Cellular Network Capacity and Coverage Enhancement with MDT Data and Deep Reinforcement Learning

Marco Skocaj[1, *], L.M. Amorosa[1], G. Ghinamo[2], G. Muratore[2], D. Micheli[2], F. Zabini[1] and R. Verdone[1]

[1]DEI, University of Bologna & WiLab - National Laboratory for Wireless Communications (CNIT)
[2]TIM Telecom Italia
*Corresponding author: marco.skocaj@unibo.it

Abstract—Recent years witnessed a remarkable increase in the availability of data and computing resources in communication networks. This contributed to the rise of data-driven over model-driven algorithms for network automation. This paper investigates a Minimization of Drive Tests (MDT)-driven Deep Reinforcement Learning (DRL) algorithm to optimize coverage and capacity by tuning antennas tilts on a cluster of cells from TIM's cellular network. We jointly utilize MDT data, electromagnetic simulations, and network Key Performance indicators (KPIs) to define a simulated network environment for the training of a Deep Q-Network (DQN) agent. Some tweaks have been introduced to the classical DQN formulation to improve the agent's sample efficiency, stability and performance. In particular, a custom exploration policy is designed to introduce soft constraints at training time. Results show that the proposed algorithm outperforms baseline approaches like DQN and best-first search in terms of long-term reward and sample efficiency. Our results indicate that MDT-driven approaches constitute a valuable tool for autonomous coverage and capacity optimization of mobile radio networks.

I. Introduction

As next-generations network management is foreseen to reach a level of complexity beyond human's ability to fully comprehend and control [1] , the last decade has been characterized by an up-growing interest in network automation-related services and applications. The requirement to meet the needs of both users and operators in a cost effective way has triggered research to add intelligence and autonomous adaptivity into future cellular networks [2]. The first network automation technology introduced by 3GPP was Self-Organizing Network (SON), initially designed for Long Term Evolution (LTE) technology [3, 4]. SON aims to reduce manual involvement in planning, configuration, management, optimization, and healing of mobile radio access networks. The next step towards the minimization of human intervention is the zero-touch paradigm, for which pervasive artificial intelligence (AI) algorithms will play a fundamental role. Recent years also witnessed an unprecedented availability of data and computing resources in many engineering domains [5]. This contributed to the rise of data-driven over model-driven AI. The benefit of data-driven approaches for next-generation networks, particularly for coverage and capacity optimization (CCO) tasks, has already

been discussed in the literature [6].

Minimization of Drive Tests (MDT) is a feature introduced in 3GPP Release 10 [7] that enables operators to utilize users' equipment (UE) to collect radio measurements and associated location information [8] (e.g., via GPS or WLAN localization (Release 16)). Collected MDT measurement reports can be aggregated into datasets with user location information (ULI) as well as statistical channel state information, providing a realistic description of the network state and allowing network troubleshooting and inference of human behavior-related statistics [9–11]. With this contribution, the authors aim to shed light on the potential of MDT data-driven algorithms for network automation. To this end, a CCO problem is addressed for a selected cluster of the TIM Italy's network by optimizing cells' antenna downtilts with the use of an MDT-driven Deep Reinforcement Learning (DRL) algorithm.

A. State of the art

CCO is a challenging optimization problem of cellular networks which has widely been studied in the literature, especially in the broader subject of SON [2, 12, 13]. One way to tackle CCO is to allow coverage regions to change by modifying cells' elevation tilts. Some early decade contributions made use of research operations methods [12, 13] to tune antennas parameters, while most recent years, motivated by the late success of Deep Learning (DL) and Reinforcement Learning (RL), have seen the rise of solutions employing DRL algorithms [14–17]. RL, indeed, seems to be a valuable tool for CCO since it can learn and adapt to the dynamics of the environment [14]. In [14], a two-step algorithm (multi-agent mean-field RL and single-agent RL) provides a scalable solution for online antenna tuning of a multi-tier network. In [15], the CCO problem is addressed by balancing under-vs-over coverage using Deep Deterministic Policy Gradient (DDPG) and Bayesian Optimization. In [17], a solution for optimizing the antenna parameters of a HetNet is proposed addressing the unstable convergence of hyperparameters. Finally, [16] proposes a safe antenna tilt update policy to avoid the execution of degrading actions and improve system's reliability. All of the

aforementioned papers are innovative, as they respectively propose solutions to improve scalability [14, 17], sample-efficiency [15] and reliability [16]. Nevertheless, none of them makes use of realistic data measurements like MDT for network state representation and training. At most, [15] employs electromagnetic simulations for Reference Signal Received Power (RSRP) map generation. In this paper, the agent is trained by direct interaction with a simulated network environment jointly employing MDT data, network KPIs, and electromagnetic simulations from the TIM cluster.

B. Paper contributions

The proposed work aims to show the potential of MDT data-driven algorithms for network automation. To the best of the authors' knowledge, the joint use of MDT data and DRL constitutes a novel approach to CCO problems. Even though being standardized by 3GPP as a key SON feature [7], MDT data have not been extensively investigated in the literature of automated communication systems. Moreover, their application is mainly circumscribed to anomalous coverage detection problems [18–20] and Quality of Service (QoS) verification [21]. The only contribution where MDT data are employed in the context of a capacity optimization problem is [22], where a two-steps algorithm - cell outage detection and cell outage compensation - is proposed for the management of a split-architecture network.
Nevertheless, only the former employs MDT-based anomaly detection to identify coverage issues, while the latter makes use of an actor-critic RL algorithm based on control/data signals to adjust antenna settings. It follows that, even in this contribution, MDT data are relegated to coverage issues detection. The benefits of employing a dataset with radio measurement reports and ULI for capacity optimization, instead, are manifolds and discussed throughout the following sections.

Another element of distinction with the literature concerns the approach adopted for the optimizer. The use of Deep-Q Networks (DQN) as Q-function $Q_\Theta(s, a)$ approximators is known and widely utilized in literature [23, 24], but is affected by sample inefficiency since the DQN agent struggles to explore the states' space efficiently and exhaustively. Consequently, some of the works in the literature employed other sample-efficient algorithms for CCO problems. In [15], the authors compare the sets of Pareto frontiers for balancing under-vs-over coverage found by using DDPG and Bayesian Optimization (BO), with the result of DDPG obtaining slightly better performance at the expense of two orders of magnitude slower exploration phase. Instead, the effort of this work is focused on identifying a method for improving a centralized DQN agent's sample efficiency. To this end, we introduced some tweaks, thoroughly described in Sec. 5, to the classical DQN formulation. In particular, a custom exploration policy, namely depth-wise-$\epsilon\eta$-greedy (DW-$\epsilon\eta$-greedy) policy is proposed to introduce soft constraints at training time while adequately balancing the exploration vs. exploitation

tradeoff. Results show that the agent's efficiency improved both in terms of training time, stability and performance. Thanks to these enhancements, it was possible to fit the antenna tuning problem to a medium-sized cluster covering a geographic area of approximately 26 $[Km^2]$.

Multi-agent systems solutions are currently capturing significant research interest because they can remarkably enhance the scalability of network optimization algorithms [14, 25–29]. The design of distributed multi-agent systems is sometimes also a necessity motivated by the partial state observability of edge network elements. Nonetheless, such design poses many challenges, like efficient communication between agents [30–32] and non-stationarity [33], oftentimes leading to unstable training and sub-optimal solutions. Be that as it may, numerous reasons motivated a centralized approach for this work:

- The use of MDT data allows full state observability with respect to other kinds of signals and data used to represent the observation space in RL-communications problems.
- MDT data are collected from TIM's network infrastructure, therefore we must abide to actual cell deployment. Given the current network density, a CCO problem can be effectively tackled with a single agent approach even for urban areas of medium-sized cities like Bologna, Italy.
- SON envisages three kinds of architectures: centralized, hybrid, distributed. A centralized solution is preferred for optimality and stationarity, whereas a distributed one is imperative to tackle the curse of dimensionality. The best tradeoff approach for larger cities or denser urban areas might envision a hybrid solution where multiple agents are responsible for variable-sized adjacent clusters instead of single e/gNodeBs. These hybrid agents might learn to maximize a common reward cooperatively based on their mild-to-severe mutual interference. To this end, the proposed contribution is also propaedeutic for future work on distributed/hybrid solutions for larger or denser geographical areas. Previous works in the literature already demonstrate that DQNs can become a practical tool for studying the decentralized learning of multiagent systems living in highly complex environments [34].

The rest of the paper is organized as follows: Sec. 2 describes the System model. Sec. 3 provides a detailed description of the simulated network environment. Sec. 4 formalizes the problem as Markov Decision Process (MDP), whereas Sec. 5 illustrates the details introduced for tweaking a DQN agent. Finally, Sec. 6 presents the obtained results and analyzes the agent's compactness and stability, while Sec. 7 draws the final remarks.

II. SYSTEM MODEL

The system model comprises a cellular network deployment for a selected area of interest, incorporating a total number of "target" and "boundary" eNodeBs. The former are the ones actively targeted by a DRL agent, whereas the latter are out of the scope of the optimization process but are still taken into account to consider boundary effects of the agent's actions. In particular, the reference scenario involves TIM's network LTE deployment for an area of approximately 26 $[Km^2]$ in the North of the city of Bologna, Italy (Fig. 1), incorporating a total of 18 eNodeBs (6 sites with tri-sectorial distribution). Among these, 9 eNodeBs have been selected as "target nodes" (pink markers, Fig. 1), while the remaining ones (grey markers) are selected as "boundary" cells. The selected area is interesting and challenging from the network configuration point of view since it yields a heterogeneous propagation scenario: highway, urban areas, and agricultural fields.
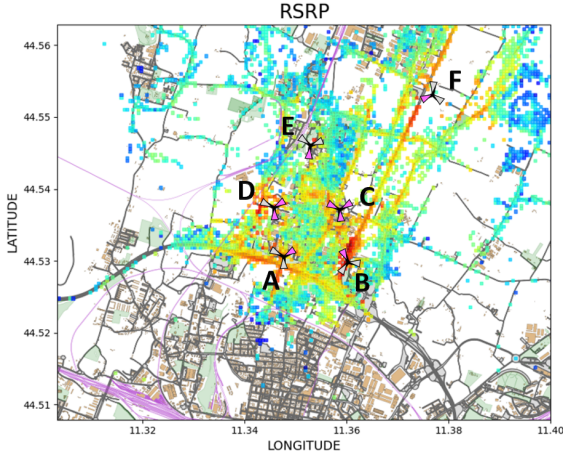

Fig. 1: TIM's Network deployment - North Bologna Area - RSRP measurements from MDT

A Reinforcement Learning agent for network parameters optimization could be trained by direct interaction with a physical network. However, this is generally very inconvenient due to slow training times and potential degrading actions performed during the exploration phase of the training. In this regard, many contributions in literature focused on risk sensitivity and damage avoidance during the exploration phase of an agent under the mantle of safe RL [16, 35].

Nevertheless, safe RL is not addressed by this work since, due to the following reasons, an offline approach was chosen for the training of the agent:

1) The simulated network environment, as described in Sec. 3, is based on MDT data processing. The possibility of relying on time-referenced and geo-located actual traffic distributions, as well as measured radio indicators, provides a far more accurate insight of the network with respect to solutions based on simulated

traffic and model-based received power. For instance, measured RSRP and Reference Signal Received Quality (RSRQ) intrinsically hold local information such as LoS/NLoS UE's condition, material and shape of surrounding buildings, UE's specific antenna model, average human-body absorption, and many more.

2) For an online training, a state space based on real-time reward signal should be employed. Consequently, it would be impossible to utilize MDT data, with the disadvantage of losing ULI.

3) Antenna tuning's dynamic is highly correlated to changes in traffic patterns. Such changes evolve at two different paces:

- *Fast*: different antenna configurations might be well-suited for different day periods (e.g., the flow of people leaving their offices and populating roads) but occur with a daily and weekly periodicity, as evident from Fig. 2c and 2a.
- *Slow*: long-term differences in daily traffic patterns are observable over extensive periods, like multiple weeks or months. Fig. 2a and 2b show autocorrelation plots of the normalized number of RRC connected UEs for the city of Bologna over few weeks (Fig. 2a) and a full year (Fig. 2c). The monotonically decreasing behavior of the autocorrelation plot denotes an increasing variability in the daily traffic patterns.

Parallel training of specialized agents might address the problem of fast dynamicity for different time ranges manifesting quasi-static behaviors and exploiting patterns' daily periodicity. The agents' policies validity would last for a much larger period than training time. Considering a scenario where trained agents are employed and running on a physical network, MDT measurement campaigns might be periodically collected to test agents' performance and proactively identify outdated optimizers.

The overall system framework is well-suited for a block description (Fig. 3).

On the left side, a simulated network environment receives MDT data, traffic KPIs, and electromagnetic simulations as inputs. The scope of the environment is to accurately represent the effect that the reconfiguration of a parameter (e.g., eNodeBs' antenna tilts), which corresponds to an agent's action, has on network performance. To this end, the environment involves a set of data processing methods, accurately described in Sec. 3. Electromagnetic simulations have been generated considering single eNodeB's antenna tilt configurations (a total of $N_{cells} \times N_{tilts}$ simulations) with the use of a TIM's proprietary tool. In contrast, MDT data and KPIs have been collected from multiple-days measurements. The simulated environment interacts with a tweaked DQN agent during training time, described in Sec. 5. The agent's goal is to eventually learn which actions would lead to a maximized cumulative future reward R_c

(a) RRC Connected UEs' autocorrelation plot - 2 weeks

(b) RRC Connected UEs' autocorrelation plot - 1 year

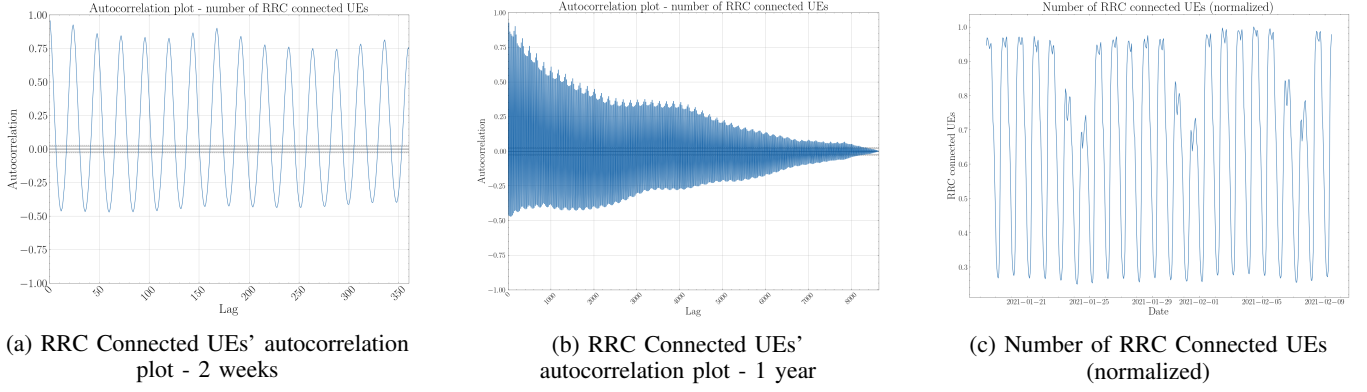(c) Number of RRC Connected UEs (normalized)

Fig. 2: Daily periodicity is evident from Fig. (a) and (c). Weekly periodicity is observable from a slight increase of the autocorrelation values over a 7 day period Fig. (a).
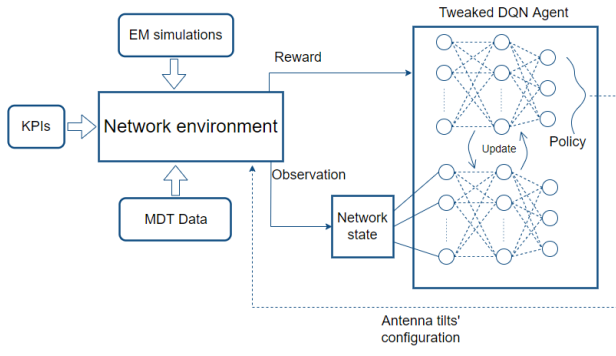

Fig. 3: System Model

based on its environment observation. The latter can be expressed as:

$$R_c = \sum_{i=t}^{\infty} \gamma^{i-t} R_{\pi(s_i)}(s_i, s_{i+1}) \qquad (1)$$

where γ is a discount factor for future rewards and $R_{\pi(s_i)}$ is the immediate reward obtained after the agent's policy π selects action a_i based on observation of state s_i.

As better described in the next two sections, the quantities in Tab. I have been collected from MDT measurement reports to build the observation space for the agent.

PCELL_RSRP	UE's measured RSRP from the serving primary cell
NCELLS_RSRP	UE's measured RSRP from up to 8 adjacent non serving cells
PCELL_RSRQ	UE's measured RSRQ from the serving primary cell
CALL_IDs	Anonymous temporary ID identifying a UE's RRC connection

TABLE I: MDT observation space

Moreover, two traffic KPIs were gathered with a cell-level spatial granularity: *average active number of UEs per Time Transmission Interval*, from now on referred to as *act_UEs*, and *average cell load ρ*.

III. SIMULATED NETWORK ENVIRONMENT

The simulated network environment is composed by 4 logical blocks performing different tasks, summarized by Fig. 4.
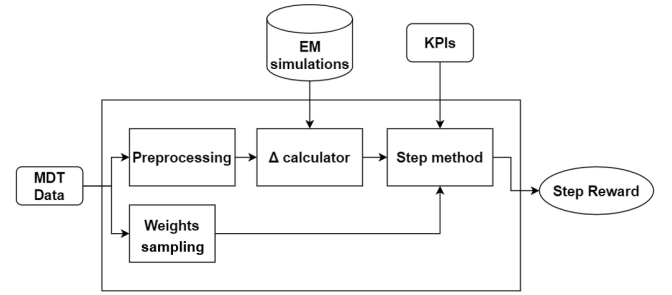

Fig. 4: Simulated environment - block system view

A. Pre-processing

The collected MDT data are divided into time slots exhibiting quasi-static traffic behavior (e.g., 2 P.M. - 6 P.M.), determined through the *act_UEs* KPI. MDT data are then aggregated into pixels of 50 $[m^2]$ dimension, aligned coherently with the ones produced by an electromagnetic simulation tool. [36] describes how the optimal area division into bins (pixels) affects three kinds of MDT measurement reports errors: 1) Positioning error, 2) Quantization error, 3) Scarcity of reports. The pixel size in this contribution has been determined prioritizing the minimization of errors 1 and 3, since a quantization error due to averaging over larger bins does not significantly affect the agent's decision on antenna tilt parameters.

Only pixels with statistical relevance are retained. Statistical relevance is determined based on a twofold condition:

1) The number of MDT samples in every pixel exceeds a pre-defined threshold. To this end, the minimization of the scarcity of reports error translates to a lower % of discarded pixels.
2) At least one MDT sample in every pixel yields measurements of a minimum of three interfering cells.

Another motivation behind the pixelization process of MDT data is to reduce the variance of the single measurements and provide a dimensional-consistent input to the DQN. Each pixel is fully characterized by three aggregated quantities that are thus computed:

1) **RSRP**: linear average of RSRP measurements, transformed in dB.

2) **WEIGHT**: The default weight is determined as the number of different CALL_IDs of a pixel in the reference time slot. As previously expressed, a CALL_ID relates to a single RRC_Connection establishment. For Immediate mode MDT, RRC_Connected UEs are configured for periodic measurement reporting (in our setup, every 5 seconds). Consequently, a very different number of reports might be produced according to each UE's connection time. Hence, assigning weights based on the number of connections over the total number of samples prioritizes fairness among users. At the beginning of a new episode, each pixel weight is sampled from a Poisson probability distribution (Weights sampling block, Fig. 4), in which the average value λ is set equal to the number of original CALL_IDs per pixel. This procedure confers generality to the environment and prevents the agent from overfitting the training data.

3) **SINR**: Signal-to-Noise and Interference Ratio (SINR) computation can be carried out following either one of the two proposed formulas:
   a) *RSRP-based SINR*:

$$\text{SINR}[dB] = \text{RSRP}_{\text{Pcell}}[\text{dBm}]$$
$$- 10 \log_{10}\left(\sum_{i=1}^{N} \text{RSRP}_{\text{Ncell},i}[\text{mW}]\right.$$
$$\left. + N_0\left[\frac{\text{mW}}{\text{Hz}}\right] \cdot B_R[\text{Hz}]\right) \quad (2)$$

   At the numerator, the average measured RSRP from the pixel's serving cell, while, at the denominator, interference is expressed as a sum of interfering cells' RSRP. Noise is accounted assuming a noise figure of 7 [dB] and reference and antenna temperature equal to 290 [K]. The noise power referred to a RE bandwidth of 15 [kHz], according to RSRP definition [37], is therefore approximately -125[dBm].

   b) *RSRQ-based SINR*:

$$\text{SINR} = \frac{12 \cdot \text{RSRQ}}{1 - \rho \cdot \text{RSRQ} \cdot 12} \quad (3)$$

   A second possible method for evaluating SINR starts from the RSRQ and average cell load ρ, indicating the % of occupied Physical Resource

Blocks (PRBs) in the corresponding serving cell. 12 is the number of OFDM sub-carriers in an LTE PRB, RSRQ at the numerator is the RSRQ aggregated on an MDT data pixel basis, in linear scale. RSRQ is defined as $\frac{N_{PRB} \cdot \text{RSRP}}{\text{RSSI}}$ [37], where N_{PRB} is the number of resource blocks of the E-UTRA carrier Reference Signal Strength Indicator (RSSI) measurement bandwidth. Since RSRP is a measurement of the demodulated Channel Reference Signal (CRS) symbols, the numerator accounts only for the co-channel serving cell signal contribution. In contrast, the denominator's RSSI is a wideband measure of co-channel serving and nonserving cells, adjacent channel interference, and noise. As a consequence, (3) is a better descriptor of the SINR when performance are limited by interference and susceptible to load variations in the cells, compared to (2), where SINR is evaluated as a ratio of reference signals demodulated powers.
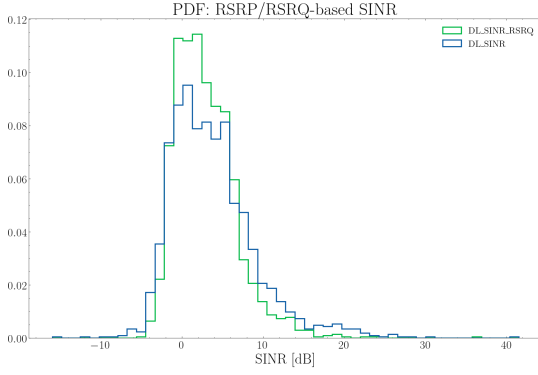
B. Electromagnetic simulator

Electromagnetic simulations are stored for every cell and each selectable downtilt parameter, totaling $P \times C$ simulations, where P is the number of downtilts and C is the number of cells in the cluster. Each simulation computes the electric field intensity [dBuV/m] for a grid of 50 $[m^2]$ pixels covering the area of interest, considering a single cell as the emitting source. The electric field intensity is then converted to RSRP [dBmW] considering a reference receiver bandwidth of one Resource Element (RE) (15 [kHz]).
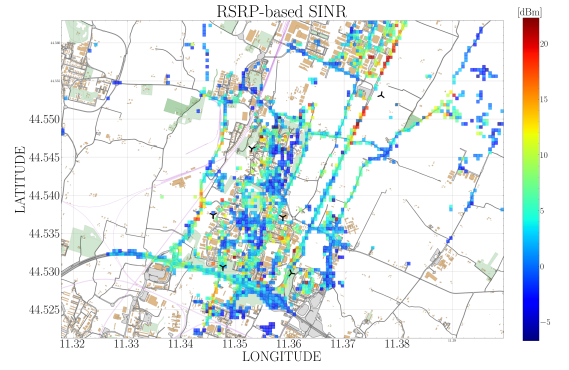Once every simulation pixel's RSRP is obtained, it is possible to compute the SINR from (2). Each pixel is associated with a serving primary cell selected with a max-RSRP criterion. Ultimately, ΔR and ΔS are computed as the dB-difference between the simulated and MDT RSRP and SINR for every pixel. These two quantities are representatives of the electromagnetic simulation tool's estimation error with respect to the UEs' measurements reports. Such error is assumed to be independent of the antenna configuration on the field, meaning that the same difference would be obtained from MDT data and simulations relative to a set of antenna parameters $A = \{P_1, P_2, ..., P_C\}$, or MDT data and simulations relative to a different set $B = \{P'_1, P'_2, ..., P'_C\}$.
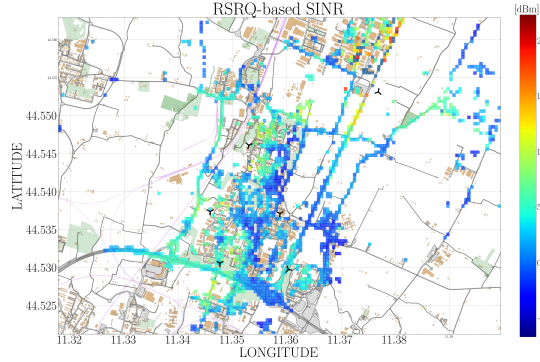
C. Step method

The step method implements an agent's action's effect on network performance, given a state observation. The immediate (or step) reward, described in Sec. 4.3., is returned as a parameter. The method entails the joint use of MDT data, em simulations, and KPIs to modify the network state. Since it is infeasible to collect MDT measurements corresponding to each agent's action, the step method defines a set of procedures to obtain synthetic MDT data (MDT') by making use of the previously computed Δ and simulation

(a) PDF: RSRP/RSRQ-based SINR

(b) RSRP-based SINR

(c) RSRQ-based SINR

Fig. 5: Comparison RSRP/RSRQ-based SINR

results. Every time the step method is called, the following procedures are executed:

1) Weight initialization: Weight sampling is performed from a Poisson distribution, as described in the section above.
2) The number of *act_UEs* is re-distributed among the pixels based on their weight. Each pixel is associated with a % of the total number of users described by the KPI.
3) A set of C simulations, containing all pixels' simulated RSRP, is retrieved from memory according to the cluster antennas configuration at step $t + 1$. The latter is fully determined by the agent's chosen actions from the beginning of an episode, as better specified in the next section.
4) Pixels Cell Reselection: each pixel's simulated RSRP is selected as the

$$max[RSRP_1, RSRP_2, ..., RSRP_C] .$$

Each vector's element indicates the simulated RSRP from one of the C cells. The serving cell is re-assigned accordingly.
5) Each pixel's simulated SINR is re-computed as (2).
6) MDT' RSRP and SINR values are obtained according to Alg. 1
7) Based on newly assigned pixels (point 4), each cell's *act_UEs* is recomputed (every pixel carries a % of active UEs).

Algorithm 1 Step method

---

for i in pixels **do**
$\quad MDT'_{RSRP,i} = SIM_{RSRP,i} + \Delta R_i$
$\quad MDT'_{SINR,i} = SIM_{SINR,i} + \Delta S_i$
end for

---

8) Step-reward computation.

IV. PROBLEM FORMULATION

A Reinforcement Learning approach requires a formulation of the optimization problem as a Markov Decision Process, in which the action-space, state-space, and reward must be appropriately defined.

A. Action Space

The agent operates in a NP-Hard setting since it has to jointly optimize P parameters' configuration on a set of C cells. The overall number of combinations grows exponentially as P^C. This formulation is represented in Fig. 6 as a single-step MDP where P^C branches generate from a root node.

Such formulation would be highly inefficient and infeasible from the computational point of view, particularly when moving to medium or large cluster sizes. Instead, it is much more convenient to represent the optimization problem with an MDP tree formulation, where the depth of the tree depends on the number of cells. At each time step of a fixed-length episode, the agent must decide 1) the selection of a
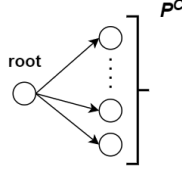
Fig. 6: single-step MDP

target cell among the c cells and 2) a proper configuration of its P parameters. The cardinality of the action set at each time step is then reduced to $P \times C$.
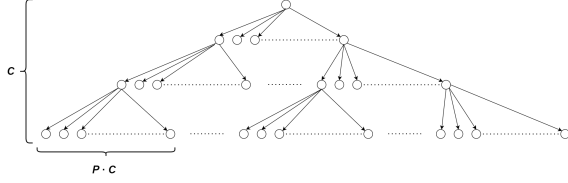

Fig. 7: multiple-step MDP

Even though the overall number of nodes in a complete tree grows as $(P \times C)^{C+1} - 1$, therefore faster than the previous formulation, this MDP formulation unlocks the potential to sequentially prune the tree based on the estimated Q-values at training time. Here lies the exploitation vs. exploration tradeoff: quickly exploiting the gained knowledge about the state space (Q-values estimates) translates to a faster tree pruning, as non-promising branches are discarded. Viceversa, an extended exploration phase allows the agent to explore the state space more exhaustively, reducing the risk of getting stuck into local optima.

To this end, the DW-$\epsilon\eta$-greedy exploration policy is presented in the next subsection as an improvement of the classical ϵ-greedy policy to introduce soft constraints at training time and to tackle this tradeoff efficiently. A set of 5 possible discrete tilts, each corresponding to an additional $-\Delta 2°$ with respect to the ground perpendicular direction, can be selected by the agent for each of the 9 target cells of the cluster. The set of new cells' parameters configuration is given by: $A = \{T_1, T_2, ..., T_p\}$ at the end of each episode.

B. State Space

At each time step, the DQN agent receives as input an observation of the network state, represented by the quantities in Tab. II. The reference area of interest can be represented by a matrix of $Y \times Z$ pixels composed of three different channels:

- RSRP is used to manage mobility and identify coverage issues.
- WEIGHT is used to distinguish between most and less relevant pixels.
- SINR is used for the computation of the throughput.

All channels are normalized before being processed by the agent's DNN.

Additionally, $(P \times C)$ one-hot encoding bits representing episode history are embedded in the state space. The i-th

bit, corresponding to the i-th action, will be set to 1 if action i is selected at the current step t. The episode history is then reset to 0 at the beginning of each new episode. The reason behind this choice is twofold:

1) Memory of the past actions can be exploited.
2) A uniform replay buffer with random sampling can be employed without losing information of episodes' past actions. Although either two contiguous steps or two uncorrelated episodes' steps are sampled during mini-batch extraction, the episode history information is fully preserved at any time (Fig. 8).
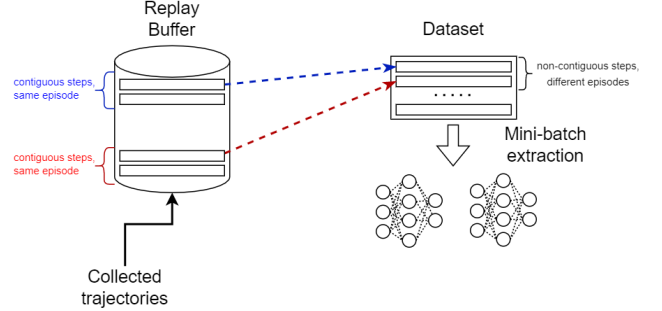

Fig. 8: Replay buffer and mini-batch extraction

C. Reward

The immediate reward function received at time step $t + 1$ after selecting action a_t must map all network performance tradeoffs in one scalar value. There is an evident twofold tradeoff of maximizing sum-rate for CCO problems without overlooking cell-edge coverage. As a consequence, two elements concur to the final reward function computation:

1) Average end-user throughput estimate.
2) A cost function for penalizing sets of actions violating coverage constraints.

The cumulative future reward is the discounted sum of all step-rewards obtained during an episode, as indicated in (1). Since each step corresponds to optimizing a particular cell among the ones in the considered cluster, and all cells share the same importance, the discount factor γ is set equal to 1. The following subsections show how the two elements are computed.

1) Average end-user throughput estimate:

$$U_{\text{avg,cell}} = \eta_{\text{MCS,cell}} \left[\frac{\text{bits/s}}{\text{Hz}} \right] \cdot N_{\text{PRB,cell}} \cdot 180[\text{KHz}] \quad (4)$$

Where $U_{avg,cell}$ is the average cell user throughput, $\eta_{MCS,cell}$ is the average cell spectral efficiency, $N_{PRB,cell}$ is the average number of scheduled PRBs per UEs per cell and 180 [KHz] is the PRB bandwidth. $\eta_{MCS,cell}$ is obtained as a weighted sum of each cell's pixels spectral efficiencies:

$$\eta_{MCS,cell} = \frac{1}{\sum_{i=1}^{N_{pixels,c}} w_i} \sum_{i=1}^{N_{pixels,c}} w_i \eta_{MCS,i} \quad (5)$$

where w_i is the pixel's weight, as indicated in the state space subsection. Pixel's spectral efficiency $\eta_{MCS,i}$ is obtained

RSRP	WEIGHT	SINR	Episode History (channel $4 \rightarrow (P \times C + 3)$)		
$Y \times Z$ pixels	$Y \times Z$ pixels	$Y \times Z$ pixels	1 hot-encoding bits		

TABLE II: State space

CQI	SINR	Modulation	Code rate	Spectral Efficiency
0	-	-	-	-
1	-6.4 dB	QPSK	0.076	0.1524
2	-4.8 dB	QPSK	0.19	0.377
3	-3.4 dB	QPSK	0.44	0.877
4	-2.2 dB	16-QAM	0.37	1.4764
5	-1.2 dB	16-QAM	0.48	1.914
6	-0.1 dB	16-QAM	0.60	2.4064
7	0.9 dB	64-QAM	0.46	2.7306
8	2.1 dB	64-QAM	0.55	3.3222
9	3.3 dB	64-QAM	0.65	3.9024
10	4.8 dB	64-QAM	0.75	4.5234
11	6.5 dB	64-QAM	0.85	5.115
12	8.5 dB	256-QAM	0.69	5.5544
13	10.9 dB	256-QAM	0.78	6.2264
14	13.8 dB	256-QAM	0.86	6.9072
15	17.1 dB	256-QAM	0.93	7.4064

TABLE III: CQI-SINR mapping. Empirical law from MDT data analysis

according to a Channel Quality Indicator (CQI)- Modulation and Coding Scheme (MCS) mapping compliant to [38], in which the maximum MCS is determined allowing a packet error rate of 10%. The CQI is not uniquely identified by SINR, as it also depends on the implementation of the UE's receiver: a sophisticated receiver can collect the incoming data at a lower SINR than a simpler one [39]. In order to derive an empirical law for the mapping between CQI and SINR, and MDT data analysis has been conducted on the reference territory. A best-fit line of polynomial degree 3 approximates the average MDT SINR values (y-axis) for different values of average CQI (x-axis) (Fig. 9, Tab. III).
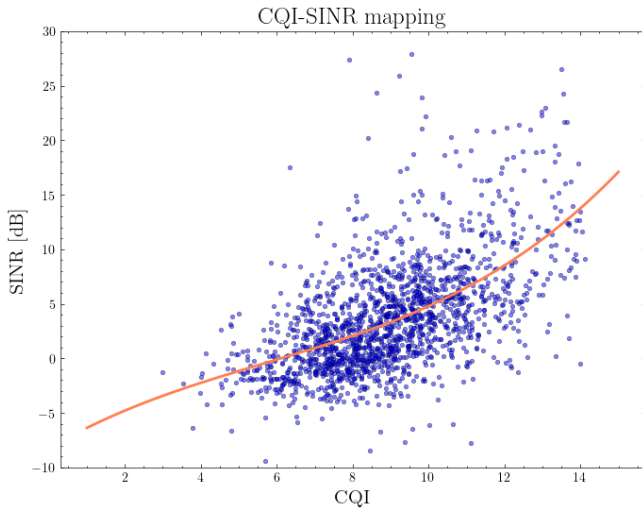

Fig. 9: CQI-SINR mapping from MDT data analysis

The average number of PRBs per UEs per cell $N_{PRB,cell}$ depends on the scheduling mechanism employed. In the following, two possible schemes are considered:

1) Round-robin scheduling

$$N_{PRB} = \frac{N_{PRB,TOT} - \text{Thr}}{N_{UE,CELL}} . \qquad (6)$$

Every UE is treated equivalently with the round-robin scheduling, regardless of its channel condition. $N_{PRB,TOT}$ is a constant indicating the number of PRBs available for each LTE frequency band (100 PRBs for the 1800 MHz band), Thr is a safety PRB occupancy threshold (e.g., 10%), and $N_{UE,CELL}$ is the average number of active users by cell.

2) Fair scheduling

$$N_{PRB} = \frac{N_{PRB,TOT} - Thr}{\sum_{i=1}^{M} \beta_i N_{UE,CELL,i}} . \qquad (7)$$

With fair scheduling, every UE is assigned a different percentage of PRBs to balance the cell-edge performance; UEs are subdivided into M classes depending on their channel conditions (the first one corresponds to the best channel condition and the last, M-th class corresponds to the worst one). The term β_i varies between β_1 and β_M. A user of the M-th class is assigned a number of PRBs which is M times greater than those assigned to a user of the first class.

The final throughput is calculated as a linear weighted average of the average user throughput per cell. The weighting term is given by the relative number of UEs per cell:

$$U = \frac{1}{N_{TOT}} \sum_{j=1}^{c} N_{UE,j} U_{avg,cell,j} . \qquad (8)$$

2) Coverage: An exponential cost function is introduced to penalize degrading sets of actions for the coverage:

$$C = \begin{cases} 0, & \text{if } 99,5\% <= \%A_{cov} <= 100\% \\ \frac{1-4,6^{\%A_{cov}-101,1}}{11,2 \times 4,6^{(\%A_{cov}-101,1)}}, & \text{if } 98\% <= \%A_{cov} < 99,5\% \\ 10, & \text{otherwhise} \end{cases} \qquad (9)$$

The % of covered area is evaluated as the weighted sum of the pixels in coverage N_{pc} over the total number of pixels N_p, where $N_{pc} <= N_p$:

$$\%A_{cov} = \frac{\sum_{i=1}^{N_{pc}} w_i}{\sum_{j=1}^{N_p} w_j} . \qquad (10)$$

As a consequence, pixels carrying more traffic are statistically more relevant. A pixel must be considered in outage if its performance are either noise limited (NL) or interference limited (IL):

1) RSRP ¡ -125 dBm $\rightarrow$ NL conditions
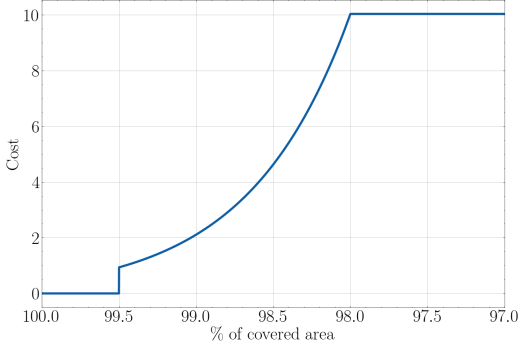2) SINR ¡ -6.4 dB (Tab. III) $\rightarrow$ IL conditions

Fig. 10: Coverage cost function

The final step-reward is the difference of the two components mentioned above:

$$R = U - C. \tag{11}$$

V. Tweaked Deep Reinforcement Learning Agent

This section is dedicated to the detailed description of the steps introduced in the definition of the DQN agent, aiming, in particular, to joint optimization of its sample efficiency and performance.

A. Tweaking of the DQN Agent

The agent has been realized by extending Tensorflow Agents API. The optimization steps introduced to the classical DQN formulation can be summarized into 4 points:

1) **Deep-tree MDP**
   As already expressed, the first improvement consists in reformulating the action space into an episode of length C, representable as a deep-tree MDP.
2) **Episode history**
   Introducing episode history directly into the state observation allows the agent to exploit the memory of its past actions up to the beginning of an episode.
3) **Reward shaping**
   Constraints of the problem have been relaxed by introducing a penalty in the reward function for sets of actions that violates them. Two constraints have been introduced to the formulation:
   a) Coverage constraint: an exponential cost function is introduced as described in the section above.
   b) Optimized cells constraint: the agent receives a negative reward if it chooses to optimize a cell that has already previously modified in the same episode.
4) **Depth-wise $\epsilon - \eta$ greedy policy**
   This custom collection policy applies soft constraints at training time to guide the agent's exploration phase. The proposed solution is effective in all those contexts where the agent must learn complex action patterns and events recurring with sparse probability. In particular, an extra degree of freedom is introduced with respect to the classical ϵ-greedy

policy: the η parameter. This parameter controls the probability of performing a "constrained random action", that is, a random action sampled from a constraint-compliant set. The pseudo-code at every episode step is presented in Algorithm 2

Algorithm 2 depth-wise $\epsilon - \eta$ greedy policy
$i \leftarrow \text{random}(0,1)$
$j \leftarrow \text{random}(0,1)$
$\epsilon \leftarrow \epsilon\_\text{scheduling}$
$\eta \leftarrow \eta\_\text{scheduling}$
if $i \geq \epsilon$ **then**:
    perform greedy action
else
    if $j \geq \eta$ **then**:
        perform constrained random action
    else
        perform random action
    end if
end if

As for the problem at hand, the desired behavior that the agent should learn is to optimize a different cell at every step of the episode. Hence, constraint (b) is enforced with probability η. The probability of randomly playing a constraint-compliant episode is given by:

$$P_{episode} = P(step_C | step_{C-1}, \ldots, step_1)$$
$$= \prod_{i=0}^{C-1} (1 - \frac{i}{C}) = \frac{(C-1)!}{C^{C-1}}. \tag{12}$$

Again, C indicates the number of target cells. Since $(C-1)! = o(C^{C-1})$, the inverse of the probability grows exponentially as the size of the problem grows. It is observable that the behavior is sparse, since $C^{C-1}/(C-1)! \approx 1067$, for C=9. This means that exploring the space randomly (i.e., with an ϵ-greedy policy) is highly inefficient, as only one episode out of 1067 on average is concluded without penalties. By controlling the probability η of choosing an action over a limited set, the number of positive experiences and penalties is balanced, and the problem of sparse positive reward is overcome. It is crucial to notice that this policy does not entirely preclude the agent from choosing an action that violates a constraint (unless $eta = 1$). Instead, the agent might find that violating a constraint provides a better long-term reward despite the immediate negative penalty. Indeed, it is a good practice not to strictly limit the agent's action space, as the programmer does not know a priori the best solution. This is particularly true if multiple constraints are applied simultaneously or the number of episodes steps is not set to a fixed number, which is left for future work. The benefit of the introduction of the η parameter, as presented in Sec. 6, is twofold:

a) Exploration phase time is drastically reduced.
b) Training is strongly stabilized.

The introduction of the η parameter is fundamental to guide the agent in acquiring complex action patterns, but is not enough for the learning process. Indeed, the advantage of a tree formulation is that of pruning the tree based on the estimated Q-values. This allows the agent to focus only on promising branches, discarding a vast portion of the state space. Since the Q-values estimate accuracy improves with the number of training episodes, the ϵ is usually scheduled to decrease monotonically during training (Fig. (11a)). Consequently, exploration leaves room for exploitation as the agent interacts and gains knowledge of the environment. Since the number of nodes grows exponentially with the depth of the tree, the first-level nodes are visited much more often than the leaf nodes. Hence, a unique scheduling of the ϵ parameter is not the most effective solution. Exploring the first-level nodes for a shorter period, instead, would be much more convenient in order to allow the tree to be pruned sequentially. To this end, a depth-wise scheduling of the ϵ parameter is proposed: at each episode step, the ϵ probability is sampled from a different scheduling function. The first level scheduling decays faster than the last layers' one (Fig. (11b)).
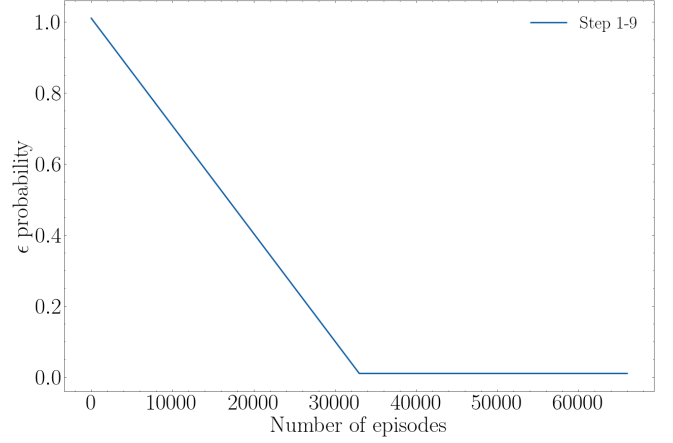This leads to sequential pruning of the tree, much more efficient in terms of convergence time and quality of the found solutions.

The parameter η can be scheduled according to the episode's step too. Since the probability that the desired behavior randomly occurs is lower at the end of the episode (more cells have already been optimized), it makes sense to have higher η values for more profound steps. In particular, a desired behavior would be to spread evenly among the tree levels the overall number of positive and negative experiences. This can be computed in terms of probabilities:
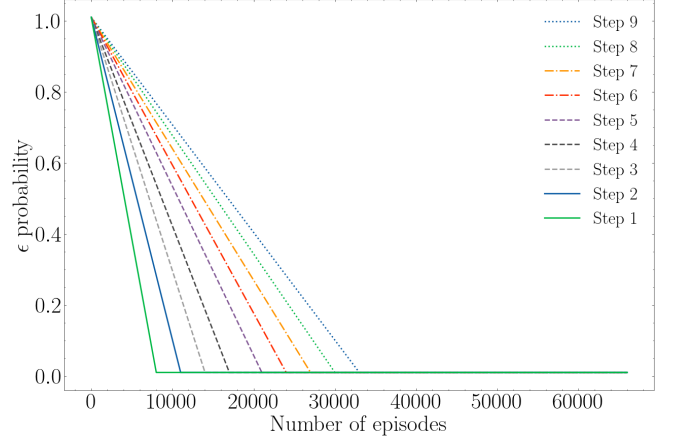
$$P_{s,step} = P_s(\bar{\eta})(1-\eta) + P_s(\eta)\eta = P_s(\bar{\eta})(1-\eta) + \eta \tag{13}$$

$$P_{s,ep} = P(s_{step,C}|s_{step,C-1}, ..., s_{step,1}) = \prod_{i=1}^{C} P_{s,i} \tag{14}$$

where $P_{s,step}$ denotes the probability of respecting constraint (b) for a single episode step, $P_{s,ep}$ the probability of respecting it for over the whole length of the episode, and $\bar{\eta}$ is the complementary probability of η. Setting a proper target balance between successful and unsuccessful episodes, e.g., $P_{s,ep} = 0.5$, assuming i.i.d episodes steps, it is easy to obtain that $P_{s,i} = P_{s,ep}^{(1/C-1)}$. For $C = 9$ and $P_{s,ep} = 0.5$, we obtain $P_{s,i} \approx 0.917$. From this, we can derive the η

(a) Monotonically decaying epsilon scheduling

(b) Depth-wise epsilon scheduling

Fig. 11: Scheduling of the ϵ parameter.

Episode step	$P_s(\bar{\eta})$	η
1	1	0
2	8/9	0.253
3	7/9	0.627
4	6/9	0.751
5	5/9	0.813
6	4/9	0.851
7	3/9	0.876
8	2/9	0.893
9	1/9	0.907

TABLE IV: depth-wise η scheduling

values for every episode step:

$$\eta = \frac{P_{s,i} - P_s(\bar{\eta})}{1 - P_s(\bar{\eta})} \tag{15}$$

Therefore, unlike ϵ, η does not decay with the number of episodes but remains constant throughout the training (Tab. IV).

B. DQN Architecture

The DQN architecture employed for the training is depicted in Fig. (12).

Hyperparameter	Value	Hyperparameter	Value
Replay buffer size	2×10^5	Conv2D layer	1, 5x5
Batch size	256	Stride	1
Epochs	66000×9	Learning rate	$2,5 \times 10^{-4}$
Optimizer	RMSprop	RMSprop's rho	0.95
Target update freq	1500	Momentum	0.0
Discount factor	1.0	Activation function	ReLU

TABLE V: Hyperparameters table

Input data are distinguished between MDT pixel at the output of the pre-processing block (Sec 3.1) and episode history. Two distinct branches of the DQN process the two data streams before being concatenated and fed to one common fully-connected layer. The former is processed by one 5x5 2D Conv layer and three fully connected layers. No pooling layers are employed since we do not want to introduce translational invariance: pixels' location matters. The latter, instead, go through a pass layer directly into the concatenate block. A detail of the chosen hyperparameters is provided in Tab. V. A fixed Q-targets variant of the algorithm is utilized, employing two DQNs as target and action networks with an update frequency of 1500 training steps (Fig. (12)).

VI. OBSERVED RESULTS

The proposed tweaked DQN algorithm has been trained for a total of 66000 episodes, or 594000 steps. The collected results show performance improvements in terms of 1) quality of the found solution (episode reward), 2) sample efficiency and 3) training stability with respect to few baseline algorithms: Best First Search (BFS) and DQN with ϵ-greedy exploration policy.
Best First Search is a graph exploration algorithm in which each branch of the tree is exhaustively explored up to the one-level deeper node. In this way, each branch at episode step n is explored with a brute force approach up until episode step $n + 1$. Hence, the best branch is selected as source node for the next iteration. As a consequence, the algorithm selects for every episode step the action corresponding to the maximum immediate reward.

We present several results related to two reward functions:

- Case 1: The coverage-constrained user throughput reward function (Eqn. (11)) expressed in Sec.4. In this case, the reward is also indicative of the gain in [Mbps] with respect to the average throughput corresponding to the default antenna configuration.
- Case 2: Weighted average sum of MDT pixels' RSRP and SINR:

$$R = \frac{1}{\sum_i w_i} \sum_{i=1}^{N_{pixels}} w_i(RSRP_i + SINR_i). \quad (16)$$

Reward shaping with constraint (b) is applied to both reward functions for the DW-$\epsilon\eta$-greedy policy.

Fig. (13) shows the training curve of the proposed DW-$\epsilon\eta$-greedy DQN compared to a classical DQN with ϵ-greedy exploration policy. The curves have been obtained by testing the two algorithms' greedy policies over 10 random seeds every 1000 episode steps during training. Additionally, their relative moving averages have been computed with a window length equal to 15 samples. Both algorithms have been trained with the same hyperparameters of Tab. V. Both Fig. (13a) and (13b) show that the proposed DW-$\epsilon\eta$ -greedy DQN registers remarkable performance improvement with respect to the ϵ-greedy DQN. In particular, the former manifest greater stability, slightly increased performance and, most importanltly, better sample efficiency, as it reaches a stable plateau in an up to 70% shorter time (Fig. (13a)).

Fig. (14) depicts the performance gain obtained by our proposed algorithm with respect to BFS. The two Fig. (14a) and (14b) illustrate the reward obtained at every episode step by the three algorithms. Results are depicted togheter with their relative 99% confidence intervals. It is particularly evident how both ϵ-greedy DQN and DW-$\epsilon\eta$ -greedy DQN agents are able to learn to efficiently sacrifice the immediate reward in favour of a better cumulative episode reward (the reward value obtained at step 9, i.e., at the end of the episode). Another interesting behavior worth noticing is that the DW-$\epsilon\eta$ -greedy DQN's policy shows more compactness than the ϵ-greedy DQN's one: since the confidence intervals of the former are narrower, this means that more environment states map to the same antenna configurations. This translates to a less frequent need of reconfiguration for the network parameters.

In RL, the variance between runs is typically enough to create statistically different distributions just from varying random seeds [40]. As such, sensibility to random seeds is a crucial aspect of RL that needs to be addressed when assessing an algorithm's performance. Boxplots of Fig. (15) show episode reward distribution for the three algorithms when tested over 50 different random seeds. DW-$\epsilon\eta$ -greedy DQN not only outperforms the baselines algorithms in terms of average episode reward, but shows a variance value comparable to that of BFS.
Finally, Fig. (16) evaluates the algorithm's stability for increasing level of randomness in the environment's traffic distribution. In order to simulate such randomness, multiple runs over 50 random seeds have been executed. In each run, the average pixels' WEIGHT is varied every episode according to a uniform distribution with increasingly larger boundaries:

$$w_i^{'} = w_{i,default} + U[-d_{range}/2, +d_{range}/2] \quad (17)$$

Where $U[a, b]$ indicates the uniform distribution between a and b. After all $w_i^{'}$ values are re-computed, the weights are then normalized in the range [0,1].
As expected, results show larger variance for increasing values of d_{range}. Nevertheless, the agent shows substantial
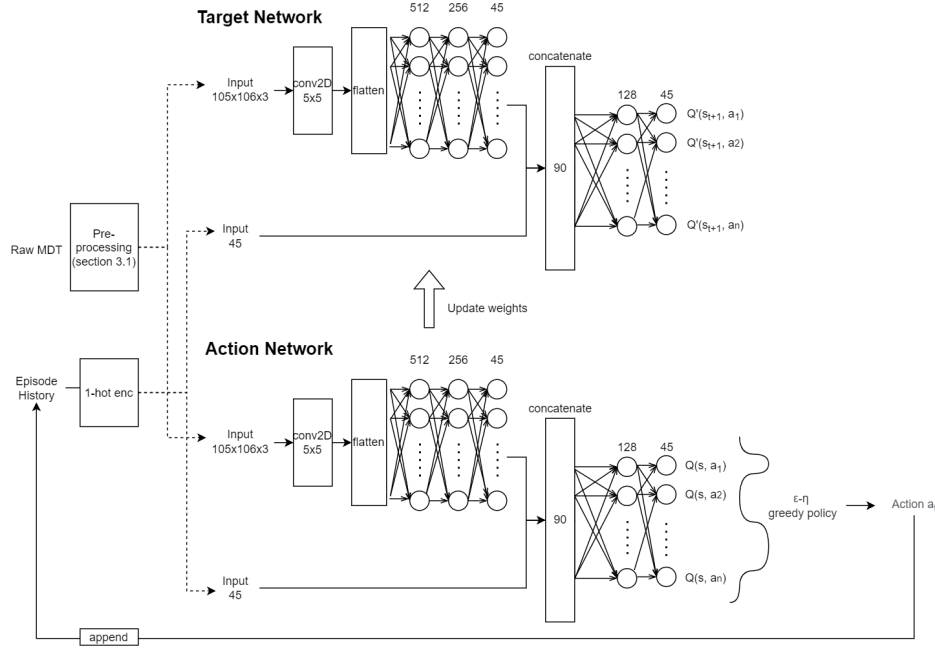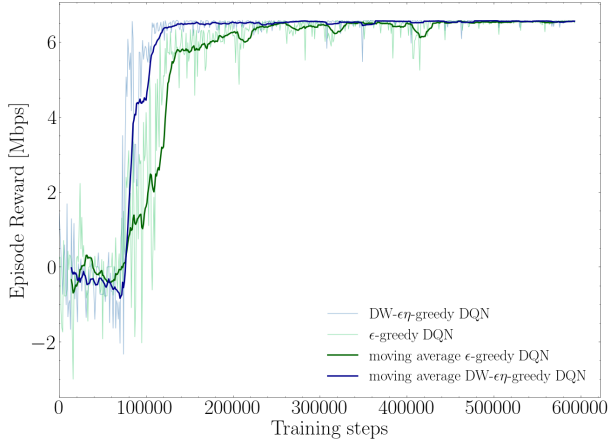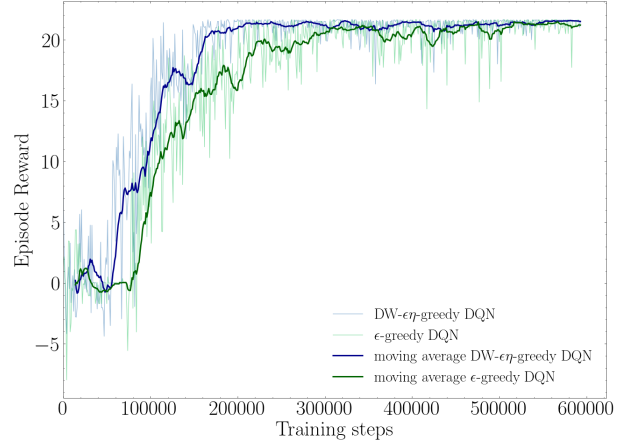
Fig. 12: Deep Q-Network architecture

(a) Reward: Coverage constrained throughput

(b) Reward: Average weighted RSRP + SINR

Fig. 13: Training curves: DW-$\epsilon\eta$ -greedy DQN vs ϵ -greedy DQN

stability, never experiencing rewards below +5.5 [Mbps] even in the worst case scenario.
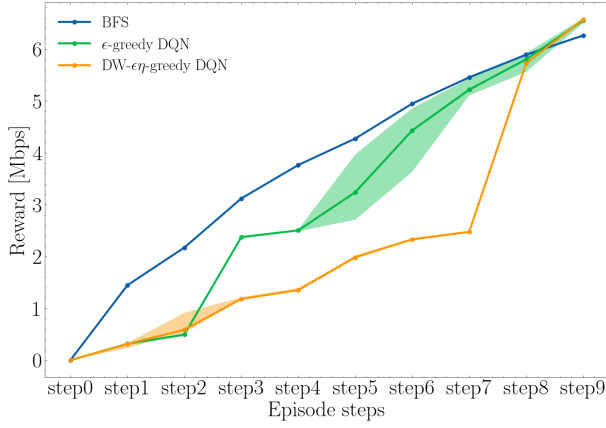
VII. CONCLUSION

In this work, we address a CCO problem in a real network scenario using an tweaked MDT-driven DRL agent to select optimal antenna downtilts configuration for a cluster of cells of TIM's network. In particular, we leveraged the joint use of MDT data, network KPIs and electromagnetic simulations to realize an accurate simulated network environment for the training of a tweaked DRL agent. The simulated network environment is able to accurately represent the effect of the change of a configuration parameter (e.g., antenna downtilts) on network performance. Furthermore, an optimized version of the DQN formulation with a custom exploration policy has been introduced. The obtained results show remarkable improvements in terms of sample efficiency, stability, and quality of found solution with respect to baseline algorithms like BFS and DQN with ϵ-greedy exploration policy.
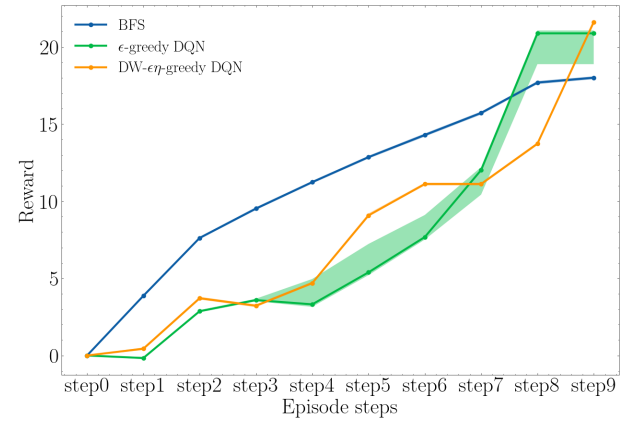
Future work might focus on the joint optimization of antenna tilts and other network parameters, as well as on a multi-agent approach to target the optimization problem of larger or denser geographical areas.

VIII. PATENTS

A patent application resulting from the work reported in this manuscript has been filed with the title "Optimization of the configuration of a mobile communications network". The same authors of the manuscript are involved in the patent application.
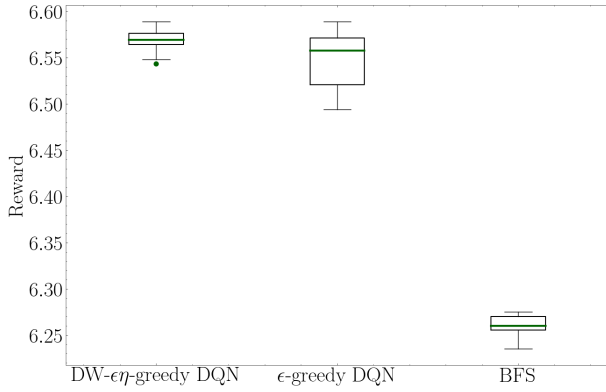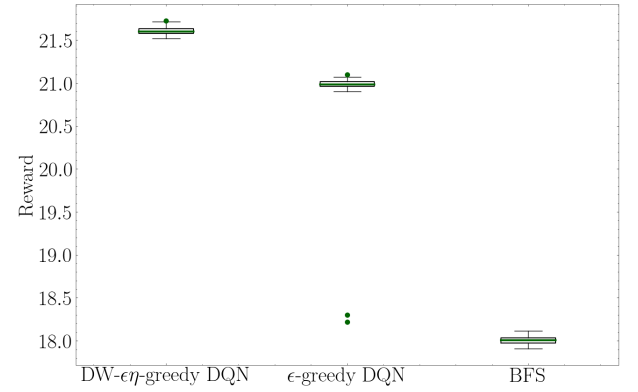
(a) Reward: Coverage constrained throughput

(b) Reward: average weighted RSRP + SINR

Fig. 14: Step reward: DW-$\epsilon\eta$ -greedy DQN vs ϵ -greedy DQN vs BFS.

(a) Reward: Coverage constrained throughput

(b) Reward: average weighted RSRP + SINR
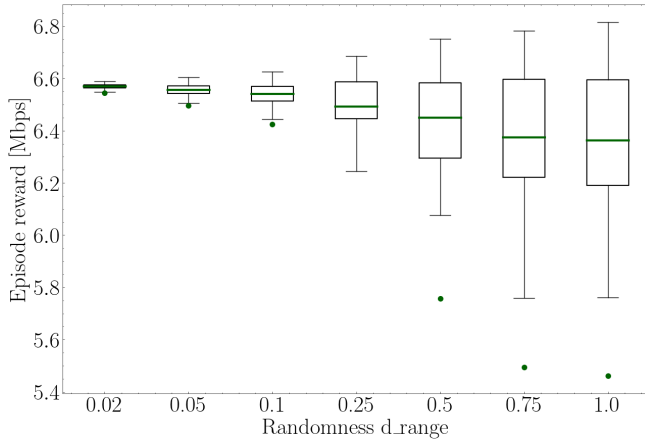
Fig. 15: Average episode reward boxplot distribution.

Fig. 16: DW-$\epsilon\eta$ -greedy DQN performance with increasing randomness in traffic distributions

REFERENCES

[1] Viktor Berggren, Rafia Inam, Leonid Mokrushin, Alberto Hata, Jaeseong Jeong, Swarup Kumar Mohalik, Julien Forgeat, and Stefano Sorrentino. Artificial intelligence in next generation connected systems. Technical report, 9 2021.

[2] Osianoh Glenn Aliu, Ali Imran, Muhammad Ali Imran, and Barry Evans. A survey of self organisation in future cellular networks. *IEEE Communications Surveys Tutorials*, 15(1):336–361, 2013. doi: 10.1109/SURV.2012.021312.00116.

[3] *TS 36.902 - Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Self-configuring and self-optimizing network (SON) use cases and solutions.* 3GPP, 4 2011. Rel 9.3.1.

[4] *TS 32.500 - Telecommunication management; Self-Organizing Networks (SON); Concepts and requirements.* 3GPP, 7 2020. Rel 16.0.0.

[5] Osvaldo Simeone. A very brief introduction to machine learning with applications to communication systems. *IEEE Transactions on Cognitive Communications and Networking*, 4(4):648–664, 2018. doi: 10.1109/TCCN.2018.2881442.

[6] Mirza Golam Kibria, Kien Nguyen, Gabriel Porto Villardi, Ou Zhao, Kentaro Ishizu, and Fumihide Kojima. Big data analytics, machine learning, and artificial intelligence in next-generation wireless networks. *IEEE*

Access, 6:32328–32338, 2018. doi: 10.1109/ACCESS. 2018.2837692.

[7] *TS 37.320 - Universal Terrestrial Radio Access (UTRA) and Evolved Universal Terrestrial Radio Access (E-UTRA); Radio measurement collection for Minimization of Drive Tests (MDT); Overall description; Stage 2*. 3GPP, 9 2021. Rel 16.6.0.

[8] Johan Johansson, Wuri A. Hapsari, Sean Kelley, and Gyula Bodog. Minimization of drive tests in 3gpp release 11. *IEEE Communications Magazine*, 50(11): 36–43, 2012. doi: 10.1109/MCOM.2012.6353680.

[9] Chiara Mizzi, Alessandro Fabbri, Sandro Rambaldi, Flavio Bertini, Nico Curti, Stefano Sinigardi, Rachele Luzi, Giulia Venturi, D. Micheli, Giuliano Muratore, Aldo Vannelli, and Armando Bazzani. Unraveling pedestrian mobility on a road network using icts data during great tourist events. *EPJ Data Science*, 7, 12 2018. doi: 10.1140/epjds/s13688-018-0168-2.

[10] Andrea Scaloni, Pasquale Cirella, Mauro Sgheiz, Riccardo Diamanti, and Davide Micheli. Multipath and doppler characterization of an electromagnetic environment by massive mdt measurements from 3g and 4g mobile terminals. *IEEE Access*, 7:13024–13034, 2019. doi: 10.1109/ACCESS.2019.2892864.

[11] Davide Micheli and Giuliano Muratore. Smartphones reference signal received power mdt radio measurement statistical analysis reveals people feelings during music events. In *2019 PhotonIcs Electromagnetics Research Symposium - Spring (PIERS-Spring)*, pages 427–437, 2019. doi: 10.1109/PIERS-Spring46901. 2019.9017606.

[12] Sascha Berger, Albrecht Fehske, Paolo Zanier, Ingo Viering, and Gerhard Fettweis. Online antenna tilt-based capacity and coverage optimization. *IEEE Wireless Communications Letters*, 3(4):437–440, 2014. doi: 10.1109/LWC.2014.2327228.

[13] Alexander Engels, Michael Reyer, Xiang Xu, Rudolf Mathar, Jietao Zhang, and Hongcheng Zhuang. Autonomous self-optimization of coverage and capacity in lte cellular networks. *IEEE Transactions on Vehicular Technology*, 62(5):1989–2004, 2013. doi: 10.1109/TVT.2013.2256441.

[14] Eren Balevi and Jeffrey Andrews. Online antenna tuning in heterogeneous cellular networks with deep reinforcement learning. *IEEE Transactions on Cognitive Communications and Networking*, PP:1–1, 08 2019. doi: 10.1109/TCCN.2019.2933420.

[15] Ryan M. Dreifuerst, Samuel Daulton, Yuchen Qian, Paul Varkey, Maximilian Balandat, Sanjay Kasturia, Anoop Tomar, Ali Yazdan, Vish Ponnampalam, and Robert W. Heath. Optimizing coverage and capacity in cellular networks using machine learning. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8138–8142, 2021. doi: 10.1109/ICASSP39728.2021. 9414155.

[16] Filippo Vannella, Grigorios Iakovidis, Ezeddin Al

Håkim, Erik Aumayr, and Saman Feghhi. Remote electrical tilt optimization via safe reinforcement learning. *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–7, 2021.

[17] Yuanjie Lin, Hui Gao, Wenjun Xu, and Yueming Lu. Dynamic antenna configuration for 3d massive mimo system via deep reinforcement learning. In *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–6, 2020. doi: 10.1109/PIMRC48278.2020.9217272.

[18] Usama Masood, Ahmad Asghar, Ali Imran, and Adnan Noor Mian. Deep learning based detection of sleeping cells in next generation cellular networks. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 206–212, 2018. doi: 10.1109/ GLOCOM.2018.8647689.

[19] Jani Puttonen, Jussi Turkka, Olli Alanen, and Janne Kurjenniemi. Coverage optimization for minimization of drive tests in lte with extended rlf reporting. In *21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1764–1768, 2010. doi: 10.1109/PIMRC.2010. 5671907.

[20] Ahmed Zoha, Arsalan Saeed, Ali Shariq Imran, Muhammad Ali Imran, and Adnan A. Abu-Dayya. A learning-based approach for autonomous outage detection and coverage optimization. *Trans. Emerg. Telecommun. Technol.*, 27:439–450, 2016.

[21] Fedor Chernogorov and Jani Puttonen. User satisfaction classification for minimization of drive tests qos verification. In *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 2165–2169, 2013. doi: 10.1109/PIMRC.2013.6666502.

[22] Oluwakayode Onireti, Ahmed Zoha, Jessica Moysen, Ali Imran, Lorenza Giupponi, Muhammad Ali Imran, and Adnan Abu-Dayya. A cell outage management framework for dense heterogeneous networks. *IEEE Transactions on Vehicular Technology*, 65(4):2097–2113, 2016. doi: 10.1109/TVT.2015.2431371.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/ nature14236.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *ArXiv*, abs/1312.5602, 2013.

[25] Navid Naderializadeh, Jaroslaw Sydir, Meryem Simsek, and Hosein Nikopour. Resource management in wireless networks via multi-agent deep reinforcement

learning. *IEEE Transactions on Wireless Communications*, PP:1–1, 01 2021. doi: 10.1109/TWC.2021.3051163.

[26] Yasar Sinan Nasir and Dongning Guo. Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks. *IEEE Journal on Selected Areas in Communications*, 37(10):2239–2250, 2019. doi: 10.1109/JSAC.2019.2933973.

[27] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. 02 2018.

[28] Pedro Iturria Rivera and Melike Erol Kantarci. Competitive multi-agent load balancing with adaptive policies in wireless networks, 10 2021.

[29] Fahime Khoramnejad, Roghayeh Joda, and Melike Erol-Kantarci. Distributed multi-agent learning for service function chain partial offloading at the edge. In *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6, 2021. doi: 10.1109/ICCWorkshops50388.2021.9473554.

[30] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *NIPS*, 2016.

[31] Tze-Yang Tung, Szymon Kobus, Joan Pujol Roig, and Deniz Gündüz. Effective communications: A joint learning and communication framework for multi-agent reinforcement learning over noisy channels. *IEEE Journal on Selected Areas in Communications*, 39(8):2590–2603, 2021. doi: 10.1109/JSAC.2021.3087248.

[32] Hao Zhou, Atakan Aral, Ivona Brandic, and Melike Erol-Kantarci. Multi-agent bayesian deep reinforcement learning for microgrid energy management under communication failures. *IEEE Internet of Things Journal*, pages 1–1, 2021. doi: 10.1109/JIOT.2021.3131719.

[33] Thanh Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Transactions on Cybernetics*, PP:1–14, 03 2020. doi: 10.1109/TCYB.2020.2977374.

[34] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PLOS ONE*, 12, 11 2015. doi: 10.1371/journal.pone.0172395.

[35] Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16(1):1437–1480, jan 2015. ISSN 1532-4435.

[36] Haneya Naeem Qureshi, Ali Imran, and Adnan Abu-Dayya. Enhanced mdt-based performance estimation for ai driven optimization in future cellular networks. *IEEE Access*, 8:161406–161426, 2020. doi: 10.1109/ACCESS.2020.3021030.

[37] *TS 36.214 - Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer; Measurements*. 3GPP, 3 2021. Rel 16.2.0.

[38] *TS 36.213 - Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures*. 3GPP, 8 2021. Rel 16.4.0.

[39] Christopher Cox. *An Introduction to LTE: LTE, LTE-Advanced, SAE, VoLTE and 4G Mobile Communications*. Wiley Publishing, 2nd edition, 2014. ISBN 1118818032.

[40] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. 09 2017.