i, \tau) \cdot \mathcal{U}_{cpu}(ue, \tau) + \beta_{\mathtt{base}}, \quad (6)$$

where $\beta_{freq}(i, \tau)$ is a constant dependent on the frequency of core i at the instant τ, β_{base} is a hardware dependent constant and $\mathcal{U}_{cpu}(ue, \tau)$ is the CPU utilization of device *ue* at the time τ, as defined by Equation 7.

$$\mathcal{U}_{cpu}(n, \tau) = \frac{\sum_{t_i \in \Delta(n)} \mathtt{SIZE}(t_i, \tau)}{\mathtt{MIPS}(n)} \quad (7)$$

Energy consumption for offloading. With offloading, input of task t_i are transferred to node n_j to be processed, and then output data are downloaded back to UE. According to [4], energy consumption of each network transfer is linear with time required by transfer between two nodes. Transfer time depends on the bandwidth and latency available on the link and if the node is an Edge or a Cloud node. Therefore, we model it as in Equation 8,

$$P_{off}(t_i, n_j, \tau) = \epsilon_{conn}(n_j) \cdot \mathcal{U}_{net}(t_i, n_j, \tau) + \mathtt{K}_{conn}(n_i), \quad (8)$$

where $\epsilon_{conn}(n_i)$ models the relationship between network utilization and instantaneous power, while $\mathcal{U}_{net}(t_i, n_j, \tau)$ is network utilization at time instant τ, namely

$$\mathcal{U}_{net}(t_i, n_j, \tau) = \frac{\mathtt{DATA}_{net}(t_i, \tau)}{\mathtt{b}(ue, n_j)}, \quad (9)$$

where $\mathtt{DATA}_{net}(t_i, \tau)$ is the amount of data offloaded at time instant τ. Time for transmission between *ue* and n_j is calculated as $\frac{\mathtt{DATA}_{in,out}}{\mathtt{b}(ue,n_j)} + \mathtt{l}(ue, n_j) \cdot dist(coords(ue), coords(n_j))$, while $\mathtt{K}_{conn}(n_i)$ represents a hardware related constant.

3.6 Deployment model

For the deployment we extend the definition in [31] to workloads. A workload $w(ue)$ is a DAG, generated by joining different applications, therefore same definitions for deployment of applications can be applied here. Since edges define a precedence order between tasks, deployment is performed in different time steps. Therefore, we first define a partial workload deployment for the time instant τ, $\Delta(w(ue), \mathcal{I}, \tau)$. In each time step we schedule only *ready* tasks, namely, the tasks $t_i \in \mathcal{T}_{w(ue)}$ such that $\delta_{in}(t_i, \tau) = \emptyset$. We define as $\mathcal{T}_{w(ue)}(\tau)$ the set of tasks in $\mathcal{T}_{w(ue)}$ that are ready at time step τ. We define the *partial* deployment of tasks $\mathcal{T}_{w(ue)}(\tau)$ as a set $\Delta(w(ue), \mathcal{I}, \tau)$ of pairs (t_i, n_j), with $\Delta(w(ue), \mathcal{I}, \tau) \subseteq \mathcal{T}_{w(ue)}(\tau) \times \mathcal{N}_{\mathcal{I}}$, such that

$$(t_i, n_j) \in \Delta(w(ue), \mathcal{I}, \tau) \iff t_i \text{ is deployed on node } n_j. \quad (10)$$

Also, the partial deployment of mobile workloads in $\mathcal{W}$, namely $\Delta(\mathcal{W}, \mathcal{I}, \tau) \stackrel{\text{def}}{=} \bigcup_{ue \in \mathcal{D}_{\mathcal{I}}} \Delta(\mathcal{W}, \mathcal{I}, \tau)$. It is possible to deploy more tasks to the same computational node n_j, as soon as $\vec{r}(t_i) \leq \vec{r}(n_j)$. We define $\Delta(n_j, \tau)$ as the set of tasks mapped to node n_j at instant τ, namely

$$t_i \in \Delta(n_j, \tau) \iff \exists j : (t_i, n_j) \in \Delta(n_j, \tau). \quad (11)$$

We define a *valid* partial deployment for all workloads $w(ue) \in \mathcal{W}$ on infrastructure $\mathcal{I}$, $\Delta(\mathcal{W}, \mathcal{I}, \tau)$. A partial deployment is valid only if (1) all the tasks are deployed only once on the infrastructure and (2) all deployments of tasks on nodes satisfy the capacity constraints of the nodes, namely:

Definition 4. *A partial deployment $\Delta(\mathcal{W}, \mathcal{I}, \tau)$ of workload $w(ue)$ on infrastructure $\mathcal{I}$ is* valid $\iff$

(1) $\bigcup_{n_j \in \mathcal{N}_{\mathcal{I}}} \Delta(n_j, \tau) \subseteq \mathcal{T}_{\mathcal{W}}(\tau)$;
(2) $\forall (t_i, n_j) \in \Delta(\mathcal{W}, \mathcal{I}, \tau)\ \mathtt{OL}(t_i) = \mathtt{FALSE} \implies n_j \in \mathcal{D}_{\mathcal{I}}$;
(3) $(t_i, n_j) \in \Delta(\mathcal{W}, \mathcal{I}, \tau) \iff$
(4) $\forall ue_i \in \mathcal{D}_{\mathcal{I}}, \sum_{i=0}^{|\mathcal{D}_{\mathcal{I}}|}, E_{ue}(t_i) \leq \mathtt{BATTERY}(ue_i)$
  (a) $\sum_{t_i \in \Delta(n_j,\tau)} \vec{d}(t_i) \leq \vec{r}(n_i)$;
  (b) $\bigcup_{t_i \in \Delta(n_j,\tau)} \delta_{in}(t_i, \tau) \subseteq \delta_{in}(n_i, \tau)$;
  (c) $\bigcup_{t_i \in \Delta(n_j,\tau)} \delta_{out}(t_i, \tau) \subseteq \delta_{out}(n_i, \tau)$;
  (d) $\forall (n_i, n_j) \in \delta_{in}(t_i, \tau)\ \mathtt{l}(n_i, n_j) \geq \mathtt{l}(n_i, n_j)$, $(n_i, n_j) \in \delta_{in}(n_i, \tau)$;
  (e) $\forall (n_i, n_j) \in \delta_{out}(t_i, \tau)\ \mathtt{l}(n_i, n_j) \geq \mathtt{l}(n_i, n_j)$ $(n_i, n_j) \in \delta_{out}(n_i, \tau)$;
  (f) $\forall (n_i, n_j) \in \delta_{in}(t_i, \tau)\ \mathtt{b}(n_i, n_j) \leq \mathtt{b}(n_i, n_j)$, $(n_i, n_j) \in \delta_{in}(n_i, \tau)$;
  (g) $\forall (n_i, n_j) \in \delta_{out}(t_i, \tau)\ \mathtt{b}(n_i, n_j) \leq \mathtt{b}(n_i, n_j$ $(n_i, n_j) \in \delta_{out}(n_i, \tau)$.

Where (1) means that the set of all the tasks allocated to nodes at time instant τ is embedded or equal to the set of nodes that are ready at time instant τ; (2) means that if a task t_i is not offloadable ($\mathtt{OL}(t_i) = \mathtt{false}$) it is allocated to a UE; (3) means that the energy consumed for execution or offloading of each task cannot be higher than the energu budget of each UE; (4) means that for each mapping (t_i, n_j) all the capacity constraints of the computational nodes are respected, in terms of CPU and storage (4.a), network connections (4.b,c) and latency and bandwidth of each link (4,e-g). With a small abuse of notation, we define also that $\Delta(t_i)$ refers to the node where task t_i is deployed, namely $\Delta(t_i) = n_j : \exists (t_i, n_j) \in \Delta(\mathcal{W}, \mathcal{I})$.

When task t_i terminates its execution at a given instant $\tau(t_i)$, it is removed from the DAG. At this point, we perform a partial deployment of tasks in the set $\mathcal{T}_{\mathcal{W}}(\tau(t_i))$. The set $\mathcal{T}_{\mathcal{W}}(\tau(t_i))$ is obtained removing from $\mathcal{L}_{\mathcal{W}}$ all the edges in $\delta_{out}(t_i, \tau)$ and adding to $\mathcal{T}_{\mathcal{W}}(\tau(t_i))$ all the tasks $t_i \in T_{\mathcal{W}}$ such that $\delta_{in}(t_i, \tau) = \emptyset$. Deployment is complete when $\mathcal{T}_{\mathcal{W}} = \emptyset$. We define a complete deployment $\Delta(\mathcal{W}, \mathcal{I}, \tau)$ as the union of all partial deployments, namely $\Delta(\mathcal{W}, \mathcal{I}, \tau) = \bigcup_{\tau=[0,\tau_{end}]} \Delta(\mathcal{W}, \mathcal{I}, \tau)$, where τ_{end} is the time instant when all the tasks have been deployed.

3.7 Problem definition

In this section we define the optimization problem we want to solve by extending the definition in [31]. Our goal is to find a deployment for each mobile workload that (1) minimizes QoS violation, defined according to different metrics, and (2) maximizes providers' profit. We employ the average QoS violation as indication of satisfaction of user requirements. Also, we define the problem of increasing providers' profits as minimizing the operational costs of provider. The problem definition can be found in in Equation 12. QoS violation and Provider cost are defined respectively in Sections 3.7.1 and 3.7.4. In Table 1 we summarize the notation used in this section.

$$\begin{cases} \min QoS\_V(\Delta(\mathcal{W}^*, \mathcal{I})) \\ \min PC(\Delta(\mathcal{W}^*, \mathcal{I})) \\ \text{With } \Delta(\mathcal{W}^*, \mathcal{I}) s.t. \text{Definition 4} \end{cases} \tag{12}$$

Notation	Description
$\Delta(w(ue), \mathcal{I})$	Deployment of $w(ue)$ on infrastructure $\mathcal{I}$.
$RT(\Delta(w(ue), \mathcal{I}))$	Response time of deployment of workload $w(ue)$ on infrastructure $\mathcal{I}$.
$UC(\Delta(w(ue), \mathcal{I}))$	User cost for deployment.
$rt(t_i, n_j)$	Response time of task t_i on node n_j.
$rt_{local}(t_i, n_i)$	Response time of task t_i excluding offloading time.
$\tau(t_i)$	Time instant when t_i becomes ready.
$OT_{up,down}(t_i, ue, n_j)$	Time for offloading/downloading task t_i from/to UE ue to/from node n_j.
$\mathtt{DATA}_{in,out}(t_i)$	Input/output data of task t_i.
$\mathcal{D}_{\mathcal{I}}$	Set of UEs in infrastructure $\mathcal{I}$
$\mathcal{E}_{\mathcal{I}}$	Set of Edge nodes in infrastructure $\mathcal{I}$
$C_{\mathcal{I}}$	Set of Cloud nodes in infrastructure $\mathcal{I}$
$\mathcal{N}_{\mathcal{I}}$	Set of computational nodes in infrastructure $\mathcal{I}$, $\mathcal{N}_{\mathcal{I}} = \mathcal{D}_{\mathcal{I}} \cup \mathcal{E}_{\mathcal{I}} \cup C_{\mathcal{I}}$
$\mathcal{L}_{\mathcal{I}}$	Set of network links between nodes in $\mathcal{N}_{\mathcal{I}}$.
$\mathcal{M}_{\mathcal{I}}(m, n)$	Map of the urban area.

Table 1: Notation summary.

3.7.1 QoS Violation. We define the requirements for a UE $\vec{qos}(ue) = \{r^*, c^*\}$ as the QoS requirements for its workload execution, respectively for workload response time and user cost. We define the average QoS violation for workload $\mathcal{W}$ as

$$QoS\_V(\Delta(\mathcal{W}, \mathcal{I})) = \frac{1}{|\mathcal{D}_{\mathcal{I}}|} \sum_{ue \in \mathcal{D}_{\mathcal{I}}} \frac{|\vec{qos}(ue) - O(\Delta(w(ue), \mathcal{I})|}{\vec{qos}(ue)} \tag{13}$$

where

$$O(\Delta(w(ue), \mathcal{I})) = \langle RT(\Delta(w(ue), \mathcal{I})), UC(\Delta(w(ue), \mathcal{I})) \rangle. \tag{14}$$

In the following sections we define $RT(\Delta(w(ue), \mathcal{I}))$ (Section 3.7.2), $UC(\Delta(w(ue), \mathcal{I}))$ (Section 3.7.3).

3.7.2 Response time. The workload response time is defined as

$$RT(\Delta(\mathcal{W}, \mathcal{I})) = \tau_{end}(\mathcal{W}), \tag{15}$$

where τ_{end} is the time when $\mathcal{T}_{\mathcal{W}} = \emptyset$. For each task t_i, $rt(t_i)$ depends on the response time of t_i on a node n_i, namely $rt_{local}(t_i, n_i)$, the time for offloading task t_i to node n_i, $OT_{up}(ue, t_i, n_i)$ and the time for downloading results of task execution, $OT_{down}(ue, t_i, n_i)$. We define $rt_{local}(t_i, n_i)$ as follows:

$$rt_{local}(t_i, n_i) = \frac{\mathtt{SIZE}(t_i)}{\mathtt{MIPS}(n_i)} \tag{16}$$

Offloading time OT_{up} depends on data transferred to $\Delta(t_i)$ and bandwidth available between user equipment ue and $\Delta(t_i)$. Offloading t_i requires transfer of input data $\mathtt{DATA}_{in}(t_i)$. Then, OT_{up} and OT_{down} are defined as

$$OT_{up,down}(ue, t_i, n_j) = \frac{\mathtt{DATA}_{in,out}}{\mathtt{b}(ue, n_j)} + \mathtt{l}(ue, n_j) \cdot dist(coords(ue), coords(n_j)). \tag{17}$$

Finally, we define the response time of a task t_i on node n_j as

$$rt(t_i, n_j) = \tau(t_i) + OT_{up}(t_i, ue, n_j) + rt_{local}(t_i, n_j) + OT_{down}(t_i, ue, n_j), \tag{18}$$

where $\tau(t_i)$ is the instant at which the task t_i is assigned to a node. Clearly, if task is executed on a UE, $rt(t_i, ue) = rt_{local}(t_i, ue)$, as $\mathtt{b}(ue, ue) = \infty$ (See Section 3.4).

3.7.3 User cost model. Execution cost of each task t_i depends on (1) location of task deployments $\Delta(t_i)$, (2) task response time $RT(t_i)$ and (3) task demands $\vec{d}(t_i)$. The cost for deploying a workload $\mathcal{W}$ is defined as the sum of the costs of execution of each task, as in

$$UC(\Delta(\mathcal{W}, \mathcal{I})) \stackrel{\text{def}}{=} \sum_{t_i \in \mathcal{T}_{\mathcal{W}}} cost(t_i, \Delta(t_i)). \tag{19}$$

Typical pricing strategies used for Cloud are not applicable to the Edge contexts [1, 2]. Therefore, cost model for execution of task t_i on a node n_j is defined as follows:

$$cost(t_i, n_j) \stackrel{\text{def}}{=} \begin{cases} (1)\ 0,\ n_j \in \mathcal{D}_{\mathcal{I}} \\ (2)\ \int_{\tau_{start}(t_i)}^{rt(t_i,n_j)} \pi(t_i, \tau, n_j)\, d\tau,\ n_j \in C_{\mathcal{I}} \\ (3)\ \int_{\tau_{start}(t_i)}^{rt(t_i,n_j)} \pi(t_i, \tau) + \int_{\tau_{start}(t_i)}^{rt(t_i,n_j)} p_e(t_i, \tau, \eta), \\ n_j \in \mathcal{E}_{\mathcal{I}} \end{cases} \tag{20}$$

where case (1) means that no price is paid for execution on UE. Otherwise, in case (2), if t_i is offloaded to the Cloud, the user will pay the instantaneous price required by the provider for using resources required by task t_i at instant τ, defined as

$$\pi(t_i, \tau, n_j) = p_{cores(\tau, n_j)} \cdot \mathtt{CPU}(t_i) + p_{storage}(\tau, n_j)(\mathtt{DATA}_{in}(t_i) + \mathtt{DATA}_{out}(t_i)). \tag{21}$$

If the task is offloaded on Edge, the user pays the price he/she would pay on the Cloud, plus a price penalty p_e. Such penalty is added because execution on Edge increases the value perceived by the user because of the reduced latency. Also, the additional cost for deploying nodes in urban areas, in proximity of the user forces providers to ask for a higher price. Therefore, this additional p_e should maximize both provider's revenue and user satisfaction. We also would like that such penalty is adjustable to the value perceived by the user. Therefore, we define p_e with an additional η parameter that models if user prefers a lower latency or a cheaper price. We define the $p_e(t_i, \tau, \eta)$ function in Equation 22 by slightly modifying the model proposed in [31].

$$p_e(t_i, \tau, \eta) = \frac{T_f(t_i)}{\eta} - \sqrt{\frac{\eta \cdot \pi(t_i, \tau, \Delta(t_i)) + T_f}{\eta^2 \cdot \min_{n_j \in \mathcal{E}_\mathcal{I}} RT(t_i, n_j)}}, \tag{22}$$

where $T_f = \sum_{n_i \in C_\mathcal{I}} \frac{1(ue, n_i)}{|C_\mathcal{I}|} + \frac{1}{\texttt{CPU}(n_i)} - \sum_{n_k \in \mathcal{E}_\mathcal{I}} \frac{1(ue, n_k)}{|\mathcal{E}_\mathcal{I}|}$ and η is a value between 0.01 and 1, where a value closer to 0.01 means that user prefers to have lower latency, while a value closer to 1 indicates that user prefers price over latency. In [50] it is shown that this function maximizes both providers' revenue and users' satisfaction, making it a possible pricing model for Edge.

3.7.4 Provider cost. In this section we define the costs for the provider. Among the different MEC infrastructure management costs, we consider only electricity cost of the infrastructure.

Energy model. Our energy model is inspired to the piecewise model in [11]. Since we consider only CPU utilization, energy consumption of a MEC infrastructure $\mathcal{I}$ is defined as the integral of instantaneous power of the computational node P_n over time τ:

$$E(\Delta(\mathcal{W}, \mathcal{I})) = \int_0^{RT(\Delta(\mathcal{W}, \mathcal{I}))} P_\mathcal{I}(\tau)\, d\tau, \tag{23}$$

The instantaneous power consumption of a MEC infrastructure $\mathcal{I}$ is given by the sum of instantaneous power consumption of each computational node in $\mathcal{I}$, as defined in Equation 24.

$$P_\mathcal{I}(\tau) = \sum_{n_i \in \mathcal{E}_\mathcal{I} \cup C_\mathcal{I}} P(n_i, t), \tag{24}$$

As defined in [11], instantaneous power consumption of a computational node is composed of a idle part, P_{idle}, and a active part, P_{active}. Therefore, power consumption of a single computational node is defined as

$$P(n_j, \tau) = P_{idle}(n_j, \tau) + P_{active}(n_j, \tau), \tag{25}$$

While P_{idle} is a hardware defined constant, we define the power consumption of a computational node as in Equation 27.

$$P_\text{l}(n_j, \tau) \stackrel{\text{def}}{=} \alpha(n_j) \cdot P_r(n_j) \cdot \mathcal{U}_{cpu}(n_j, \tau) \tag{26}$$

$$P_\text{h}(n_j, \tau) \stackrel{\text{def}}{=} \gamma(n_j) \cdot P_r(n_j) + (1 - \gamma(n_j)) \cdot P_r(n_j) \cdot \cdot \mathcal{U}_{cpu}(n_j, \tau);$$

$$P(n_j, \tau) = \begin{cases} P_l(n_j, \tau), & \mathcal{U}_{cpu}(n_j, \tau) \leq \mathcal{T}(n_j); \\ P_h(n_j, \tau), & \text{otherwise.} \end{cases} \tag{27}$$

where $\mathcal{T}(n_j)$ is the load at which the trend changes on node n_j, $\mathcal{U}_{cpu}(n_j, t)$ is the CPU utilization as defined in Equation 7, $P_r(n_j) = P_\text{max}(n_j) - P_\text{idle}(n_j)$, where $P_\text{max}(n_j)$ and $P_\text{idle}(n_j)$ are maximum and idle power consumption of node n_j, and $\alpha(n_j)$ and $\beta(n_j)$ are the coefficients for low (i.e. $\leq \mathcal{T}(n_j)$) and high (i.e. $> \mathcal{T}(n_j)$) CPU load.

Energy cost. according to [46], costs for energy change over time, depending on different factors such as temperature and fluctuations of the local energy market. In this work, we assume that price depends on the time instant τ and the location of the computational node. Therefore, we first define instantaneous energy cost of a single computational node as

$$EC(n_j, \tau) = p_{energy}(\texttt{loc}(n_j), \tau) \cdot P(n_j, \tau). \tag{28}$$

Afterwards, by applying Equation 23, we can obtain the energy cost of the whole execution of deployment $\Delta(\mathcal{W}, \mathcal{I})$. Since providing a pricing model for energy is outside the scope of this work, we simulate values of $p_{energy}(\texttt{loc}(n_j), \tau)$ by reading real-world electricity traces coming from different markets, as described in Section 5.

4 MULTI-OBJECTIVE ALGORITHM

Finding an infrastructure plan $\mathcal{I}$ given (1) a set of computational nodes $\mathcal{N}_I$, (2) the links between each node $\mathcal{L}_I$ and (3) the map $\mathcal{M}(m, n)$ is an optimization problem with two objectives, as defined in Equation 12. Since it is not possible to find a single solution that minimizes provider costs and QoS Violation at the same time, we want to choose our solution among a set of *non-dominated* solutions called *Pareto-set*. Definitions of dominance and Pareto-set can be found in [12, 18]. A Pareto-set can be found with multi-objective meta-heuristics, such as MOPSO [23] and NSGA-II [12]. We focus on on NSGA-II meta-heuristic, due to the better performance in comparison with other meta-heuristics [48]. The pseudocode of our method is described in Algorithm 1. The input for our algorithm is the mobile workload, the available links between UE and other computational nodes and the number and geographical positions of Cloud nodes. The output is the number and the locations on the map $\mathcal{M}_\mathcal{I}(m, n)$ of the Edge nodes in $\mathcal{E}_\mathcal{I}$. Since this planning is performed at the time MEC infrastructure is deployed, we focus on obtaining a robust plan, rather than on the response time of the algorithm. First, we perform several samplings of the infrastructure in parallel.

Algorithm 1 Monte-Carlo method

```
1: function FindPlan(W, N_I, L_I, M_I)
2:     E* ← ∅                          ▷ the set of all Edge plans generated by the algorithm
3:     i ← 0
4:     while i < Iter do
5:         L_I ← sampleQoS(L_I)
6:         N_I ← sampleUEPosition(D_I)
7:         E* ← E* ∪ MOPLAN(W, N_I, L_I, M_I)
8:         nIter ← nIter + 1
9:     end while
10:    H_E ← computeHistogram(E*)
11:    H_E ← filter(H_E, thr)          ▷ Remove if frequency < thr
12:    E_I ← aPosterioriDecision(E_I)
13: return I = I ∪ E_I
14: end function
```

In each sampling, we randomly generate (1) the number of cores available on Cloud nodes and Edge nodes, (2) the QoS provided by the links between each UE and the computational nodes and (3) the coordinates of UEs in the map $\mathcal{M}_\mathcal{I}(m, n)$. After this, we generate a set of solutions for the Edge nodes planning, that in the following

iterations are combined using evolutionary operators, like crossover and mutation, described in the following sections. At each iteration, the fitness of each solution is evaluated and used to decide which one will be kept for the future iterations. At the end of the iterations, we compute ranking and crowding distance, as described in [12], to identify the Pareto set. At the end of all the iterations, we calculate the histogram of possible plans, by computing the frequency of each solution contained in the Pareto-sets generated in each iteration. Afterwards, we select the solutions with the higher frequency in the histogram and perform a-posteriori decision making to select our deployment. MOPLAN algorithm is described in Algorithm 2. The parameters we employ are summarized in Table 2. In the next sections, we describe each phase of the MOPLAN algorithm.

Parameter	Value
Population size	25
Parent selection	Binary tournament
Crossover operator	Uniform crossover
Crossover probability	0.9
Offspring	2
Mutation probability	$\frac{1}{m \cdot n}$
Max iterations	50

Table 2: NSGA-II parameters

Algorithm 2 NSGA-II based planning of Edge nodes

1: **function** MOPLAN($\mathcal{W}$, $\mathcal{N}_I$, $\mathcal{L}_I$)
2: $\mathcal{E}^*_0 \leftarrow generatePopulation(populationSize, \mathcal{M}_I)$
3: $evaluateFitness(\mathcal{W}, \mathcal{N}_I \cup \mathcal{E}^*_0)$
4: $nIter \leftarrow 0$
5: **while** $nIter < maxIter$ **do**
6: $\mathcal{E}^*_{nIter,0} \leftarrow crossover(\mathcal{E}^*_{nIter}, crossoverProbability)$
7: $\mathcal{E}^*_{nIter,1} \leftarrow mutate(\mathcal{E}^*_{nIter,0}, mutationProbability)$
8: $evaluateFitness(\mathcal{E}^*_{nIter,1})$
9: $\mathcal{E}^*_{nIter} \leftarrow selection(\mathcal{E}^*_{nIter,0}, \mathcal{E}^*_{nIter,1})$
10: $nIter \leftarrow nIter + 1$
11: **end while**
12: **return** $\mathcal{E}^*_{nIter}$
13: **end function**

Generation of initial population. First of all, we have to describe how the initial population is generated. In this phase, we need to use a very efficient algorithm, allowing us to use it all the time MOPLAN is invoked, and also allowing to explore the whole search space. For this reason we employ Algorithm 3. We generate *populationSize* random setup of Edge nodes on the map. Initialization of Edge nodes (line 9) depends on the specifications available for each Edge nodes, as defined in Section 3.4.

Fitness evaluation. To evaluate the fitness of each plan, we have to calculate QoS violation and Provider cost obtained by scheduling workload $\mathcal{W}$ with the given plan. To do this, we simulate scheduling of $\mathcal{W}$ by generating a topological sort of the tasks in $\mathcal{W}$ using the rank function described by Equation 29.

$$rank(t_i) = \begin{cases} \mathtt{rt(t_i)} \text{ if } t_i \text{ is the exit task.} \\ \mathtt{rt(t_i)} + \max_{t_j \in \delta_{out}(t_i)}(\mathtt{OT(t_j)} + rank(t_j)) \end{cases} \quad (29)$$

Algorithm 3 Generation of initial population

1: **function** GENERATEPOPULATION($populationSize$, $\mathcal{M}_I$)
2: $\mathcal{E}^* \leftarrow \emptyset$
3: $p, i, j, k \leftarrow 0$
4: **for** $p < populationSize$ **do**
5: $\mathcal{E}_I \leftarrow \emptyset$
6: **for** $i < m$ **do**
7: **for** $j < n$ **do**
8: **if** $rand() < 0.5$ **then**
9: $e_k \leftarrow initEdgeNode()$
10: $coords(e_k) \leftarrow (i, j)$
11: $\mathcal{E}_I \leftarrow \mathcal{E}_I \cup e_k$
12: $k \leftarrow k + 1$
13: **end if**
14: **end for**
15: **end for**
16: $\mathcal{E}^* \leftarrow \mathcal{E}^* \cup \mathcal{E}_I$
17: **end for**
18: **return** $\mathcal{E}^*$
19: **end function**

Where $\mathtt{RT(t_i)}$ is the average running time of task t_i calculated over each node in $\mathcal{N}_I$, namely $\mathtt{RT(t_i)} = \frac{\sum_{n_j \in \mathcal{N}_I} rt_{local}(t_i, n_j)}{|\mathcal{N}_I|}$, while $\mathtt{OT(t_i)} = \frac{\sum_{n_j \in \mathcal{N}_I} OT(t_i, ue, n_j)}{|\mathcal{N}_I|}$ is the average offloading time for task t_i. The ranks for each task determine the order in which each node has to be scheduled for execution. After this, each task is put in a priority queue and extracted according to the rank. Afterwards, each task is scheduled according to the Earliest Finish Time, as in [42], considering the nodes that respects constraints in Definition 11.

Crossover. Two solutions (parents) in the population $\mathcal{E}^*$ are combined to generate two new ones (offspring). Parents are selected using Binary Tournament Selection [6]. After selection, a random number is generated: if it is greater than crossover probability, the two parents are returned as offspring; otherwise, offspring are generated by using a uniform crossover [40].

Mutation. Mutation operator works by changing the setting of a cell in the map. First, it picks a random cell in the map: if there is no Edge node in that cell, it adds to a node e_{k+1} to the solution. If there is a node instead, it removes the node assigned to that cell from the current solution. Mutation operator is applied with probability, $m_p = \frac{1}{m \cdot n}$, to ensure that in average one assignment of the map $\mathcal{M}_I(m, n)$ is modified.

Selection. The goal of this phase is to select the best solutions inside the old population and the offspring generated in current iteration. To do this, the best n solutions available in both populations are selected to be part of the new population, discarding the others. Selection criteria are ranking and crowding distance [12].

A-posteriori decision making. Finally, we select one solution among all the solutions in Pareto set. We select the solution who gives the less cost for the provider. If there are two or more solution with the same cost, we choose the one with the lower QoS violation. Since this solution is selected among the Pareto set obtained by the solution of the problem defined in Equation 12, it is going to be a solution that minimizes the cost for the provider, but at the same time is satisfactory for the user, being a non-dominated solution.

Livelab Entry	DAG	Probability
'com.facebook.Facebook'	Facebook	0.45
'com.apple.Maps'	Navigator	0.30
'com.mymobileapps.ichesseval'	Chess	0.10
'com.apple.mobileslideshow-Photos'	Facerec	0.10
'com.shavedham.pantiescannerlite'	Antivirus	0.05

Table 3: Probability distribution of each application according to their frequency in LiveLab traces.

5 EXPERIMENTAL SETUP

5.1 Simulation framework

Evaluation is performed using simulation based on real-world traces, due to unavailability of a MEC infrastructure at the time we write. After evaluation of different simulators for Edge, like iFogSim [21] and EdgeCloudSim [39], we decided to base our simulation on the extended version of FogTorchPi described in [31]. The reason of this choice is that it provides validated models for both mobile applications and Edge/Cloud infrastructure. We extend this version by adding support for (1) mobile workloads based on real-world traces, (2) multiple user environment and (3) price model for electricity based on US electricity market. The simulator is available online (https://bitbucket.org/vindem/fogtorchpi-extended). The input for the simulator consists in real-world traces for mobile workload and the infrastructure setup, consisting of description computational nodes and network connections parameters. The simulation covers one year of MEC infrastructure operations. Each simulation is run 100 times, to ensure that the average value for each metric falls in a confidence interval of 95%. Since many valid deployments can be generated in each iteration, we store them in a histogram and consider only the most frequent.

5.2 Mobile workload

In [31] there are different types of DAG simulating mobile applications, but it is not described how to use such DAGs to simulate a real-world execution of a mobile application. To this end, we employ the LiveLab mobile application usage traces [36], available online[3]. These traces collect data about usage of different mobile applications on iOS over 12 months from 25 undergraduate users from Rice University, therefore are representative of the typical workload of a mobile user. Based on these data, we randomly generate for each user trace of one year execution based on the probabilities in Table 3, where probability is calculated based on the frequency of the selected application over all the entries of the Livelab database. When we do not have an exact match between the LiveLab entry and the DAGs in [31], we try to find the closest match between the entry: for example, since we could not find an Antivirus application in Livelab database, to calculate its frequency we use another application that is used for scanning malware who exhibits a similar functioning. Concerning the requirements of each application, we generate them using the exponential distribution with the parameters identified by [31].

[3]http://livelab.recg.rice.edu/traces.html

Area name	Extension (km^2)	Number of UEs
Hernals	11.35	900
Leopoldstadt	19.27	2025
Simmering	23.23	3600

Table 4: Data about each area.

5.3 Geographical distribution

Figure 4: Cloud data center locations

For simulation of geographic distribution we imagine three urban areas of different size, selected among the neighborhoods in the city of Vienna. Data about these areas are summarized in Table 4. The embedding of SCC in the urban area is done by generating the smallest square grid that is able to cover that area and considering only the cells that are included in that area. The number of UEs is calculated as $25 \cdot \frac{AREA}{2}$, where 25 is the number of users per cell that is used in LiveLab traces, $AREA$ is the size of the area we are considering in km^2 and 2 is the area of each SCeNB in km^2, as estimated in several works targeting SCC [1]. Concerning instead the locations of Cloud data centers, we use the locations defined in [30], that are shown in Figure 4. For electricity costs of Edge nodes, we consider the prices for St. Ghislain, the closest to Vienna.

5.4 Computational nodes

We assume that CPU, RAM and storage specifications of Cloud nodes, as well as UE specifications, do not change during each different run of the simulation. This is because in real world scenarios, hardware configuration of computational nodes is rarely changing during one single application execution. We assume that Edge nodes have less capabilities than Cloud nodes in terms of cores, MIPS, RAM and storage [7]. The hardware specifications and hardware resources cost for each node are shown in Table 5. Concerning UE, we need to consider also energy consumption for the calculation of battery lifetime. We use the energy model defined in [31], with the energy consumption coefficients specified by [4]. Concerning energy consumption of computational nodes, we use data coming from [11] for the Equations 25. Regarding energy prices, we use traces available online from the U.S. electricity market, since they provide us granularity on the level of 5 minutes. For non-U.S. locations, we use the same data, shifted accordingly to time zone, as done also by [30]. Coefficients are summarized in Table 6.

5.5 Network infrastructure

Due to the unreliability of connections in MEC infrastructure (caused by mobility, environmental factors and reduced availability of Edge nodes), we need to accurately model the unreliable connections

$n_j \in \mathcal{N}_\mathcal{I}$	CPU	MIPS	CPU cost	MEM cost
c-*	64	15	0.03	0.01
e-*	16	15	0.03	0.01
ue-*	2	4	0	0

Table 5: Hardware configuration.

Coefficient	Value
$\beta_{\texttt{freq}}$	6.9320
$\beta_{\texttt{base}}$	$625.25e-6$
ϵ_{3g}	$0.025e-6$
$\mathcal{K}_{3g}$	$3.5e-6$
ϵ_{wifi}	$0.007e-6$
K_{wifi}	$5.9e-6$
α	5.29
γ	0.68
P_{idle}	501
P_{max}	840
$\mathcal{T}(h)$	0.12

Table 6: Energy coefficients for Equations 6 and 8.

between UEs and Edge/Cloud nodes to perform an accurate simulation. The QoS provided by each link is modelled $l_i \in \mathcal{L}_\mathcal{I}$ as a random variable $r'(l_i) = \langle \mathsf{l}(l_i), \mathsf{b}(l_i) \rangle$. By our assumption, each node is reachable by the mobile device using two different types of connections: 3G and WiFi. Which connection is available at a given time is determined by a uniform random variable. If both are available during the execution, the faster between the two is selected. For the QoS, we use the probability distribution of FogTorchPI [8] which is summarized in Table 7. $\mathsf{l} = \infty$ and $\mathsf{b} = 0$ means that there is no connection between the two computational nodes. For Cloud offloading, also Internet transmission delay has to be considered, that is estimated by [14] to be between 100 and 300 milliseconds. Since we expect this value to be close to the average, we model it as a Gaussian random variable with $\mu = 200$ and $\sigma = 33.5$.

Connection	Availability	QoS profile		Probability
		Latency (ms)	Bandwidth (Mbps)	
3G	0.75	54	7.2	0.9957
		∞	0	0.043
WiFi	0.25	15	32	0.9
		15	4	0.09
		∞	0	0.01

Table 7: Network availability distribution.

6 EVALUATION

We define the two evaluation metrics used in this section, derived from the objectives defined in Section 3.7. First, we define *provider profit* as the difference between user and provider cost, namely

$$PP(\Delta(\mathcal{W},\mathcal{I})) \stackrel{\text{def}}{=} \frac{UC(\Delta(\mathcal{W},\mathcal{I})) - PC(\Delta(\mathcal{W},\mathcal{I}))}{PC(\Delta(\mathcal{W},\mathcal{I}))}. \quad (30)$$

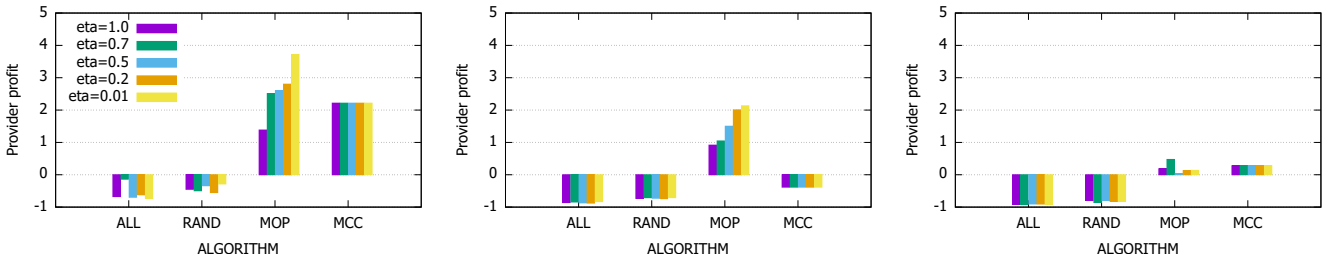
also, we define *user satisfaction* as the opposite of QoS violation,

$$US(\Delta(\mathcal{W},\mathcal{I})) \stackrel{\text{def}}{=} 1 - QoS\_V(\Delta(\mathcal{W},\mathcal{I})). \quad (31)$$

6.1 Results

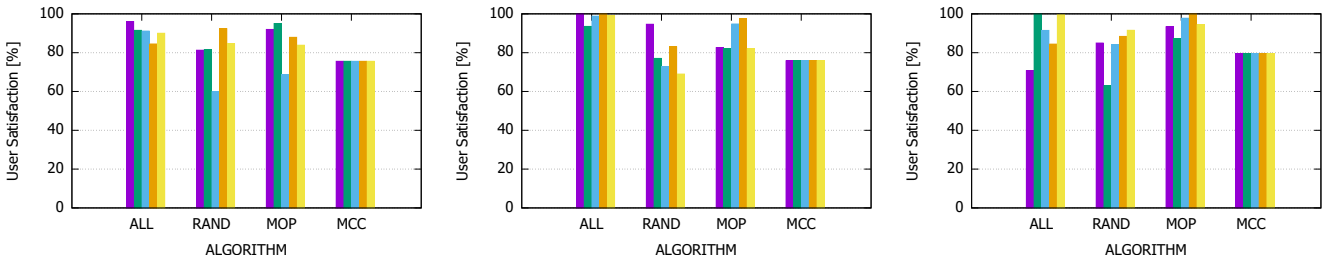
The comparison is performed between our proposed method (MOPLAN) and three other methods, respectively (1) MCC, an approach where no Edge node is used. Offloading is performed only on Cloud nodes; (2) ALLCELLS, where an Edge node is placed on each SCeNB in the map; (3) RANDOM, where Edge nodes are placed using the random algorithm described in Algorithm 3. From our theoretical modeling in Section 3, the main parameters affecting user satisfaction are the workload runtime and the price of Cloud resources. These parameters affect also providers' profit, since users are willing to pay the cost of additional computational resources only if this helps them in obtaining better performance. For this reason, we perform an analysis of the user cost model (Section 6.1.1) and of the runtime (Section 6.1.2). Both definitions can be found in Sections 3.7.2 and Section 3.7.3. In all the following experiments, provider profit is calculated according to Equation 30, while for user satisfaction we employ Equation 30. The $\vec{qos}(ue)$ considered for calculation of user satisfaction is obtained by assigning to rt^* and uc^* respectively to the minimum value for runtime and user cost among all the solutions obtained by the selected algorithms. To simulate different amount of system load, we vary the number of UEs for each of the selected areas as in Table 4.

6.1.1 Price model analysis. We evaluate the behavior of the proposed algorithms using different pricing models. The different pricing is simulated by varying the η parameter described in Section 3.7.3. We vary it in the range $\{0.01, 0.2, 0.5, 0.7, 1.0\}$, which allows us to simulate a wide range of prices for Edge resources. Since our focus is on the pricing model, we set the demand vector of each user to $\langle \infty, 0 \rangle$, meaning that the user does not impose any constraint on minimum latency and maximum bandwidth. Concerning results for provider profit (see Figures 5a-5c), we see that our algorithm is able to reach a profit up to five times more in comparison to the electricity cost, according to the cost model selected. The highest profit is reached when $\eta = 0.01$, to which however corresponds the lowest degree of user satisfaction, due to the higher prices for Edge resources. Also, we can observe significant losses when using the other two algorithms, ALLCELLS and RANDOM. In the first case this is because the algorithm overestimates the number of Edge nodes that is necessary for the number of users, while in the latter is because the position of Edge nodes is selected randomly and therefore might not be able to provide geographical proximity to users. This is even clearer when looking at Figure 7a, where we see how many Edge nodes are selected by the algorithm for each area. As we can see, regardless of the comparable number of Edge nodes deployed by MOPLAN and RANDOM, the user satisfaction is significantly higher when employing MOPLAN algorithm, showing that the geographical proximity to the user is more important than the number of Edge nodes used. Results for User satisfaction are summarized in Figures 6a-6c. As we can see, our Multi-Objective algorithm (MOP) is capable of achieving a user satisfaction above 80% in most of the cases, sometimes getting very close to the value obtained by the ALLCELL algorithm, that is the one providing the highest quality of service, since it deploys Edge nodes in each SCeNB in the map, without thinking about provider costs for deployment. It must also be noted that MCC, the case where no Edge node is deployed, has a user satisfaction always less than 80%. This is because the additional latency required to reach Cloud nodes significantly reduces the quality of service provided to the users and therefore theirs satisfaction in using the service,



(a) HERNALS. **(b) LEOPOLDSTADT.** **(c) SIMMERING.**

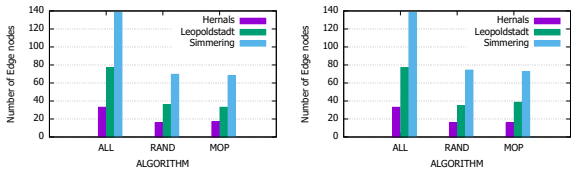
Figure 5: Provider profit for different areas $d = \langle \infty, 0 \rangle$, $\eta = \{0.01, 0.2, 0.5, 0.7, 1.0\}$.



(a) HERNALS. **(b) LEOPOLDSTADT.** **(c) SIMMERING.**

Figure 6: User satisfaction for different areas $d = \langle \infty, 0 \rangle$, $\eta = \{0.01, 0.2, 0.5, 0.7, 1.0\}$.

showing the benefits of deploying Edge services in this context. Also, according to our experiments, the best compromise between user satisfaction and provider profit is $\eta = 0.7$.



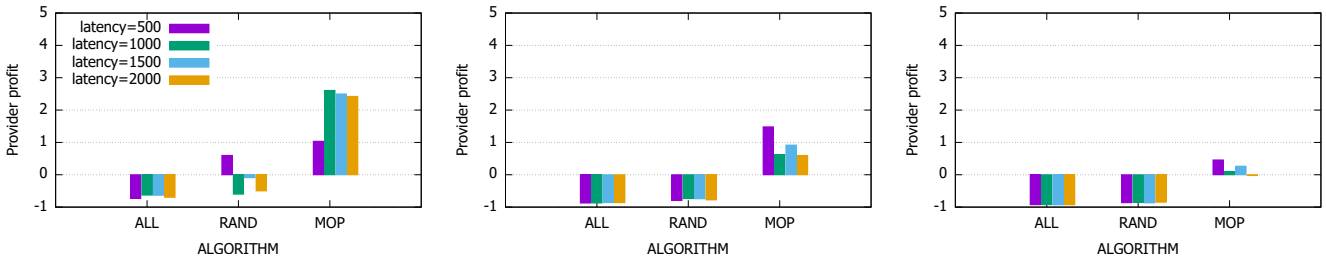
(a) Price model evaluation. **(b) Response time evaluation.**

Figure 7: Number of Edge nodes for both experiments

6.1.2 Response time evaluation. In this section, we perform an evaluation of how user satisfaction and provider profits change according to users' requirements. To this end, in this experiment, we simulate strict latency requirements for each mobile workload by setting the latency requirement of each link to different values. Requirements are generated using an exponential distribution with $\lambda = \{500, 1000, 1500, 2000\} ms$. These values were selected in order to simulate different latency requirements typical of Edge applications. Bandwidth is set to 0.1 Mbps. Concerning price model, we set $\eta = 0.7$, as in our Section 6.1.1 seems to be the best choice both for user satisfaction and for provider profit. Results for user satisfaction are summarized in Figures 9a-9c, while results for provider profit can be found in Figures 8a-8c. We do not show results for MCC in this case, since for the latency requirements settings it was impossible to find a valid deployment in all the cases. As we see, MOPLAN clearly outperforms the other two algorithms in terms of providers' profit, getting close to achieve a profit of almost 3 times more the electricity cost. Concerning user satisfaction, we see that MOPLAN gets really close to the best result, obtained by ALLCELL, with an average difference of 10% from it. However, as we see from Figure 7b, where we show the number of Edge nodes used by each algorithm, this is achieved through a strong overprovisioning of the infrastructure, with a negative effects on profit. It is also of note that the average number of Edge nodes required by each algorithm is slightly higher in comparison with the previous experiment: this is because the stronger latency requirements need a higher number of nodes in proximity to the UE to be satisfied.

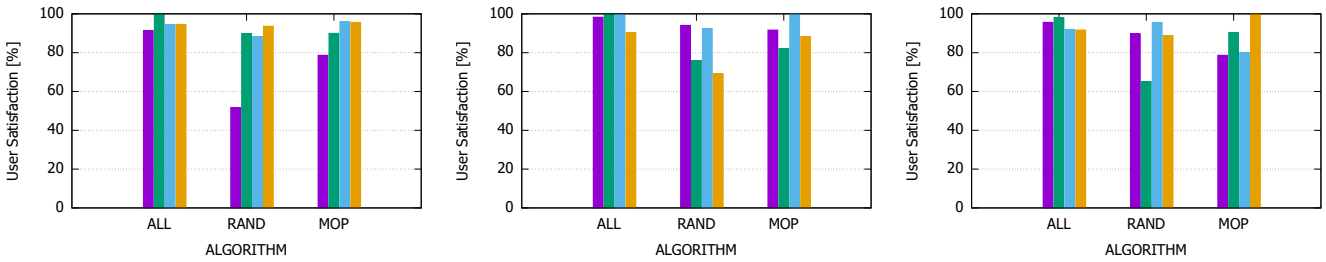
7 RELATED WORK

MEC is described as a key technology towards 5G by [25] and it is of great interest for scenarios like IoT [34]. In [32], a dynamic offloading scheme for MEC has been proposed, but focused on energy-harvesting devices. Works like [9, 26] focus on MCC, and more on the side of offloading than on provisioning. Other works discuss proposals of MEC architectures, such as Small Cell Cloud (SCC) [29], Mobile Micro Cloud (MMC) [45], Follow Me Cloud (FMC) [44] and CONCERT [28]. In this work, we focus on SCC. Multicell MEC has been discussed in works like [35], from the point of view of computational and radio resources optimization. Concerning computation offloading in the context of MEC, works like [32, 33] investigate partial offloading of single applications, while works like [15, 41, 49] analyze the benefits of offloading for



(a) HERNALS. **(b) LEOPOLDSTADT.** **(c) SIMMERING.**

Figure 8: Provider profit for different areas, $d = \langle\{500, 1000, 1500, 2000\}, 0.1\rangle$, $\eta = 0.7$



(a) HERNALS. **(b) LEOPOLDSTADT.** **(c) SIMMERING.**

Figure 9: User satisfaction for different areas, $d = \langle\{500, 1000, 1500, 2000\}, 0.1\rangle$, $\eta = 0.7$

real-world mobile applications, such as augmented reality, accelerated web browsing and speech recognition, respectively. Regarding provisioning, there are several works for provisioning in Cloud [22], but very few focus on MCC. In [27], a multi-objective provisioning approach for UltraScale systems is discussed. Concerning Edge computing, works like [16, 38] focus on resource provisioning and management for IoT services, while in [37, 47] author focus on provisioning considering only QoS as an objective. Workload models have been used by works like [17] for evaluation of resource management systems.

8 CONCLUSION AND FUTURE WORK

The main contributions of this paper are the following: first, we describe our vision of MEC in a SCC scenario. Then, we design a model for MEC in SCC by extending the work in [31] to consider multiple users, geographically distributed data centers and provider profits. After, we design a multi-objective algorithm based on NSGA-II metaheuristic to identify a Pareto-front solution between user satisfaction and providers' profit. Then, we evaluate the proposed solution using Monte-Carlo simulations based on real workload traces from LiveLab [36] and real prices from electricity market. Our results show that our algorithm on MEC can ensure a user satisfaction of above 80% and a profit up to 4 times providers' electricity cost. As future work, we plan to extend this work by considering different MEC systems, such as Mobile Micro Cloud (MCC) [45], Follow Me Cloud (FMC) [44] and CONCERT [28]. Also, in the current work we did not consider mobility of UEs. We would like to extend it by including mobility models for wireless networks, as in [19], or also for vehicular ad hoc networks [24]. Also, to improve accuracy of our simulations, we will investigate different types of simulation frameworks, such as agent-based simulations [10]. Finally, at the current state, our algorithm is used only for planning the number of Edge nodes required for a given area and the physical location for each node. In the future, we plan to extend it to make it capable of dynamically change the state of the Edge (e.g. shutting down nodes to save energy, perform migration for resilience reasons) according to load conditions and different user requirements.

ACKNOWLEDGMENTS

The work described in this paper has been funded through the Haley project (Holistic Energy Efficient Hybrid Clouds) as part of the TU Vienna Distinguished Young Scientist Award 2011 and Rucon project (Runtime Control in Multi Clouds), FWF Y 904 START-Programm 2015.

REFERENCES

[1] A. Ahmed and E. Ahmed. [n. d.]. A survey on mobile edge computing. In *2016 10th International Conference on Intelligent Systems and Control (ISCO)*. 1–8.
[2] May Al-Roomi, Shaikha Al-Ebrahim, Sabika Buqrais, and Imtiaz Ahmad. 2013. Cloud Computing Pricing Models: A Survey. *International Journal of Grid and Distributed Computing* 6, 5 (2013), 93–106.
[3] Farhan Azmat Ali, Pieter Simoens, Tim Verbelen, Piet Demeester, and Bart Dhoedt. 2016. Mobile device power models for energy efficient dynamic offloading at runtime. *Journal of Systems and Software* 113 (2016), 173 – 187.
[4] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. 2009. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 280–293.
[5] Michael Till Beck, Martin Werner, Sebastian Feld, and S Schimper. 2014. Mobile edge computing: A taxonomy. In *Proc. of the Sixth International Conference on*

Advances in Future Internet. Citeseer, 48–55.

[6] Tobias Blickle and Lothar Thiele. [n. d.]. A Comparison of Selection Schemes Used in Evolutionary Algorithms. *Evol. Comput.* 4, 4 ([n. d.]), 361–394.

[7] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing (MCC '12)*. ACM, 13–16.

[8] A. Brogi, S. Forti, and A. Ibrahim. [n. d.]. How to Best Deploy Your Fog Applications, Probably. In *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*. 105–114.

[9] Valeria Cardellini, Vittoria De Nitto Persone, Valerio Di Valerio, Francisco Facchinei, Vincenzo Grassi, Francesco Lo Presti, and Veronica Piccialli. 2016. A game-theoretic approach to computation offloading in mobile cloud computing. *Math. Program.* 157, 2 (2016), 421–449.

[10] G. Cordasco, C. Spagnuolo, and V. Scarano. 2016. Toward the New Version of D-MASON: Efficiency, Effectiveness and Correctness in Parallel and Distributed Agent-Based Simulations. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 1803–1812. https://doi.org/10.1109/IPDPSW.2016.52

[11] Vincenzo De Maio, Gabor Kecskemeti, and Radu Prodan. 2016. An Improved Model for Live Migration in Data Centre Simulators. In *Proceedings of the 9th International Conference on Utility and Cloud Computing (UCC '16)*. ACM, 108–117.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.

[13] Maofei Deng, Hui Tian, and Bo Fan. [n. d.]. Fine-granularity based application offloading policy in cloud-enhanced small cell networks. In *2016 IEEE International Conference on Communications Workshops (ICC)*. 638–643.

[14] M. DeVirgilio, W. David Pan, L. L. Joiner, and D. Wu. [n. d.]. Internet delay statistics: Measuring internet feel using a dichotomous Hurst parameter. In *2013 Proceedings of IEEE Southeastcon*. 1–6.

[15] J. Dolezal, Z. Becvar, and T. Zeman. 2016. Performance evaluation of computation offloading from mobile device to the edge of mobile network. In *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*. 1–7.

[16] Konstantinos Douzis, Stelios Sotiriadis, Euripides G. M. Petrakis, and Cristiana Amza. 2018. Modular and generic IoT management on the cloud. *Future Generation Comp. Syst.* 78 (2018), 369–378.

[17] Martyn Ellison, Radu Calinescu, and Richard F. Paige. 2018. Evaluating cloud database migration options using workload models. *J. Cloud Computing* 7 (2018), 6.

[18] Hamid Mohammadi Fard, Radu Prodan, and Thomas Fahringer. 2014. Multi-objective list scheduling of workflow applications in distributed computing infrastructures. *J. Parallel and Distrib. Comput.* 74, 3 (2014), 2152 – 2165.

[19] Ansgar Fehnker, Peter Höfner, Maryam Kamali, and Vinay Mehta. 2013. Topology-Based Mobility Models for Wireless Networks. In *Quantitative Evaluation of Systems*, Kaustubh Joshi, Markus Siegle, Mariëlle Stoelinga, and Pedro R. D'Argenio (Eds.). Springer Berlin Heidelberg, 389–404.

[20] S. Govindasamy and I. Bergel. 2018. Uplink Performance of Multi-Antenna Cellular Networks With Co-Operative Base Stations and User-Centric Clustering. *IEEE Transactions on Wireless Communications* 17, 4 (April 2018), 2703–2717.

[21] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya. 2016. iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments. *CoRR* abs/1606.02007 (2016).

[22] Kishaloy Halder, Umesh Bellur, and Purushottam Kulkarni. 2012. Risk Aware Provisioning and Resource Aggregation Based Consolidation of Virtual Machines. In *2012 IEEE Fifth International Conference on Cloud Computing, Honolulu, HI, USA, June 24-29, 2012*. 598–605.

[23] Jiang Hao, Zheng Jin-hua, and Chen liang jun. [n. d.]. Multi-Objective Particle Swarm Optimization Algorithm Based on Enhanced ϵ-Dominance. In *2006 IEEE International Conference on Engineering of Intelligent Systems*. 1–5.

[24] J. Harri, F. Filali, and C. Bonnet. 2009. Mobility models for vehicular ad hoc networks: a survey and taxonomy. *IEEE Communications Surveys Tutorials* 11, 4 (Fourth 2009), 19–41. https://doi.org/10.1109/SURV.2009.090403

[25] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. 2015. Mobile edge computingâĂŤA key technology towards 5G. *ETSI white paper* 11, 11 (2015), 1–16.

[26] M. H. Jiang, Otto W. Visser, I. S. W. B. Prasetya, and Alexandru Iosup. [n. d.]. A mirroring architecture for sophisticated mobile games using computation-offloading. *Concurrency and Computation: Practice and Experience* 30, 17 ([n. d.]), e4494.

[27] Dragi Kimovski, Sashko Ristov, Roland Mathá, and Radu Prodan. 2018. Multi-objective Service Oriented Network Provisioning in Ultra-Scale Systems. In *Euro-Par 2017: Parallel Processing Workshops*, Dora B. Heras, Luc Bougé, Gabriele Mencagli, Emmanuel Jeannot, Rizos Sakellariou, Rosa M. Badia, Jorge G. Barbosa, Laura Ricci, Stephen L. Scott, Stefan Lankes, and Josef Weidendorfer (Eds.). Springer International Publishing, Cham, 529–540.

[28] J. Liu, T. Zhao, S. Zhou, Y. Cheng, and Z. Niu. 2014. CONCERT: a cloud-based architecture for next-generation cellular systems. *IEEE Wireless Communications* 21, 6 (2014), 14–22.

[29] F. Lobillo, Z. Becvar, M. A. Puente, P. Mach, F. Lo Presti, F. Gambetti, M. Goldhamer, J. Vidal, A. K. Widiawan, and E. Calvanese. 2014. An architecture for mobile computation offloading on cloud-enabled LTE small cells. In *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. 1–6.

[30] Drazen Lucanin and Ivona Brandic. 2016. Pervasive Cloud Controller for Geotemporal Inputs. *IEEE Trans. Cloud Computing* 4, 2 (2016), 180–195.

[31] V. De Maio and I. Brandic. 2018. First Hop Mobile Offloading of DAG Computations. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. 83–92.

[32] Y. Mao, J. Zhang, and K. B. Letaief. 2016. Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices. *IEEE Journal on Selected Areas in Communications* 34, 12 (2016), 3590–3605.

[33] O. MuÃśoz, A. Pascual Iserte, J. Vidal, and M. Molina. [n. d.]. Energy-latency trade-off for multiuser wireless computation offloading. In *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. 29–33.

[34] Dario Sabella, Alessandro Vaillant, Pekka Kuure, Uwe Rauschenbach, and Fabio Giust. 2016. Mobile-edge computing architecture: The role of MEC in the Internet of Things. *IEEE Consumer Electronics Magazine* 5, 4 (2016), 84–91.

[35] S. Sardellitti, G. Scutari, and S. Barbarossa. 2015. Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing. *IEEE Transactions on Signal and Information Processing over Networks* 1, 2 (2015), 89–103.

[36] Clayton Shepard, Ahmad Rahmati, Chad Tossell, Lin Zhong, and Phillip Kortum. 2011. LiveLab: Measuring Wireless Networks and Smartphone Users in the Field. *SIGMETRICS Perform. Eval. Rev.* 38, 3 (Jan. 2011), 15–20.

[37] Olena Skarlat, Matteo Nardelli, Stefan Schulte, Michael Borkowski, and Philipp Leitner. 2017. Optimized IoT service placement in the fog. *Service Oriented Computing and Applications* 11, 4 (2017), 427–443.

[38] O. Skarlat, S. Schulte, M. Borkowski, and P. Leitner. 2016. Resource Provisioning for IoT Services in the Fog. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*. 32–39. https://doi.org/10.1109/SOCA.2016.10

[39] C. Sonmez, A. Ozgovde, and C. Ersoy. [n. d.]. EdgeCloudSim: An environment for performance evaluation of Edge Computing systems. In *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. 39–44.

[40] Gilbert Syswerda. 1989. Uniform Crossover in Genetic Algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2–9.

[41] N. Takahashi, H. Tanaka, and R. Kawamura. 2015. Analysis of Process Assignment in Multi-tier mobile Cloud Computing and Application to Edge Accelerated Web Browsing. In *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. 233–234.

[42] Haluk Topcuouglu, Salim Hariri, and Min-you Wu. [n. d.]. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. *IEEE Trans. Parallel Distrib. Syst.* 13, 3 ([n. d.]), 260–274.

[43] David Villegas, Athanasios Antoniou, Seyed Masoud Sadjadi, and Alexandru Iosup. 2012. An Analysis of Provisioning and Allocation Policies for Infrastructure-as-a-Service Clouds. In *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012, Ottawa, Canada, May 13-16, 2012*. 612–619.

[44] Kaiqiang Wang, Minwei Shen, Junguk Cho, Arijit Banerjee, Jacobus Van der Merwe, and Kirk Webb. 2015. MobiScud: A Fast Moving Personal Cloud in the Mobile Network. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges (AllThingsCellular '15)*. 19–24.

[45] Shiqiang Wang, Guan-Hua Tu, Raghu Ganti, Ting He, Kin Leung, Howard Tripp, Katy Warr, and Murtaza Zafer. 2013. Mobile micro-cloud: Application classification, mapping, and deployment. In *Proc. Annual Fall Meeting of ITA (AMITA)*.

[46] RafaÅĆ Weron. 2014. Electricity price forecasting: A review of the state-of-the-art with a look into the future. *International Journal of Forecasting* 30, 4 (2014), 1030 – 1081.

[47] Ashkan Yousefpour, Ashish Patil, Genya Ishigaki, Inwoong Kim, Xi Wang, Hakki C. Cankaya, Qiong Zhang, Weisheng Xie, and Jason P. Jue. 2018. QoS-aware Dynamic Fog Service Provisioning. *CoRR* abs/1802.00800 (2018).

[48] Cristian Zambrano-Vega, Antonio J. Nebro, José García-Nieto, and José Francisco Aldana Montes. 2017. Comparing multi-objective metaheuristics for solving a three-objective formulation of multiple sequence alignment. *Progress in Artificial Intelligence* 6 (2017), 195–210.

[49] Yuan Zhang, Hao Liu, Lei Jiao, and Xiaoming Fu. 2012. To offload or not to offload: An efficient code partition algorithm for mobile cloud computing. In *2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET)*. 80–86.

[50] Tianchu Zhao, Sheng Zhou, Xueying Guo, Yun Zhao, and Zhisheng Niu. [n. d.]. Pricing policy and computational resource provisioning for delay-aware mobile edge computing. *2016 IEEE/CIC International Conference on Communications in China, ICCC 2016* ([n. d.]).