DeepCoMP: Coordinated Multipoint Using Multi-Agent Deep Reinforcement Learning

Stefan Schneider, Holger Karl, Ramin Khalili, and Artur Hecker

***Abstract*—Macrodiversity is a key technique to increase the capacity of mobile networks. It can be realized using coordinated multipoint (CoMP), simultaneously connecting users to multiple overlapping cells. Selecting which users to serve by how many and which cells is NP-hard but needs to happen continuously in real time as users move and channel state changes. Existing approaches often require strict assumptions about or perfect knowledge of the underlying radio system, its resource allocation scheme, or user movements, none of which is readily available in practice.**

Instead, we propose three novel self-learning and self-adapting approaches using model-free deep reinforcement learning (DRL): DeepCoMP, DD-CoMP, and D3-CoMP. DeepCoMP leverages central observations and control of all users to select cells almost optimally. DD-CoMP and D3-CoMP use multi-agent DRL, which allows distributed, robust, and highly scalable coordination. All three approaches learn from experience and self-adapt to varying scenarios, reaching 2x higher Quality of Experience than other approaches. They have very few built-in assumptions and do not need prior system knowledge, making them more robust to change and better applicable in practice than existing approaches.

***Index Terms*—Mobility Management, Coordinated Multipoint, CoMP, Cell Selection, Resource Management, Reinforcement Learning, Multi Agent, Self-Learning, Self-Adaptation, QoE**

I. Introduction

Modern cellular networks coordinate resources across multiple terminals, base stations, cells, and access network entities. An example is coordinated multipoint (CoMP), a kind of cooperative MIMO that leverages macrodiversity by allowing user equipment (UE) to connect to and receive data from multiple cells simultaneously. CoMP was introduced in 3GPP LTE rel. 11 [1] but will play an even more important role in 5G and 6G with many small and partially overlapping cells [2].

We see CoMP as an interesting and relevant real-world use case to study coordinated resource management. In the considered scenario, UEs move around such that their channel state continuously changes due to path loss, shadowing, reflections, etc. Especially UEs at the cell edge experience strong path loss and benefit from connecting to multiple cells. At the same time, connections must be balanced across different cells as all UEs served by a cell compete for limited resources, e.g., physical resource blocks (PRBs) in LTE. With more UEs connected to a cell, its available PRBs per UE decrease. Hence, a UE may connect to more cells to increase its effective data rate but thereby increases competition at these additional cells, possibly reducing the available PRBs and consequently the data rate of other connected UEs. To navigate this trade-off and adapt to UE movement, changing load, and channel state, it is crucial to dynamically select how many and which cells should serve which UEs.

We focus on dynamic multi-cell selection in downlink CoMP with joint transmission and coordinated scheduling [1]. Instead of maximizing mere data rate, our goal is to also support maximizing Quality of Experience (QoE) for all UEs. QoE depends on the considered scenario and may, for example, require reaching a certain data rate threshold or show diminishing returns with increasing data rate [3], [4]. Multi-cell selection is significantly more complex than single-cell selection and the resulting problem is NP-hard and cannot be reasonably approximated [5]. Despite this complexity, it must happen quickly online as user movement affects channel state and achievable data rates.

Existing approaches for multi-cell selection typically use heuristic algorithms or solve mixed-integer linear programs (Sec. II). Often, these approaches build on rigid models or rely on perfect knowledge of the underlying system, including radio model, resource allocation scheme (i.e., intra-cell allocation of PRBs to connected UEs), or even user movement [5]–[7]. Such detailed knowledge may only be approximated or even be completely unavailable in practice (e.g., vendor-specific, unknown PRB allocation), limiting applicability of these approaches. Simpler approaches based on channel measurements and simple rules, e.g., [7], [8], may require different manual configurations for different scenarios and often lead to suboptimal results (Sec. VII).

Instead, we propose three novel self-learning and self-adaptive approaches, DeepCoMP, DD-CoMP, and D3-CoMP, using model-free deep reinforcement learning (DRL). DRL excels at sequential decisions to optimize long-term rewards [9] and is thus particularly useful here, where we want to optimize UEs' long-term QoE over multiple sequential cell assignments. Moreover, our model-free approaches require very few built-in assumptions or system knowledge and learn from experience of previous actions. They do not need explicit and detailed system information but adapt to scenarios with varying and even heterogeneous resource allocation schemes and cell density. Unlike existing work [10]–[12], they support multiple moving UEs and autonomously adapt to varying UE numbers and

Manuscript received October 29, 2021.

S. Schneider is with the Computer Networks Group at Paderborn University, Germany (email: stefan.schneider@upb.de). H. Karl is with Hasso Plattner Institute, University of Potsdam, Germany. R. Khalili and A. Hecker are with Huawei Technologies Munich Research Center, Germany.

This work was supported in part by the German Research Foundation within the Collaborative Research Centre "On-The-Fly Computing" (SFB 901) and the European Commission under 5G-PPP project FUDGE-5G (H2020-ICT-42-2020 call, grant 957242). The expressed views are the authors' and do not necessarily represent these projects.

movement. Our approaches can continuously adapt to ongoing changes in the scenario through online transfer learning. Similarly, they support transfer learning for fine-tuning pretrained models to new, unseen situations without requiring expensive retraining from scratch (cf. Google AutoML [13]). Overall, our DRL approaches are not explicitly aware of the underlying wireless system details, which they abstract away, and instead self-adapt to any given scenario through indirect feedback. Hence, we expect the proposed techniques to also carry over to other tasks in network and service management, relying only on available observations to learn which management actions optimize the desired goal.

In more detail, DeepCoMP (Sec. IV) is a centralized DRL approach that jointly coordinates all UEs. By central observation and control, DeepCoMP achieves close-to-optimal solutions but requires more training for larger scenarios with more UEs. To address scenarios without global observations or with many UEs, we propose two variants: DD-CoMP and D3-CoMP (Sec. V). Both are multi-agent DRL approaches that observe and control UEs individually, in a distributed fashion. DD-CoMP trains a single neural network that is later replicated for distributed inference. D3-CoMP distributedly trains separate neural networks. While the proposed DRL approaches learn without human intervention, designing the underlying Markov decision process (MDP) is known to be challenging for practical problems [9], [14], [15], which we *solved by applying domain knowledge to the MDP design*. For example, we carefully design DD-CoMP's and D3-CoMP's observation and action spaces to be invariant in the number of UEs, allowing fast training even in large scenarios. For the observations and reward of each agent, we also take locally available information of surrounding UEs into account to encourage cooperative behavior, which is an open challenge in multi-agent DRL [16], [17]. *Overall, our contributions are:*

- In Sec. IV, we propose DeepCoMP as a centralized DRL approach, jointly selecting multiple cells for all UEs.
- In Sec. V, we propose DD-CoMP and D3-CoMP as distributed multi-agent DRL approaches that coordinate UEs individually and converge more quickly.
- In Sec. VI, we discuss architecture and options for real-world deployment of DeepCoMP, DD-CoMP, and D3-CoMP, leveraging domain knowledge for designing practical observations, actions, and reward.
- In Sec. VII, we evaluate adaptability, robustness, and scalability of our three DRL approaches and show that they consistently and significantly outperform existing approaches (2x higher QoE).
- Our code is publicly available [18] to encourage reproduction and extension of our work.

II. Related work

A. Conventional Approaches without DRL

Xenakis et al. [19] and Giust et al. [20] survey approaches for mobility management and dynamic cell selection, which are often addressed together with problems like resource allocation or power control. While many authors consider selection of a single cell [21]–[23], we focus on approaches that select multiple serving cells for CoMP. Qamar et al. [24] review different CoMP modes and survey corresponding literature. One of their identified open research issues is fairness in CoMP, which we encourage by optimizing a logarithmic utility function (representing QoE) [25]. We focus on CoMP joint transmission (CoMP-JT) with coordinated scheduling (CoMP-CS) but believe that our approaches could also learn to effectively select cells for other CoMP modes (e.g., CoMP-CB) if trained in a corresponding environment.

Marsch and Fettweis [7] statically cluster cells into fixed groups of given size, optimizing CoMP for a priori known UE positions. Vijayarani and Nithyanandan [6] dynamically optimize the number of serving cells for each UE without selecting the cells as such, but evaluating the expected throughput for all possible numbers of cooperating cells. All these examples require detailed knowledge of the underlying system and environment dynamics (e.g., UE positions, data rates, resource allocation). Our approaches, without such knowledge, dynamically decide both how many and which cells to select and consistently outperform algorithms with a fixed number of cells (Sec. VII).

Similar to our approach, Amzallag et al. [5] optimize how many and which cells to select for CoMP and also consider fairness. They use an offline approach that requires a priori knowledge of UEs, cells, and system details over all time steps. You and Yuan [11] also dynamically select multiple cells for CoMP and additionally optimize resource allocation. While their problem is very similar to ours, our approaches neither require detailed, possibly unavailable system knowledge nor are they tied to any specific resource allocation scheme.

Beylerian and Ohtsuki [8] propose a simpler, rule-based approach for multi-cell selection where users connect to all cells above a signal-to-interference-and-noise ratio (SINR) threshold. As we show in Sec. VII, the threshold parameter strongly influences performance and the best value varies between scenarios, again requiring human expertise for proper configuration. In contrast, our proposed DRL approaches adapt to different and even heterogeneous schemes without explicit knowledge of these schemes through self-learning and consistently outperform this approach.

B. Self-Learning DRL Approaches

Self-learning approaches, particularly reinforcement learning [26], have become popular as they can be applied without detailed system knowledge and should adapt to different scenarios. Nasir and Guo [27] propose multi-agent DRL for dynamic power control. Their approach is comparable to our DD-CoMP as it also relies on local observations to make distributed decisions after centralized training. We also propose D3-CoMP, which supports distributed training. They assume each UE to be connected to a single fixed cell and then control transmission power. We do not consider power control but focus on cell selection. Our approaches are hence complementary. Similarly, Ozturk et al. [28] use supervised learning to predict UE movement, which could be combined with our approach to focus training on expected UE positions and movement. Ayala-Romero et al. [29] jointly allocate radio

and compute resources using autoencoders and DRL without requiring a priori system knowledge but adapting to different and even heterogeneous systems. Our approaches could be combined for self-adaptive multi-cell selection and resource allocation.

More related to our problem, Chen et al. [10] use DRL to select base stations for multi-access edge computing (MEC) but only consider a single UE and connections to a single cell. Elsayed et al. [30] use tabular Q-learning for cell selection and interference mitigation in 5G by connecting UEs to a single cell, ignoring CoMP. Furthermore, tabular Q-learning does not scale to large or continuous observations and is limited to very small and simple threshold-based observations, possibly leading to suboptimal results and restricting applicability to small scenarios. Niyato and Hossain [31] dynamically select a single network in HetNets (comparable to single-cell selection), rely on tabular Q-learning, and do not consider user mobility.

The work by Zhou et al. [12] is most related to ours. We both consider self-learning approaches for dynamic cell selection with UEs moving across many small, partially overlapping cells. Zhou et al. address the non-stationary nature of the problem by proposing a piece-wise stationary approach using bandits for regret minimization. To this end, they assume that UEs only move abruptly for short durations and otherwise stand still. In contrast, we support constantly moving UEs, making a piecewise stationary approach inapplicable. Our proposed on-policy DRL approaches can learn continuously online and adjust to changes in the environment (unlike typical off-policy approaches [32]). Moreover, the authors focus on a single UE, only consider connections to one cell at a time, and maximize throughput directly. We support multiple moving UEs and simultaneous connections using CoMP. Rather than optimizing throughput, we derive and optimize UEs' QoE.

In general, we go beyond existing work by proposing three different DRL approaches geared towards different, realistic scenarios (number of UEs, available training time, etc.). In particular, we are among the first in the area to successfully apply cooperative multi-agent DRL, which is known to be challenging [17], as well as online transfer learning. We make fewer limiting assumptions, e.g., about the number or movement of UEs, and optimize a more complex utility (QoE instead of throughput), which we believe makes our approach more meaningful and generally applicable.

III. Problem Statement

Our self-learning DRL approaches require very few assumptions and only readily available information about UEs and cells in the area (current connections, SINR, and QoE per UE as detailed in Sec. IV-A1 and V-B1). We intentionally omit a formalization of radio models or other system details that are irrelevant for our DRL approaches. We describe the example model we use for evaluation in Sec. VII-B.

A. Parameters

We consider M UEs $u_j \in U$ and N cells $c_i \in C$, transmitting in discrete time steps $t = 1, 2, ..., T$, synchronized among cells. We focus on the downlink and assume each cell $c_i \in C$ to schedule its PRBs (or other units of radio resources) to connected UEs autonomously and transparently using CoMP coordinated scheduling (CoMP-CS) [33]. Inside one cell, a PRB is assigned to at most one UE, but one UE can be assigned multiple PRBs from one or several cells. We make no other assumptions about cells' resource allocation scheme, i.e., how or how many PRBs they assign to each connected UE. Considering multi-cell selection for uplink communication (using CoMP joint processing [34]) is interesting future work; we expect similar problems and solutions as described here for downlink.

UEs $u_j \in U$ move around freely with varying direction and velocity, unknown to the system. Movement affects channels between UEs and connected cells, fluctuating the measured SINR over time. Here, in the context of cell selection, we focus on fluctuations on the order of seconds and assume that fast-fading effects on the order of sub-milliseconds are handled and averaged out in the physical layer. We denote by $\text{SINR}_{i,j}(t)$ the measured SINR from cell c_i to UE u_j at time t. UE u_j can only connect to and receive data from cell c_i if the SINR is above a given threshold γ_{SINR}. Assuming it can connect, the effective downlink data rate $r_{i,j}(t)$ from cell c_i to UE u_j at time t depends on cell-internal resource allocation (e.g., power, PRBs, antennas) [35]. Our DRL approaches are neither explicitly aware of $r_{i,j}(t)$ nor of the cells' resource allocation but learn to adapt to a given scenario through feedback of UEs' QoE.

B. Decision Variables

We express cell selection by binary decision variable $x_{i,j}(t) \in \{0, 1\}$, indicating whether cell c_i serves UE u_j at time t. To allow $x_{i,j}(t) = 1$, the corresponding SINR must be above threshold γ_{SINR}. A UE's total rate is simply the sum of its cell rates (considering CoMP-CS [35]): $r_j(t) = \sum_{i=1}^{N} x_{i,j}(t) r_{i,j}(t)$. Deciding which cells serve which UEs may be driven completely by the network or UEs may trigger or assist cell connections themselves (Sec. VI). To limit signaling overhead, we restrict UEs to connecting to or disconnecting from at most one cell per time step. Frequent connection changes are still possible as time steps are typically quite short, depending on the acceptable signaling overhead in a given scenario.

C. Objective: Maximize QoE

Rather than simply maximizing the data rate per UE, as commonly done in related work [6], [12], [30], we are interested in optimizing UEs' QoE, which better reflects users' satisfaction and determines, e.g., operators' financial success [36]. UEs' QoE typically increases roughly logarithmically with increasing data rate, i.e., with diminishing returns [3], [4]. We quantify QoE of UE u_j by its *utility* $U_j(t)$. Using a logarithmic function as example, $U_j(t)$ could be formalized as

$$U_j(t) = \min\left\{U_j^{\max}, \max\left\{U_j^{\min}, w_1 log_{w_2}(w_3 + r_j(t))\right\}\right\} \quad (1)$$

where w_1, w_2, w_3 are configurable weights and limits $U_j^{\min}$, $U_j^{\max}$ ensure that the utility is finite and well defined. An

advantage of using logarithmic utility is that maximizing the sum of logarithms is equivalent to ensuring a natural concept of fairness, namely proportional fairness [25].

Nonetheless, logarithmic utility is just an example based on previous studies [3], [4]. Our approaches are not tied to this particular utility function but also adapt successfully to other utility functions as we show in Sec. VII-C4. In fact, they do not even require the utility function to be known. Instead, it suffices to approximate the instantaneous QoE of each UE (i.e., their utility), which is possible from the network side without relying on UEs reporting their QoE truthfully [37].

Our overall goal is, therefore, to maximize long-term QoE, averaged over all UEs and time steps:

$$\text{Avg. QoE:} \qquad \max \lim_{T\to\infty} \frac{1}{T}\frac{1}{M} \sum_{t\in T, j\in\{1,\dots,M\}} U_j(t) \qquad (2)$$

Our self-learning DRL approaches approximate and optimize this long-term utility through typical discounted rewards. We present the details of our DRL approaches next: DeepCoMP in Sec. IV, DD-CoMP and D3-CoMP both in Sec. V, and a discussion of architecture and deployment options in Sec. VI.

IV. DeepCoMP: Centralized DRL

DeepCoMP is a centralized DRL approach that simultaneously observes and controls all UEs in an area with a single DRL agent. While this requires centrally collecting observations of all UEs as well as centralized control, it enables DeepCoMP to learn close-to-optimal policies. These observations contain SINR and QoE estimates, but no detailed system information, which may be unavailable. Sec. IV-A details DeepCoMP's observations, actions, and reward and Sec. IV-B presents the overall algorithm. In Sec. VI, we discuss how DeepCoMP could be deployed in practice.

A. Markov Decision Process

Due to the complexity of the mobile scenario, it is unrealistic to capture the complete environment state, even for a centralized agent. Instead, we focus on partial observations that could be available in practice. The partially observable Markov decision process (POMDP) consists of tuple $(\mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ with observations $\mathcal{O}$, actions $\mathcal{A}$, unknown environment dynamics $\mathcal{P}$, and reward function $\mathcal{R}$, defined next. Fig. 1 shows an example of DeepCoMP's observations, actions, and reward.

1) Observations $\mathcal{O}$: In each time step, DeepCoMP only observes the UEs' current connections, their SINR, and their utility, which are readily available (Sec. VI). Specifically, the DeepCoMP agent observes $O = \langle\langle X_j, \widehat{\text{SINR}}_j, \hat{U}_j\rangle | \forall j \in \{1,\dots,M\}\rangle$ for each cell c_i and UE u_j. Observations are grouped into vectors X_j, $\widehat{\text{SINR}}_j$, and $\hat{U}_j$ for each UE u_j, containing elements for each cell c_i as detailed below. We normalize all observations to be in range $[-1, 1]$ (or $[0, 1]$), which is important for effective training and generalization of deep neural networks [38]. Otherwise, stronger observation signals with a larger range could drown out weaker signals with a smaller range, making the DRL agent "blind" to such observations.

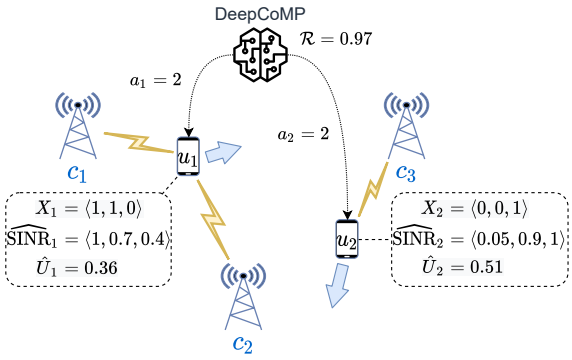


Fig. 1: Illustration of DeepCoMP's POMDP.

a) Current Connections: $X_j = \langle x_{i,j}(t) | \forall i \in \{1,\dots,N\}\rangle \in \{0,1\}^N$ indicates to which cells c_i a UE u_j is currently connected and directly corresponds $x_{i,j}(t)$ (Sec. III-B). UEs hold their connections unless they are told explicitly to connect or disconnect or until they move away from a connected cell.

b) SINR: $\widehat{\text{SINR}}_j = \langle \frac{\text{SINR}_{i,j}(t)}{\max_{i'} \text{SINR}_{i',j}(t)} | \forall i \in \{1,\dots,N\}\rangle \in [0,1]^N$ is the normalized $\text{SINR}_{i,j}(t)$ between each cell c_i and UE u_j. Specifically, $\text{SINR}_{i,j}(t)$ is normalized by the maximum SINR of all cells for UE u_j at time t. This maps the observed SINR of the strongest cell to 1 and highlights the differences in SINR between the available cells, simplifying cell selection. In the special case that a UE is far off any cells (with zero SINR), we set the vector to all zeros to avoid division by zero.

c) Utility: The normalized utility of each UE u_j is $\hat{U}_j = 2 \cdot \frac{U_j(t) - U_j^{\min}}{U_j^{\max} - U_j^{\min}} - 1 \in [-1, 1]$, i.e., $U_j(t)$ scaled to range $[-1, 1]$ based on bounds $U_j^{\min}, U_j^{\max}$ (Sec. III-C). If these bounds are not known explicitly, they can be approximated by keeping track of the lowest and highest $U_j(t)$ over all UEs j and time steps t so far. Observation $\hat{U}_j$ provides valuable information about each UE's current QoE, which is affected by the observed SINR but also by other factors that are not directly observable, e.g., cells' resource allocation and UE movement. Again, note that the current utility $U_j(t)$ of each UE can be approximated locally without relying on UEs to report their QoE truthfully [37].

2) Actions $\mathcal{A}$: To limit protocol overhead, each UE can either connect to or disconnect from at most one cell per time step. This also simplifies and shrinks the action space considerably compared to alternatives like choice of arbitrary subsets of cells. We design the corresponding action space as $\mathcal{A} = \langle a_j | \forall j \in \{1,\dots,M\}\rangle \in \{0, 1, \dots, N\}^M$. Hence, DeepCoMP selects for each UE u_j either a specific cell c_i, where it toggles the connection status, or a no-op. Action $a_j = i \in \{1,\dots,N\}$ means that u_j's connection to cell c_i should be toggled, i.e., establishing a connection if u_j and c_i are not yet connected or disconnecting otherwise. Alternatively, $a_j = 0$ indicates a no-op, where all of u_j's connections remain unchanged.

3) Reward $\mathcal{R}$: DeepCoMP's reward in time step t is the avg. of current utilities over all UEs, using $\hat{U}_j \in [-1, 1]$ as defined in Sec. IV-A1c for the reward: $\mathcal{R} = \frac{1}{M} \sum_{j\in\{1,\dots,M\}} \hat{U}_j$. This corresponds to our goal of maximizing the avg. QoE over

Algorithm 1 DeepCoMP

1: initialize π_θ, V_ϕ, b
2: **for** $t \in \{1, ..., T\}$ **do**
3: **for** $u_j \in U$ **do**
4: $o_j, r_j \leftarrow$ `get_obs_and_r`(u_j)
5: $o_t \leftarrow \langle o_j | \forall j \in \{1, ..., M\} \rangle \cup \langle 0 | \forall j \in \{M{+}1, ..., M^{\max}\} \rangle$
6: $r_t \leftarrow \frac{1}{M} \sum_{j \in \{1,...,M\}} r_j$
7: $b \xleftarrow{\text{add}} (o_{t-1}, a_{t-1}, r_t, o_t)$
8: $a_t \leftarrow \pi_\theta(o_t)$
9: **for** $a_j \in a_t$ with $j \in \{1, ..., M\}$ **do**
10: $x_{i,j}(t) \leftarrow$ `set_conn`(u_j, a_j)
11: **if** training **and** b is full **then**
12: train V_ϕ using temporal difference updates [9]
13: train π_θ maximizing $\mathbb{E}[\sum_i \gamma^i r(o_{t+i}, a_{t+i})]$

all UEs as defined in Sec. III-C. Internally, the DRL agent maximizes the sum of discounted rewards to optimize long-term utility.

4) Varying Number of UEs: Since DeepCoMP observes and controls all UEs, the size of its observation and action space depends on the number of active UEs in the area, which may change over time. Observations and actions, on the other hand, are used as inputs and outputs of a deep neural network and, thus, need to be of fixed size. Even if the number M of active UEs in the area varies over time, the size of these observations and actions must not change. To this end, we define $M^{\max}$ as the maximum number of supported UEs in the area and set the size of the observations and actions to $M^{\max}$. If $M < M^{\max}$, related observations are padded with zeros to ensure consistent size of observations and to indicate which UEs are currently missing. Resulting actions for $j \in \{M+1, ..., M^{\max}\}$, i.e., for non-existing UEs, are simply ignored. Overall, the observation and action space of DeepCoMP grow linearly with $M^{\max}$. The distributed DRL approaches (Sec. V) are invariant in $M^{\max}$ and therefore suitable for larger scenarios.

B. DeepCoMP Algorithm

We leverage proximal policy optimization (PPO) [39], which is a state-of-the-art actor-critic DRL algorithm, to train our DeepCoMP DRL agent. Alg. 1 shows our centralized training and inference procedure for DeepCoMP. We start training by initializing stochastic policy π_θ (actor), value function V_ϕ (critic), and mini-batch b (ln. 1). DeepCoMP then observes and acts in each time step for all UEs simultaneously. First, it collects observations and rewards and constructs the observation vector and total reward as defined in Sec. IV-A with function `get_obs_and_r`(u_j) for each UE u_j (ln. 3–6). It then adds the experience (observation, action, reward, next observation) to mini-batch b (ln. 7) and selects an action a_t for the observation vector o_t according to policy π_θ (ln. 8). Action a_t is a vector that specifies the cell selection for all UEs and is applied to the environment by setting decision variable $x_{i,j}(t)$ accordingly (ln. 9–10). Function `set_conn`(u_j, a_j) applies action a_j for UE u_j by dis-/connecting from/to the selected cell c_j if $a_j > 0$ and the selected cell is in range (a_j *toggles* connectivity).

During training, critic V_ϕ's weights are updated whenever mini-batch b is full (ln. 12). Using standard temporal difference updates [9], critic V_ϕ is trained to approximate long-term value $V_\phi(o)$ of receiving an observation o and then following policy π_θ. In turn, $V_\phi(o)$ is used to calculate advantage $A(o, a)$, which is the relative value of taking an action a after receiving observation o compared to all other actions. Finally, advantage A is used inside the policy update when training actor π_θ to optimize the long-term discounted reward (ln. 13), which corresponds to maximizing the avg. QoE, i.e., our objective (Sec. III-C) [40].

We repeat this procedure over many episodes to train DeepCoMP offline and then switch to quick online inference upon training convergence. We also support continuous online training, which we evaluate in Sec. VII-D. Here, training (ln. 11–13) happens asynchronously to avoid blocking online inference. With a fixed number of hidden units, time and space complexity for inference with DeepCoMP is in $O(M^{\max} \cdot N)$ based on the size of the observation and action space (Sec. IV-A1 and IV-A4).

TABLE I: Proposed DRL approaches

DRL Approach	Training	Inference	Deployment
DeepCoMP (Sec. IV)	Centralized	Centralized	Core network
DD-CoMP (Sec. V)	Centralized	Distributed	Network or UEs
D3-CoMP (Sec. V)	Distributed	Distributed	Network or UEs

V. DD- & D3-CoMP: Distributed DRL

DeepCoMP leverages global knowledge and simultaneous control of all UEs to learn highly optimized policies. In practice, however, observations from all UEs may not be available centrally or only with significant overhead, preventing fast centralized inference. Furthermore, its observation and action space grows with the (max.) number of active UEs in the network. We hence propose distributed DeepCoMP (DD-CoMP), which only requires local observations and control per UE. DD-CoMP is a multi-agent DRL approach that trains a *logically centralized* neural network, which is *replicated* for distributed inference. We also propose D3-CoMP, which *trains separate* neural networks for each UE. Table I summarizes these differences in training and inference between the three DRL approaches as well as deployment options, which we discuss in Sec. VI.

A. Trade-offs and Design Choices

1) Decisions per Cell vs. per UE: Using multi-agent DRL, an obvious idea would be to assign one agent per cell, controlling its connections. However, this leads to the same complications as discussed in Sec. IV-A4, where the number of UEs fluctuates but the size of the observation and action spaces are fixed and thus need to be padded. The alternative of assigning one agent per UE appears more suitable as the number of cells is rather fixed. As UEs arrive and depart over

time, DRL agents can be spawned/terminated on demand. Using separate agents per UE does not predicate the networking architecture/location of control (Sec. VI).

Using separate DRL agents for each UE, we can limit each agent to only observe a single UE instead of all UEs, where relevant observations are available locally at the UE (Sec. V-B1). Similarly, the agent's actions determine the UE's cell selection, which can be applied locally without much communication overhead. In large areas with many cells, the observation and action space could be reduced by limiting agents' observations and actions to each UE's k closest cells, which are most relevant for cell selection. A remaining challenge with this approach is that each agent only observes and controls a single UE. This easily leads to greedy and unfair behavior and ultimately lower avg. QoE if agents do not consider or observe other UEs' utility. To overcome this challenge, we slightly adjust DeepCoMP's POMDP for DD-CoMP and D3-CoMP so that it takes other, nearby UEs into account (Sec. V-B).

2) DD-CoMP vs. D3-CoMP: DD-CoMP and D3-CoMP only differ in their training architecture. DD-CoMP trains *a single, logically centralized* neural network for actor π_θ and critic V_ϕ, respectively. It collects experience from all DRL agents in joint mini-batches. While DD-CoMP's training is logically centralized, it could be implemented in a distributed fashion with minimal adjustments. For example, each agent may update its neural network locally and only share the computed gradient updates to synchronize the neural network weights across all DRL agents (cf. federated learning [41]).

D3-CoMP trains *separate, logically decentralized* neural networks for each agent/UE. D3-CoMP can hence learn individual cell selection policies per UE, allowing DRL agents to adapt to heterogeneous UE characteristics that affect QoE but are not explicitly observed, e.g., UE velocity or movement patterns. Furthermore, D3-CoMP does not exchange agents' experiences or gradients during training, lowering overhead. Training decentralized DRL agents independently easily leads to greedy or adversarial policies, which we address by coupling the reward of nearby, competing UEs (cf. best response strategy [42]). However, DD-CoMP's central neural network is trained with more diverse data (from all UEs) than each of D3-CoMP's distributed DRL agents. Hence, DD-CoMP often learns more robust policies (Sec. VII).

B. Markov Decision Process

For DD-CoMP and D3-CoMP, we adjust Sec. IV-A's POMDP. Instead of observation/action vectors for all UEs, agents only observe and control cell selection for a single UE u_j. To allow some awareness of nearby UEs and improve fairness, we extend the observation space and adjust the reward function.

1) Observations O: $O = \langle X_j, \widehat{\text{SINR}}_j, \hat{U}_j, \hat{U}^{\text{avg}}, \hat{M} \rangle$ includes $X_j, \widehat{\text{SINR}}_j, \hat{U}_j$ as defined in Sec. IV-A1. Additionally, we assume that cells collect and broadcast aggregated information $\hat{U}^{\text{avg}}$ and $\hat{M}$ about their connected UEs. Specifically, cells approximate the instantaneous QoE of their connected UEs [37] (or rely on QoE reported by the UEs) and calculate their corresponding utility $\hat{U}_j$ as defined in Sec. IV-A1c. Each cell c_i then broadcasts the avg. utility of all connected UEs as $\hat{U}_i^{\text{avg}} = \frac{\sum_{j'} x_{i,j'}(t) \cdot \hat{U}_{j'}}{M_i}$, where $M_i = \sum_j x_{i,j}(t)$ denotes the number of currently connected UEs at c_i with $j, j' \in \{1, ..., M\}$. If no UEs are currently connected ($M_i = 0$), we set $\hat{U}_i^{\text{avg}}$ to zero. Similarly, if the observing UE u_j is too far away from a cell c_i to receive its broadcast (i.e., $\text{SINR}_{i,j}(t) < \gamma_{\text{SINR}}$), the UE assumes $M_i = 0$ and $\hat{U}_i^{\text{avg}} = 0$ as observations. Finally, vector $\hat{U}^{\text{avg}} = \langle \hat{U}_i^{\text{avg}} | \forall i \in \{1, ..., N\} \rangle \in [-1, 1]^N$ contains the observed avg. utility of all cells. Hence, vector $\hat{U}^{\text{avg}}$ can be observed locally by each DRL agent and helps the agent to select suitable cells with high $\hat{U}_i^{\text{avg}}$ to avoid competing for radio resources at cells with already low avg. utility.

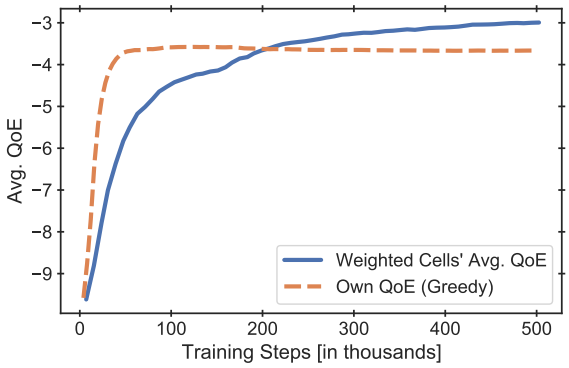


Fig. 2: Learning curves with different reward.

To give the DRL agent a better sense of the load at different cells, cells also broadcast the number of currently connected UEs M_i as defined above. Based on the received M_i (if not received, $M_i = 0$), the DRL agent calculates the normalized load per cell $\hat{M}_i = \frac{M_i}{\sum_{i' \in \{1,...,N\}} M_{i'}}$, setting $\hat{M}_i = 0$ if $M_i = 0$. It then observes vector $\hat{M} = \langle \hat{M}_i | \forall i \in \{1, ..., N\} \rangle \in [0, 1]^N$ containing the normalized amount of connected UEs at each cell. This helps to avoid congested cells and identify free cells without any UEs, i.e., without competition for radio resources. The signaling overhead for $\hat{U}^{\text{avg}}$ and $\hat{M}$ is small and constant (Sec. VI-B). Centralized DeepCoMP does not require additional observations $\hat{U}^{\text{avg}}$ and $\hat{M}$ as it observes all UEs' connections and their QoE individually.

2) Actions $\mathcal{A}$: Actions of DD-CoMP and D3-CoMP are similar to DeepCoMP but refer to a single UE. Particularly, each DRL agent makes an action $\mathcal{A} = a \in \{0, 1, ..., N\}$, controlling cell selection for its UE u_j as defined in Sec. IV-A2.

3) Reward $\mathcal{R}$: An obvious choice for the reward function of a DRL agent controlling UE u_j would be $\mathcal{R} = \hat{U}_j$, i.e., rewarding high utility of the corresponding UE. However, this reward would encourage agents to greedily optimize their UE's utility: Agents would tend to increase their UE's data rate even if it only marginally increases their utility. As a consequence, this could significantly harm other UEs' utilities and overall avg. utility, which is our main objective (Sec. III-C).

Instead, we define $\mathcal{R} = \sum_{i \in \{1,...,N\}} \hat{U}_i^{\text{avg}} \cdot \hat{M}_i \in [-1, 1]$ as the avg. utility at all cells in range (i.e., with received broadcasts), weighted by their normalized amount of connected UEs $\hat{M}_i$.

Algorithm 2 DD-CoMP

1: initialize π_θ, V_ϕ, b ▷ Training
2: **for** $t \in \{1, ..., T\}$ **do**
3: **for** $u_j \in U$ in parallel **do**
4: $o_j^t, r_j^t \leftarrow$ `get_obs_and_r`(u_j)
5: $b \xleftarrow{\text{add}} (o_j^{t-1}, a_j^{t-1}, r_j^t, o_j^t)$
6: $a_j^t \leftarrow \pi_\theta(o_j^t)$
7: $x_{i,j}(t) \leftarrow$ `set_conn`(u_j, a_j^t)
8: **if** b is full **then**
9: train V_ϕ using temporal difference updates [9]
10: train π_θ maximizing $\mathbb{E}[\sum_{j,k} \gamma^k r(o_j^{t+k}, a_j^{t+k})]$
11: Deploy a copy π_{θ_j} of π_θ for each UE $u_j \in U$ ▷ Inference
12: **for** $t \in \{1, ..., T\}$ **do**
13: **for** $u_j \in U$ in parallel **do**
14: $o_j^t, r_j^t \leftarrow$ `get_obs_and_r`(u_j)
15: $a_j^t \leftarrow \pi_{\theta_j}(o_j^t)$
16: $x_{i,j}(t) \leftarrow$ `set_conn`(u_j, a_j^t)

If UE u_j is connected to some cell(s), its utility is already included in the sum above. Otherwise, if u_j is not connected to any cells, we explicitly include its weighted utility in the reward and define $\mathcal{R} = \sum_{i \in \{1,...,N\}} \hat{U}_i^{\text{avg}} \cdot \hat{M}_i + \frac{\hat{U}_j}{\sum_{i' \in \{1,...,N\}} M_{i'}}$. The UE's own utility is therefore always considered but does not dominate the reward. Hence, this reward function encourages DRL agents to maximize the avg. utility over all UEs and not just the utility of their own UE.

Fig. 2 illustrates the avg. QoE (Sec. III-C) for the two reward formulations in an example scenario. Using just the own UE's QoE as reward leads to greedy behavior, which is easier to learn and leads to faster yet clearly suboptimal convergence. Our more sophisticated reward based on weighted avg. QoE at surrounding cells requires negligible signaling overhead (Sec. VI-B) and still leads to quick convergence and ultimately to a significantly better policy (20 % higher QoE).

C. DD-CoMP Algorithm

DD-CoMP agents observe and control each UE's connections separately in parallel (Alg. 2, ln. 3–7) rather than jointly for all UEs like DeepCoMP. Still, experiences from all UEs are added to the same mini-batch b and train a central neural network for actor π_θ and critic V_ϕ (ln. 8–10). After logically centralized training, DD-CoMP performs distributed inference with separate DRL agents, each using a copy of the trained actor network (ln. 11) for local inference (ln. 12–16).

To support continuous training, all DRL agents need to continue sending their UEs' experiences to central batch b to train the the logically centralized neural network. Again, sharing and training happens asynchronously to avoid blocking inference. For quick inference, the distributed DRL agents could regularly update their local copies π_{θ_j}. Assuming a fixed number of hidden units, space and time complexity of inference itself is in $O(N)$, i.e., it is invariant in the number of UEs and linear only in the number of cells. If N is large, space and time complexity could further be reduced to be constant by only considering the k closest cells per UE (Sec. V-A1).

Algorithm 3 D3-CoMP

1: initialize $\pi_{\theta_j}, V_{\phi_j}, b_j \quad \forall j \in \{1, ..., M\}$
2: **for** $t \in \{1, ..., T\}$ **do**
3: **for** $u_j \in U$ in parallel **do**
4: $o_j^t, r_j^t \leftarrow$ `get_obs_and_r`(u_j)
5: $b_j \xleftarrow{\text{add}} (o_j^{t-1}, a_j^{t-1}, r_j^t, o_j^t)$
6: $a_j^t \leftarrow \pi_{\theta_j}(o_j^t)$
7: $x_{i,j}(t) \leftarrow$ `set_conn`(u_j, a_j^t)
8: **if** training **and** b_j is full **then**
9: train V_{ϕ_j} using temporal difference updates [9]
10: train π_{θ_j} maximizing $\mathbb{E}[\sum_k \gamma^k r(o_j^{t+k}, a_j^{t+k})]$

If the number of UEs in the area varies over time, the number of DRL agents is adjusted accordingly. If a new UE arrives, a new DRL agent is instantiated, and removed again when the UE leaves the area. All DRL agents access the logically centralized neural network and thus directly start with a good policy, benefiting from experience of other UEs. Note that, while all UEs share the same neural network weights, their experiences are never shared with each other directly.

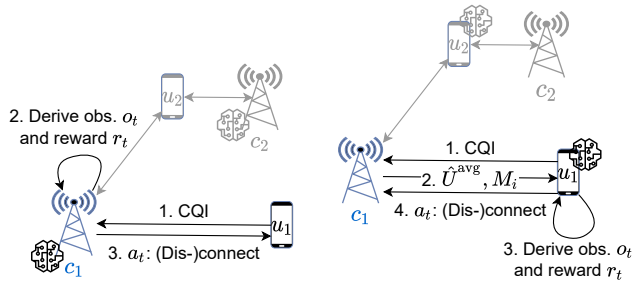
D. D3-CoMP Algorithm

Alg. 3 shows the distributed D3-CoMP training and inference procedure. While the main procedure is similar to DeepCoMP and DD-CoMP, D3-CoMP directly initializes M different actor π_{θ_j} and critic V_{ϕ_j} networks for each UE u_j (ln. 1). Experiences from different UEs are added to different mini-batches b_j (ln. 5). Consequently, actor and critic of each DRL agent are trained only on the corresponding UE's experiences (ln. 8–10). Similar to DD-CoMP, inference time and space complexity is in $O(N)$.

D3-CoMP also automatically adjusts the number of DRL agents to the number of UEs in the area, starting new DRL agents when UEs arrive and terminating them once UEs leave. As D3-CoMP trains separate policies per UE, it typically initializes new neural networks for newly spawned DRL agents. However, trained neural network weights for a UE could be stored when the UE leaves the network and loaded again (e.g., based on a UE identifier) when it returns to avoid retraining from scratch. New UEs (with a previously unseen ID) could start with a trained and stored policy of another UE (e.g., with similar velocity or movement pattern). A hybrid solution between DD-CoMP and D3-CoMP is also conceivable, where a central default policy is trained up front by DD-CoMP and then used as starting point for new DRL agents in D3-CoMP when new UEs arrive in the network.

VI. Architecture and Deployment Options

There are different architectural options for deploying DeepCoMP, DD-CoMP, and D3-CoMP, summarized in Table I. All three approaches can be deployed in the network for network-initiated cell selection. Depending on available resources and latency requirements, deployment could be in the core, edge,



(a) Network-based Deployment (b) UE-based Deployment

Fig. 3: Architecture and deployment options: a) DRL agents deployed in the network for network-initiated cell selection. b) Deployed at each UE for UE-initiated cell selection.

or access network [43]. Alternatively, DD-CoMP and D3-CoMP also support distributed deployment of DRL agents directly at the UEs, enabling UE-initiated cell selection. Both network-controlled and UE-based cell selection are supported and relevant in 5G [44]. In both cases, the DRL approaches can leverage already available data with low overhead. In the following, we detail the two architectural options (Sec. VI-A and VI-B), which are illustrated in Fig. 3, and discuss training and inference in practice (Sec. VI-C).

A. Network-based Deployment

The centralized DeepCoMP approach is designed to be deployed on the network side for network-initiated cell selection. Here, the agent collects available data and exchanges control messages between cells for centralized observations and actions.

The entire network could be split into different areas (e.g., based on cell coverage or business aspects) that are controlled independently by separate DeepCoMP agents. Within each area, a single DeepCoMP agent is deployed, which should be pretrained (e.g., in simulation) as further discussed in Sec. VI-C. Inside an area, the responsible DeepCoMP agent observes and controls all active UEs.

For the agent's observations, the network could keep track of each UE's current connections and collect the UEs' SINR and QoE (step 1 in Fig. 3a). We emphasize, again, that our approach does not assume any predefined, known utility function (e.g., a logarithmic function) but only requires instantaneous approximations of UEs' QoE. To do so, UEs could either report their SINR and QoE directly or cells could use *already available information*, e.g., UEs' channel quality indicators (CQI) and QoS measurements, to approximate SINR and QoE locally [37], [45]. In doing so, cells can locally process available data to derive the necessary observations (Sec. IV-A1) and reward (Sec. IV-A3) in step 2. In step 3, the processed observations are passed through DeepCoMP's actor neural network to obtain the next action, which selects suitable cells for all UEs and triggers necessary connections or disconnections.

DD-CoMP and D3-CoMP can also be deployed in the network for network-initiated cell selection. This works similarly as with DeepCoMP but, here, with multiple DRL agents being deployed for the different active UEs in the area, spawning and terminating DRL agents on demand (as discussed in Sec. V-C and Sec. V-D). These agents could be deployed at different cells, where each one only requires observations and reward based on its corresponding UE rather than data from all UEs. If DD-CoMP and D3-CoMP agents are deployed in the network, cells can locally compute and exchange their avg. utility $\hat{U}^{\text{avg}}$ and load $\hat{M}$, which are required as observations and for the reward (Sec. V-B) and do not need to broadcast them.

B. UE-based Deployment

Alternatively, each DRL agent could be deployed directly at a UE for user-initiated cell selection. Deploying DeepCoMP at the UEs is theoretically feasible but impractical since it requires each UE to collect observations from all other UEs. Instead, DD-CoMP and D3-CoMP are suitable for UE-based deployment because they rely on local observations and control. Each cell c_i still needs the QoE of its connected UEs, which the cells can approximate locally based on UEs' reported CQIs [37], [45] (step 1 in Fig. 3b). In step 2, cells calculate and broadcast their avg. utility $\hat{U}_i^{\text{avg}}$ and number M_i of connected UEs (from which $\hat{M}$ can be derived). In step 3, the DRL agents at each UE derive the current observations and reward (Sec. V-B) based on the received broadcasts and locally observed X_j, $\widehat{\text{SINR}}_j$, and $\hat{U}_j$. Finally, each DRL agent uses its actor neural network to choose an action for its UE and initiates the corresponding connections or disconnections (step 4).

Cells' broadcasts of $\hat{U}_i^{\text{avg}}$ and M_i have small, constant size and could be included in existing system information broadcasts (SIB) [46] without requiring extra signaling messages. Also, no communication between UEs is necessary during inference. Hence, DD-CoMP and D3-CoMP require negligible signaling overhead; much smaller than existing approaches that require detailed system knowledge (e.g., [5], [11]). Furthermore, this approach allows DRL agents to collect all relevant observations locally at the UEs without requiring cells to exchange information with each other directly, e.g., for scenarios with cells of different operators. While UE-based cell selection enables UEs to indicate which cells they want to connect to, the final connection decision can still be controlled by the network (cf. UEs' connection requests in 5G [44]).

C. Training and Inference in Practice

In practice, we suggest to train a reasonable policy in simulations up front and then copy and deploy the trained neural networks at the network or UE side. The pretrained neural networks can then be used for fast online inference with low resource requirements. Our trained neural networks are lightweight (less than 5 MB in size) and can be further optimized using TensorFlow Lite for efficient inference with minimal space, compute, memory, and power consumption [47].

As we show in Sec. VII-D1, it is important to train on randomized simulation scenarios for learning robust policies. To further support bridging the gap from simulation to real

deployment ("sim2real gap" [48]), our DRL approaches support online transfer learning, where they continuously fine-tune their pretrained policy in the actual deployment scenario. We explore and evaluate online transfer learning in Sec. VII-D2. This online training could happen at edge computing centers close to the cells as proposed in 5G [43]. To facilitate online learning in practice, the reward signal for DeepCoMP, DD-CoMP, and D3-CoMP can be directly and locally derived from the collected observations without requiring additional information or overhead (Sec. IV-A3 and V-B3). During training, DD-CoMP agents periodically share their experiences and update their copy of the joint neural network, but D3-CoMP agents train locally without extra communication. To reduce communication overhead for DD-CoMP during training, agents could locally calculate their gradient updates and only share those using federated learning [41]. The computed gradients are smaller and therefore require less communication overhead than sharing the agents' full experiences, including observations, actions, and reward. Federated learning is practically feasible on mobile devices and already being used for commercial applications (cf. Google Gboard [49]).

VII. Performance Evaluation

A. Prototype Implementation

We implemented prototypes of DeepCoMP, DD-CoMP, and D3-CoMP using Python 3.8, TensorFlow 2, and RLlib [50]. To encourage reproduction and reuse, our source code is publicly available [18]. It also contains the light-weight simulation environment we used for training and evaluation, which others can use to study and improve their approaches, too.

B. Evaluation Setup

We evaluate DeepCoMP, DD-CoMP, and D3-CoMP in different scenarios, comparing their avg. QoE over $T = 100$ time steps (Sec. III-C). In Sec. VII-C, we train our approaches from scratch in different mobile scenarios and evaluate how well they adapt to each scenario through self-learning without changing their hyperparameters or making any manual adjustments. In Sec. VII-D, we investigate how well our pretrained DRL approaches generalize to new scenarios without any further training and how continuous online transfer learning can boost performance and training efficiency. Finally, Sec. VII-E compares the scalability of the different approaches.

1) Base Scenario: We vary a base scenario with three partially overlapping cells and three moving UEs, using typical LTE parameters [51] and the Okumura-Hata model [52] for path loss in urban areas with carrier frequency 2.5 GHz, bandwidth $H = 9$ MHz, thermal noise density -90 dBm/Hz, transmit power $p_i = 30$ dBm, SINR threshold $\gamma_{\text{SINR}} = -77$ dB, cell tower height 50 m, cell-to-cell distance 100 m, and UE height 1.5 m. Since we focus on downlink, there is no intra-cell interference. Our approaches control cell selection but not resource allocation (e.g., scheduling of PRBs), which the cells coordinate transparently themselves using CoMP coordinated scheduling, avoiding inter-cell interference [24], [33].

UEs move according to the improved random waypoint model [53] with uniformly distributed velocity of 1 m/s–3 m/s and pause 2 time steps at waypoints. Each UE u_j has utility $U_j(t) = 10 \log_{10}(r_j(t))$ with $U_j^{\min} = -10$ and $U_j^{\max} = 10$, quantifying its QoE [3], [4]. As evaluation metric, we consider the avg. QoE over all time steps and UEs as defined in Sec. III-C, where positive values close to $U_j^{\max} = 10$ indicate excellent QoE, values around 0 indicate satisfactory QoE, and negative values close to $U_j^{\min} = -10$ indicate bad QoE. The chosen utility function and limits are examples and could also be chosen differently. In Sec. VII-C4, we show that our DRL approaches also adapt to other objectives and utility functions.

2) DRL Hyperparameters: To avoid time and resource-intensive hyperparameter tuning, we used the PPO default settings [50]: 1) Fully connected neural networks with two hidden layers (256 nodes, tanh activation). 2) Discount factor $\gamma = 0.99$. 3) Learning rate $\alpha = 5 \cdot 10^{-5}$. 4) Mini-batch size $|b| = 4000$ with 30 training epochs per mini-batch. 5) PPO clipping parameter of 0.3. 6) Kullback-Leibler (KL) coefficient initialization of 0.2 and target of 0.01. 7) Generalized advantage estimation (GAE) parameter $\lambda = 1$. 8) Value function parameter 1 and entropy coefficient 0. These settings are largely in line with general recommendations for practical DRL [54]. Tuning hyperparameters automatically for each scenario could further improve performance.

3) Baseline Solutions: We compare DeepCoMP, DD-CoMP, and D3-CoMP against three other approaches:

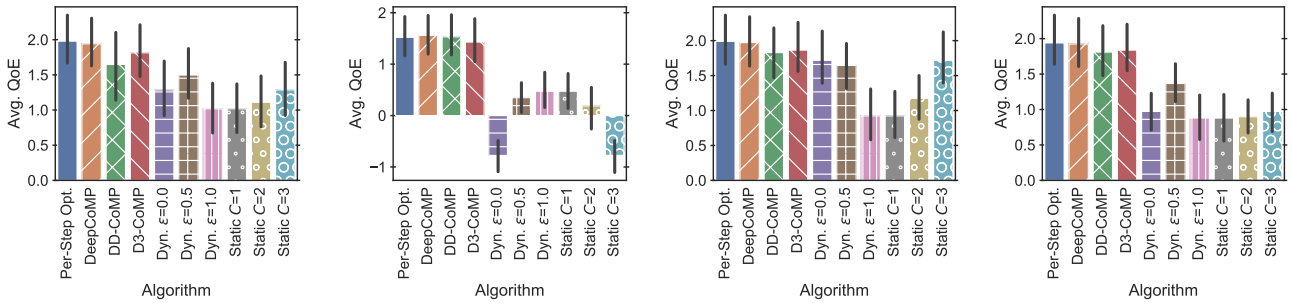
- *Dyn.:* A dynamic UE-centric multi-cell selection heuristic by Beylerian and Ohtsuki [8], which connects a UE u_j to all cells c_i with $\text{SINR}_{i,j}(t) \geq \epsilon \cdot \text{SINR}_j^{\max}(t)$, where $\text{SINR}_j^{\max}(t)$ is the SINR of the strongest cell. We evaluate $\epsilon \in \{0, 0.5, 1\}$. With $\epsilon = 1$, the heuristic only connects to the strongest cell, similar to common 3GPP-based approaches for single-cell selection [12].
- *Static:* A static clustering approach similar to [7]. It groups all cells into fixed clusters of size C and then connects UEs to all cells of the strongest cluster. We consider $C \in \{1, 2, 3\}$.
- *Per-Step Opt.:* A brute-force approach that has full knowledge and checks all actions to select the best one in each time step. The selected action is optimal *per step* but not necessarily in the long term. This approach is clearly impractical and only serves as a comparison.

All approaches have the same action space, but only our DRL approaches learn through feedback from their actions.

4) Execution and Plots: We repeated all experiments with 10 different seeds for training and testing, running on machines with 20 core Intel Xeon W-2145 CPUs and 100 GB RAM. Figures show the mean and 90 % confidence interval of the avg. QoE as defined in Sec. III-C.

C. Self-Adaption to Varying Scenarios

1) Varying Resource Allocation: We first consider the base scenario (Sec. VII-B1) with varying resource allocation schemes applied by the cells to allocate their available PRBs to connected UEs. In particular, we use the following schemes: 1) Resource-fair: All cells allocate their available PRBs equally among connected UEs [55]. 2) Rate-fair: Each cell ensures the same downlink data rate for its connected UEs by assigning more PRBs to UEs that are farther away.



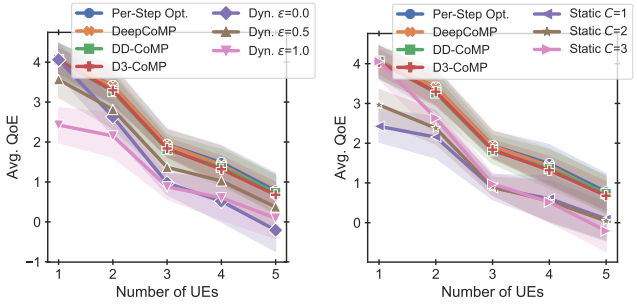
(a) Resource-Fair (b) Rate-Fair (c) Proportional-Fair (d) Heterogeneous

Fig. 4: Our DRL approaches adapt to varying resource allocation schemes and outperform other algorithms.

3) Proportional-fair: Cells allocate PRBs in a proportional-fair way, based on UEs' instantaneous and historical data rates [56]. In addition, we also consider a *heterogeneous* scenario with all three schemes used in parallel, each by one of the three cells.

Fig. 4 shows the resulting avg. QoE under the different schemes. While the avg. QoE is mostly satisfactory (around 0 or above) in these scenarios, there are significant differences between the algorithms. DeepCoMP's results are almost identical to the brute-force approach (within 0.4 % on avg.) and clearly outperform all other approaches (on avg. 2x better than the dynamic and 2.3x better than the static heuristic over all parameter settings). In fact, DeepCoMP is even slightly better (2.5 %) than the brute-force approach for the rate-fair scheme. Recall that brute force is only *per step* optimal but not necessarily long-term. Particularly, DeepCoMP learns to favor actions that may be suboptimal in short-term (e.g., connecting to a far-away cell) but pay off in long-term QoE. In contrast, the brute-force approach may get stuck at myopic optima since we only allow one connection or disconnection per UE and time step to limit overhead (Sec. III-B). With rate-fair allocation, switching to a far-away cell is very resource-expensive but may pay off when reducing competition at a closer cell.

DD-CoMP and D3-CoMP only have local observations and control for each UE and are thus slightly worse than DeepCoMP but still much better than the dynamic and static heuristics. DD-CoMP is on avg. 87 % better than the dynamic heuristic and 107 % better than the static heuristic. D3-CoMP is comparable to DD-CoMP but learns slightly better policies in some resource-fair scenarios where different UEs benefit from heterogeneous policies. Note that the performance of both heuristics strongly varies with their parameter (ϵ and C) and that the best parameter depends on the specific resource allocation scheme. For example, the dynamic heuristic works best with $\epsilon = 1$ in the rate-fair case, with $\epsilon = 0$ in the proportional-fair case, and with $\epsilon = 0.5$ in the other two cases. Similarly, the static heuristic works best with different parameter settings in different scenarios. Hence, for best results, these heuristics require knowledge of the resource allocation schemes and corresponding (possibly manual) configuration. In contrast, our three self-learning approaches autonomously



(a) DRL vs. Dynamic Heuristc (b) DRL vs. Static Heuristic

Fig. 5: DRL adapts to varying number of UEs.

adapt to the different resource allocation schemes and even the heterogeneous mix of schemes successfully from training experience. We emphasize again that the agents have no explicit knowledge about which resource allocation scheme is employed inside the cells!

2) Varying Number of UEs: We again consider the base scenario (Sec. VII-B1) with heterogeneous resource allocation and now vary the number of active UEs. Fig. 5 shows the avg. QoE for five different scenarios with 1–5 UEs, respectively. With few UEs (1 or 2), there is little competition for radio resources. Here, the static heuristic with $C = 3$ works well as it connects UEs to many cells and uses resources effectively. Conversely, selecting fewer cells ($C = 1$) leads to better results in scenarios with higher load (4 and 5 UEs) as it reduces competition among UEs by limiting a UE to a single cell. Overall, avg. QoE decreases with more UEs as competition increases. The dynamic heuristic works best with $\epsilon = 0.5$ across all scenarios (except for 1 UE) but is still clearly worse than our DRL approaches (e.g., DeepCoMP is 35 % better on avg.). Our three DRL approaches self-adapt to the number of UEs and learn cell selection policies that are very close to the per-step optimal brute-force approach. They consistently and considerably outperform both heuristics—in low-load as well as high-load scenarios and with any parameter setting.

3) Varying Cell Density: We now vary the cell-to-cell distance (80 m–120 m) in the base scenario (three UEs, heterogeneous resource allocation), where cell size is generally

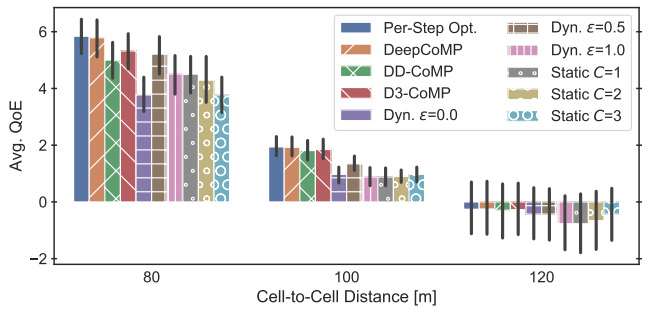
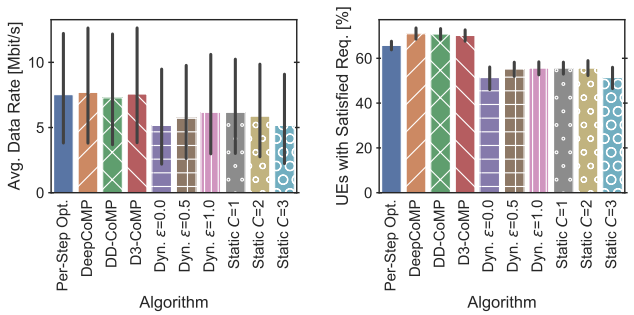


Fig. 6: DRL adapts to varying cell density.



(a) Maximizing Data Rate (b) Hard Data Rate Requirements

Fig. 7: DRL adapts to varying objectives and utility functions.

small due to strong path loss in the considered urban scenario (Sec. VII-B1). While the avg. QoE naturally decreases with sparser cells (from very good to satisfactory), Fig. 6 shows that varying cell density affects the heuristics with different parameters differently: The static heuristic works best with $C = 1$ in denser cells (80 m), where a single serving cell can achieve good QoE, and with $C = 3$ in sparser cells (120 m), where multiple connections are required for cell-edge UEs. The dynamic heuristic is also affected by the cell density ($\epsilon = 0$ vs. $\epsilon = 1$) but works best with $\epsilon = 0.5$ across all scenarios and is comparable to DD-CoMP and D3-CoMP at 80 m. However, our DRL approaches adapt much better to other cell densities and overall outperform both the dynamic and the static heuristic by 6 %–73 % on avg., depending on the parameter setting (ϵ and C).

4) Varying Objective and Utility Function: We consider our base scenario (Sec. VII-B1) with heterogeneous resource allocation again and investigate the impact of using different utility functions. Instead of the logarithmic utility function, we first consider maximizing UEs' data rate directly rather than their QoE by setting $U_j(t) = r_j(t)$ with $U_j^{\min} = 0$ and $U_j^{\max} = 1000$. We choose $U_j^{\max} = 1000$, i.e., capping data rates above 1000 Mbit/s, as an example for supporting even upcoming highly demanding and immersive services [57]. Fig. 7a shows UEs' resulting avg. data rate for each algorithm. Since UEs' data rates fluctuate heavily with their movement and distance to connected cells (due to strong path loss), the error bars are comparably large here. Still, the results clearly show that our proposed DRL approaches achieve similar avg. data rates as the per-step optimal brute force approach and outperform all other algorithms (by 18 %–49 % on avg., depending on ϵ and C). Particularly, they learn to only connect one UE per cell with the highest channel capacity. While this is unfair towards other UEs, e.g., at the cell edge, it maximizes the overall data rate and therefore the avg. utility in this case. This illustrates why maximizing only data rate, as commonly done in related work, often does not reflect user satisfaction and easily leads to undesired behavior.

As another example, in Fig. 7b, we consider the case that UEs have a hard data rate requirement (here 1 Mbit/s) that must be met. Hence, the utility follows a step function, where we set $U_j(t) = -10$ if $r_j(t) < 1$ Mbit/s and $U_j(t) = 10$ otherwise. Fig. 7b shows the resulting avg. percentage of UEs with satisfied data rate requirement, i.e., with $r_j(t) \geq 1$ Mbit/s. Our DRL approaches adapt to this scenario exceptionally well and outperform all other approaches (26 %–38 % more satisfied UEs than the dynamic and static heuristic). They learn to select cells such that UEs' data rate is just high enough (≥ 1 Mbit/s) and there are sufficient remaining radio resources to satisfy the data rate requirements of as many UEs as possible. Note that our DRL approaches are neither explicitly aware of UEs' required data rate nor the utility function but learn to adapt through feedback from UEs' instantaneous utility. The dynamic and static heuristic are also not aware of the required data rate but, unlike our DRL approaches, do not learn from experience. Instead, they tend to waste radio resources by either providing unnecessarily high data rates to some UEs or connecting cell edge users whose data rate still remains under the required threshold. Ultimately, this leads to many UEs with unfulfilled data rate requirements.

Fig. 7b shows that our DRL approaches even outperform the brute force approach. Again, this is possible because the brute force approach is only optimal *per step* but not necessarily in the long term. Indeed, this leads to suboptimal cell selection in this setting where one cell uses proportional fair resource allocation, which depends on UEs' historic long-term data rate. While our DRL approaches are not explicitly aware of the resource allocation schemes at the different cells, they implicitly learn to select actions with possibly suboptimal instantaneous utility but that optimize the avg. QoE in the long term. Overall, our DRL approaches effectively self-adapt to different objectives and utility functions and outperform all other algorithms. Again, we used these utility functions merely as examples in our evaluation. In practice, our approaches do not require defining a utility function at all and it suffices to approximate UEs' instantaneous QoE (e.g., with machine learning [37]).

D. Generalization and Transfer Learning

1) Generalization: In Sec. VII-C, we train our approaches from scratch in different scenarios and test how well they adapt to each scenario, using identical UE positions and movement during training and testing. Here, we consider pretrained DRL agents and evaluate their capability to generalize to unseen, randomized UE movement without further training. Note that our approaches are only indirectly aware of UEs' position and movement via observed SINR and QoE.

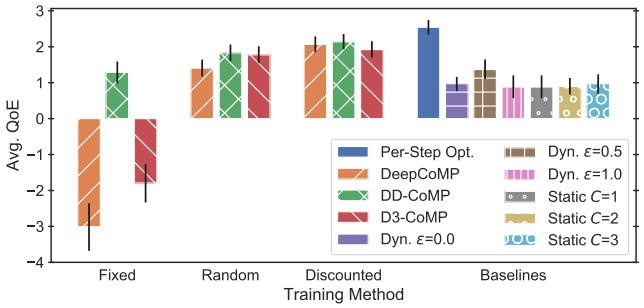
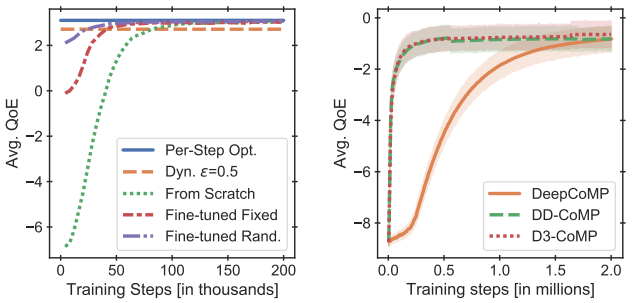


Fig. 8: Generalization to unseen UE movement.



(a) Online Transfer Learning (b) Scalability (20 UEs)

Fig. 9: DRL learning curves.

When trained on a single UE movement pattern, Fig. 8 ("Fixed") shows that DeepCoMP and D3-CoMP generalize poorly to new UE movement patterns (bad avg. QoE indicated by negative values) while DD-CoMP generalizes much better. This is because DeepCoMP and D3-CoMP learn cell selection separately for each UE, whereas DD-CoMP combines UEs' experiences and learns a joint policy for all UEs. Based on the diverse experience from these UEs, the joint policy is naturally more robust to different UE locations and movement. Accordingly, Fig. 8 ("Random") shows that training on randomly varying UE movement greatly improves generalization for DeepCoMP and D3-CoMP but also for DD-CoMP.

A challenge when training with randomized UE movement is the resulting high variance of experience collected during training, which is known to negatively affect performance of DRL approaches [58]. While randomized UE movement during training helps generalization, a simple option to reduce variance is to increase discounting of rewards by smaller γ (cf. Alg. 1, ln. 13). This decreases the weight of subsequent rewards and, thus, also reduces impact of variance in later time steps. Fig. 8 ("Discounted") shows that stronger discounting with $\gamma = 0.5$ (instead of 0.99) increases avg. QoE when training our DRL approaches on randomized UE movement, significantly outperforming the two heuristics by 40 %–143 % on avg. ("Baselines").

2) Online Transfer Learning: As UE movement patterns may change over time, our pretrained DRL approaches can adapt online to the current pattern using transfer learning. In fact, fine-tuning a pretrained policy to new UE movement with transfer learning is much more efficient than training a new policy from scratch, i.e., starting with a randomly initialized neural network. For DeepCoMP, Fig. 9a shows that such fine-tuning helps to quickly converge towards a very good policy (similar to Per-Step Opt.). Compared to training from scratch, it requires roughly half the amount of training steps until convergence—both for policies pretrained on fixed and randomized UE movement.

In practice, a safe and efficient approach would be to pretain a robust policy on randomized scenarios and then quickly and continuously fine-tune it for highly optimized QoE in the current scenario (comparable to "Fine-tuned Rand." in Fig. 9a). This approach quickly reaches and exceeds the quality of the best heuristic (dynamic with $\epsilon = 0.5$) within just 6 training iterations or 24 k training steps (batch size $|b| = 4000$; Sec. VII-B2). With ongoing transfer learning, the approach exceeds the best heuristic by ultimately 13 %. DD-CoMP and D3-CoMP converge even faster than DeepCoMP as we show in Sec. VII-E. For a production system, the efficiency of our prototype implementation could be further improved (e.g., more efficient implementation, hyperparameter tuning), leading to even faster online transfer learning.

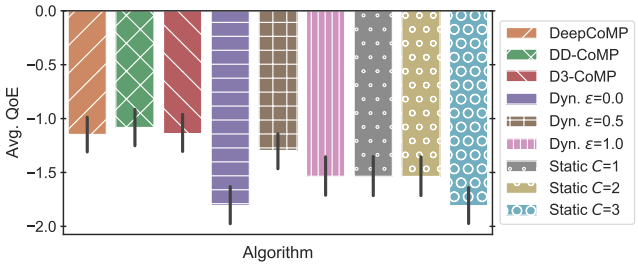


Fig. 10: DRL adapts to dynamic arrival of 15 UEs.

E. Scalability

1) Dynamic UE Arrival: We now explore a scenario similar to our base scenario (Sec. VII-B1) but with up to 15 UEs that arrive dynamically over time. Due to the significantly higher number of UEs and the resulting combinatorial explosion, the brute-force approach becomes intractable and is therefore omitted. Overall, the avg. QoE is much lower than in previous scenarios due to the higher load and increased competition between UEs (Fig. 10). With dynamically arriving UEs, DeepCoMP's observation and action space needs to be configured for the maximum number of UEs, here $M^{\max} = 15$ (Sec. IV-A4). In contrast, DD-CoMP and D3-CoMP have much smaller observation and action spaces that are invariant in the number of active UEs and can simply spawn new DRL agents whenever a new UE arrives (Sec. V-C and V-D). Consequently, DD-CoMP and D3-CoMP scale better to many UEs and converge much faster (investigated further in Sec. VII-E2). Still, DeepCoMP also learns a good cell selection policy within the given training steps (here, 1 million). All three DRL approaches adapt successfully to the dynamically changing number of active UEs and outperform both heuristics with all parameter settings by 14 %–67 %.

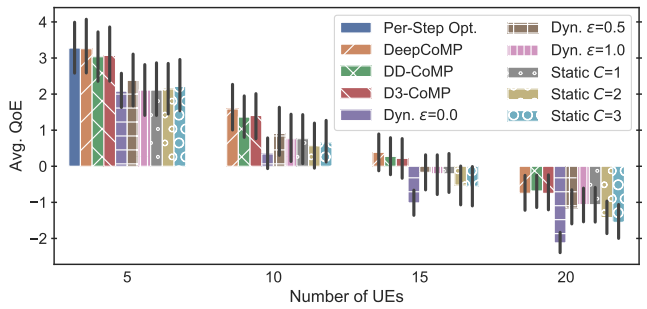


Fig. 11: Scalability to 7 cells and 5–20 UEs.

2) Large Scenario: We further investigate the scalability of our DRL approaches in a larger scenario with 7 cells and up to 20 moving UEs. Fig. 9b shows the learning curves of our approaches for 20 UEs, trained from scratch. DD-CoMP and D3-CoMP converge rapidly even in large scenarios since their observation and action spaces are small and invariant in the number of UEs. In contrast, DeepCoMP's observation and action spaces grow linearly with more UEs, leading to higher complexity for larger scenarios, and indeed to much slower training convergence. At 1 million training steps, DeepCoMP is still significantly worse than DD-CoMP and D3-CoMP but keeps learning with more training and eventually catches up with DD-CoMP and D3-CoMP.

Fig. 11 shows the final results of the DRL approaches trained for 2 million training steps, compared to the other algorithms. Again, avg. QoE decreases from rather good (> 0) to rather bad (< 0) with more UEs as competition increases. Due to the escalating complexity, the optimal brute-force approach is intractable and omitted for 10 or more UEs. Thanks to its global view and control of all UEs, DeepCoMP learns highly optimized cell selection and even exceeds DD-CoMP and D3-CoMP for 5–15 UEs. As shown in Fig. 9b for 20 UEs, it reaches comparable performance within 2 million training steps but is not yet fully converged and would likely further improve with even more training. In contrast, DD-CoMP and D3-CoMP converge rapidly and still clearly outperform both heuristics with all parameter settings. For example, DD-CoMP is 69 % better on avg. than the best heuristic (Dyn. $\epsilon = 0.5$) at 20 UEs.

VIII. Conclusion

We propose three self-learning DRL approaches that effectively self-adapt to various scenarios and objectives. They consistently and considerably outperform existing approaches without requiring detailed system knowledge or human instructions. DeepCoMP leverages its global view and control to learn highly optimized results and is useful for network-initiated cell selection when long training times are acceptable. Alternatively, DD-CoMP and D3-CoMP are suitable for either network-initiated or mobile-initiated cell selection, converge rapidly, and are particularly useful in practical large-scale scenarios. DD-CoMP learns a single, robust policy and D3-CoMP learns separate policies that can adapt to heterogeneous UEs. Our source code is available online [18] and can be used as a platform by others to study and evaluate their own solutions.

In future work, DeepCoMP, DD-CoMP, and D3-CoMP could be combined into a hybrid solution that dynamically switches to the most suitable approach. DD-CoMP and D3-CoMP could be further improved through recent advances in cooperative multi-agent DRL, e.g., curriculum learning [17]. Overall, we believe that our proposed DRL approaches are an important step towards self-adaptive, effective CoMP and higher QoE in practice, leading to happier customers and more profit for operators. We also expect the proposed techniques to carry over to other network control tasks, paving the way to zero-touch network management.

References

[1] 3GPP, "LTE Release 11: 3GPP TR 36.819," 3rd Generation Partnership Project (3GPP), Tech. Rep., 2012, version 11.1.0. [Online]. Available: https://www.3gpp.org/specifications/releases/69-release-11

[2] Y. Al-Eryani and E. Hossain, "The D-OMA method for massive multiple access in 6G: Performance, security, and challenges," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 92–99, 2019.

[3] M. Fiedler and T. Hoßfeld, "Quality of experience-related differential equations and provisioning-delivery hysteresis," in *ITC Specialist Seminar on Multimedia Applications-Traffic, Performance and QoE*. IEICE, 2010.

[4] S. Khirman and P. Henriksen, "Relationship between quality-of-service and quality-of-experience for public internet service," in *Workshop on Passive and Active Measurement*, 2002.

[5] D. Amzallag, R. Bar-Yehuda, D. Raz, and G. Scalosub, "Cell selection in 4G cellular networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 7, pp. 1443–1455, 2013.

[6] R. Vijayarani and L. Nithyanandan, "Dynamic cooperative base station selection scheme for downlink CoMP in LTE-advanced networks," *Wireless Personal Communications*, vol. 92, no. 2, pp. 667–679, 2017.

[7] P. Marsch and G. Fettweis, "Static clustering for cooperative multi-point (CoMP) in mobile communications," in *IEEE International Conference on Communications (ICC)*. IEEE, 2011, pp. 1–6.

[8] A. Beylerian and T. Ohtsuki, "Multi-point fairness in resource allocation for C-RAN downlink CoMP transmission," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, no. 1, pp. 1–10, 2016.

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

[10] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2018.

[11] L. You and D. Yuan, "Joint CoMP-cell selection and resource allocation in fronthaul-constrained C-RAN," in *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*. IEEE, 2017, pp. 1–6.

[12] Y. Zhou, C. Shen, and M. van der Schaar, "A non-stationary online learning approach to mobility management," *IEEE Transactions on Wireless Communications*, vol. 18, no. 2, pp. 1434–1446, 2019.

[13] R. Thomas, "Google's AutoML: Cutting Through the Hype," https://www.fast.ai/2018/07/23/auto-ml-3/ (accessed June 15, 2021), 2018.

[14] M. A. Alsheikh, D. T. Hoang, D. Niyato, H.-P. Tan, and S. Lin, "Markov decision processes with applications in wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1239–1267, 2015.

[15] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, pp. 1–50, 2021.

[16] C. Zhang and V. Lesser, "Coordinating multi-agent reinforcement learning with limited communication," in *International Conference on Autonomous Agents and Multi-agent Systems*, 2013, pp. 1101–1108.

[17] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 2017, pp. 66–83.

[18] S. Schneider, "DeepCoMP, DD-CoMP, and D3-CoMP GitHub Repository," https://github.com/CN-UPB/DeepCoMP, 2021.

[19] D. Xenakis, N. Passas, L. Merakos, and C. Verikoukis, "Mobility management for femtocells in LTE-advanced: Key aspects and survey of handover decision algorithms," *IEEE Communications surveys & tutorials*, vol. 16, no. 1, pp. 64–91, 2013.

[20] F. Giust, L. Cominardi, and C. J. Bernardos, "Distributed mobility management for future 5G networks: Overview and analysis of existing approaches," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 142–149, 2015.

[21] S. Mosleh, L. Liu, and J. Zhang, "Proportional-fair resource allocation for coordinated multi-point transmission in LTE-advanced," *IEEE Transactions on Wireless Communications*, vol. 15, no. 8, pp. 5355–5367, 2016.

[22] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[23] A. Tolli, H. Ghauch, J. Kaleva, P. Komulainen, M. Bengtsson, M. Skoglund, M. Honig, E. Lahetkangas, E. Tiirola, and K. Pajukoski, "Distributed coordinated transmission with forward-backward training for 5G radio access," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 58–64, 2019.

[24] F. Qamar, K. B. Dimyati, M. N. Hindia, K. A. B. Noordin, and A. M. Al-Samman, "A comprehensive review on coordinated multi-point operation for LTE-A," *Computer Networks*, vol. 123, pp. 19–37, 2017.

[25] H. Kim and Y. Han, "A proportional fair scheduling for multicarrier transmission systems," *IEEE Communications Letters*, vol. 9, no. 3, pp. 210–212, 2005.

[26] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[27] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2239–2250, 2019.

[28] M. Ozturk, M. Gogate, O. Onireti, A. Adeel, A. Hussain, and M. A. Imran, "A novel deep learning driven, low-cost mobility prediction approach for 5G cellular networks: The case of the control/data separation architecture (CDSA)," *Neurocomputing*, vol. 358, pp. 479–489, 2019.

[29] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs, and J. J. Alcaraz, "vrAIn: A deep learning approach tailoring computing and radio resources in virtualized rans," in *International Conference on Mobile Computing and Networking (MobiCom)*, 2019, pp. 1–16.

[30] M. Elsayed, K. Shimotakahara, and M. Erol-Kantarci, "Machine learning-based inter-beam inter-cell interference mitigation in mmWave," in *IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[31] D. Niyato and E. Hossain, "Dynamics of network selection in heterogeneous wireless networks: An evolutionary game approach," *IEEE Transactions on Vehicular Technology (TVT)*, vol. 58, no. 4, 2008.

[32] A. Xie, J. Harrison, and C. Finn, "Deep reinforcement learning amidst lifelong non-stationarity," *arXiv preprint arXiv:2006.10701*, 2020.

[33] A. Marotta, D. Cassioli, C. Antonelli, K. Kondepu, and L. Valcarenghi, "Network solutions for CoMP coordinated scheduling," *IEEE Access*, vol. 7, pp. 176 624–176 633, 2019.

[34] R. Irmer, H. Droste, P. Marsch, M. Grieger, G. Fettweis, S. Brueck, H.-P. Mayer, L. Thiele, and V. Jungnickel, "Coordinated multipoint: Concepts, performance, and field trial results," *IEEE Communications Magazine*, vol. 49, no. 2, pp. 102–111, 2011.

[35] Y. L. Lee, T. C. Chuah, J. Loo, and A. Vinel, "Recent advances in radio resource management for heterogeneous LTE/LTE-A networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2142–2180, 2014.

[36] R. Jain, "Quality of experience," *IEEE MultiMedia*, vol. 11, no. 1, pp. 96–95, 2004.

[37] V. Menkovski, G. Exarchakos, and A. Liotta, "Online QoE prediction," in *International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE, 2010, pp. 118–123.

[38] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 448–456. [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.html

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

[40] M. Hutter, "General discounting versus average reward," in *International Conference on Algorithmic Learning Theory*. Springer, 2006, pp. 244–258.

[41] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[42] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote, "A survey of learning in multiagent environments: Dealing with non-stationarity," *arXiv preprint arXiv:1707.09183*, 2017.

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

[44] Network and E. I. L. N. S. Q. Signaling Working Group: Verizon, Cisco, "Verizon 5G TF; Network and Signaling Working Group; Verizon 5th Generation Radio Access; Overall Description (Release 1)," Tech. Rep., 2016.

[45] P. Casas, A. D'Alconzo, F. Wamser, M. Seufert, B. Gardlo, A. Schwind, P. Tran-Gia, and R. Schatz, "Predicting QoE in cellular networks using machine learning and in-smartphone measurements," in *International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2017, pp. 1–6.

[46] 3GPP, "Release 15; NR; Radio Resource Control (RRC); Protocol specification; TS 38.331," 3rd Generation Partnership Project (3GPP), Tech. Rep., 2021, version 15.13.0.

[47] T. Contributors, "TensorFlow Lite," https://www.tensorflow.org/lite, 2021.

[48] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra, "Sim2Real predictivity: Does evaluation in simulation predict real-world performance?" *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6670–6677, 2020.

[49] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[50] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica, "RLlib: Abstractions for distributed reinforcement learning," in *International Conference on Machine Learning*, 2018, pp. 3053–3062.

[51] A. Toskala and H. Holma, *WCDMA For UMTS: HSDPA Evolution And LTE*. Wiley, 2007.

[52] A. Medeisis and A. Kajackas, "On the use of the universal Okumura-Hata propagation prediction model in rural areas," in *Vehicular Technology Conference (VTC)*, vol. 3. IEEE, 2000, pp. 1815–1818.

[53] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," in *IEEE International Conference on Computer Communications (INFOCOM)*, vol. 2. IEEE, 2003, pp. 1312–1321.

[54] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski *et al.*, "What matters in on-policy reinforcement learning? a large-scale empirical study," *arXiv preprint arXiv:2006.05990*, 2020.

[55] A. Saxena and R. Sindal, "Strategy for resource allocation in LTE-A," in *International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*. IEEE, 2016, pp. 29–34.

[56] J. Yang, Z. Yifan, W. Ying, and Z. Ping, "Average rate updating mechanism in proportional fair scheduler for HDR," in *IEEE Global Telecommunications Conference (GLOBECOM)*. IEEE, 2004, pp. 3464–3466.

[57] Y. Qi, M. Hunukumbure, M. Nekovee, J. Lorca, and V. Sgardoni, "Quantifying data rate and bandwidth requirements for immersive 5G experience," in *2016 IEEE International Conference on Communications Workshops (ICC)*. IEEE, 2016, pp. 455–461.

[58] E. Greensmith, P. L. Bartlett, and J. Baxter, "Variance reduction techniques for gradient estimates in reinforcement learning," *Journal of Machine Learning Research*, vol. 5, no. Nov, pp. 1471–1530, 2004.

Stefan Schneider obtained his master's degree in 2017 at the Paderborn University, Germany. He is currently pursuing his Ph.D., working as a research associate at the university's computer networks group. His main research interests are network softwarization, cloud & edge computing, 5G and beyond—particularly in combination with machine learning. He has worked on multiple large research projects with industry partners and has been leading his own research project (RealVNF) in 2018–2021.

Holger Karl leads, since July 2021, the Internet Technology and Softwarization research group at Hasso Plattner Institute, University of Potsdam; before that, he was professor for Computer Networks at Paderborn University. He has two main research interests; the first one is advanced wireless network, e.g., cooperative diversity techniques and resource management in factory-floor automation. His second interest is future Internet, specifically unifying concepts like SDN and NFV with edge computing across different scenarios.

Ramin Khalili received his B.Sc. from Shiraz University, his M.Sc. from the Sharif University of Technology, both in Iran, and his Ph.D. in computer networks and distributed systems from UPMC, France. He was with the University of Massachusetts at Amherst, EPFL, and the Telekom Innovation Laboratories in Berlin, before joining the Huawei Research Center in Munich, Germany. Ramin published over fifty scientific papers in topics related to wireless networking, optimization, and machine learning and received multiple best paper awards.

Artur Hecker (MSc Universität Karlsruhe and PhD ENST, Paris) is Director of networking research at the Advanced Wireless Technology Laboratory of Huawei Munich Research Center. From 2006 to 2013, Artur was Associate Professor at Télécom ParisTech, where he was leader of Security and Networking research. Overall, Artur looks back at almost 20 years of entrepreneurial, academic and industry experience in networks, systems and system security.