Mobility Management for Computation-Intensive Tasks in Cellular Networks with SD-RAN

Anna Prado, Zifan Ding, Fidan Mehmeti, Wolfgang Kellerer
Chair of Communication Networks, Technical University of Munich, Germany.
{anna.prado, zifan.ding, fidan.mehmeti, wolfgang.kellerer}@tum.de

***Abstract*—With the rapid increase in the amount of exchanged traffic over cellular networks, stemming partly from computation-intensive tasks, and the highly mobile nature of the users, mobility management exhibits considerable challenges in next-generation cellular networks. A way to alleviate these problems is by using Software-defined Radio Access Networks (SD-RAN), where a centralized controller with a complete overview of the network topology (distribution of users across base stations and their channel conditions) can make decisions on the user assignment and resource allocation. To that end, in this paper, we formulate an optimization problem with the objective of maximizing the network utility, where computation-intensive tasks are sent from the users to edge clouds, taking into account the communication constraints (uplink and downlink bandwidth) as well as the finite storage and processing capabilities of edge clouds. Moreover, we provide a user rate guarantee to satisfy an additional application for all users. The problem is NP-hard, therefore, we propose to use Deep Reinforcement Learning (DRL) to solve it. Extensive realistic simulations show that our approach is close to the optimal solution, where the latter is obtained using a solver, while outperforming a benchmark by up to 65%.**

***Index Terms*—Handover, mobility management, SD-RAN, 5G.**

I. Introduction

The traffic over 5G cellular networks has experienced a tremendous increase in the last years [1]. This is partly due to the rapid increase of the number of smartphones and other high-tech gadgets, offering services that are data-intensive. Such examples are Augmented Reality (AR), Virtual Reality (VR), Extended Reality (XR), online gaming, live video streaming [2], etc.

Coping with the traversal of this large amount of data through a network is not straightforward because of the finite resources, despite the fact that in 5G there are more resources available. Besides the Radio Access Network (RAN) resources, i.e., the resources needed to transmit/receive those data, the aforementioned applications require considerable computational power as well. For example, when running some AI-related tasks a large amount of data have to be processed. Therefore, in-network computing [3] has become one of the features of today's cellular networks, with data being processed at the entities known as *edge clouds*. Usually, edge clouds are collocated with Base Stations (BSs).

An additional issue is the high mobility of the users, which requires changing frequently over time the serving BS. Given the huge amount of data that need to be uploaded, which may take seconds or minutes, the users would be changing their service BSs multiple times. Therefore, the users will be sending the (large) data to different BSs, and therefore multiple edge clouds. So, the important question that arises is *how are those data going to be assembled and processed?*

In the next step, the data need to be processed, which will take some time depending on their processing requirement, the capacity of the edge clouds, and the number of competing users for these resources. After this has been done, the results would need to be downloaded to the user. As it has been some time since the user sent the request and data, it is to be expected that it has changed the position (or experiences a blockage of the Line of Sight (LoS) to the serving BS) and would at that time be assigned to another BS compared to the time it started uploading its data. In case there are a large amount of data associated with the results, the user while downloading them would be changing the serving BSs, which adds to the challenge of impeccable network operation.

A potential way to alleviate the aforementioned problems is by using Software-defined Radio Access Networks (SD-RANs), which are the adaptation of Software-defined Networks (SDNs) [4] in RAN. SD-RAN represent a paradigm shift in the operation of cellular networks in general, and in how resource management is handled, in particular. In SD-RAN, the control is decoupled from the data plane and transferred to a centralized unit, known as *SD-RAN controller*. With this feature, the controller can dynamically assign the Physical Resource Blocks (PRBs)[1] to BSs, with the instructions/decisions on how to further allocate them to the users within the coverage area of each BS.

The increased flexibility in resource allocation brought by SD-RAN can be exploited in improving the mobility management in cellular networks. Namely, while the data-intensive task is being uploaded, performing handovers would not be good as then the remaining part of task would need to be sent to another edge cloud after the user is handed over to the next BS, increasing thus the complexity of the operation. On the other hand, having a centralized knowledge of all the network (which the SD-RAN controller does), to avoid frequent handovers, the controller has an extra degree of freedom by increasing the amount of resources (in terms of the number of PRBs) that can be assigned to a user while uploading the task and while downloading results. This would lead to fewer interruptions and would reduce both signaling and data

[1]PRBs are the unit of resource allocation in 5G.

information exchange, leaving more resources available for serving more users or providing better level of service of the existing ones.

There are some important questions that arise related to the mobility management for data-intensive tasks with SD-RAN:

- If successfully serving a task brings some utility to the operator, how does one decide which tasks to serve when there are not enough resources, given the heterogeneous network requirements of the task, so that the total network utility is maximized?
- What resource allocation policy should be followed both in terms of the RAN and edge cloud resources?

To answer these questions, in this paper we formulate an optimization problem, which has the time component, whose objective is to maximize the overall network utility by deciding which tasks to serve (each task has its own utility), given the limited network resources in terms of the upload bandwidth, download bandwidth, storage in edge clouds, and finite computational capacity, whose solution would determine the resource allocation and user assignment policy when handling data-intensive tasks. Given the complexity of the problem, we then propose a Deep Reinforcement Learning (DRL) approach, which yields solid performance. Our approach is particularly important for network operators in deciding which (computation-intensive) tasks to admit and how to allocate the resources. The main message is that one cannot directly use the objective with penalties for constraint violations to train a well-performing DRL agent. Instead, the reward function needs to be carefully designed to guide the agent.

Specifically, our main contributions in this work are:

- We propose using SD-RAN to handle the problem of communication, computation, and mobility management in a cellular network when dealing with data-intensive tasks. To that end, we propose an optimization problem with the time component whose objective is to maximize the total utility while taking into account the possible constraints in terms of communication resources (both in uplink and downlink) and computation resources.
- As the problem exhibits exponential complexity, we propose an approach relying on DRL, which is shown to perform very closely to the optimal solution, considerably outperforming state of the art.
- We provide directions on how this approach can be implemented in practice.

The remainder of this paper is organized as follows. In Section II, the formulation of the optimization problem is given. This is followed by a detailed description of the proposed approach in Section III. In Section V, the performance of our approach is assessed and compared against state of the art that is presented previously in Section IV. Section VI presents some related work. Finally, Section VII concludes the paper.

II. Problem Formulation

In this section, we present the system model first and then provide the optimization formulation.

TABLE I: Notation

Variable	Explanation
$\mathcal{U}$	Set of all users in the network
$\mathcal{B}$	Set of all Base Stations (BSs) in the network
T	Simulation time (slots)
U_u	The utility of user's u task
a_u	The arrival time of user's u task
d_u	The deadline of user's u task
s_u	The storage requirement (data size) of user's u task
K_u	The computation requirement of user's u task
s'_u	The result size of user's u task
r_u	The minimum required application rate of user u
$R_{u,b}$	Per PRB rate of user u at BS b
$\eta_{u,b}$	Overhead due to handover of user u to BS b
d_u^{UL}	The upload time of user's u task
d_u^{CP}	The computational processing time of user's u task
d_u^{DL}	The download time of user's u result
$\sigma_u(t)$	The amount of task data of user u sent at slot t
$\sigma'_u(t)$	The amount of results of user u sent at slot t
$\kappa_u(t)$	The amount of computing resources allocated to user u at slot t
C_b	The total computational capacity of BS b
S_b	The total storage capacity of BS b
K_b^{PRB}	The total number of PRBs at BS b
N_b^{CP}	The total number of computational resources at BS b
$x_{u,b}$	User u to BS b assignment decision variable
z_u	Processing of user's u task decision variable
$y_{u,b}$	Number of PRBs that user u obtains from BS b for uploading the task
$y'_{u,b}$	Number of PRBs that user u obtains from BS b for downloading the task
$y_{u,b}^{app}$	Number of PRBs that user u obtains from BS b for its application rate r_u
$k_{u,b}$	The amount of computational resources that user u receives from edge cloud b for processing its task

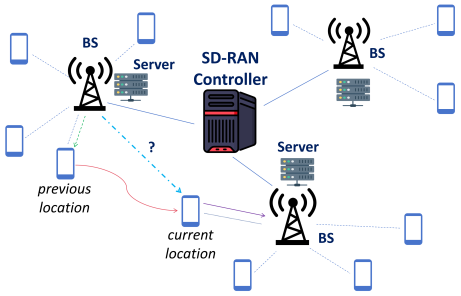


Fig. 1: Illustration of the system model.

A. System Model

The network consists of multiple BSs, which are managed by a *central controller*. The sets $\mathcal{U}$ and $\mathcal{B}$ denote the sets of all users and BSs (macro and micro) in the network, respectively. The problem is considered over a time horizon of a duration T time slots. The system model is depicted in Fig. 1. Mobile users move across the area covered by the network and require two types of services: edge computing services to solve computation-intensive tasks and constant bit rate services, such as video calls.

Each user can have at most one task that they want to be processed. Serving each task consists of the following three steps: (1) uploading the data to one of the BSs/edge clouds; (2) processing the task at the edge cloud; and (3) sending the

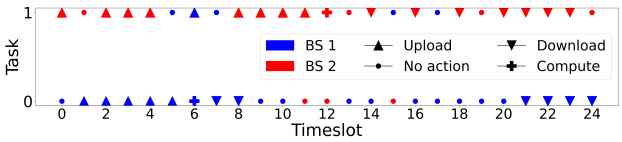


Fig. 2: Serving the task: upload, processing, and download.

result data back to the user. We assume that edge clouds are co-located with the BSs. For task processing, the entire task must be uploaded before processing can begin. This is known as *batch processing* [5]. For instance, a complete video upload is necessary before it can be analyzed. We allow tasks to span multiple time slots, reflecting a realistic scenario.

The arrival time of the task is denoted as a_u; its deadline, by which the task must be served, is d_u. The utility achieved if a task is completely processed by its deadline is U_u. Task u has a storage requirement (or task data size) of s_u, which is split into small chunks of a fixed size that are transmitted one by one. Task's total computational requirement is K_u, and a result size that needs to be sent back to the user after processing is s'_u (that is also split into fixed size chunks). The total storage capacity of the edge cloud located at BS b is denoted as S_b, while its total computational capacity is C_b units per slot.

The network manages the following resources: (1) bandwidth resources (PRBs) at macro and micro BSs that are shared between uplink and downlink transmissions; (2) storage resources at edge clouds to store the task until the processing can start and the result is sent back to the user; and (3) computing resources at edge clouds. The BSs are connected among each other and can forward the data instantly. This is especially beneficial if a user has to switch from one BS to another, the process called a *handover*, where user's data need to be forwarded from the previous BS to the target one.

The wireless resource unit, a *PRB* is defined as 12 consecutive subcarriers in the frequency domain and one slot in the time domain. We denote by $R_{u,b}$ the rate user $u \in \mathcal{U}$ receives from every allocated PRB from BS $b \in \mathcal{B}$. As such, $R_{u,b}$ depends on the Signal to Noise Ratio (SNR)[2] of the user, the type of BS b and its bandwidth. We calculate $R_{u,b}$ using Shannon's formula, as in [6], [7]. Additionally to tasks, the focus is on a constant bit rate application as well. Therefore, the aim is to ensure that each user receives a minimum data rate, denoted as r_u that satisfies the traffic requirements.

Computation resources are measured as the amount of data that can be processed per unit of time (expressed in this work as MB/s), while the storage resources are defined as the required storage for the task to run on the edge cloud. We assume that the total computational and storage resources can be split into a finite number of units of integer size, which can be then allocated to a task. Table I summarizes the notation used throughout this paper.

[2]We refer to it as SNR, and not Signal to Interference plus Noise Ratio (SINR) because the reuse factor is 1 in our simulations.

B. Handover Approaches

To the best of our knowledge, the 3GPP standard does not define when handovers should be performed in applications such as computational intensive task processing. Several questions may rise, e.g., shall a user be forced to stay connected to its serving BS until the task is completed or shall the user be able to switch to another BS at any time? In this work, we consider three scenarios regarding the handover execution. A handover is possible 1) only after the entire task is completed, 2) after a part of the task (e.g., uploading of s_u or processing) is completed, or 3) at any time (e.g., after a chunk of data s_u was uploaded or chunk of s'_u was downloaded, and at any time during computing). We evaluate these scenarios in Section V, but one can already notice that in millimetre Waves (mmWaves) forcing the user to stay connected not to the best BS might result in a Radio Link Failure (RLF). Moreover, depending on the task size and available resources, the task processing can take a long time during which a user might leave the coverage of its serving BS or obtain a LoS to another BS, thus, also a better SNR. Next, in this section, we formulate an optimization problem for the latter case, when a handover is possible at any time (i.e., after transmitting a chunk of data). We omit problem formulations for two other cases due to space limitation. An example of serving two tasks with chunk-based handover policy is presented in Fig. 2, where user 1 executes a handover from BS 2 to BS 1 at time slot $t = 5$ while uploading their data. Then, the user switches back to BS 2 at $t = 8$. This handover was necessary because the user experiences a blockage of the LoS and could not communicate with BS 2.

C. Optimization Formulation

Next, we introduce four decision variables: (i) $x_{u,b}(t)$ that states whether user u is assigned to BS b; (ii) $z_u(t)$ that states whether the task of user u is being processed in time slot t; (iii) $k_{u,b}(t)$ that denotes the amount of computing resources allocated to task u by edge cloud b in slot t; (iv) $y_{u,b}(t)$ and $y'_{u,b}(t)$ indicate the amount of PRBs allocated for task uploading and downloading, respectively; (v) $y^{app}_{u,b}(t)$ represents the amount of PRBs allocated to satisfy the running application's required rate; and (vi) d^{UL}_u, d^{CP}_u, and d^{DL}_u, whose values represent the time it takes to upload and process the task, as well as to download the result, respectively.

The optimization formulation is similar to [5]. Nevertheless, differently from [5], we consider mobility management, studying the impact of handovers, as well as we provide a guarantee of an additional sensitive-rate service. The indicator variable $\theta_u(t)$ reserves the necessary storage for a task since the start of its upload until it is completely processed:

$$\theta_u(t) = \begin{cases} 1, & \text{if } \min\{a_u + t_t | \kappa_u(a_u + t_t) > 0\} \leq t \\ & \leq \max\{a_u + t_t | \kappa_u(a_u + t_t) > 0\} \\ 0, & \text{otherwise,} \end{cases} \quad (1)$$

where t_t is a time slot when the task is completely processed ($0 \leq t_t \leq d_u - 1$).

In line with the objective of this paper, the formulation of the optimization problem is as follows:

$$\max \sum_{u=1}^{|U|} U_u z_u \tag{2}$$

$$\text{s.t.} \quad \sum_{b=1}^{|B|} x_{u,b}(t) = 1, \quad \forall u \in U, \forall t \in T, \tag{3}$$

$$\sum_{b=1}^{|B|} x_{u,b} y_{u,b}^{app} R_{u,b}(1-\eta_{u,b}) \geq r_u, \quad \forall u \in U, \forall t \in T, \tag{4}$$

$$\sum_{t=1}^{T} \sigma_u(t) = s_u z_u, \quad \forall u \in U, \tag{5}$$

$$\sum_{t=1}^{T} \kappa_u(t) = K_u z_u, \quad \forall u \in U, \tag{6}$$

$$\sum_{t=1}^{T} \sigma'_u(t) = s'_u z_u, \quad \forall u \in U, \tag{7}$$

$$\sigma_u(t) = \sum_{b=1}^{|B|} x_{u,b} z_u y_{u,b} R_{u,b}(1-\eta_{u,b}), \quad \forall u \in U, \forall t \in T, \tag{8}$$

$$\sigma'_u(t) = \sum_{b=1}^{|B|} x_{u,b} z_u y'_{u,b}(t) R_{u,b}(1-\eta_{u,b}), \quad \forall u \in U, \forall t \in T, \tag{9}$$

$$\eta_{u,b}(t) = (1 - x_{u,b}(t-1))\frac{T_{HIT}}{T_{slot}}, \quad \forall b \in B, \forall u \in U, \forall t \in T \backslash \{0\}, \tag{10}$$

$$\kappa_u(t) = \frac{\sum_{b=1}^{|B|} x_{u,b} z_u k_{u,b} C_b}{N_b^{CP}}, \quad \forall u \in U, \forall t \in T, \tag{11}$$

$$\sum_{u=1}^{|U|} x_{u,b} z_u s_u \theta_{p,u} \leq S_b, \quad \forall b \in B, \forall t \in T, \tag{12}$$

$$\sum_{u=1}^{|U|} x_{u,b} z_u k_{u,b} \leq N_b^{CP}, \quad \forall b \in B, \forall t \in T, \tag{13}$$

$$\sum_{u=1}^{|U|} \left(x_{u,b} z_u (y_{u,b} + y'_{u,b}) + x_{u,b} y_{u,b}^{app} \right) \leq K_b^{PRB}, \quad \forall b \in B, \forall t \in T, \tag{14}$$

$$x_{u,b}(t) \in \{0,1\}, \quad \forall b \in B, \forall u \in U, \forall t \in T, \tag{15}$$

$$z_u(t) \in \{0,1\}, \quad \forall u \in U, \forall t \in T, \tag{16}$$

$$k_{u,b}(t) \in \left\{0,1,\cdots,N_b^{CP}\right\}, \quad \forall u \in U, \forall t \in T, \tag{17}$$

$$y_{u,b}(t) = \left\{0,1,\cdots,N_b^{PRB}\right\}, \quad \forall b \in B, \forall u \in U, \forall t \in T, \tag{18}$$

$$y'_{u,b}(t) = \left\{0,1,\cdots,N_b^{PRB}\right\}, \quad \forall b \in B, \forall u \in U, \forall t \in T, \tag{19}$$

$$y_{u,b}^{app}(n) = \left\{0,1,\cdots,N_b^{PRB}\right\}, \quad \forall b \in B, \forall u \in U, \forall t \in T, \tag{20}$$

$$\eta_{u,b}(t=0) = 0, \quad \forall b \in B, \forall u \in U, \tag{21}$$

$$\sigma_u(t) = 0, \quad \forall u \in U, t = \left\{1,\cdots,a_u-1, a_u+d_u^{UL},\cdots,T\right\}, \tag{22}$$

$$\sigma_u(t) \geq 0, \quad \forall u \in U, t = \left\{a_u,\cdots,a_u+d_u^{UL}-1\right\}, \tag{23}$$

$$\kappa_u(t) = 0, \quad \forall u \in U, \; t = \left\{1,\cdots,a_u+d_u^{UL}-1, a_u+d_u^{UL}+d_u^{CP},\cdots,T\right\}, \tag{24}$$

$$\kappa_u(t) \geq 0, \quad \forall u \in U, \; t = \left\{a_u+d_u^{UL},\cdots,a_u+d_u^{UL}+d_u^{CP}-1\right\}, \tag{25}$$

$$\sigma'_u(t) = 0, \quad \forall u \in U, \; t = \left\{1,\cdots,a_u+d_u^{UL}+d_u^{CP}-1, a_u+d_u,\cdots,T\right\}, \tag{26}$$

$$\sigma'_u(t) = 0, \quad \forall u \in U, \; t = \left\{1,\cdots,a_u+d_u^{UL}+d_u^{CP}-1, a_u+d_u,\cdots,T\right\}, \tag{27}$$

$$\sigma'_u(t) \geq 0, \quad \forall u \in U, \; t = \left\{a_u+d_u^{UL}+d_u^{CP},\cdots,a_u+d_u-1\right\}, \tag{28}$$

$$d_u^{UL} + d_u^{CP} + d_u^{DL} \leq d_u, \quad \forall u \in U, \tag{29}$$

$$d_u^{UL} = \{1,2,\cdots,d_u-2\}, \quad \forall u \in U, \tag{30}$$

$$d_u^{CP} = \{1,2,\cdots,d_u-2\}, \quad \forall u \in U, \tag{31}$$

$$d_u^{DL} = \{1,2,\cdots,d_u-2\}, \quad \forall u \in U. \tag{32}$$

The objective (2) is to maximize the network sum utility of all tasks. Constraint (3) states that the user must be connected to exactly one BS in every time slot t. Constraint (4) ensures that user u receives a certain data rate r_u required for a constant-rate application (in addition to task processing). Constraint (5) defines the total amount of uploaded data that needs to be stored if the task is served, while constraints (6) and (7) state the amount of total required computational resources for processing and the amount of the result data that should be transmitted in the downlink. Serving a task consists of multiple steps captured by constraints (22)-(28), i.e., the uploading can start only after the task has arrived, the processing can start only after the uploading has finished, while the transmission of the results in the downlink requires the processing to be finished in the previous slot.

Constraints (8) and (9) limit the amount of transmitted data in the uplink and downlink, which depends on the per-PRB rate $R_{u,b}$ and the amount of allocated PRBs $y_{u,b}$. Also, the handover overhead is included in these constraints, which is defined by Eq. (10). It depends on the previous user-to-BS assignment; namely, on $x_{u,b}(t-1)$. The handover overhead at the first time slot $t = 0$ is zero since there are no handovers possible (see constraint (21)).

Constraint (11) captures the amount of data processed at slot t. The storage capacity, computational, and wireless resources of every BS are limited as stated in constraints (12)-(14).

Constraint (15) and (16) define the binary decision variables $x_{u,b}$ and z_u, while the constraints (18)-(20) define the variable

that states the number of PRBs allocated for task upload and download, as well as for the application rate.

Finally, the task must be served before the deadline d_u expires, which is captured by constraint (29). It consists of uploading, processing, and downloading times, and is defined by constraints (30)-(32).

This optimization problem is solved once for the entire time horizon T, assuming the complete knowledge of future tasks, including their arrival times, sizes, and utilities. Therefore, we refer to this solution as the *Oracle*, given the assumed knowledge of all the parameters of interest. The problem (2)-(32) is an Integer Non-linear Program (INLP). Therefore, it is NP-hard [8]. Because of that, we can obtain the optimal solution using a solver, like Gurobi, assuming that the information about all users in the network is available at time slot $t = 0$, only for a very small scenario. Therefore, an approach that provides results faster, although they would be sub-optimal, is needed. To that end, in the next section we propose a practical DRL-based solution and compare its performance to the Oracle in Section V.

III. DRL Approach

This section explains the proposed DRL-based approach for handover management as well as its state space, action space, and reward function.

A. PPO as a DRL Algorithm

In Reinforcement Learning (RL), an agent learns a policy by interacting with the environment by *trial and error*, receiving feedback in the form of rewards based on its actions. The goal of the RL algorithm is to predict the expected reward given the current state. By continuously refining its predictions and strategies, the agent can perform well even in complex environments.

To overcome the curse of dimensionality in tabular RL, DRL algorithms utilize Neural Networks (NNs) to approximate various functions involved in the learning process [9]. Proximal Policy Optimization (PPO) [10] uses two NNs to approximate the value and policy functions. DRL generalizes well to unseen states, and an NN does not need to visit every state [9]. This allows creating a well-performing agent for users with various task parameters, channel conditions, and mobility patterns.

PPO [10] is a state-of-the-art model-free DRL algorithm that belongs to the policy gradient methods. It outperforms other popular algorithms such as Deep Q Network (DQN) and Asynchronous Advantage Actor-Critic (A3C), yet it is simple to implement and tune [10]. The current $\pi_\theta(a_t|s_t)$ is being updated, while the old $\pi_{\theta_{\text{old}}}(a_t|s_t)$ is used to generate new trajectories of state, action, reward, next state; specifically, $\{(s_t, a_t, r_t, s_{t+1})\}$. This improves sample efficiency. After several iterations, the current policy network is synchronized with the old one. To measure the difference between the new and old policies, the ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is computed. The objective function clips the estimated advantage $\hat{A}_t$ if the new policy deviates significantly from the old one.

The main objective function is expressed as [10]

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right],$$

where ϵ is a small hyperparameter that controls the range of the clipping. The second term clips the probability ratio, so that r_t is inside the interval $[1-\epsilon, 1+\epsilon]$. Finally, the minimum of the clipped and unclipped objectives is taken, and the overall objective becomes a lower bound on the unclipped objective. PPO clips the objective so that the updates to the policy do not cause excessively large policy updates, which ensures stable learning. To summarize, the advantages of PPO are stability, simplicity, and sample efficiency.

Algorithm 1 Reward Design

```
The agent outputs b, y_{u,b}, and k_{u,b} for user u in time slot t
Compute y^{app}_{u,b} based on b and R_{u,b}
if handover then
    Reduce user's task and application rates by HIT η_{u,b} as in Eqs. (4), (8)-(9)
end if
Set reward = 0
if handover & SINR_b(t) ≤ SINR_{b'}(t−1) − th_{handover} then
    reward += p_{handover}
end if
if wireless resource violation because of PRBs to satisfy user' rate then
    reward += p^{app}_{resource−violation}
else
    Allocate y^{app}_{u,b} to user u from the serving b
    if y^{app}_{u,b} > th_{PRB} then
        reward += p_{too−many−PRBs}
    end if
end if
if wireless or compute resource violation for tasks then
    reward += p^{task}_{resource−violation}
else if one task is completed then
    reward += U_u
else if one task is processed in any stage then
    Compute the progress ratio of upload/processing/download progress
    reward += progress/10
end if
```

B. PPO-based Approach

1) Resource and mobility management as an RL problem: We deploy one agent at the network controller that can communicate with low latency to all BSs. This agent makes task processing- and application rate-related decisions for all users in the network. Serving BSs signal measured SNR values of their connected users to this controller as well as the amount of available resources (wireless/communication, computational, and storage). The agent makes assignment and

resource allocation decisions and signals them back to the serving BSs. The agent actions correspond to the decision variables (per user per slot) in the optimization problem in Section II-C. Note that our approach re-uses most of the signaling proposed by 3GPP in [11].

The Oracle from Section II-C knows everything about future tasks, such as their arrival and deadline times, as well as task parameters. Although this is a very unrealistic assumption, it provides the best possible performance. Another unrealistic assumption is that the Oracle considers the information from all users and outputs decision variables for all of them at once. Firstly, this creates a combinatorial problem, leading to an exploding action and state space size, making learning infeasible. Secondly, the number of users in the network is expected to vary over time, making it impractical to assume that the number of users is fixed, and that the decision for all of them must be made at the same time slot. Therefore, our agent makes assignment and resource allocation decisions for every user one by one, and each user is a separate vector. The architecture of our NN is independent from the number of users in the network.

2) Design: The effectiveness of RL algorithms is significantly impacted by the reward function, state space, and action space design. Our PPO-based approach uses the existing information at the BSs and users, such as measurement reports, task parameters, task completion status, and the available resources to output its decision.

State space: The state space includes channel measurements of the BSs, their available resources, as well as task parameters and the current progress of task processing. So, the state of user u can be expressed as a flattened array of the following arrays: (i) the measured SNR values per BS; (ii) the id of the serving BS; (iii) task parameters (arrival a_u, deadline d_u, its utility U_u); (iv) task requirements (data size s_u, computation K_u, and results' data size s'_u); (v) the available PRBs per BS; (vi) the available compute resources per BS; (vii) the remaining number of users to be served; (viii) the current user stage (upload, process, download, or no task); (ix) the remaining task size to be served, which is an updated tuple from (iv); and (x) the current time slot.

It is worth mentioning that before feeding SNR values and the resource arrays to the agent, they are normalized to be in the range $[0, 1]$ to speed up learning.

Action space: The action space consists of the set of all BSs along the user trajectory and the set of wireless and computational resources. Hence, the action consists of the index of the next serving BS b for user u, the number of allocated wireless/communication $y_{u,b}$ or $y'_{u,b}$, and compute resources $k_{u,b}$. We compute $y^{app}_{u,b}$ after the agent outputs b based on $R_{u,b}$, r_u, and handover overhead. Note that this is not an integer value and must be rounded up. One could also let the agent output $y^{app}_{u,b}$ directly, but this complicates the learning as our training results show.

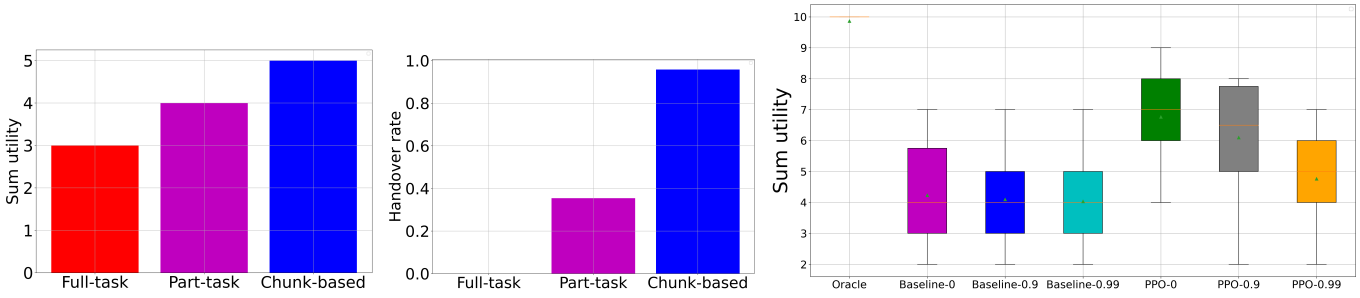
Reward function: As the goal is to maximize the task utility and satisfy users' application rate, we provide a positive reward for task completion and penalize the agent if the user rate is not satisfied (due to the lack of wireless resources or a poor choice of the serving BS). One key finding of this work is that directly using the objective function from the optimization formulation with penalties for constraint violations does not result in a well-performing DRL agent. The agent needs to complete three steps in the right order, which are task upload, processing, and download, before the deadline expires in order to receive the utility value as the reward. The rewards are sparse and delayed, which is a well-known problem in RL [9]. Eventually, the agent learns to act safely to avoid penalties, but achieves a very low utility because it does not do anything beyond satisfying the constraints.

Thus, we propose a more complex reward design summarized in Algorithm 1, where we guide the agent by providing a small reward for some progress. Initially, we set the reward to zero. We penalize the agent for handovers by considering Handover Interruption Time (HIT) in the user's data rate (Line 4 of Algorithm 1). Additionally, for handovers to a target BS where the SNR is worse by a threshold $th_{handover}$ (e.g., 2 dB) compared to the serving BS's SNR, we apply a small penalty $p_{handover}$ to reduce the handover rate. For the wireless and computational violation of resources allocated to tasks, we assign the penalty $p^{task}_{resource-violation}$, whereas $p^{app}_{resource-violation}$ is given for wireless resource violations allocated to satisfy the user's rate. We differentiate between these penalties so that it is easier for the agent to learn. We also discourage the agent from connecting the user to a BS with a poor channel by giving a penalty if too many PRBs have to be allocated to satisfy user's rate (Line 15 of Algorithm 1). Finally, if there was any progress on task completion, e.g., a part of the task was uploaded or processed, the agent receives an immediate reward $progress$ (Line 24 of Algorithm 1). When a task is completed, the reward is incremented by the full utility U_u.

IV. SNR-based Baseline Model

Next, we describe the baseline model against which we are going to compare the performance of our approach later in Section V. The user periodically measures the channel and sends the Measurement Report (MR) to its serving BS, which contains the signal strength of the serving and neighboring BSs. We assume that users signal their measured channels to their serving BSs periodically. In the baseline handover algorithm, the network makes handover decisions and signals them to the users [11]. Before reporting the measurements, the user applies Layer-3 filtering and averages Reference Signal Received Power (RSRP) or SNR values over 200 ms [11]. Based on these measurements, the serving BS selects the target BS that should be prepared for handover. The network initiates a handover when a neighboring BS becomes better than the serving BS by a certain margin (e.g., 3 dB) during a certain period of time (e.g., during 320 ms), similarly to [12]. We use SNR measurements to make handover decisions, hence, we refer to this baseline algorithm as the SNR-based handover.

Next, after the user assignment was performed, the baseline algorithm allocates to all users the minimum required number



(a) (b) (c)

Fig. 3: a) and b): The utility and handover rate with Oracle with full-, part-task and chunk-based approaches; c) the utility with Oracle, Baseline and PPO-based algorithms for handover overhead values (HIT) of 0, 0.9, and 0.99.

TABLE II: Performance with direct and curriculum learning.

Learning	Task utility	Handover rate	User rate satisfaction
Direct	4.167	1.069	0.940
Curriculum	6.1	0.392	0.963

of PRBs to satisfy the user's rate if there are enough resources available. In the second step, the remaining PRBs are allocated to the tasks to maximize the utility. Here, we prioritize tasks according to their arrival time, i.e., the tasks that arrive first receive wireless resources to upload their data first. We allocate to a user $k_u^{alloc} = \min\{k_{req}, k_b\}$, where k_{req} are the required resources to upload/process/download the task, k_b are the available resources at BS b.

V. Performance Evaluation

First, we describe the simulation setup. Then, we evaluate the performance with network scenarios of 10 and 15 users.

A. Simulation Setup

We evaluated the algorithms with 4 BSs (one macro and three micro BSs) over 1000 s. We consider a two-tier network with urban channel models for macro and micro cells, respectively, from 3GPP 5G Release 14 [13] when simulating the channel. We model the path loss and shadowing for LoS and no LoS as in [13]. Macro BS operates at 2.5 GHz, while micro BSs at 28 GHz. Random Waypoint mobility model is used to generate the mobility-related data [14] for pedestrian users, bikes, and cars. An RLF occurs when the user's SNR falls below an RLF T_{out} threshold during *T310* timer. We consider one cluster of BSs and the frequency reuse factor is 1, hence, there is no interference from the first neighbors.

We set the following parameters [5], [15]: $U_u = 1$, $C_b = 7.2$ Mbps, $T_{slot} = 100$ ms, $T_{HIT} = 80$ ms, $N_b^{CP} = 100$ units, $r_u = 1$ Mbps. Other parameters are normally distributed: κ_u with the mean $\mu = 4.7$ Mbit and standard deviation $\sigma = 0.15$; S_b with $\mu = 3.6$ GB and $\sigma = 0.3$; d_u with $\mu = 270$ time slots and $\sigma = 2$; s_u and s'_u with $\mu = 50$ MB and $\sigma = 3.125$; a_u with $\mu = 2$ tasks per slot and $\sigma = 1$.

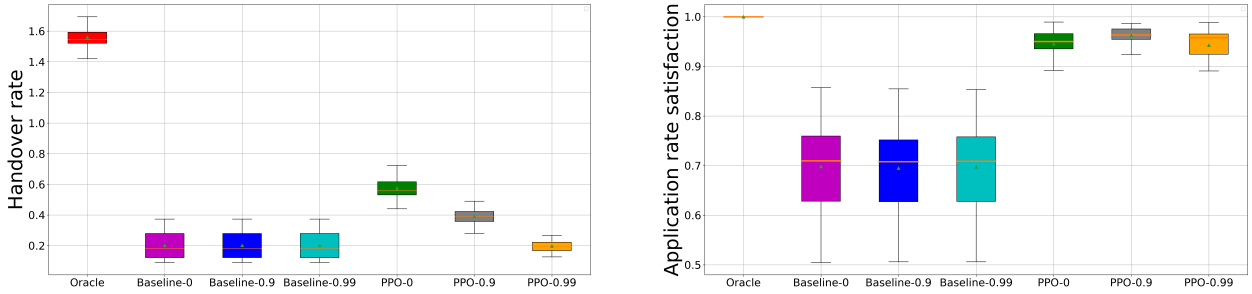
B. Evaluations of Three Handover Approaches with Oracle

Fig. 3a and Fig. 3b present the results of Oracle for three different handover approaches described in Section II-B. The first, denoted as *full-task* is characterized by the fact that once the task processing starts, the complete task needs to be processed (uploaded, computed, and downloaded) before a handover to another BS can be performed. In this case, we assume that the BSs cannot exchange task-related information with each other and the task processing must be completed at one BS. In the second, denoted as *part-task*, one of the three parts needs to be completed before a handover, i.e., a handover can be executed after the uploading finishes. An example of this scenario is when the previous BS can send the result data $s_{'u}$ after processing to another BS. In the third case, called *chunk-based*, the user can be handed over to another BS after transmitting a small chunk of data, e.g., 10 MB. This scenario is especially well suited for fast-changing radio conditions, to avoid RLFs and benefit from high SNR after a handover.

As can be observed from Fig. 3a, the chunk-based approach achieves the highest sum utility. Moreover, the other two approaches cause RLFs in case of fast changing radio conditions, especially for large tasks or when very few bandwidth or computing resources are allocated to the user, i.e., when the task completion time is high. So, with our DRL-based solution, we focus on the chunk-based case. Note that the handover rate in Fig. 3b is very high because Oracle assumes instantaneous handovers. As will be shown in Section V-D, the handover rate can be reduced by considering the handover overhead.

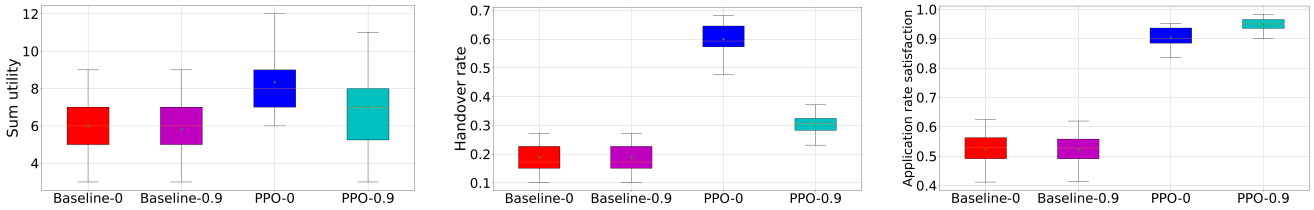
C. Impact of Curriculum Learning

To reduce the handover rate, the handover overhead must be considered in the user rate calculation. Training an agent with a high HIT value is more challenging because the agent keeps the handover rate very low (to avoid the penalty for it) and completes only a small number of tasks. Therefore, we apply curriculum learning [16], which is a technique in RL, where the agent is trained first on easier samples and then on harder ones. In our case, the agent first trains in an environment with HIT $= 0$ and then we increase the difficulty level by setting HIT = 0.9. As the results in Table II show, the average value of utility increases by 46%, the handover rate reduces by 63%, and the user rate satisfaction increases by 2% when using curriculum learning. Note that the number of the training episodes is kept the same with both types of learning. Therefore, in all the following results with DRL, the agent was trained with curriculum learning.



(a) (b)

Fig. 4: Performance with Oracle, Baseline and PPO-based algorithms for handover overhead values (HIT) of 0, 0.9, and 0.99.



(a) (b) (c)

Fig. 5: The performance with the Baseline and PPO-based algorithms for 15 users with an overhead of 0 and 0.9.

D. Performance Evaluation for a Scenario with 10 *Users*

Fig. 3c and Fig. 4 portray the performance results of Oracle, the SNR-based baseline (denoted as Baseline) and the proposed PPO-based algorithm for different overhead values: 0, 0.9, and 0.99. The Oracle achieves the highest utility thanks to its global and future knowledge about all users and tasks. It is followed by PPO-0 with the optimality gap of 30% because PPO-0 trades-off a high handover rate for the increased utility. The PPO-based algorithm improves the sum utility by up to 65% and achieves a user rate satisfaction of $\approx$ 96% by increasing the user rate y 35%. The handover rate with the Baseline and PPO-0.99 reduces only slightly (0.2 with Baseline vs. 0.19 with PPO-0.9).

E. DRL Solution Evaluation for a Scenario with 15 *Users*

As shown in Fig. 5, PPO-based solution increases the utility by $\approx$ 40% while increasing the user rate satisfaction from 52% to 96% at the cost of slightly higher handover rate. If the operator wants to keep the handover rate even lower, a larger handover overhead like in Fig. 4 can be set. Interestingly, with PPO, a higher handover overhead increases the user rate satisfaction (Fig. 5c), as handovers result in outages. Higher handover rate results though in a larger utility value, hence, there is a trade-off between the handover rate, utility, and user rate. Even though the Baseline prioritizes satisfying user rate to utility maximization (see Section IV), it achieves a very low user rate satisfaction (see Fig. 5c) because users are not well-distributed across BSs.

VI. Related Work

The authors in [1] show that handovers negatively impact the performance of 5G networks, with the throughput dropping significantly after a handover, and the latency skyrocketing up to 14.5×. While video conferencing is a popular service that 5G should easily support, conferencing over currently deployed 5G networks remains challenging for mobile users due to handovers, even though videoconferencing applications like Zoom require less than 1 Mbps. Another application which performance was evaluated is volumetric video streaming. The video bitrate reduces by 59%, while the latency increases by over 100% in mmWaves [1].

Numerous solutions have been proposed to enhance mobility management, i.e., adjusting handover parameters [17], [18] performing jointly resource allocation and handover management [6], [7], reducing mobility-related signaling [19], applying DRL to select the target BS [20], and using lower level signaling for mobility [12]. In [17], the authors tune handover margin and Time-to-Trigger (TTT) based on the user speed and measured channel. The authors in [18] also adjust handover parameters based on the user speed as well as on the cell load, and perform load-balancing between neighboring cells. However, handovers in dense scenarios are not only caused by mobility, but also by LoS blockages and load-balancing decisions. The authors in [20] also apply the PPO algorithm for the handover management problem. Their objective is to assign the user to the strongest BS while at the same time avoiding RLFs and ping-pong handovers. However, in [20] the available resources are neglected in the decision process.

The problem of running computation-intensive tasks has been considered in [21], [22], where the best options when to solve a task locally or to send it to a cloud have been investigated for different optimization goals. However, this was done for users that do not change their serving BSs. Running computation-intensive tasks, where the limitations in the uplink/downlink bandwidth and finite storage and computation capacities in the clouds were fully incorporated in

the problem setup, with different optimization objectives and different processing structure have been considered in [5] and [23]. For example, in [5] the authors assign tasks to edge clouds using a bidding approach in a distributed manner. They consider two scenarios: pipeline, where the data units can be processed as they arrive, and batch processing, where all data must be uploaded before processing can begin. However, their study does not consider handover management, which is a crucial component and must be included in the process of resource allocation as prior works [1], [6] have shown.

As far as SD-RAN is concerned, its potential in improving various aspects of network performance has already been documented in [24] with improving the total network throughput compared to the traditional resource allocation in cellular networks, in [25] with providing proportional fairness, and in [26] with guaranteeing delay fairness among all the users in the network. However, in none of these works the possibility of users moving across different cells is not considered. Furthermore, the computing aspect is not considered either. On the other hand, in the current work we show the great potential of SD-RAN in handling data-intensive tasks with highly-mobile cellular users, leading to a maximization of the network utility, beneficial to cellular operators.

Finally, we have proposed in [27] a two-level wireless resource allocation using SD-RAN with the goal of maximizing network sum throughput, ensuring a minimum user rate and minimizing the number of handovers. However, as opposed to the current work where we consider both communication and computation resource allocation, in [27] the analysis is confined to communication resources only. To our best knowledge, there are no other works that propose the utilization of SD-RAN in handling mobility management for computation-intensive tasks in cellular networks.

VII. Conclusion

In this paper, we considered the problem of joint mobility management and resource allocation (communication, computation, and storage) using SD-RAN. To that end, we formulated an optimization problem, which is NP-hard. Because of its complexity, we proposed a DRL-based solution and showed that it significantly outperforms another baseline with an optimality gap of 30% to the Oracle, where the latter assumes complete future knowledge of everything in the network. Furthermore, we provide a user rate guarantee up to 96% to satisfy an additional constant bit rate application. In the future, we plan to consider a cross-layer optimization in the mobility management, by considering also the transmission power.

References

[1] A. Hassan, A. Narayanan, A. Zhang, W. Ye, R. Zhu, S. Jin, J. Carpenter, Z. M. Mao, F. Qian, and Z.-L. Zhang, "Vivisecting mobility management in 5G cellular networks," in *Proc. of ACM SIGCOMM*, 2022.

[2] A. Hazarika and M. Rahmati, "Towards an evolved immersive experience: Exploring 5G-and beyond-enabled ultra-low-latency communications for augmented and virtual reality," *Sensors*, vol. 23, no. 7, 2023.

[3] S. Kianpisheh and T. Taleb, "A survey on in-network computing: Programmable data plane and technology specific applications," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, 2022.

[4] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "FlexRan: A flexible and programmable platform for software-defined radio access networks," in *Proc. of ACM CoNEXT*, 2016.

[5] C. Rublein, F. Mehmeti, M. Towers, S. Stein, and T. F. La Porta, "Online resource allocation in edge computing using distributed bidding approaches," in *Proc. of IEEE MASS*, 2021.

[6] A. Prado, F. Stoeckeler, F. Mehmeti, K. Patrick, and W. Kellerer, "Enabling proportionally-fair mobility management with reinforcement learning in 5G networks," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 6, 2023.

[7] A. Prado, D. Gölitz, F. Mehmeti, and W. Kellerer, "Proportionally Fair Resource Allocation Considering Geometric Blockage Modeling for Improved Mobility Management in 5G," in *Proc. of ACM Q2SWinet*, 2022.

[8] R. Srikant and L. Ying, *Communication Networks: An Optimization, Control, and Stochastic Networks Perspective*. Cambridge University Press, 2013.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[11] 3GPP, "NR; NR and NG-RAN Overall description; Stage-2," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.300, 3 2021, version 16.5.0. [Online]. Available: http://www.3gpp.org/DynaReport/38300.htm

[12] A. Gündogan, A. Badalıoğlu, P. Spapis, and A. Awada, "On the modelling and performance analysis of lower layer mobility in 5G-advanced," in *Proc. of IEEE WCNC*, 2023.

[13] 3GPP, "Study on channel model for frequencies from 0.5 to 100 GHz," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 38.901, 1 2020, version 16.1.0. [Online]. Available: http://www.3gpp.org/DynaReport/38901.htm

[14] X. Lin, R. K. Ganti, P. J. Fleming, and J. G. Andrews, "Towards understanding the fundamentals of mobility in cellular networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 4, 2013.

[15] N. Naz, A. Haseeb Malik, A. B. Khurshid, F. Aziz, B. Alouffi, M. I. Uddin, and A. AlGhamdi, "Efficient processing of image processing applications on CPU/GPU," *Mathematical Problems in Engineering*, vol. 2020, pp. 1–14, 2020.

[16] X. Wang, Y. Chen, and W. Zhu, "A survey on curriculum learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, 2022.

[17] A. Alhammadi, M. Roslee, M. Y. Alias, I. Shayea, S. Alraih, and K. S. Mohamed, "Auto tuning self-optimization algorithm for mobility management in LTE-A and 5G HetNets," *IEEE Access*, vol. 8, 2019.

[18] A. Hatipoğlu, M. Başaran, M. A. Yazici, and L. Durak-Ata, "Handover-based load balancing algorithm for 5G and beyond heterogeneous networks," in *Proc. of ICUMT*, 2020.

[19] A. Prado, F. Mehmeti, and W. Kellerer, "Cost-efficient mobility management in 5G," in *Proc. of IEEE WoWMoM*, 2023.

[20] P. J. Gu, J. Voigt, and P. M. Rost, "A deep reinforcement learning-based approach for adaptive handover protocols in mobile networks," *arXiv preprint arXiv:2401.14823*, 2024.

[21] N. Felemban, F. Mehmeti, H. Khamfroush, Z. Lu, S. Rallapalli, K. Chan, and T. L. Porta, "PicSys: Energy-efficient fast image search on distributed mobile networks," *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, 2021.

[22] K. S. Wheatman, F. Mehmeti, M. Mahon, H. Qiu, K. S. Chan, and T. F. L. Porta, "Optimal resource allocation for crowdsourced image processing," *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, 2023.

[23] C. Rublein, F. Mehmeti, T. D. Gunes, S. Stein, and T. F. La Porta, "Scalable resource allocation techniques for edge computing systems," in *Proc. of ICCCN*, 2022.

[24] F. Mehmeti, A. Papa, and W. Kellerer, "Maximizing network throughput using SD-RAN," in *Proc. of IEEE CCNC*, 2023.

[25] F. Mehmeti and W. Kellerer, "Proportionally fair resource allocation in SD-RAN," in *Proc. of IEEE CCNC*, 2023.

[26] ——, "Delay fairness in 5G networks with SD-RAN," in *Proc. of ICCCN*, 2023.

[27] A. Prado, M. Ciki, F. Mehmeti, and W. Kellerer, "Enhanced mobility management with SD-RAN in 5G networks," in *Proc. of IFIP/IEEE Networking*, 2024.