# Dynamic Allocation of Computing and Communication Resources in Multi-Access Edge Computing for Mobile Users

Jan Plachy, *Student Member, IEEE*, Zdenek Becvar, *Senior Member, IEEE*, Emilio Calvanese Strinati, *Member, IEEE*, and Nicola di Pietro

***Abstract*—The Multi-Access Edge Computing (MEC) constitutes computing over virtualized resources distributed at the edge of mobile network. For mobile users, an optimal allocation of communication and computing resources changes over time and space, and the resource allocation becomes a complex problem. Moreover, for delay constrained applications, the resource allocation to mobile users cannot be solved by approaches designed for static users, as a solution would not be obtained within a desired time. Thus, in this paper, we propose a low-complexity computing and communication resource allocation for offloading of real-time computing tasks generated with a high arrival rate by the mobile users. We exploit probabilistic modeling of the users' movement to pre-allocate the computing resources at base stations and to select suitable communication paths between the users and the base station with the pre-allocated computing resources. The simulations show that the proposed algorithm keeps the offloading delay below 100 ms for the small tasks even with the arrival rate of five tasks per second per user, while the state-of-the-art algorithms can handle only up to 0.5 tasks per second per user. Thus, the proposal enables an exploitation of the MEC for various real-time applications even if the users are moving.**

***Index Terms*—Mobile networks, multi-access edge computing, offloading, mobility management, resource allocation, prediction, real-time.**

## I. Introduction

INCREASING requirements on computing capabilities of mobile devices motivate a convergence of communication and computing to a single concept of Multi-access Edge Computing (MEC) [1], also known as Mobile Edge Computing [2]. The MEC can exploit communication over multiple types of wireless technologies, such as LTE-A, LTE-A Pro, 5G, or WiFi. The concept of MEC further evolves Mobile Cloud Computing (MCC) [3] by moving computing resources to the edge of mobile network, i.e., to base stations (gNBs), small cells, remote radio heads, etc., [4]. Therefore, a communication delay between a user equipment (UE) and its allocated computing resources in the MEC is, in principle, lowered in comparison to the MCC [5], [6]. Moreover, the MEC reduces a backhaul load of the gNBs, since the UEs' data is processed directly in a MEC server located close to (or even at) the gNBs without a need to forward the UEs' data to a centralized computing server farm via the backhauls of the gNBs. Thus, the MEC provides a low offloading delay for many applications, such as content caching, augmented or virtual reality applications [7], vehicular networks [8], Internet of Things [9] or Industrial Internet of Things [10]. However, running computation demanding real-time applications, such as augmented reality, virtual reality, or games on mobile devices is possible only with radio resource and mobility management guaranteeing strict requirements on the offloading delay.

The processing of the UEs' applications in the MEC exploits virtualized computing resources in a form of either containers [11] or Virtual Machines (VMs) [12]. Both the VMs and the containers exploit physical computing resources, such as computing time of Central Processing Unit (CPU) or Random Access Memory (RAM), and virtualize them. In the case of containers, the virtualized resources are provided by the containers sharing host's Operating System (OS). In the case of VMs, the physical computing resources are virtualized by a hypervisor running either on the host OS or directly on the host hardware. Therefore, the VMs and the host OS are completely isolated from each other. This isolation provides a certain level of security, but the security comes at the cost of an additional overhead [13]. The overhead affects a performance and a startup time of the VMs and makes the VMs less efficient comparing to the containers [14], [15].

An application to be processed in the MEC, denoted as the offloaded application in this paper, is run on the virtualized resources (VMs or containers). The UE sends data to be processed (denoted as the offloaded task) to the offloaded application in the MEC. For example, the offloaded task by an augmented reality application contains information on the UE's position, its cone of vision, etc., [16].

Manuscript received June 18, 2020; revised January 9, 2021 and April 2, 2021; accepted April 5, 2021. Date of publication April 12, 2021; date of current version June 10, 2021. This work has been supported by project No. LTT18007 funded by Ministry of Education, Youth and Sports, Czech Republic and the grants of Czech Technical University in Prague No. SGS17/184/OHK3/3T/13 and No. SGS20/169/OHK3/3T/13. The associate editor coordinating the review of this article and approving it for publication was Q. Ling. *(Corresponding author: Jan Plachy.)*

Jan Plachy and Zdenek Becvar are with the Department of Telecommunication Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, 166 27 Prague, Czech Republic (e-mail: jan.plachy@fel.cvut.cz; zdenek.becvar@fel.cvut.cz).

Emilio Calvanese Strinati is with the Minatec Campus, CEA-Leti, 38000 Grenoble, France (e-mail: emilio.calvanese-strinati@cea.fr).

Nicola di Pietro was with CEA-Leti, 38000 Grenoble, France. He is now with Athonet Srl, 36050 Bolzano Vicentino, Italy (e-mail: nicola.dipietro@athonet.com).

Digital Object Identifier 10.1109/TNSM.2021.3072433

The offloading process is not always feasible or beneficial due to latency constraints, computation complexity, energy consumption, or memory requirements [17]. Thus, the offloading process is preceded by a decision whether to offload or not [17], [18], [19]. If this decision is positive, the whole offloading process goes through the following stages: i) transmission of the offloaded task to the MEC server where the computing resources are allocated, ii) processing of the offloaded task by the offloaded application running over the allocated computing resources, and iii) transmission of the computing results back to the UE. Each stage introduces a delay contributing to the overall offloading delay perceived by the UEs. Note that we assume that the MEC server is collocated with the gNB as outlined in, e.g., [20], [21]. Apart from the offloading delay, also an energy consumed at the UE for the offloading itself (in this paper denoted as the offloading energy) can be considered in the offloading decision [17], [22], [23], [24], [25].

To guarantee that the offloading delay meets the requirements imposed by the offloaded application, the communication and computing resources should be allocated jointly. If the MEC is serving static UEs, the VM can be kept at a single gNB during the whole offloading process. However, for the moving (mobile) UEs, the resource allocation complexity becomes significant, as the resources should be reallocated over time to adapt to the changes in quality of communication channels (due to UE mobility) and to an availability of the computing and communication resources. The allocation of the communication and computing resources can be adapted to the UE mobility via the UE mobility and channel quality prediction [26], [27]. However, these works do not address joint communication and computing resource allocation.

The UE's mobility can be handled by moving the virtualized resources allocated to the UE from the MEC server collocated with one gNB to another. This process is known as a VM migration or a container migration [28]. The VM migration consumes relatively long time [14] and can even lead to a service interruption, thus, it is not suitable for real-time services. The mobility of the UEs can be also facilitated by a selection of a new communication route, represented in mobile networks by a selection of new gNB that delivers the computation results to the UE [29], [30], [31]. However, even this approach cannot guarantee a low offloading delay if the connection among the gNBs is not of a very high quality. Another option of the mobility support is to deploy an entirely new VM at the new gNB and start the computing over [32]. This approach is denoted as a VM deployment. A similar approach is possible for the containers, which can exploit their advantage of a lower startup time compared to the VMs [14]. For the containers, the process is denoted as a container deployment [11]. The deployment of the new VM or container at another gNB, however, leads to a wasting of energy and the computing resources of the MEC servers. Consequently, the performance of such approach becomes limited in scenarios with a heavy computation load.

None of the above mentioned papers provides a solution that allows an efficient computing and communication resource allocation for the mobile (moving) UEs exploiting the real-time applications with a high arrival rate of the offloaded tasks. Therefore, in this paper, we focus on a computing and communication resource allocation for the moving UEs exploiting real-time applications with a high arrival rate.

The contribution and novelty presented in this paper are summarized as follows:

- We propose a novel algorithm for the dynamic communication and computing resource allocation (DCCRA) for MEC systems. Unlike the existing works, we target the offloading of the real-time applications by the moving UEs. Such scenario implies requirements on a very low offloading delay and a consideration of an impact of UEs' handover (in terms of the communication) and VM migration (in terms of the computing). We solve the offloading via two cooperating sub-algorithms, one for the dynamic selection of the communication path (i.e., the gNB that serves the UE) and one for the VM placement.
- To facilitate the proposed DCCRA, we develop a framework for a realistic and practical prediction of the UE's mobility and the channel quality based on a probabilistic model of the UE's movement. The mobility prediction is first illustrated in a scenario with one degree of mobility freedom and, then, we generalize it for multiple degrees of mobility freedom. The UE's mobility prediction is, afterwards, exploited for an efficient allocation of the communication and computing resources.
- Via simulations, we show that the proposed algorithm enables the offloading of the real-time applications by the mobile UEs and keeps the offloading delay under 100 ms even for a high arrival rate of up to five tasks per second per UE. Such performance is notably superior to the existing works and it facilitates an exploitation of the MEC services by the real-time applications even for the moving UEs. Furthermore, we show that the performance of the proposed solution in terms of the offloading delay and the energy consumption of the UEs is significantly improved comparing to the existing solutions and it is even close to the case with a perfect prediction of the channel quality.

The rest of this paper is organized as follows. In the next section, related work in the area of MEC resource allocation is presented. In Section III, the resource allocation problem is formulated and assumptions along with a system model are described. In Section IV, the framework for the mobility and channel prediction suitable for the proposed resource allocation is outlined. The proposed resource allocation algorithm is defined in Section V. Then, in Section VI, the environment and models for simulations are presented, and the simulation results are discussed in Section VII. Last, Section VIII concludes the paper and outlines future work.

## II. Related Work

This section describes related work in the area of resource allocation for the MEC services in scenarios with static and mobile UEs.

The problem of joint computing and communication resource allocation for the MEC services can be solved by an iterative algorithm as proposed in [33]. An extension of the iterative algorithm towards a distributed solution, which can be run on each gNB separately is presented in [34]. In [34], the authors show that the performance of the distributed solution is close to the centralized one while collection of all information at the central control node in the network is not required. The computing and communication resource allocation with an interference management is proposed in [35]. The developed offloading decision is followed by a resource allocation with interference management exploiting graph coloring. In [36], the authors design two solutions for an optimal VM placement, based on the integer linear programming, to minimize the number of pre-allocated VMs and the degradation of quality of experience. The energy consumption of the UEs is considered in [37] and [38]. The authors of [38] formulate the resource allocation problem as a convex optimization problem for a minimization of the UEs' energy consumption under a constraint on the computation latency and on the fairness of resource allocation. Then, an optimal policy for the resource allocation is derived. The UE's energy consumption is further minimized via exploitation of the Non-orthogonal Multiple Access (NOMA) in [39]. Communication resource allocation is considered in [40], where the authors exploit device-to-device communication of mobile UEs. However, the authors assume that the UEs and their channel quality are static during offloading. In [41], the authors propose a Q-learning-based algorithm for the resource allocation. The algorithm learns how to allocate the resources, and allocates these resources based on the actual state of the VM. Furthermore, in [42], the authors consider also a content caching for the resource allocation in order to improve the performance of the MEC. The caching is exploited to keep the content requested by the UE at the gNB to alleviate the gNB's backhaul. The authors formulate their optimization task as a convex problem, which is then transformed into a distributed convex optimization problem.

All the above-mentioned papers [33], [34], [35], [36], [37], [38], [39], [41], [42] focus on the static UEs or the UEs with a very low mobility. However, a support for the mobility management of the UEs during the offloading is a key feature required to ensure seamless exploitation of the MEC services [43]. The solutions developed for the static UEs in [33], [34], [35], [36], [37], [38], [42] cannot be easily extended to support the UEs' mobility as the VM placement would have to be determined every time the UEs' positions change. Similar approach for determination of an optimal VM placement every time the UEs' positions change is considered in [44]. The authors exploit only the computing resources of the UEs. This is, however, not an easy task (if not infeasible) for the moving UEs due to the computation complexity of these algorithms.

To handle the UE's mobility in the MEC, dynamic algorithms are required. The dynamic algorithms based on the VM placement and considering the mobility of UEs are outlined, e.g., in [45] and [46]. The VM is migrated to a new gNB whenever the UE changes its serving gNB. This means that the VM is migrated to remain in a proximity of the UE in order to reduce the communication delay. A similar solution, which decides whether and where to migrate the VM is proposed in [47]. The authors consider the Euclidean distance as a sole metric for the decision on the VM migration. This work is further enhanced in [48], where a mobility prediction with a fixed accuracy is assumed, and the computation load of the gNB is considered as the decision metric on the top of the Euclidean distance. Another approach for the decision on the VM migration exploiting mobility prediction is presented in [49], where the authors propose a Q-learning-based algorithm determining the time when the VM migration should be started. The prediction of the UEs' mobility is critical for the VM migration, since the migration is both computation and communication resource demanding. Therefore, the VM migration-based solutions, exploited in [45], [46], [47], [48], [49] impose a significant delay (in order of seconds), which prevents their exploitation for the real-time applications [50].

The requirement of a low delay for real-time applications can be handled by a pre-allocation of the virtualized resources (in the form of VM or containers) [51]. The VM pre-allocation for handling the UEs' mobility is exploited in our prior work [52], where the algorithm dynamically allocating computing and communication resources for the computation offloading is proposed. In [52], it is assumed that the VMs are pre-allocated on all gNBs. This is, however, a limiting factor for the real networks as it introduces redundant reservation of the gNBs' CPU, memory, and storage space, leading to high requirements on the deployed hardware making the solution impractical. Moreover, although the prediction with the fixed accuracy is considered in our prior work [52] and in [48], such assumption is too strong for realistic scenarios. Thus, we develop a solution for the prediction of mobility and channel quality suitable for the proposed resource allocation algorithm, which can work even under unreliable mobility and channel predictions. The mobility prediction for MEC is exploited in several existing works, such as [22], [26], [27], [53], [54], [55], [56], [57]. However, these works, in comparison to the proposed solution do not exploit the channel quality (data rate) prediction. Moreover, computing resource allocation is not considered in [22], [26], [53], [55], communication resource allocation is not considered in [27], [54], [56] and the authors of [57] do not consider impact of handover and VM migration. The proposed algorithm is motivated by a need for a low complexity solution allowing the resource allocation and VM management for the mobile UEs exploiting the real time applications, as emphasized in [36] and [43].

## III. System Model and Problem Formulation

In this section, we define the system model exploited for the proposed algorithm, we formulate the computing and communication resource allocation problem for the offloading, and we summarize the main assumptions for the resource allocation algorithm.

### A. System Model

We consider a set $S = \{s^1, s^2, \ldots, s^M\}$ of the gNBs and a set $U = \{u^1, u^2, \ldots, u^N\}$ of the UEs. The serving gNB

for the UE at the discrete time *t*, denoted $s_t \in S$, is selected as the gNB providing the highest Received Signal Strength (RSS). As the UE moves, the serving gNB is updated following a conventional hard handover procedure based on the RSS considering also a handover interruption with a duration of $t_{HO}$. This means that the serving gNB is updated if there exists the gNB $s' \in S$, where $s' \neq s_t$, for which $RSS(s') > RSS(s_t)+\Delta_{HO}$, where $\Delta_{HO}$ is the handover hysteresis (see, e.g., [58] for more details about the conventional hard handover and hysteresis in mobile networks). Based on the serving gNBs determined for the UEs, we define $n_t^R(s)$ as the number of UEs sharing the radio communication resources of the *s*-th gNB at the time *t*, respectively. Furthermore, in a similar way, we define $n_t^B(s)$ as the number of UEs sharing the backhaul communication resources of the *s*-th gNB at the time *t*, respectively.

Then, we define $s_t^* \in S$ as the gNB where the VM or the container for the UE is placed at the time *t*. We assume the possibility to pre-allocate the VMs or the containers on multiple gNBs to alleviate the issue of an unreliable mobility and channel predictions. Nevertheless, only one VM or container is exploited by the application offloaded by each UE at any given time. The time required to start the VM at the gNB is denoted as $t_{VM}$. This time includes the VM pre-allocation and the time required to start the processing of the offloaded tasks. Next, we define $\omega_t(s)$ as the amount of available processing resources of the *s*-th gNB in Millions Instructions Per Second (MIPS) at the time *t*. Then, the requirement on computing resources (in terms of MIPS) of the application offloaded by the *u*-th UE is labeled as $\omega(u)$. The *s*-th gNB is considered for the VM placement of the *u*-th UE at the time *t* only if the following condition holds:

$$\omega_t(s) > \omega(u). \tag{1}$$

Note, that the offloaded application resource requirements are not limited to the computing power, but also memory and/or hard drive capacity are considered. These resource requirements can be formulated in the same way as for the computing power requirements and an extension of the condition (1) to these parameters is straightforward. Therefore, without loss of generality and in order to keep our notation simple, we consider (1) as the only resource restriction of our problem. The MEC system model with the gNB communication and computing load is shown in Fig. 1.

The offloaded task is defined by the amount $L_{\mathrm{O}}$ of offloaded data (in bits), the amount $L_{\mathrm{C}}$ of collected data representing the computation results (again in bits), and the number $L_{\mathrm{P}}$ of instructions of the offloaded task to be executed at an gNB. The offloaded tasks are generated by the offloaded application with a task arrival rate $\lambda$, representing the number of offloaded tasks generated per second. The offloaded task can be delivered from the UE to the computing gNB directly via radio if $s_t = s_t^*$ or indirectly via the serving gNB $s_t$, if the serving gNB is different from the computing gNB $s_t^*$, i.e., if $s_t \neq s_t^*$. The latter case can appear, for example, in the situation when the serving gNB is not able to offer a sufficient computing power to the UE. The latter case assumes to exploit the backhaul connections of the $s_t^*$ and the $s_t$ for a transfer of the offloaded

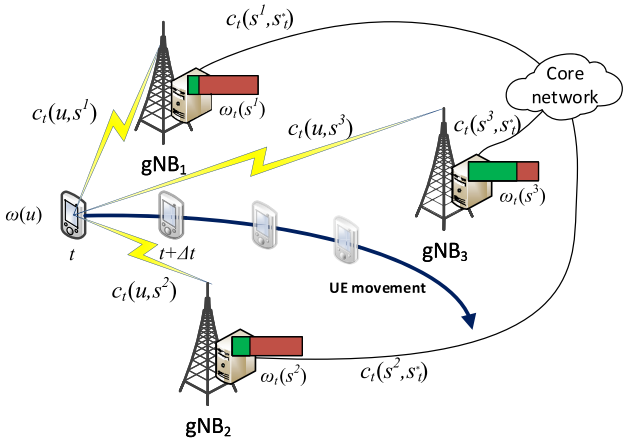


Fig. 1. MEC system model with one mobile UE.

task between the serving and the computing gNBs. Note that the backhaul communication between the $s_t$ and the $s_t^*$ is assumed to be routed via an operator's core network as it is done in conventional mobile networks [59], [60].

Furthermore, we define the set $Q_t(u) \subseteq S$ as the subset of all gNBs with which the *u*-th UE can communicate at the time *t*. In particular, this set contains only the gNBs to which the UE has Signal to Interference plus Noise Ratio (SINR) above a minimum SINR level required for the wireless communication ($SINR_{min}$). For the communication between the UE and the serving gNB, we assume LTE or 5G-based radio interface with radio resources shared equally among all UEs connected to the same gNB. Thus, the radio communication data rate between the *u*-th UE and the *s*-th gNB is calculated as:

$$c_t(u,s) = \nu_u \rho_u \frac{n^{RB}(s)}{n_t^R(s)}, \tag{2}$$

where $\nu_u$ is the number of bits per symbol for a given modulation scheme, $\rho_u$ is the code rate used for the radio communication between the gNB and the *u*-th UE, and $n^{RB}(s)$ is the number of Resource Blocks (RBs) available at the *s*-th gNB. Both $\nu_u$ and $\rho_u$ are derived from the channel quality according to the SINR of the *u*-th UE (see [61] for more details).

When $s_t \neq s_t^*$, the data rate expected on the backhaul connection between the serving gNB $s_t$ and the computing gNB $s_t^*$ (see Fig. 1) is defined as:

$$c_t(s,s^*) = \min\left\{\frac{c_s}{n_t^B(s)}, \frac{c_{s^*}}{n_t^B(s^*)}\right\}, \tag{3}$$

where $c_s$ and $c_{s^*}$ denote the available backhaul capacity of the serving gNB $s_t$ and the computing gNB $s_t^*$, respectively. Note that the "min" in (3) indicates that different data rates can be expected on the backhauls belonging to $s_t$ and $s_t^*$.

The communication data rate available for a delivery of the offloaded task from the UE to the $s_t^*$ either directly via radio (if $s_t = s_t^*$) or indirectly via the radio of $s_t$ and the backhauls of $s_t$ and $s_t^*$ (if $s_t \neq s_t^*$) is derived as:

$$c_t^{UL}(u,s,s^*) = \begin{cases} c_t(u,s) & \text{if } s_t = s_t^* \\ min\{c_t(u,s), c_t(s,s^*)\} & \text{otherwise.} \end{cases} \tag{4}$$

### *B. Problem Formulation*

Our objective is to find an allocation strategy of the computing and communication resources that minimizes the total offloading metric, represented by the offloading delay. Thus, the objective is to find the resource allocation strategy that minimizes the total offloading delay for the UE (denoted as $t_{\mathrm{MEC}}$). Minimization of the total offloading delay enables offloading of the real-time tasks, as these tasks require a very low delay. The total offloading delay consists of:

i) the time required to deliver the offloaded task from the UE to the gNB that starts the computation, determined as:

$$t_{\mathrm{O}} = \frac{L_{\mathrm{O}}}{c_t^{UL}(u, s, s^*)}, \tag{5}$$

ii) the time required to process the offloaded task, calculated as:

$$t_{\mathrm{P}} = \frac{L_{\mathrm{P}}}{\omega_t(s^*)}, \tag{6}$$

iii) the time required to deliver the processed data from the gNB that finishes the computation to the UE, defined as:

$$t_{\mathrm{C}} = \frac{L_{\mathrm{C}}}{c_t^{DL}(u, s, s^*)}, \tag{7}$$

with data rate $c_t^{DL}(u, s, s^*)$, derived in line with (4), but for the downlink, as the results of computation are received by the UE,

iv) the time consumed by the handover process, defined as:

$$t_{\mathrm{H}} = \sum_{i=1}^{n^{\mathrm{H}}} t_{\mathrm{HO}}^i, \tag{8}$$

where $t_{\mathrm{HO}}^i$ is the duration of the *i*-th handover and $n^{\mathrm{H}}$ is the number of handovers due to the UE changing serving gNB.

v) the time of the VM starts (including obtaining UE's application which processes offloaded tasks) during the offloading, determined as:

$$t_{\mathrm{M}} = n^{\mathrm{VM}} t_{\mathrm{VM}}, \tag{9}$$

where $n^{\mathrm{VM}}$ is the number of VM starts (equal to 0 if no VM starts is needed or VMs are pre-allocated) taking place during the offloading process. Note that the gNB that receives the offloaded data and the gNB that delivers the results back to the UE may not be the same due to the UE's mobility. The total offloading delay experienced by the UE is then calculated as:

$$t_{\mathrm{MEC}} = t_{\mathrm{O}} + t_{\mathrm{P}} + t_{\mathrm{C}} + t_{\mathrm{M}} + t_{\mathrm{H}}. \tag{10}$$

We can treat the minimization problem as a pair of joint problems. The first problem is the determination of the sequence of gNBs $\{s_t^*\}_t^{\mathrm{opt}}$, where the VM (or the container) should be placed for the *u*-th UE at each time *t*. The second problem is the selection of the communication path, identified with the serving gNBs sequence $\{s_t\}_t^{\mathrm{opt}}$. Merging both problems, we formulate the objective as:

$$\{s_t^*\}_t^{\mathrm{opt}}, \{s_t\}_t^{\mathrm{opt}} = \underset{\{s_t^* \in S\}_t, \{s_t \in S\}_t}{\operatorname{argmin}} t_{\mathrm{MEC}}. \tag{11}$$

However, solving (11) is, in general, difficult and impractical as both computing and communication resource allocation have to be done together for each *t* leading to a complex problem. Furthermore, the offloading delay (10), consisting of the time to offload the task (5) and the time to collect the processed results (7), depend on the data rates defined in (4). These data rates are not always known and should be predicted. This complicates the possibility to reach the global optimum. Moreover, finding the global optimum at each *t* leads to allocation of the computing resources (VMs) at different gNBs due to variation of the channel quality over time. Exploiting the VMs on different gNBs then leads to a high number of VM starts ($n^M$) and handovers ($n^H$) as shown in [52]. Thus, even though the global optimum is known, it may be impossible to reach it in practice, as the VMs would be constantly started over and over again.

Since the problem (11) is impractical and cannot be directly solved, we simplify the problem and transform it into the maximization of the communication data rate $c_t^{UL}(u, s, s^*)$ due to the constant $L_{\mathrm{O}}$ and $L_{\mathrm{C}}$, while considering (1), (3), (4), and (10). We focus on the uplink communication rate, because the uplink is commonly assumed to be of a lower data rate than the downlink. The extension to consider the downlink communication rate is trivial and we leave it out to simplify the notations. Therefore, we transform the problem into the following:

$$\{s_t^*\}_t^{\mathrm{opt}}, \{s_t\}_t^{\mathrm{opt}} = \underset{\{s_t^* \in S\}_t \{s_t \in S\}_t}{\operatorname{argmax}} \{c_t(u, s, s^*)\}_t \tag{12}$$

$$\text{s.t.} \quad \omega_t(s^*) > \omega(u). \tag{13}$$

where the constraint (13) is defined to avoid placing the VMs on the gNBs, which do not have enough computing power to host the VM for the UE. The constraint (13) considers computing power of the gNBs, since each gNB can have different computing power. Since $\{s_t^*\}_t^{\mathrm{opt}}$ and $\{s_t\}_t^{\mathrm{opt}}$ can be different, the transformed problem can be solved as two subproblems via two proposed cooperative algorithms.

### *C. Assumptions*

In this paper, we assume that every task is offloaded, as assumed in [48]. The assumption of offloading every task represents the case of the UE that does not have enough computing resources to process tasks itself and is forced to offload them. Note that introducing the offloading decision, such as in [62], simply leads to a lower amount of tasks to be processed in the MEC servers, as only some tasks would be offloaded. Thus, the proposed solution is applicable to any offloading decision algorithms and its impact on performance is proportional to changes in the task arrival rate $\lambda$ investigated later in the paper.

In [48] and [52], the authors suppose that the communication data rate is predicted with a pre-defined fixed accuracy, but this assumption is quite strong. More realistically, here, we assume that the prediction accuracy is unknown and varies in time, or even that a prediction is unavailable at all. This reflects the unreliability of the UEs' mobility prediction strategies, even when they are based on a significant amount of information about the UEs [63]. A suitable approach is

to exploit probabilistic models or probabilistic-free models as in [64]. To design and implement such approach, we assume that the knowledge of users' contextual information, such as scheduled meetings, favorite places, etc. as exploited, e.g., in [63], [64] is *not* available. Such assumption complicates the prediction and potentially negatively impacts on the performance of the developed algorithm. However, this assumption is motivated by questionable willingness of the users to provide this type of information to the network operator due to privacy issues. Thus, we expect the availability of only the information that is typically available to the network or which is commonly shared by the users. More specifically, we exploit anonymized UEs positions and SINR at those positions. As this type of information can be easily anonymized, the privacy risks are significantly lowered with respect to [63], [64].

For clarity and simplification of explanation, we assume the architecture where the MEC servers are collocated with the gNBs as proposed in [20], [21]. Note that a placement of the MEC servers to other network nodes, such as core network elements, increases $t_{\mathrm{O}}$ and $t_{\mathrm{C}}$. Since the proposed algorithm considers $t_{\mathrm{O}}$ and $t_{\mathrm{C}}$ in terms of the communication data rate, it can be simply extended to consider also different placements of the MEC servers. However, this extension is omitted here for the sake of clarity.

The data of the offloaded task are processed via an UE application in the MEC. For this, we assume, that the UE's application at the MEC server (represented by a gNB or SCgNB) is obtained from a cloud storage during the VM start time.

## IV. Mobility and Channel Quality Prediction

To solve the problem formulated in the previous section, we develop the mobility and channel quality predictions with a low complexity to derive the expected communication data rate. The predicted data rate is then exploited by the proposed computing and communication resource allocation algorithm. Note that the main novelty of this paper is the computing and communication resource allocation, while the mobility and channel quality prediction are tools designed to fit the needs of the proposed computing and communication resource allocation. The effectiveness of different mobility prediction approaches depends on the application scenarios. Therefore, we split the description of the mobility prediction into two cases with: i) one degree of mobility freedom (e.g., movement along a sidewalk or street with no possibility to turn away), and ii) multiple degrees of movement freedom with possibility to change the direction (e.g., crossroads, open spaces, squares, etc.). These two cases are explained in the next two subsections. Then, the last part of this section describes the proposed channel quality prediction strategy, which is further exploited for the communication data rate prediction.

### A. Mobility Prediction With one Degree of Mobility Freedom

If the UE's mobility is limited to one degree of mobility freedom (i.e., UE following a sidewalk or street) an extrapolation of the UE's movement is a suitable approach to predict the UE's future position following the assumption of the limited knowledge about the UEs as defined in Section III-C. The prediction of the UE's movement can be divided into two subcases: i) the UE moving along a straight path and ii) the UE moving along a curved path. In the first subcase, we can simply extrapolate the future movement of the UE from its past movement. However, in the second case, a linear motion extrapolation of the UE's movement would lead to an inaccurate prediction of its position, crossing the environment boundaries such as sidewalks, streets or walls, as shown in Fig. 2. Therefore, after describing the extrapolation of the UE's movement, we also outline how to exploit the knowledge of the environment to obtain a more accurate prediction of the UE's movement.

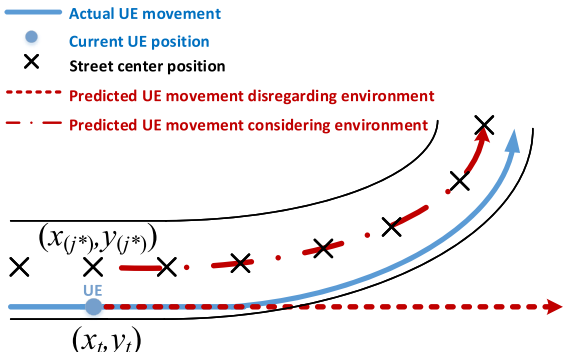


Fig. 2. Example of UE mobility prediction with one degree of mobility freedom following a curved street with known street center positions.

The position of the UE at the discrete time $t$ is represented by the coordinates $(x_t, y_t)$. From the current time instant $t$ and the previous time instant $t - \Delta t$, we obtain the UE's approximated velocity vector $(\Delta x, \Delta y)$ where:

$$\Delta x = \frac{x_t - x_{(t-\Delta t)}}{\Delta t}, \tag{14}$$

$$\Delta y = \frac{y_t - y_{(t-\Delta t)}}{\Delta t}. \tag{15}$$

The predicted UE's position at the time $t + k\Delta t$, where $k = \{1, 2, \ldots, K\}$ and $K\Delta t$ being the prediction window in seconds (typically ranging up to tens of seconds [43]), is calculated as:

$$x_{t+k\Delta t} = x_t + k\Delta x \Delta t \tag{16}$$

$$y_{t+k\Delta t} = y_t + k\Delta y \Delta t. \tag{17}$$

Now we describe an extension of the simple linear extrapolation defined in (16) and (17) by exploiting a knowledge of the environment. Let the UE be located at the position $(x_t, y_t)$ (indicated by a dot in Fig. 2) and let the UE follow a curved street as shown in Fig. 2. In our model, the street is represented by a discrete set of street centers $X \times Y = \{(x_{(j)}, y_{(j)})\}_{j \in J}$, $J \subseteq \mathbb{Z}$, indicated by the crosses in Fig. 2. To exploit the knowledge of the environment, the UE's position is mapped to the closest street center, identified

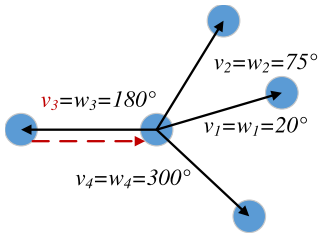


Fig. 3. Example of the UE with multiple degrees of mobility freedom, arriving from angle $v_3$ (red dashed line) with multiple options for the departure angle $w$ (solid lines).

by the index $j^*$ determined as follows:

$$j^* = \underset{j \in J}{\operatorname{argmin}} \sqrt{\left(x_t - x_{(j)}\right)^2 + \left(y_t - y_{(j)}\right)^2}. \quad (18)$$

In Fig. 2, the closest street center to the UE is $(x_{(j^*)}, y_{(j^*)})$. Based on the knowledge of the environment, the UE's position at $t + k\Delta t$ is then mapped to the street center indexed by $j^* + \kappa(k)$ as:

$$x_{t+k\Delta t} = x_{(j^*+\kappa(k))}, \quad (19)$$

$$y_{t+k\Delta t} = y_{(j^*+\kappa(k))}, \quad (20)$$

where $\kappa(k) = \lfloor k\Delta t \frac{\sqrt{\Delta x^2+\Delta y^2}}{\Delta j} \rceil$ approximates the number of street centers run over by the UE during $k$ time instants and $\Delta j$ is the distance between any two consecutive street centers, which we consider constant and can be computed as $\Delta j = \sqrt{(x_{(j+1)} - x_{(j)})^2 + (y_{(j+1)} - y_{(j)})^2}$.

### *B. Mobility Prediction With Multiple Degrees of Movement Freedom*

Now, we extend our mathematical formulation to the case where the UE has multiple degrees of mobility freedom. The set of degrees of freedom for the UE movement is denoted as $W$. This set includes the angles $w$ that the UE can select for its future direction. The set of arrival angles $V$ includes the angles $v$ from which the UE has arrived to the current position. In a general scenario, the UE can select any departure (arrival) angle between $0°$ and $360°$. To limit the complexity, we discretize the angles in a similar way as in [64]. This means that a range of nearby angles is represented by a single departure angle. For example, the discretization with angle difference of $1°$ results in 360 elements (arrival and departure angles) in the both sets $V$ and $W$. An example of the UE with four degrees of freedom, i.e., $|V| = |W| = 4$, is shown in Fig. 3, where the UE arrives from the angle $v_3$ and can depart in the direction of any angle from the set $\{w_1, w_2, w_3, w_4\}$.

In our model, among all the departure angles in $W$, we consider only those with non-zero probability of selection. The adopted probabilistic model is based on Markov chains with underlying Hidden Markov Model (HMM), which is suitable for systems with multiple states and transitions between those states. The important property of Markov chains is that the conditional probability distribution of future states depends solely on the present state. This property is valid for our model as we can legitimately suppose that the departure angle selected by the UE depends only on the arrival angle $v$ and the current UE's position.

The HMM model consists of states and transition probabilities between the states. The states of the HMM model (represented by the departure angles) are learned from an environment layout in the form of a map, (e.g., openstreetmaps.org [65]) in line with [66]. Note that the learning of HMM states can be done either offline in the cloud or in real-time. The offline learning minimizes the delay imposed by the learning. Nevertheless, as the learning of the HMM states is not a computationally intensive task [66], even the real-time learning is suitable for practical use cases. Exploitation of the environment maps for mobility prediction is considered for example in [63] or [64]. Therefore, with the known states, only the transition probabilities (probability of the transition from the arrival to departure angles) of the Markov chain need to be estimated. The transition probabilities represent the probability that each departure angle is chosen [67]. Estimation of the transition probabilities is then done by estimation of the transition probabilities of the Markov chain as described in [67]. To this end, the number of transitions from each arrival angle $v \in V$ to each departure angle $w \in W$ is counted. Note that, if the number of states in unknown, the estimation of the HMM states and transition probabilities is done via the Maximum Likelihood Estimation (MLE) [68].

As the time to learn the transition probabilities between each arrival and departure angle can be high, we consider the transition model aggregated over all the UEs altogether, which reduces the learning time for the estimation of the transition probabilities between the states in the Markov chain. The cost of this aggregation is a slightly lower accuracy of the learned model. However, once enough transitions for each UE are collected, the transition model of the individual UE can be used to replace the aggregated model. It is worth to mention that the learned aggregated transition model still guarantees good results, because the main purpose of the model is to avoid the transitions with very low probabilities. Moreover, by exploiting the aggregated transition model, the transition probability of any UE, including those with unknown transition model, can be predicted.

The probability that the UE at the position $(x_t, y_t)$ selects a departure angle $w$ conditioned by the arrival angle $v$ is denoted $P(w|v, (x_t, y_t))$. This probability, representing the transition probability in the Markov chain [67], is calculated as:

$$P(w|v, (x_t, y_t)) = \frac{N(v, w, (x_t, y_t))}{\sum_{w' \in W} N(v, w', (x_t, y_t))}, \quad (21)$$

where $w \in W$ is the selected departure angle, $N(v, w, (x_t, y_t))$ is the number of transitions from the arrival angle $v$ to the departure angle $w$ at the UE's position $(x_t, y_t)$ summed up till the current time $t$ and for all UEs. Notice that we do not exclude the possibility that the UE stops at the crossroad or departs via the arrival angle (i.e., $w = v$). In this case, the VM placement remains constant, because frequent re-deployments or migrations would overload the network and lead to a disruption in the MEC service. However, the communication path selection is exploited to provide sufficient connectivity considering also channel changes.

Based on the probabilistic model (21), we extend the prediction of mobility by considering the departure angles $w$ with non-zero probability. When, the surrounding environment is unknown, for a given $w$, (16) and (17) are modified as follows:

$$x^w_{t+k\Delta t} = x_t + k\Delta t\sqrt{\Delta x^2 + \Delta y^2}\cos(w), \quad (22)$$

$$y^w_{t+k\Delta t} = y_t + k\Delta t\sqrt{\Delta x^2 + \Delta y^2}\sin(w), \quad (23)$$

where $\Delta x$ and $\Delta y$ are calculated via (14) and (15), respectively. Furthermore, in the case when the movement is predicted with environment knowledge, we extend (18) to:

$$j^*_w = \underset{j\in J_w}{\operatorname{argmin}} \sqrt{\left(x^w_{t+\Delta t} - x_{(j)}\right)^2 + \left(y^w_{t+\Delta t} - y_{(j)}\right)^2}, \quad (24)$$

where $J_w$ is the set of street centers along the departure angle $w$. Note that $j^*_w$ represents the closest street center to the UE's position at the time $t + \Delta t$, when the departure angle is $w$. Accordingly, (19) and (20) are generalized as follows:

$$x^w_{t+k\Delta t} = x_{(j^*_w+\kappa(k-1))}, \quad (25)$$

$$y^w_{t+k\Delta t} = y_{(j^*_w+\kappa(k-1))}, \quad (26)$$

with $\kappa(0) = 0$.

From the estimated positions of the UE at the times $t+k\Delta t$, we calculate the corresponding Euclidean distances to the gNBs. These distances replace the communication data rate as the offloading metric whenever the data rate is unknown or impossible to predict. The predicted Euclidean distance between the UE and the $s$-th gNB located at ($x(s)$, $y(s)$) at the time $t + k\Delta t$ is calculated as:

$$d_{t+k\Delta t}(s, w) = \sqrt{\left(x^w_{t+k\Delta t} - x(s)\right)^2 + \left(y^w_{t+k\Delta t} - y(s)\right)^2}. \quad (27)$$

### C. SINR and Communication Data Rate Prediction

After predicting the UEs' future movement, the communication data rate is calculated based on the estimated future SINR values. Future SINR is predicted either from SINR maps [13] or is extrapolated from the past SINR values if the SINR map is not learned yet. First, we describe the exploitation of the SINR map, and then we describe the extrapolation of the SINR based on the past SINR values.

The SINR map, shared by all the gNBs is represented by a matrix $\Psi$ containing the SINR levels $\Psi_{x,y}$ observed by the UEs at discrete and quantized coordinates $x \in \mathbb{N}$ and $y \in \mathbb{N}$. The SINR map is updated each time when the SINR measurement is received from the UE at the coordinates $(x, y)$ and stored in $\psi_{x,y}$. The update of the SINR map is implemented as a weighted average of the current SINR map value $\Psi_{x,y}$ and the SINR map measurement $\psi_{x,y}$. Thus, the SINR map is updated so that:

$$\Psi_{x,y} \leftarrow \big((1-\chi)\Psi_{x,y} + \chi\psi_{x,y}\big), \quad (28)$$

where $\chi$ is the weight of the new input value to the SINR map. Note that $\chi$ can be optimized based on the performance in a real deployment. Due to the dependency of $\chi$ on the real deployment, we leave the optimization of $\chi$ for future research.

If the SINR map is not learned yet, the SINR is extrapolated based on ARIMA [69], because it enables prediction of non-stationary SINR as required in our case. The SINR is non-stationary due to its time variance caused by varying power levels of received and interference signals.

The generic ARIMA $(P, D, G)$ model is defined by the order of autoregressive part $P$, the degree of the first differencing $D$, the order of the moving average part $G$, and the model parameters: the autoregression $\theta_i$ and the differencing and moving average terms $\phi_i$ (index $i$ indicates terms of autoregression and terms of moving average) [69]. As the SINR does not periodically change values, we leave out the seasoning difference, which is a common part of the generic ARIMA, but it is exploited only if the predicted time series depends on the month, hour, and so on. For our purposes of the SINR level prediction, we define the ARIMA model for SINR prediction as:

$$B^P SINR_t = SINR_{t-P}, \quad (29)$$

$$\phi_i(B) = 1 - \sum_{i=1}^{G} \phi_i B^i, \quad (30)$$

$$\theta_i(B) = 1 - \sum_{i=1}^{P} \theta_i B^i, \quad (31)$$

$$SINR_t = \frac{\theta_G(B)e_t}{\phi_P(1-B)^D}, \quad (32)$$

where $SINR_t$ is the SINR time series, $B^i$ is the lag operator of the $i$-th order, and $e_t$ is the error term of the ARIMA model.

The ARIMA model and the coefficients of autoregression, moving average, and lag operator are estimated from the past samples of SINR by MLE following [69]. Then, the future SINR levels $SINR_{t+\Delta t}$, $SINR_{t+2\Delta t}$, $\ldots$, $SINR_{t+K\Delta t}$ are calculated based on the estimated ARIMA model and the coefficients from (32). The communication data rate is then predicted from SINR levels at times $(t+\Delta t, t+2\Delta t, \ldots, t+K\Delta t)$ via (2). Note that $K$ represents the number of predicted SINR samples.

## V. Proposed Dynamic Communication and Computing Resource Allocation Algorithm

In our previous work [52], we have shown that if a prediction with a fixed accuracy is available and the VMs are pre-allocated on all gNBs, the communication and computing resource allocation can handle offloading of the tasks with the arrival rate $\lambda$ up to 1 task per second in the considered scenario. However, the hypothesis of fixed prediction accuracy is not reasonable for real networks. Thus, we propose an algorithm, denoted Dynamic Communication and Computing Resource Allocation (DCCRA), which exploits the probabilistic UEs' mobility prediction approach described in Section IV.

The DCCRA is composed of two cooperating algorithms: one for the computing and one for the communication resource allocation. The computing part targets a proper VM placement (computing resource allocation) while the communication part

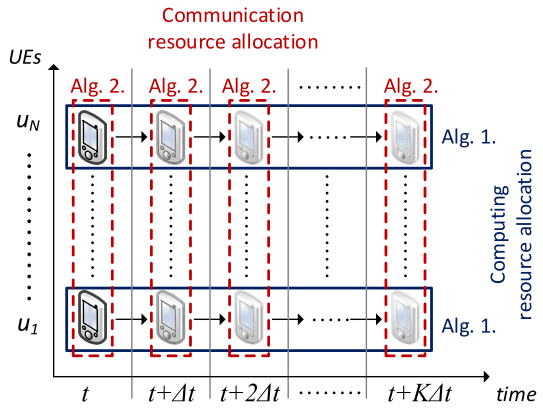


Fig. 4. Cooperation of the proposed DCCRA algorithms.

consists in selection of a proper communication path (communication resource allocation). The cooperation of the proposed algorithms is shown in Fig. 4 for a prediction window of $K\Delta t$. First, the VM placement is determined for each UE via Algorithm 1 over the duration of $K\Delta t$. In parallel, a proper communication path is selected by Algorithm 2 for individual UEs in every time interval $t$. Both algorithms exploit and are joined by the predicted channel quality. The computing resource allocation is done over all gNBs in the proximity of the UE considering the predicted channel quality. At the same time, the communication resource allocation considers the predicted channel quality together with the communication resources availability. Therefore, both low complexity algorithms work together to achieve the joint computing and communication resource allocation. Both parts of the DCCRA are described in the following sections, followed by a complexity analysis.

Note that the computing resources can be allocated either in the form of the VMs or the containers. To simplify the following text, we describe the algorithm for the VMs, but these are interchangeable with the containers in the proposed algorithm. Furthermore, the proposed algorithm is designed for a generic case with multiple degrees of mobility freedom, as described in Section IV-B.

### *A. Computing Resource Allocation*

The computing resource allocation part of the proposed DCCRA algorithm, deciding where and when to allocate the VMs for each UE, is described in this section.

In general, the DCCRA selects the most suitable gNB $s_t^*$ for the placement of computing resources in terms of the VMs. However, in the case with multiple degrees of movement freedom, the VM is pre-allocated on multiple gNBs. Both availability of the computing resources and the quality of all potentially involved communication links are considered. To alleviate the gNBs' backhaul load for the placement of the computing resources, we restrict the list of available links exclusively to the gNBs from set $Q_t(u)$ with which the UE can communicate directly at the time $t$. This restriction leads to a lower overhead, as the SINR information from the gNBs in $S \backslash Q_t(u)$ is not required.

The management of VMs (starting, terminating, and adapting VMs to the actual movement and channel quality of the UEs) is done in a Mobile edge orchestrator (MEO), located in the core network [2], [70]. Whenever the information about the UEs' movement and the channel quality is available, the MEO adapts the VM initialization based on the actual UE's velocity and position. The adaptation of the VM initialization, consists of changing the allocation of the computing resources for the UE to other gNB if the real movement of the UE differs from the predicted one. On the other gNB, the pre-allocated VM is exploited if available, otherwise the VM is started. Furthermore, the MEO terminates VMs that are no longer needed. More details about exploitation of the MEC for various offloaded applications, such as VR or AR are provided in [7]. The computing resources are allocated every $K\Delta t$ seconds to update the VM placement. The value of $K\Delta t$ can be adapted to each environment, e.g., $K\Delta t$ is set to a high value (tens of seconds) in an area with very few crossings whereas in an area with a high number of crossings, such as city center, $K\Delta t$ is set to a low value (few seconds or less). Thus, the computing resource allocation can be dynamically adapted to various environments and various UE's mobility characteristics (walking, in a car, train, etc.).

The process of computing resource allocation is shown in Algorithm 1. The algorithm is designed for the case of UEs with several degrees of mobility freedom (line 2). At first, the UE's velocity vector is predicted (line 1). Then, if the environment is known (line 3), it is exploited to predict the closest street centers (line 4). Afterwards, the computing resources are allocated for every time instant $t + k\Delta t$ until $t + K\Delta t$ (lines 6 to 31). Only the gNBs with enough available computing resources are considered for the VM placement (lines 13 and 14).

In the next steps, the offloading metric $\alpha_\tau^w(s)$ is determined. The offloading metric $\alpha_\tau^w(s)$ is derived from the communication data rate (according to (2)), either from SINR map (line 17) provided that the SINR map is available (line 16), or from SINR predicted from known previous SINR levels by ARIMA (line 23) if SINR can be predicted (line 19). If SINR to the $s$-th gNB cannot be predicted due to a lack of information for the prediction (i.e., if SINR map is not trained or not enough known previous SINR levels are available) the offloading metric $\alpha_\tau^w(s)$ is set based on the distance $d_\tau(s, w)$ defined in (27) (line 20).

The sequence of the gNBs that maximizes the decision metric $\alpha_\tau^w(s)$ is selected for the VM placement $\{s_\tau^*(w)\}_t^{t+K\Delta t}$ (line 30). The sequence $\{s_\tau^*(w)\}_t^{t+K\Delta t}$ is then exploited at the MEO to manage the initialization of the VMs. The management of the VMs includes determination of the time instances when the VM is started ($t_S$) and ended ($t_E$). Between these two times, the VM on the gNBs should be up and running. The time instances $t_S$ and $t_E$ are derived based on $\{s_\tau^*(w)\}_t^{t+K\Delta t}$ and are equal to the first and the last occurrence of $s$ in the sequence $\{s_\tau^*(w)\}_t^{t+K\Delta t}$, respectively. Furthermore, we avoid the pre-allocation of the VM to the gNBs, where the VM would be exploited for less than the VM startup time $t_{VM}$ (line 33). The gNBs for which $t_E - t_S < t_{VM}$ are removed from $\{s_\tau^*(w)\}_t^{t+K\Delta t}$ and these are not considered for the VM

**Algorithm 1** Allocation of Computing Resources

1: Calculate UE's $\Delta x$ and $\Delta y$ velocity via (14) and (15)
2: **for** $w \in W$ such that $P(w|v,(x_t,y_t)) > 0$
3: **if** there exists a known map of street centers
4: Calculate $j_w^*$ via (24) from (22) and (23)
5: **end if**
6: **for** $\tau = t, t+\Delta t, \ldots, t+K\Delta t$
7: **if** there exists a known map of street centers
8: Calculate $(x_\tau^w, y_\tau^w)$ via (25) and (26)
9: **else**
10: Calculate $(x_\tau^w, y_\tau^w)$ via (22) and (23)
11: **end if**
12: **for** $s \in Q_\tau(u)$
13: **if** $\omega_\tau(s) < \omega(u)$
14: $Q_\tau(u) \leftarrow Q_\tau(u) \backslash s$
15: **else**
16: **if** $\Psi_{x,y} \neq 0, \forall \lfloor x_\tau^w \rceil, \lfloor y_\tau^w \rceil$
17: $SINR_\tau(s) \leftarrow \Psi_{x_\tau^w, y_\tau^w}$
18: **else**
19: **if** $SINR_\tau(s)$ is not predictable
20: $\alpha_\tau^w(s) \leftarrow \frac{1}{d_\tau(s,w)}, \forall s \in Q_\tau(u)$
21: break
22: **else**
23: Predict SINR by (32)
24: **end if**
25: **end if**
26: Calculate $c_\tau(u,s)$ via (2)
27: $\alpha_\tau^w(s) \leftarrow c_\tau(u,s)$
28: **end if**
29: **end for**
30: $s_\tau^*(w) \leftarrow \text{argmax}_{s \in Q_\tau(u)} \alpha_\tau^w(s)$
31: **end for**
32: **end for**
33: Remove $s$ with VM exploited for less than $t_{VM}$

placement. Instead, already running VMs are exploited to handle the offloading. Thus, the computing load of the gNBs is decreased and the gNBs can be exploited for the VMs of the other UEs.

An example of the Algorithm 1 determining VM pre-allocation is shown in Fig. 5. In this example, three gNBs ($\text{gNB}_1$, $\text{gNB}_2$, $\text{gNB}_3$) and one UE are located on a crossroad with three possible future directions $w_1$, $w_2$, and $w_3$. For each direction and time step, the gNBs are ordered according to $\alpha$ (see table in the middle part in Fig. 5). Then, $t_S$ and $t_E$ are determined as the first and the last occurrence of each gNB in the first row of the table in Fig. 5 over all departure angles.

### B. Selection of Communication Path

To further reduce the offloading delay, we propose also an algorithm reducing the communication delays $t_O$ and $t_C$. The algorithm forces the UE to perform handover to the gNB that provides the fastest delivery of the offloaded task to the VM considering radio as well as backhaul data rates. The algorithm is inspired by our previous work, Path Selection with Handover (PSwH) algorithm [29], [30]. The PSwH maximizes

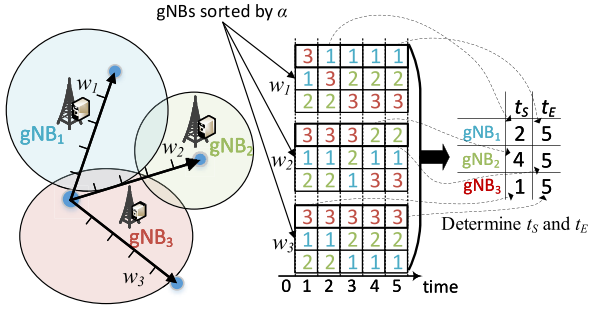


Fig. 5. Example of VM placement by Algorithm 1 for three gNBs (number of rows in each table in the middle part of the figure) over five time instants (columns in each table in the middle part of the figure). For each departure angle, represented by individual table, a sequence (row) of the gNBs maximizing $\alpha$ is chosen and then exploited to determine $t_S$ and $t_E$ for each gNB.

the communication data rate of the UEs. However, in the PSwH, the UEs do not cooperate and the algorithm does not consider the prediction of the channel quality for resource allocation. Thus, we propose the algorithm that efficiently handles the rapid changes in the UEs' communication data rates. The selection of the communication paths for the UEs is made by an iterative update of the serving gNBs every $\Delta t$ as shown in Algorithm 2, assuming fixed $s_t^*$ for every $u$ during given time interval $\langle t, t+\Delta t \rangle$. The communication resource allocation by the PSwH can be implemented in to the mobile networks via network slicing and orchestration in a similar way as indicated in [71].

The algorithm for selection of communication path starts with a determination of the serving gNBs based on the SINR of the UEs (line 1). Then, the current data rates in uplink (following (4)) and downlink (by adapting (2)) are derived from the known SINR and from the number of connected UEs. Then, a set of the gNBs $\hat{S}$ is created by sorting the gNBs in descending order based on their radio communication load $n_t^R(s)$ (line 3). The algorithm then goes through the gNBs in $\hat{S}$ that have more than one connected UE (lines 4 and 5). Four variables are defined for the communication path selection: i) minimal gain of handover to avoid exploiting handover for the UEs with a minor data rate improvement $\epsilon$, ii) the UE $u^H$ with the highest benefit from handover to any gNB (set initially to 0, see (line 6)), iii) the gNB $s^H$ selected by the UE $u^H$ as a candidate for the handover (set to 0 in initial phase), and iv) maximal achievable handover gain of all the UEs $\beta$ (also set to 0 in initial phase). The algorithm iteratively searches for the UE, which can benefit the most from handover, i.e., the UE that maximizes $\beta$ (lines 9 to 17). The auxiliary handover gain $\beta^{\text{a}}$ (exploited to find $\beta$) is determined for each pair of the UE and the gNB that can communicate with SINR above $SINR_{min}$ (lines 9 and 10). The gain is defined as the difference between the achievable communication data rates (in both uplink and downlink) when the UE is connected to its serving gNB $s_t$ and to a target gNB from the set $S \backslash s_t$ (line 11). If $\beta^{\text{a}}$ is higher than $\beta$, $u^H$ and $s^H$ are updated. Then, if $\beta$ is equal to or higher than the threshold $\epsilon$, the UE $u^H$ is handed over to the gNB $s^H$ (line 19) and the UEs' data rates are updated (line 20).

**Algorithm 2** Allocation of Communication Resources

Determine serving gNBs maximizing SINR.
Calculate the uplink and downlink data rates $c_t^{UL}(u,s,s^*)$ and $c_t^{DL}(u,s,s^*)$ for each UE and gNB.
Sort the gNBs in descending order based on $n_t^R(s)$ to $\hat{S}$
**for** $\hat{s} \in \hat{S}$
  **if** $n_t^R(\hat{s}) > 1$
    Initialize $u^H = 0$, and $s^H = 0$.
    **while** (true)
      $\beta = 0$
      **for** $u \in U$ such that $s_t = \hat{s}$
        **for** $s \in Q_\tau(u)$
          $\beta^{\mathrm{a}} = min\{c_t^{UL}(u,s,s^*) - c_t^{UL}(u,\hat{s},s^*), c_t^{DL}(u,s,s^*) - c_t^{DL}(u,\hat{s},s^*)\}$
          **if** $\beta^{\mathrm{a}} > \beta$
            $u^H = u$, $s^H = s$.
            $\beta = \beta^{\mathrm{a}}$
          **end if**
        **end for**
      **end for**
      **if** $\beta \geq \epsilon$
        $s_t(u^H) = s^H$
        Update $c_t^{UL}(u,s,s^*)$ and $c_t^{DL}(u,s,s^*)$.
      **else**
        break
      **end if**
    **end while**
  **end if**
**end for**

### C. Complexity

The minimization of the offloading delay by the joint selection of the VM placement and the communication path leads to a combinatorial formulation. The total complexity of the DCCRA for $N$ UEs and considering $|W|$ is $\mathcal{O}(N|S||Q_t|K + N|W||Q_t|K)$, where $|Q_t| = \max_{u \in U} |Q_\tau(u)|$. The state of the art algorithm presented in [48], further denoted as VM Online Approximation Placement (VM-OAP) algorithm, has complexity $\mathcal{O}(N|S|^2K)$. However, the VM-OAP algorithm is designed only for one degree of freedom. When $|W| = 1$ and all gNBs being considered for the path selection, i.e., $|Q_t| = |S|$ (worst case scenario), the DCCRA is of the same complexity as the VM-OAP algorithm in the worst case.

## VI. Simulation Scenario and Models

In this section, we describe simulation models and scenarios for performance evaluation carried out in MATLAB. The main simulation parameters, presented in Table I, are in line with recommendations for mobile networks with small cells as defined by 3GPP in [72]. We also follow the specifications of the physical layer and frame structure parameters for LTE-A mobile networks defined in the same document. The signal propagation over radio channel is modeled according to 3GPP [72] with path loss model $PL = 128.1 + 37.6log_{10}d$, where $d$ is the distance between the UE and the gNB. We consider the mapping function between SINR and MCS defined in [61] for Bit Error Rate (BER) of 10 %. The minimal SINR required to enable communication, $SINR_{min}$, is set to $-6.9$ dBm, according to [61]. We set the weighing factor for SINR map updates, $\chi$, to 0.5 so that the SINR changes due to varying environment are quickly propagated in our model. Note that, in a real deployment, $\chi$ can be adjusted based on the environment. The backhaul of the gNBs is modeled as an optical fiber with randomly generated available capacities following a normal distribution with the mean $\mu = 100$ and the variance $\sigma^2 = 2$ (in Mbit/s).

Since we target the real-time applications, the offloaded tasks with sizes of 20 and 200 kB are considered [73]. Moreover, the offloaded task with a size of 2000 kB is investigated as well, to show performance even for larger tasks. The VM startup time $t_{VM}$, representing the time required to initialize the VM and to prepare it to process the offloaded tasks is 4.5 s for the VMs [32]. This corresponds to the time between the moment when the VM pre-allocation begins to the moment when the offloaded application is run. Note that the $t_{VM}$ contributes to the offloading delay only when the VM is not prepared on the gNB on time. The radio and backhaul resources are allocated to the UEs by round-robin scheduling.

The simulation area, as shown in Fig. 6, represents a part of Prague, Czech Republic. The environment is similar to the one

TABLE I
Simulation Parameters

| Parameter | Value |
|---|---|
| Simulation area | 650m x 370m |
| Carrier frequency | 2 GHz |
| Bandwidth of uplink/downlink | 10/10 MHz |
| Tx power of gNB/SCgNB/UE ($S_{Tx}$) | 27/15/10 dBm |
| $SINR_{min}$ | -6.9 dB |
| Weighting factor | 0.5 |
| Number of gNB/SCgNB | 4/30 |
| VM startup time $t_{VM}$ | 4.5 s |
| Prediction window $K$ | 200 |
| ARIMA number of past samples | 20 |
| Offloaded task size $L_{\mathrm{O}}$ = results size $L_{\mathrm{C}}$ | 20/200/2000 kB |
| Offloaded task number of instructions $L_{\mathrm{P}}$ | 1e6 instructions |
| gNB/SCgNB CPU | 3300 MIPS |
| Shadowing factor | 6 dB |
| Handover interruption duration $t_{HO}$ | 30 ms |
| Threshold $\epsilon$ | 100 kbit/s |
| Number of UEs | 30/60/90 |
| Speed of users | 1 m/s |
| Backhaul capacity – Normal distribution | $\mu$=100, $\sigma^2$=2 |
| Simulation time/Number of simulation drops | 3 600 s/ 20 drops |
| Simulation step | 100 ms |


Fig. 6. Simulation model with deployment of gNBs and SCgNBs.

in [64], where the authors consider arrival and departure angles difference of 90°. In this area, four gNBs (represented by blue discs in Fig. 6) are deployed according to the real position of the gNBs of a mobile operator [74]. In addition, 30 small cell base stations (SCgNBs), divided into two sets with different transmission frequencies are randomly deployed (denoted as orange crosses in Fig. 6). To show the impact of network load on the performance, 30, 60, and 90 UEs are randomly dropped into the simulation area. The UEs follow the realistic mobility model with crossroad direction probabilities defined in [74]. The number of UEs per gNB in our scenario is about 0.9, 1.8, and 2.7, which is higher than that in [48] (where they assume roughly 0.55 UEs per gNB). We consider also the higher UEs' density, i.e., 60 and 90 UEs, to evaluate the performance with highly loaded gNBs (MEC servers). Thus, we show the behaviour of the proposed algorithm in even more challenging scenarios. Furthermore, the bandwidth for the wireless communication is 10 MHz in uplink and 10 MHz in downlink allocated in Frequency Division Duplex manner. We exploit a common handover procedure based on SINR, as described in [75], to keep the UE connected to the gNB with the highest SINR. In the simulations, the UEs move with the same speed. This might be seen as an optimistic assumption, however, the proposed DCCRA takes into account the speed of UE via (9) and (10). Thus, the DCCRA can handle easily different speeds of UEs without any degradation of performance.

The energy consumption model of the UEs follows an empirical model defined in [76]. This energy consumption model considers the power consumption of the UE communication modem being turned on $P_{ON} = 853$ mW, the uplink communication power $P_{UL}$, and the downlink communication power $P_{DL}$. Both uplink ($P_{UL}$) and downlink ($P_{DL}$) communication powers consist of the signal processing parts $P_{TxBB}$ and $P_{RxBB}$, the radio parts $P_{TxRF}$ and $P_{RxRF}$, and the consumption of the transmitter (in uplink) and receiver (in downlink) circuits $P_{TxON}$ and $P_{RxON}$, respectively. Hence, the power consumed for the uplink communication ($P_{UL}$) is calculated as:

$$P_{UL} = P_{TxON} + P_{TxRF} + P_{TxBB} \text{ [mW]}, \tag{33}$$

where $P_{TxON} = 29.9$ mW, $P_{TxBB} = 0.62$ mW, and $P_{TxRF}$ is calculated as:

$$P_{TxRF} = \begin{cases} 0.78S_{Tx} + 23.6 & S_{Tx} \leq 0.2 \\ 17S_{Tx} + 45.4 & 0.2 < S_{Tx} < 11.4 \\ 5.9S_{Tx}^2 - 118S_{Tx} + 1195 & 11.4 < S_{Tx}, \end{cases} \tag{34}$$

where $S_{Tx}$ is the transmission power of the UE in dBm.

The power consumed for downlink communication ($P_{DL}$) is calculated as:

$$P_{DL} = P_{RxON} + P_{RxRF} + P_{RxBB} \text{ [mW]}, \tag{35}$$

where $P_{RxON} = 25.1$ mW, and $P_{RxBB}$ is calculated as:

$$P_{RxBB} = 0.97R_{Rx} + 8.16 \text{ [mW]}, \tag{36}$$

where $R_{Rx}$ is the downlink throughput in Mbit/s, and $P_{RxRF}$ is calculated as:

$$P_{RxRF} = \begin{cases} -0.04S_{Rx} + 24.8 & S_{Rx} \leq -52.5 \\ -0.11S_{Rx} + 7.86 & S_{Rx} > -52.5, \end{cases} \tag{37}$$

where $S_{Rx}$ is the power received at the UE from the gNB in dBm.

The energy consumption of the UE is then calculated by multiplying the required power by the transmission time:

$$E = E_{UL} + E_{DL} = P_{ON}(t_{\mathrm{O}} + t_{\mathrm{C}} + t_H) + \\ + P_{DL}t_{\mathrm{C}} + P_{UL}t_{\mathrm{O}} \text{ [J]}. \tag{38}$$

## VII. Performance Evaluation and Discussion of Results

The performance of the proposed algorithm (DCCRA) is compared with two state-of-the-art approaches:

- *Serving Only (SO)* according to [77] - where the VM is kept at the serving gNB, so the VM is migrated each time handover is performed.
- *VM Online Approximation Placement (VM-OAP)* according to [48] – where the VM placement is based on predicted future costs (in terms of channel quality) of its placement.

In addition to these two competitive solutions, we also show the performance of the DCCRA under perfect mobility and channel quality prediction with the VM pre-allocation on only one gNB (denoted as DCCRA-perfect in the paper) to see potential improvement if the prediction would be ideal.

In Fig. 7, we show the mean offloading delay over the task arrival rate for 30 UEs with the offloaded task size of 20 kB (Fig. 7a), 200 kB (Fig. 7b), and 2000 kB (Fig. 7c). Note that the mean offloading delay consists of the time consumed to deliver the offloaded task to the MEC, the time required to process the offloaded task, the time spent by a delivery of the processed data back to the UE, the time consumed due to handover process and the time of the VM starts. In each figure, we see that a higher arrival rate generally leads to a higher offloading delay because more offloaded tasks per second are generated and more communication and computation resources are consumed. Furthermore, it is shown that increasing the offloaded task size results in a higher offloading delay, as more data have to be transmitted. The SO algorithm supports offloading up to $\lambda = 0.2$ for $L_{\mathrm{O}} = L_{\mathrm{C}}$ up to 200 kB. The SO algorithm does not enable a higher $\lambda$ as it does not exploit any prediction or pre-allocation of the VMs, thus, the resources become unavailable even for a very light computation load. The VM-OAP algorithm enables $\lambda$ up to 2 for $L_{\mathrm{O}} = L_{\mathrm{C}} = 20$ kB. However, for a higher $L_{\mathrm{O}}$ and $L_{\mathrm{C}}$, the VM-OAP algorithm can handle $\lambda$ only up to 0.33. The VM-OAP exploits channel quality prediction, but it does not pre-allocate the VMs. The DCCRA outperforms both compared algorithms by enabling the offloading of the tasks with $\lambda$ up to 5 for $L_{\mathrm{O}} = L_{\mathrm{C}}$ up to 200 kB, and $\lambda$ up to 0.5 for $L_{\mathrm{O}} = L_{\mathrm{C}} = 2000$ kB. This means that the DCCRA enables offloading with almost twice higher $\lambda$ than the VM-OAP. Comparing the DCCRA to the DCCRA perfect, we can see that the performance of both is very similar and

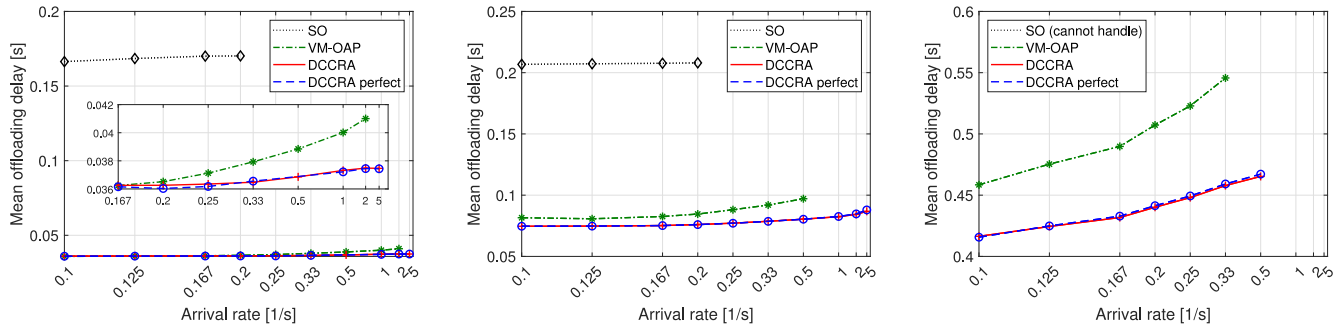


Fig. 7. Mean times required to offload, compute, and collect results of the offloaded task for 30 UEs with $L_O = L_C = 20$ kB (a), $L_O = L_C = 200$ kB (b), and $L_O = L_C = 2000$ kB (c).

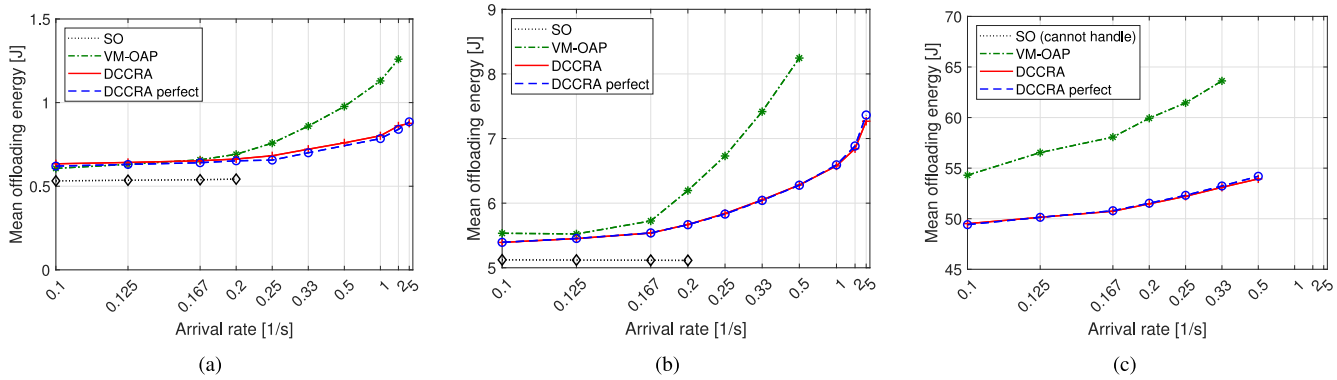


Fig. 8. Mean offloading energy for 30 UEs with $L_O = L_C = 20$ kB (a), $L_O = L_C = 200$ kB (b), and $L_O = L_C = 2000$ kB (c).

the ideal prediction does not lead to any notable reduction in the offloading delay.

The proposed DCCRA reduces the offloading delay by up to 78 % comparing to the SO algorithm for $\lambda = 0.2$. In comparison to the VM-OAP, the DCCRA reduces the offloading delay by 8.2 % for $L_O = L_C = 20$ kB and $\lambda = 2$. Furthermore, increasing $L_O = L_C$ to 2000 kB leads to an increased gain (15.2 % for $\lambda = 0.33$) of the DCCRA in comparison to the VM-OAP algorithm. The offloading delay reduction is achieved by optimizing the placement and pre-allocation of the VMs, as well as the selection of the communication path. Increasing $L_O = L_C$ leads to a higher offloading delay for all compared algorithms, but the DCCRA keeps the offloading delay below 100 ms for small sized tasks ($L_O = L_C$ below 2000 kB).

Fig. 8 shows the mean energy consumed by the UEs for the transmission of the offloaded task and the reception of the computing results with the offloaded task size of 20 kB (Fig. 8a), 200 kB (Fig. 8b), and 2000 kB (Fig. 8c). In all these figures, any increase in $\lambda$ or $L_O = L_C$, leads to an increase in the energy consumed per the offloaded task, because the network load rises. The higher energy consumption is caused by increased communication time due to the communication and computing load of the gNBs, and the relation between the energy and the transmission time (see (38)). In comparison to the SO algorithm, the proposed DCCRA increases the consumed energy by less than 23 %, however, it is only 0.1 J, for $L_O = L_C = 20$ kB and 8.3 %, i.e., 0.5 J, for $L_O = L_C = 200$ kB. Note that this increase is largely compensated by a significant reduction in the offloading delay by up to 78 % and by enabling the offloading of tasks with $L_O = L_C = 2000$ kB, as shown in Fig. 7. Furthermore, the DCCRA reduces the energy consumed for the offloading by up to 35 % compared to the VM-OAP algorithm. The reduction in the offloading energy is achieved by avoiding the overloaded communication paths and by pre-allocation of the VMs. Avoiding the overloaded communication links is done by the proposed selection of communication path, while the VMs are pre-allocated to minimize the delay of VM startup (migration). Again, we see that the DCCRA provides a similar performance as the DCCRA perfect.

In Fig. 9, we show the mean offloading delay over the task arrival rate for 60 UEs with the offloaded task size of 20 kB (Fig. 9a), 200 kB (Fig. 9b), and 2000 kB (Fig. 9c). The results follow the same trends as shown in Fig. 7 for 30 UEs. The DCCRA increases the gain in comparison to the VM-OAP to 31 % for $\lambda = 0.33$ and $L_O = L_C = 2000$ kB. This is caused by the fact that the DCCRA balances the computing and communication loads among the gNBs.

Similar changes due to the increased number of UEs are seen in mean offloading energy, as shown in Fig. 10 with the offloaded task size of a 20 kB (Fig. 10a), 200 kB (Fig. 10b), and 2000 kB (Fig. 10c). However, a higher offloading delay leads to an increased energy consumption. The DCCRA leads to a similar energy consumption as the DCCRA perfect for $L_O = L_C = 200$ kB or less. In the case of

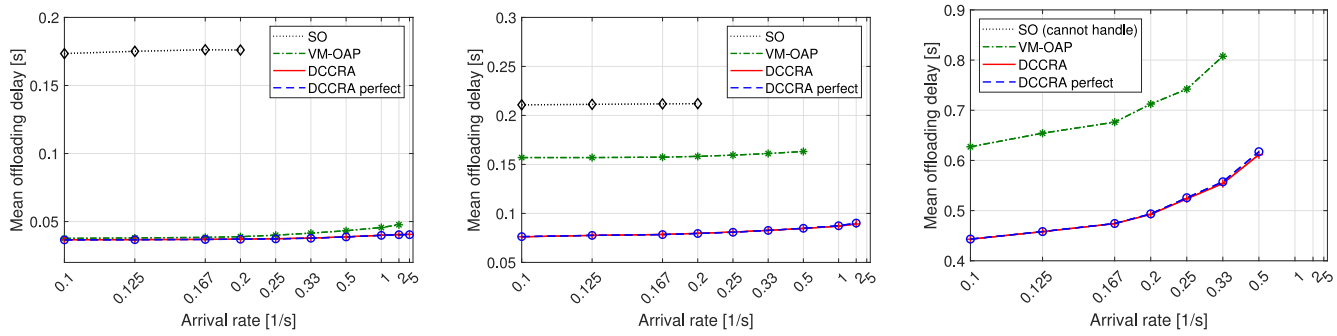


Fig. 9. Mean times required to offload, compute, and collect results of the offloaded for 60 UEs with $L_O = L_C = 20$ kB (a), $L_O = L_C = 200$ kB (b), and $L_O = L_C = 2000$ kB (c).

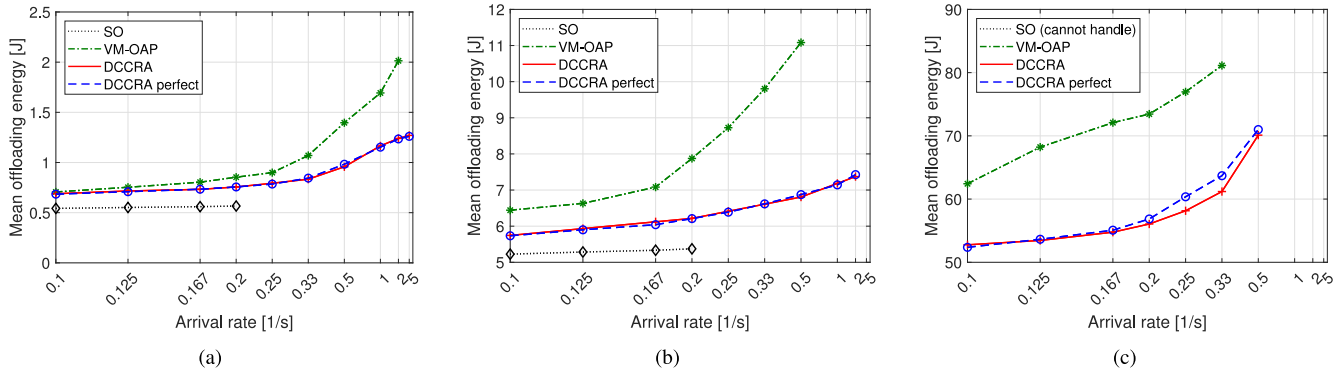


Fig. 10. Mean offloading energy for 60 UEs with $L_O = L_C = 20$ kB (a), $L_O = L_C = 200$ kB (b), and $L_O = L_C = 2000$ kB (c).

$L_O = L_C = 2000$ kB, the DCCRA slightly lowers the energy consumption with respect to the DCCRA perfect. This is caused by the pre-allocation of slightly more VMs for each UE by the DCCRA comparing to the DCCRA perfect. These additional pre-allocated VMs by the DCCRA are exploited to avoid the overloaded gNBs. Note that the DCCRA perfect does not predict the number of connected UEs, thus, the predicted data rate as well as the overloading of the gNBs is not predicted perfectly. Therefore, the DCCRA provides a minor improvement over the DCCRA perfect, but at the cost of pre-allocating a higher number of VMs.

The mean offloading delay for 90 UEs is shown in Fig. 11 with the offloaded task size of 20 kB (Fig. 11a), 200 kB (Fig. 11b), and 2000 kB (Fig. 11c). The increased number of UEs, again, leads to an increased offloading delay. The SO algorithm cannot handle the offloading for 90 UEs due to keeping the VM on the serving gNB. Furthermore, the VM-OAP cannot handle the offloading for 90 UEs and $L_O = L_C$ above 20 kB, as shown in Fig. 11b and Fig. 11c, as it does not exploit pre-allocation. The DCCRA enables offloading with $\lambda$ equal to 5, 2, and 0.5 for $L_O = L_C$ equal to 20, 200, and 2000 kB, respectively. From Fig. 7, Fig. 9, and Fig. 11, we see that the DCCRA keeps the offloading delay for small offloaded tasks (below 200 kB) under 100 ms, which is not possible with any of the competitive algorithms.

The energy consumed for the offloading for 90 UEs is shown in Fig. 12 with the offloaded task size of 20 kB (Fig. 12a), 200 kB (Fig. 12b), and 2000 kB (Fig. 12c). Again, the increased number of the UEs leads to an increased energy consumption. The results for the SO algorithm are not shown, as the algorithm cannot handle such high number of UEs. Furthermore, the VM-OAP is shown only for $L_O = L_C = 20$ kB, as it cannot handle larger offloaded task sizes with 90 UEs. The DCCRA consumes slightly less energy than the DCCRA perfect for $L_O = L_C = 2000$ kB due to the same reason as for 60 UEs (Fig. 10c).

The mean amount of data transmitted over the backhaul due to delivery of the offloading task to the computing VM and collection of the results at the UE is shown in Fig. 13a for 20 kB, in Fig. 13b for 200 kB, and in Fig. 13c for 2000 kB. Since the SO algorithm places the VMs exclusively on the serving gNB, no data is transmitted over the backhaul. Thus, the SO is not included in these figures. For the VM-OAP algorithm, the amount of data transmitted over the backhaul is constant over all investigated task arrival rates for all numbers of UEs, and for all offloaded task sizes. For the proposed DCCRA, the amount of data transmitted over the backhaul is slightly decreasing with increasing $\lambda$. This is caused by the need for a closer placement of the VMs to minimize the communication delay when the time between two consecutive offloaded tasks is low (i.e., for a high $\lambda$). The proposed algorithm transmits 40 % less data over the backhaul comparing to the VM-OAP. This reduction is achieved by allocating the VMs in a proximity of the UEs to reduce the offloading delay and to alleviate the backhaul communication load. The DCCRA perfect transmits slightly more data over the backhaul comparing to the

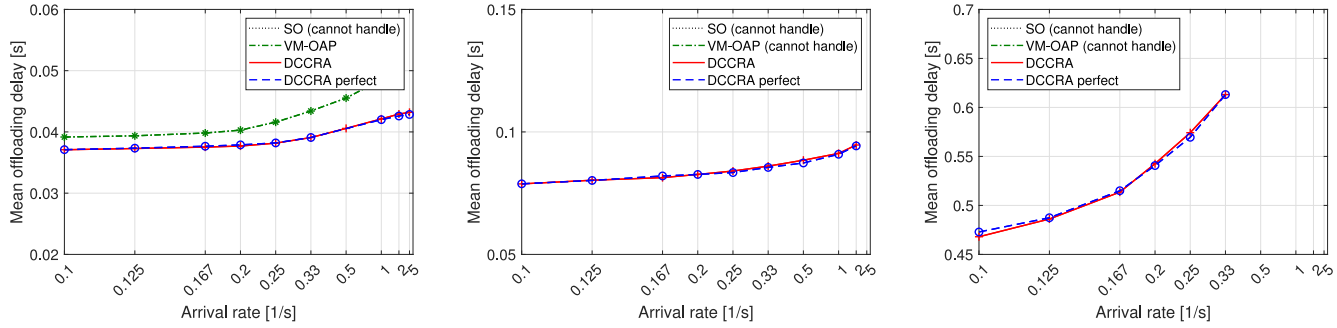


Fig. 11. Mean times required to offload, compute, and collect results of the offloaded for 90 UEs with $L_O = L_C = 20$ kB (a), $L_O = L_C = 200$ kB (b), and $L_O = L_C = 2000$ kB (c).

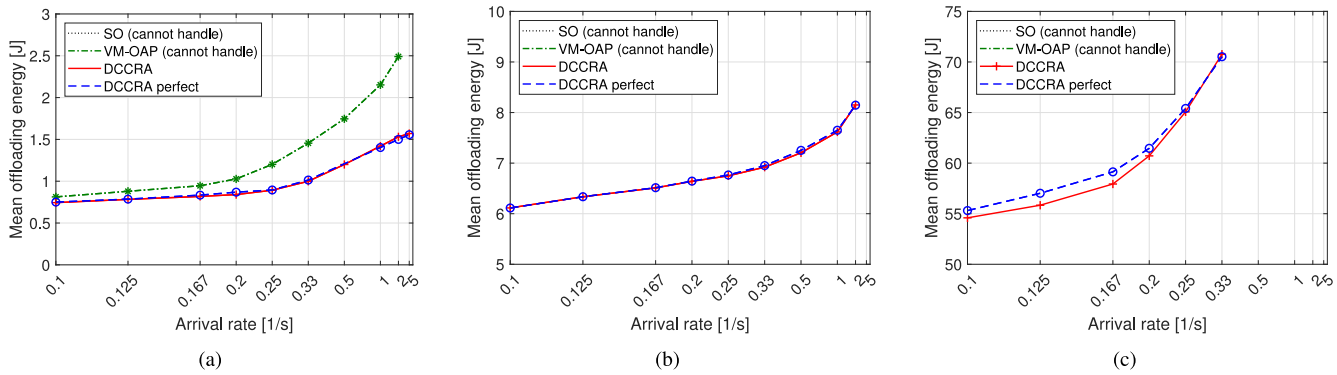


Fig. 12. Mean offloading energy for 90 UEs with $L_O = L_C = 20$ kB (a), $L_O = L_C = 200$ kB (b), and $L_O = L_C = 2000$ kB (c).

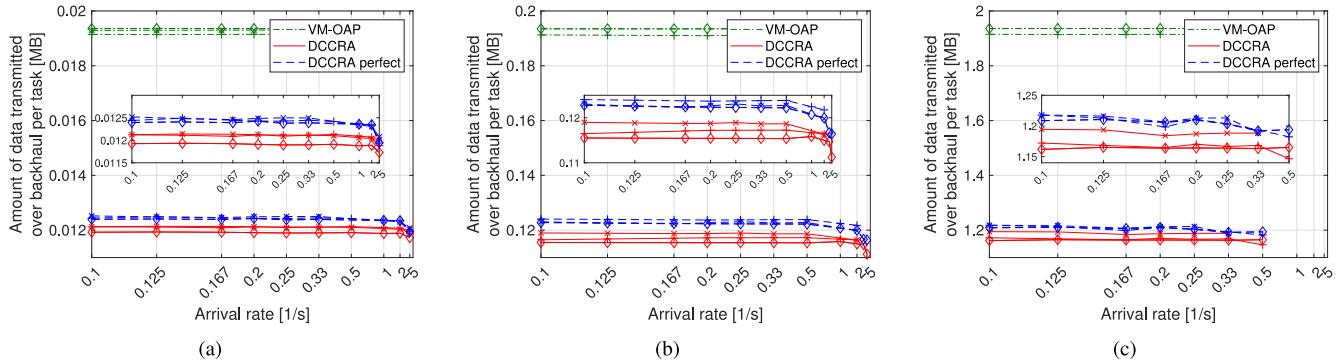


Fig. 13. Amount of data transmitted over backhaul per task for $L_O = L_C = 20$ kB (a), $L_O = L_C = 200$ kB (b), and $L_O = L_C = 2000$ kB (c), diamond marker ($\diamond$) represents 30 UEs, plus marker ($+$) 60 UEs, and x marker ($x$) 90 UEs.

DCCRA. This is caused by the pre-allocation of the VM on a lower number of the gNBs, as shown in Fig. 14. Comparing the impact of the size of offloaded task (i.e., comparing sub-figures Fig. 13a, Fig. 13b, and Fig. 13c), we can see that the amount of data transmitted over the backhaul is increasing proportionally to the offloaded task size. Note that the lines for 30, 60 and 90 UEs are partially overlapping. This is expected, as we show amount of data transmitted over backhaul per one offloaded task.

The number of VMs deployed for all UEs during the simulation run is shown in Fig. 14. The sub-figures represent results for the tasks with a size of 20 kB (Fig. 14a), 200 kB (Fig. 14b), and 2000 kB (Fig. 14c), respectively. To provide an insight into performance of our prior work [52], the number of pre-allocated VMs is equal to the number of UEs (60) multiplied by the number of gNBs (34), i.e., 2040 VMs in our scenario. This number is many times higher than the number of gNBs where the VM is pre-allocated by the DCCRA, thus, we do not show the lines for 2040 VMs in the figure. Since only our proposed algorithm exploits the possibility to deploy more than one VM per UE, other algorithms are not depicted. The SO and VM-OAP algorithms deploy the same number of VMs as the number of UEs offloading their tasks, i.e., 30 VMs for 30 UEs, 60 VMs for 60 UEs, and 90 VMs for 90 UEs. To show the impact of mobility and channel prediction, we compare the DCCRA to the DCCRA perfect for $\lambda = 2$ with 30, 60, and 90 UEs. From the Fig. 14, we see that just after the simulation starts, there is a steep increase in the number

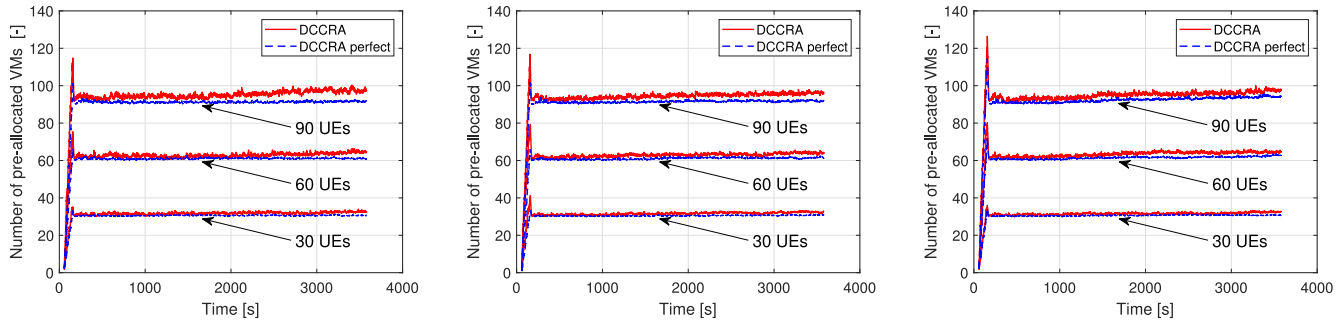


Fig. 14. Number of pre-allocated VMs for the proposal during the simulation run. for $L_O = L_C = 20$ kB (a), $L_O = L_C = 200$ kB (b), and $L_O = L_C = 2000$ kB (c), solid line represents 30 UEs, dashed 60 UEs and dotted 90 UEs.

of the deployed VMs, as the number of UEs offloading their tasks increases. However, when all the UEs are offloading their tasks, the number of deployed VMs stabilizes at 32.4 VMs for 30 UEs, 65 VMs for 60 UEs, and 98 VMs for 90 UEs. In case of the DCCRA perfect, the number of deployed VMs is 31 VMs for 30 UEs, 61.5 VMs for 60 UEs, and 93 VMs for 90 UEs. On the average, there are 1.08 and 1.03 VMs pre-allocated per UE for the DCCRA and the DCCRA perfect, respectively, for all sizes of the offloaded tasks. The difference in the number of pre-allocated VMs between the DCCRA and the DCCRA perfect is caused by the need to pre-allocate more VMs for the DCCRA to compensate for the mobility and channel prediction inaccuracies. The minor fluctuation in the number of pre-allocated VMs over time is caused by the fact that the UEs are selecting from multiple future angles at irregular time instants. With pre-allocation of 8 % more VMs than the number of UEs, the DCCRA enables the offloading of the real-time task with very high arrival rate.

## VIII. Conclusion

In this paper, we have proposed a novel algorithm for dynamic allocation of computing and communication resources for Multi-Access Edge Computing. The algorithm dynamically allocates VMs considering the computation load of gNBs and selects the best communication path between the UE and the gNB with allocated VM. For the proposed algorithm, we have designed a suitable mobility channel prediction with a low complexity.

Comparing to state of the art approaches, the proposed algorithm reduces the offloading delay by up to 64 %, while reducing the UE's energy consumption by up to 39 %. The proposed algorithm enables offloading of tasks with arrival rate up to 5 tasks per second per UE for small task sizes. The competitive algorithms do not surpass 2 and 0.5 tasks per second for very small and small task sizes. The proposed algorithm, also provides offloading delay below 100 ms for small sized offloaded tasks, making it suitable for real-time offloading. Furthermore, we show that the performance of the proposed algorithm is similar to the case with perfect mobility and channel prediction.

Future research should consider to develop a distributed solution for the communication and computing resource allocation and the solution should be extended towards optimization of the energy consumption.

## References

[1] 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 Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1657–1681, 3rd Quart., 2017.

[2] "Mobile edge computing (MEC); Framework and reference architecture," Eur. Telecommun. Stand. Inst. (ETSI), Sophia Antipolis, France, Rep. GS MEC 003, 2016.

[3] S. S. Qureshi, T. Ahmad, K. Rafique, and S. Ul Islam, "Mobile cloud computing as future for mobile applications—Implementation methods and challenging issues," in *Proc. IEEE Int. Conf. Cloud Comput. Intell. Syst.*, 2011, pp. 467–471.

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing a key technology towards 5G," Sophia Antipolis, France, ETSI, White Paper, 2015.

[5] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, 2013.

[6] A. Huang, N. Nikaein, T. Stenbock, A. Ksentini, and C. Bonnet, "Low latency MEC framework for SDN-based LTE/LTE-A networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2017, pp. 1–6.

[7] "Mobile edge computing (MEC); Technical requirements," Eur. Telecommun. Stand. Inst. (ETSI), Sophia Antipolis, France, Rep. GS MEC 002, 2018.

[8] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.

[9] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.

[10] W. Sun, J. Liu, Y. Yue, and H. Zhang, "Double auction-based resource allocation for mobile edge computing in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4692–4701, Oct. 2018.

[11] B. I. Ismail *et al.*, "Evaluation of Docker as edge computing platform," in in *Proc. IEEE Conf. Open Syst. (ICOS)*, 2015, pp. 130–135.

[12] Y. Wang, W. Shi, and M. Hu, "Virtual servers co-migration for mobile accesses: Online versus off-line," *IEEE Trans. Mobile Comput.*, vol. 14, no. 12, pp. 2576–2589, Dec. 2015.

[13] Z. Li, M. Kihl, Q. Lu, and J. A. Andersson, "Performance overhead comparison between hypervisor and container based virtualization," in *Proc. IEEE Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, 2017, pp. 955–962.

[14] K.-T. Seo, H.-S. Hwang, I.-Y. Moon, O.-Y. Kwon, and B.-J. Kim, "Performance comparison analysis of linux container and virtual machine for building cloud," *Adv. Sci. Technol. Lett.*, vol. 66, pp. 105–111, Dec. 2014.

[15] D. Bernstein, "Containers and cloud: From LXC to Docker to Kubernetes," *IEEE Cloud Comput.*, vol. 1, no. 3, pp. 81–84, Sep. 2014.

[16] J. Dolezal, Z. Becvar, and T. Zeman, "Performance evaluation of computation offloading from mobile device to the edge of mobile network," in *Proc. IEEE Conf. Stand. Commun. Netw. (CSCN)*, 2016, pp. 60–66.

[17] J. Oueis, E. C. Strinati, and S. Barbarossa, "Multi-parameter decision algorithm for mobile computation offloading," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, vol. 3, 2014, pp. 3005–3010.

[18] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3056–3069, Nov. 2017.

[19] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? The bandwidth and energy costs of mobile cloud computing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2013, pp. 1285–1293.

[20] F. Lobillo *et al.*, "An architecture for mobile computation offloading on cloud-enabled LTE small cells," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, 2014, pp. 1–6.

[21] M. A. Puente, Z. Becvar, M. Rohlik, F. Lobillo, and E. C. Strinati, "A seamless integration of computationally-enhanced base stations into mobile networks towards 5G," in *Proc. IEEE 81st Veh. Technol. Conf. (VTC Spring)*, vol. 2015, 2015, pp. 1–5.

[22] F. Yu, H. Chen, and J. Xu, "DMPO: Dynamic mobility-aware partial offloading in mobile edge computing," *Future Gener. Comput. Syst.*, vol. 89, pp. 722–735, Dec. 2018.

[23] W. Zhan, C. Luo, G. Min, C. Wang, Q. Zhu, and H. Duan, "Mobility-aware multi-user offloading optimization for mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3341–3356, Mar. 2020.

[24] H. Ke, J. Wang, L. Deng, Y. Ge, and H. Wang, "Deep reinforcement learning-based adaptive computation offloading for MEC in heterogeneous vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7916–7929, Jul. 2020.

[25] S. Yu, X. Chen, L. Yang, D. Wu, M. Bennis, and J. Zhang, "Intelligent edge: Leveraging deep imitation learning for mobile edge computation offloading," *IEEE Wireless Commun.*, vol. 27, no. 1, pp. 92–99, Feb. 2020.

[26] Z. Wang, Z. Zhao, G. Min, X. Huang, Q. Ni, and R. Wang, "User mobility aware task assignment for mobile edge computing," *Future Gener. Comput. Syst.*, vol. 85, pp. 1–8, Aug. 2018.

[27] Z. Zhao *et al.*, "Mobility management with transferable reinforcement learning trajectory prediction," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2102–2116, Dec. 2020.

[28] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Live service migration in mobile edge clouds," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 140–147, Feb. 2018.

[29] Z. Becvar, J. Plachy, and P. Mach, "Path selection using handover in mobile networks with cloud-enabled small cells," in *Proc. IEEE 25th Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, vol. 2014, 2014, pp. 1480–1485.

[30] J. Plachy, Z. Becvar, and P. Mach, "Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network," *Comput. Netw.*, vol. 108, Oct. 2016, pp. 357–370.

[31] N. Wang and J. Wu, "Opportunistic WiFi offloading in a vehicular environment: Waiting or downloading now?" in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[32] K. Razavi, G. V. D. Kolk, and T. Kielmann, "Prebaked $\mu$VMs: Scalable, instant VM startup for IaaS clouds," in *Proc. IEEE 35th Int. Conf. Distrib. Comput. Syst.*, vol. 2015, Jul. 2015, pp. 245–255.

[33] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile cloud computing," in *Proc. IEEE Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2014, pp. 354–358.

[34] S. Sardellitti, G. Scutari, and S. Barbarossa, "Distributed joint optimization of radio and computational resources for mobile cloud computing," in *Proc. IEEE Int. Conf. Cloud Networking (CloudNet)*, 2014, pp. 211–216.

[35] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7432–7445, Aug. 2017.

[36] I. Farris, T. Taleb, M. Bagaa, and H. Flick, "Optimizing service replication for mobile delay-sensitive applications in 5G edge network," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2017, pp. 1–6.

[37] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[38] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[39] Y. Pan, M. Chen, Z. Yang, N. Huang, and M. Shikh-Bahaei, "Energy-efficient NOMA-based mobile edge computing offloading," *IEEE Commun. Lett.*, vol. 23, no. 2, pp. 310–313, Feb. 2019.

[40] S. Yu, B. Dab, Z. Movahedi, R. Langar, and L. Wang, "A socially-aware hybrid computation offloading framework for multi-access edge computing," *IEEE Trans. Mobile Comput.*, vol. 19, no. 6, pp. 1247–1259, Jun. 2020.

[41] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Trans. Emerg. Topics Comput.*, early access, Mar. 4, 2019, doi: 10.1109/TETC.2019.2902661.

[42] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

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

[44] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3887–3901, Dec. 2016.

[45] A. Ksentini, T. Taleb, and M. Chen, "A Markov decision process-based service migration procedure for follow me cloud," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2014, pp. 1350–1354.

[46] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proc. IEEE/ACM Int. Conf. Clust. Cloud Grid Comput.*, 2010, pp. 826–831.

[47] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *Proc. IFIP Netw. Conf.*, 2015, pp. 1–9.

[48] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1002–1016, Apr. 2017.

[49] Z. Gao, Q. Jiao, K. Xiao, Q. Wang, Z. Mo, and Y. Yang, "Deep reinforcement learning based service migration strategy for edge computing," in *Proc. IEEE Int. Conf. Service-Oriented Syst. Eng. (SOSE)*, 2019, pp. 116–1165.

[50] M. V. Barbera, S. Kosta, A. Mei, V. C. Perta, and J. Stefa, "Mobile offloading in the wild: Findings and lessons learned through a real-life experiment with a new cloud-aware system," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2014, pp. 2355–2363.

[51] P. Papadimitriou, O. Maennel, A. Greenhalgh, A. Feldmann, and L. Mathy, "Implementing network virtualization for a future Internet," in *Proc. 20th ITC Specialist Seminar Netw. Virtualization Concept Perform. Aspects*, 2009.

[52] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *Proc. IEEE Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, 2016, pp. 1–6.

[53] G. Yu and J. Wu, "Content caching based on mobility prediction and joint user prefetch in mobile edge networks," *Peer-to-Peer Netw. Appl.*, vol. 13, no. 5, pp. 1839–1852, 2020.

[54] C.-L. Wu, T.-C. Chiu, C.-Y. Wang, and A.-C. Pang, "Mobility-aware deep reinforcement learning with glimpse mobility prediction in edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–7.

[55] A. Rago, P. Ventrella, G. Piro, G. Boggia, and P. Dini, "Towards an optimal management of the 5G cloud-ran through a spatio-temporal prediction of user's dynamics," in *Proc. Mediterr. Commun. Comput. Netw. Conf. (MedComNet)*, 2020, pp. 1–4.

[56] U. Fattore, M. Liebsch, B. Brik, and A. Ksentini, "AutoMEC: LSTM-based user mobility prediction for service management in distributed MEC resources," in *Proc. Int. ACM Conf. Model. Anal. Simulat. Wireless Mobile Syst.*, 2020, pp. 155–159.

[57] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 360–374, Jan. 2021.

[58] "Technical specification group technical specification group core network and terminals; Handover procedures (Release 13)," 3GPP, Sophia Antipolis, France, Rep. TS 23.009, 2015.

[59] J. Oueis, E. Calvanese-Strinati, A. De Domenico, and S. Barbarossa, "On the impact of backhaul network on distributed cloud computing," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, 2014, pp. 12–17.

[60] M. Jaber, M. A. Imran, R. Tafazolli, and A. Tukmanov, "5G backhaul challenges and emerging research directions: A survey," *IEEE Access*, vol. 4, pp. 1743–1766, 2016.
[61] C. Mehlführer, J. C. Ikuno, M. Šimko, S. Schwarz, M. Wrulich, and M. Rupp, "The Vienna LTE simulators—Enabling reproducibility in wireless communications research," *EURASIP J. Adv. Signal Process.*, vol. 29, no. 1, p. 29, 2011.
[62] W. Sun, J. Liu, and Y. Yue, "AI-enhanced offloading in edge computing: When machine learning meets industrial IoT," *IEEE Netw.*, vol. 33, no. 5, pp. 68–74, Sep./Oct. 2019.
[63] N. Samaan and A. Karmouch, "A mobility prediction architecture based on contextual knowledge and spatial conceptual maps," *IEEE Trans. Mobile Comput.*, vol. 4, no. 6, pp. 537–551, Nov./Dec. 2005.
[64] A. Nadembega, A. Hafid, and T. Taleb, "A destination and mobility path prediction scheme for mobile networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 6, pp. 2577–2590, Jun. 2015.
[65] *OpenStreetMap*, OpenStreetMap Found., Cambridge, U.K., 2013.
[66] Y. I. Parish and P. Müller, "Procedural modeling of cities," in *Proc. ACM Special Interest Group GRAPHics Interact. Techn. (SIGGRAPH)*, 2001, pp. 301–308.
[67] D. Stynes, K. N. Brown, and C. J. Sreenan, "A probabilistic approach to user mobility prediction for wireless services," in *Proc. Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, 2016, pp. 120–125.
[68] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Royal Stat. Soc. B, Methodol.*, vol. 39, no. 1, pp. 1–22, 1977.
[69] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. Hoboken, NJ, USA: Wiley, 2015.
[70] S. Kekki *et al.*, "MEC in 5G networks," Sophia Antipolis, France, ETSI, White Paper, 2018.
[71] A. Reid *et al.*, "OSM deployment and integration," OSM, Sophia Antipolis, France, White Paper, 2020.
[72] "Evolved universal terrestrial radio access (E-UTRA); Further advancements for E-UTRA physical layer aspects," 3GPP, Sophia Antipolis, France, Rep. 36.814, 2010.
[73] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proc. Int. Conf. Mobile Syst. Appl. Services (MobiSys)*, 2014, pp. 68–81.
[74] Z. Becvar, M. Vondra, and P. Mach, "Dynamic optimization of neighbor cell list for femtocells," in *Proc. IEEE Veh. Technol. Conf. (VTC Spring)*, 2013, pp. 1–6.
[75] V. M. Nguyen, C. S. Chen, and L. Thomas, "Handover measurement in mobile cellular networks: Analysis and applications to LTE," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2011, pp. 1–6.
[76] M. Lauridsen, L. Noël, T. B. Sørensen, and P. Mogensen, "An empirical LTE smartphone power model with a view to energy efficiency evolution," *Intel® Technol. J.*, vol. 18, no. 1, pp. 172–193, 2014.
[77] K. Wang, M. Shen, J. Cho, A. Banerjee, J. Van der Merwe, and K. Webb, "MobiScud: A fast moving personal cloud in the mobile network," in *Proc. Workshop All Things Cell. Oper. Appl. Challenges*, 2015, pp. 19–24.

**Jan Plachy** (Student Member, IEEE) received the B.Sc. and M.Sc. degrees in telecommunication engineering from the Czech Technical University in Prague, Czech Republic, in 2012 and 2014, respectively, where he is currently pursuing the Ph.D. degree with the Department of Telecommunication Engineering, with a topic Allocation of Communication and Computation Resources in Mobile Networks.

He was on internships with CEA-Leti, France, in 2014, EURECOM, France, in 2016 and 2019, and Bar-Ilan University, Israel, in 2017. In 2015, he has joined the 5G Mobile Research Lab funded by the Czech Technical University of Prague, focusing on key aspects and challenges related to future mobile networks and emerging wireless technologies. He has coauthored seven conference papers and four journal papers. His research interests cover optimization of mobility management and radio resource management of both communication and computing resources in future mobile networks.

**Zdenek Becvar** (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in telecommunication engineering from the Czech Technical University in Prague, Czech Republic, in 2005 and 2010, respectively.

He is currently an Associate Professor with the Department of Telecommunication Engineering, Czech Technical University in Prague. From 2006 to 2007, he joined Sitronics R&D Center, Prague, focusing on speech quality in VoIP. Furthermore, he was involved in research activities of Vodafone R&D center, Czech Technical University in Prague in 2009. He was on internships with Budapest Politechnic, Hungary, in 2007, CEA-Leti, France, in 2013, and EURECOM, France, in 2016. From 2013 to 2017, he was a representative of the Czech Technical University in Prague in ETSI and 3GPP standardization organizations. In 2015, he founded 5Gmobile Research Lab, CTU in Prague, focusing on research towards 5G mobile networks and beyond. He is a member of more than 20 program committees at international conferences or workshops and he has published four book chapters and more than 70 conference or journal papers. He works on development of solutions for future mobile networks with special focus on optimization of mobility and radio resource management, device-to-device communication, self-optimization, power control, architecture of radio access network, and small cells.

**Emilio Calvanese Strinati** (Member, IEEE) received the master's degree in engineering from the University of Rome 'La Sapienza' and the Ph.D. degree in engineering science in 2005.

He started working with the Motorola Labs, Paris, in 2002. In 2006, he joint CEA/LETI as a Research Engineer. In 2007, he became a Ph.D. Supervisor and since 2018 holds the French Research Director Habilitation (HDR). Since February 2021, he has been the Director of the Nano Electronic and Wireless for 6G Initiative, dedicated to the required convergence between microelectronic and telecom, hardware and software, network and equipment for upcoming 6G technologies. He has published around 150 papers in international conferences, journals and books chapters, given more than 200 international invited talks, keynotes and tutorials. He is the main inventor or coinventor of more than 65 patents. He has organized more than 100 international conferences, workshops, panels and special sessions on green communications, heterogeneous networks and cloud computing hosted in international conferences, such as IEEE GLOBCOM, IEEE PIMRC, IEEE WCNC, IEEE ICC, IEEE VTC, EuCnC, IFIP, EUCNC, and European Wireless. He is the General Chair of EUCNC 2022.

**Nicola di Pietro** received the B.S. degree in mathematics from the University of Padova, Italy, in 2008, the M.S. degree in mathematics jointly from the University of Padova, Italy, and the University of Bordeaux, France, in 2010, within the framework of the international ALGANT program, and the Ph.D. degree in mathematics from the University of Bordeaux, France, in 2014. He is currently a System Engineer with Athonet, Italy. During the years of his doctoral studies, he was a Research Engineer with the European R&D Center, Mitsubishi Electric, Rennes, France. From 2014 to 2016, he was an Associate Postdoctoral Fellow with Texas A&M University, Qatar. From 2017 to 2021, he was a Research Engineer with CEA LETI, Grenoble, France. His research interests are information theory, lattice error-correcting codes, caching, and edge cloud computing for 5G networks.