# Sensitivity of Dynamic Network Slicing to Deep Reinforcement Learning Based Jamming Attacks

Feng Wang, M. Cenk Gursoy, and Senem Velipasalar
Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, 13244
E-mail: fwang26@syr.edu, mcgursoy@syr.edu, svelipas@syr.edu

***Abstract*—In this paper, we consider multi-agent deep reinforcement learning (deep RL) based network slicing agents in a dynamic environment with multiple base stations and multiple users. We develop a deep RL based jammer with limited prior information and limited power budget. The goal of the jammer is to minimize the transmission rates achieved with network slicing and thus degrade the network slicing agents' performance. We design a jammer with both listening and jamming phases and address jamming location optimization as well as jamming channel optimization via deep RL. We evaluate the jammer at the optimized location, generating interference attacks in the optimized set of channels by switching between the jamming phase and listening phase. We show that the proposed jammer can significantly reduce the victims' performance without direct feedback or prior knowledge on the network slicing policies.**

***Index Terms*—Network slicing, dynamic channel access, deep reinforcement learning, multi-agent actor-critic, adversarial learning, jamming attacks.**

## I. Introduction

Network slicing is one of the key enablers of 5G networks that can support novel applications with heterogeneous requirements [1]–[4]. In network slicing, service flexibility is attained by dividing the the physical resources into multiple virtual network slices and allocating the desired subset of network slices to the users/applications with the goal to satisfy their quality-of-service requirements [5]. Recently, deep reinforcement learning (deep RL) based approaches have been considered to optimize slice selection in challenging dynamic settings with resource interplay and user mobility [6]–[9]. In the literature, it is typically assumed that the slice state is identical for all different slices over time, and network slicing is reduced to just determining the number of slices to be assigned to each request. In [10], we considered a more practical scenario with multiple base stations and multiple users with random mobility patters in a dynamic interference environment, and studied dynamic network slicing. Specifically, we developed a multi-agent deep RL based learning and decision-making framework with multiple actors and a centralized critic (MACC) [11]–[13] that aims at maximizing the overall performance. We introduced the pointer network [14] to implement the actor policy to handle the varying observations of the deep RL agents. We provided comparisons with independent and decentralized actor-critic multi-agent models as well as feed-forward neural network (FNN) based actor structures along with statistical algorithms, and demonstrated performance improvements with the proposed MACC framework.

The work was supported by the National Science Foundation under Grant CNS-2221875.

Despite these advances in learning based network slicing methods, it is important to note that due to being highly data driven, deep neural networks are vulnerable to minor but carefully crafted perturbations, and it is known that adversarial samples with such perturbations can cause significant loss in accuracy, e.g. in inference and classification problems in computer vision [15]–[17]. Given the broadcast nature of wireless communications, deep learning based adversarial attacks have also recently been attracting increasing attention in the context of wireless security [18], [19]. Motivated by these considerations, we in this paper consider MACC-based multi-agent network slicing and design a deep RL based jammer agent with jamming and listening phases. Different from most existing works, we analyze how the jamming location and channel selection are optimized without direct feedback on the victims' performance. We further analyze the performance degradation in network slicing agents in the presence of jamming attacks and identify their sensitivity in an adversarial environment.

The remainder of the paper is organized as follows. In Section II, we provide the preliminaries on the multi-agent deep RL based network slicing agent. Subsequently, in Section III, we devise the deep RL based jammer agent, and introduce the two operation phases, jamming location optimization, and jamming channel optimization. In this section, we further provide the details of the actor-critic implementation. In Section IV, we conduct numerical analysis and evaluate the performance to show the effectiveness of proposed listening measurement based on the impact factor. Finally, we conclude the paper in Section V.

## II. Preliminaries – Multi-Agent Reinforcement Learning with Pointer Networks for Network Slicing

For the sake of completeness, we in this section provide the preliminaries on deep RL based network slicing. In [10], we have presented a multi-agent deep RL framework for network slicing in an environment with multiple base stations, multiple mobile users, and randomly generated requests. In particular, we have developed an MACC (multiple actor centralized critic) based deep RL approach to optimize the performance across all base stations instead of pursuing local optimization. The actors are implemented as pointer networks to fit the varying dimension of input.

In the considered setting, the features of request $k$ includes the minimum transmission rate $m_k$, lifetime $l_k$, and initial payload $p_k$. In time slot $t$, $\mathsf{r}_k(t)$ bits of the remaining payload of request $k$ is transmitted with the allocated resources. If the minimum rate constraint $\mathsf{r}_k(t) \geq m_k(t)$ or the remaining lifetime constraint $l_k(t) > 0$ of request $k$ is not met, then this request will be terminated and be marked as failed. If the payload is completed within the lifetime, the request $k$ is considered as completed and marked as success. In case of success, the network slicing agent at the base station receives a positive reward $R_k$ that is equal to the initial payload $p_k$. If a failure occurs, the agent has a negative reward of $R_k = -p_k$. Subsequently, base station $b$ records the latest transmission rate history into a 2-dimensional matrix $H^b \in \mathbb{R}_{\geq 0}^{N_u \times N}$, and updates this matrix over time.

The objective of the network slicing agent at each base station is to find a policy $\pi$ that selects $n_c$ channels and assigns them to $n_r$ requests such that the sum reward of all requests at all base stations is maximized over time:

$$\underset{\pi}{\operatorname{argmax}} \sum_{t'=t}^{\infty} \left( \gamma^{(t'-t)} \sum_{b=1}^{N_B} \sum_{k \in K'_b} R_k \right), \quad (1)$$

where $K'_b$ is the set of completed or terminated requests for base station $b$ at time $t$, and $\gamma \in (0,1)$ is the discount factor.

## *A. Multi-Agent Deep RL with Multiple Actors and Centralized Critic (MACC)*

To address network slicing and solve (1), we, as also noted above, proposed an MACC-based multi-agent deep RL algorithm. The goal is to attain the maximal reward across all base stations by choosing the optimal subset of channels to allocate to each request. We denote the full observation at base station $b$ as $\mathcal{O}_b$, the observation over all base stations as $\mathcal{O} = \cup_{b=1}^{N_B} \mathcal{O}_b$, and the channel selection at base station $b$ as a matrix of actions $\mathcal{A}_b \in \{0,1\}^{n_r \times N}$, where each element in $\mathcal{A}_b$ is an indicator whether channel $c$ is assigned to request $k$.

In MACC, we have decentralized actors parameterized with $\theta$ and policy $f^\theta(\mathcal{O}_b)$, and only one centralized critic with parameter $\phi$ and policy $g^\phi(\mathcal{O})$. The critic aims to maximize the sum reward by updating each decentralized actor. Via the feedback of the single critic, agents can effectively interact among each other and also with the environment and learn coordinated strategies despite the scarcity of information sharing in the training phase.

## *B. Pointer Network Architecture for the Actor*

In the proposed MACC agent, the actor policy $f^\theta$ is implemented as a pointer network. Similar to sequence-to-sequence learning, pointer network utilizes two recurrent neural networks (RNNs) called encoder and decoder whose output dimensions are $h_e$ and $h_d$. The former encodes the sequence $\{\mathcal{O}_b^{e1}, \mathcal{O}_b^{e2}, \ldots, \mathcal{O}_b^{en_r}\}$ with $\mathcal{O}_b^{ek} \subset \mathcal{O}_b$ (where the index $k \in \{1,2,\ldots,n_r\}$ corresponds to requests), and produces the sequence of vectors $\{e_1, e_2, \ldots, e_{n_r}\}$ with $e_k \in \mathbb{R}^{h_e}$ at the output gate in each step. The decoder inherits the encoder's hidden state and is fed by another sequence $\{\mathcal{O}_b^{d1}, \mathcal{O}_b^{d2}, \ldots, \mathcal{O}_b^{dN}\}, \mathcal{O}_b^{dc} \subset \mathcal{O}_b$ (where the index $c \in \{1,2,\ldots,N\}$ corresponds to channels), and produces the sequence of vectors $\{d_1, d_2, \ldots, d_N\}, d_c \in \mathbb{R}^{h_d}$.

Furthermore, pointer network computes the conditional probability array $\mathcal{P}^c \in [0,1]^{n_r}$ where

$$\mathcal{P}^c[k] = P(\mathcal{A}_b[k,c] = 1 | \mathcal{O}_b^{d1}, \ldots, \mathcal{O}_b^{dc}, \mathcal{O}_b^{e1}, \ldots, \mathcal{O}_b^{en_r}) \quad (2)$$

using attention mechanism as follows:

$$\begin{aligned} u_k^c &= \tanh(W_1 e_k + W_2 d_c), \\ \mathcal{P}^c &= \operatorname{softmax}([u_1^c, u_2^c, \ldots, u_{n_r}^c]^T) \end{aligned} \quad (3)$$

where softmax normalizes the vector of $u_k^c$ to be an output distribution over the dictionary of inputs, and vectors $W_1$ and $W_2$ are trainable parameters of the attention model. Thus, we obtain the decision of $\mathcal{A}_b$ by picking $n_c$ maximum elements in $\mathcal{P}^c$.

## III. Deep RL Based Jammer

In this section, we introduce an actor-critic deep RL agent that performs the jamming attack on the aforementioned victim network slicing users (introduced in Section II) and aims at minimizing the victims' transmission rate. We assume the jammer has the geometric map of all BSs, but it does not have any information on the channel states, users' locations, requests, victim reward or the victim policy. This deep RL jammer may jam multiple channels to reduce the transmission rate and potentially may lead to request failures, or observe the environment and record the interference power in each channel to speculate the victims' actions. We demonstrate a jammer that can significantly degrade the aforementioned victim users' performance even though it lacks critical information on the victims.

## *A. System Model*

In the considered channel model, the fading coefficient of the link from the jammer to the user equipment (UE) $u$ in a certain channel $c$ is denoted by $h_c^{J,u}$, and the fading coefficient of the link from the base station $b$ to the jammer in a certain channel $c$ is denoted by $h_c^{b,J}$. We consider $h_c^{b,u}$, $h_c^{J,u}$ and $h_c^{b,J}$ to be independent and identically distributed (i.i.d.) and vary over time according to the Jakes fading model [20]. Once the jammer is initialized at horizontal location $\{x_J, y_J\}$ with height $h_J$, it can choose in any given time slot one of the two operational phases: jamming phase and listening phase.

*1) Jamming phase:* In this phase, the jammer jams $n_J \leq N_J$ channels simultaneously with jamming power $P_J$ in each channel without receiving any feedback. With the additional interference from the jammer, we can express the transmission rate $\mathsf{r}_c^{b,u}$ from base station $b$ to UE $u$ as

$$\mathsf{r}_{c,J}^{b,u} = \log_2 \left( 1 + \frac{P_B L^{b,u} |h_c^{b,u}|^2}{\sum_{b' \neq b} \mathcal{N}_c^{b,b'} + \mathcal{N}_c^{b,J} + \sigma^2} \right), \quad (4)$$

$$\mathcal{N}_c^{b,J} = \mathbf{1}_c^{b,J} P_J L^{J,u} |h_c^{J,u}|^2, \quad (5)$$

where $P_J$ is the jamming power, $\mathcal{N}_c^{b,J}$ is the jamming interference in channel $c$, $\mathbf{1}_c^{b,J}$ is the indicator function for both base station $b$ and jammer choosing channel $c$ , and $L^{J,u}$ is the path loss:

$$L^{J,u} = \left(h_J^2 + (x_J - x_u)^2 + (y_J - y_u)^2\right)^{\alpha/2}. \quad (6)$$

In (4), $P_B$ is the transmission power of the base stations, $h_c^{b,u}$ is the fading coefficient of the link between base station $b$ and UE $u$ in channel $c$, $\mathcal{N}_c^{b,b'}$ is the interference from base station $b'$ in channel $c$ (if base station $b'$ is also transmitting in channel $c$ based on the network slicing decisions), $\sigma^2$ is the variance of the additive white Gaussian noise, and $L^{b,u}$ is the path loss in the link between base station $b$ and UE $u$

$$L^{b,u} = \left(h_B^2 + (x_b - x_u)^2 + (y_b - y_u)^2\right)^{\alpha/2} \quad (7)$$

where $h_B$ is the height of each base station, $\alpha$ is the path loss exponent, and $\{x_b, y_b\}$ and $\{x_u, y_u\}$ are the 2-D coordinates of base station $b$ and UE $u$, respectively.

By degrading the transmission rate $\mathsf{r}_{c,J}^{b,u}$ with the jamming interference in channel $c$, the jamming attack may lead to a number of request failures. The jammer may further amplify the impact by intelligently choosing a preferable subset of channels to jam. The listening phase is introduced to learn such information from the environment.

*2) Listening phase:* In this phase, the jammer does not jam any channel, but only listens the (interference) power in each channel $c$ among $N$ channels:

$$\mathcal{N}_c^{\text{listen}} = \sum_b \mathbf{1}_c^b P_B L^{b,J} |h_c^{b,J}|^2 + \sigma^2, \quad (8)$$

where $\mathbf{1}_c^b$ is the indicator function which has a value of $1$ if there is a transmission at base station $b$ to any UE in channel $c$, and $L^{b,J}$ is the path loss:

$$L^{b,J} = \left((h_B - h_J)^2 + (x_b - x_J)^2 + (y_b - y_J)^2\right)^{\alpha/2}. \quad (9)$$

Due to the jammer's lack of prior information, we consider an approximation and assume that the listened power $\mathcal{N}_c^{\text{listen}}$ from all base stations transmitting in channel $c$ is a rough estimate of the sum of jamming interferences $\mathcal{N}_{c,\text{est}}^{b,J}$ if jammer were in the jamming phase and chose channel $c$ to inject interference, i.e.,

$$\mathcal{N}_c^{\text{listen}} \approx \sum_b \mathcal{N}_{c,\text{est}}^{b,J} + \sigma^2. \quad (10)$$

Therefore, with this assumption, the jammer anticipates that the higher $\mathcal{N}_c^{\text{listen}}$ being observed/listened, the more likely that jamming in channel $c$ degrades the victim users' performance. Given this, we introduce how we optimize the subset of channel to attack during jamming phase in Section III-C.

Another benefit of the listening phase is that no jamming power is consumed in this phase, and consequently average power consumption is reduced. In the remainder of this paper, we assume the jammer only switches from listening phase to jamming phase by the end of each period with $T_J \in \mathbb{R}^+$ time slots, and thus it has an average power consumption of $n_J P_J / T_J$.

## B. Jamming Location Optimization

The jammer aims at minimizing the performance of victim users, but it does not have any information on channel fading, UE locations or rewards provided to different requests. Therefore, the jamming location is optimized by minimizing the expected sum transmission rate for given UE $u$ integrated over the service area when the channels for transmission coincide with the channels being jammed. More specifically, we have the following optimization:

$$\{x_J^*, y_J^*\} = \underset{\{x_J, y_J\}}{\operatorname{argmin}} \, \mathbb{E}_h \left( \sum_b \sum_c \iint_{D_h^b} \mathsf{r}_{c,J}^{b,u} dx_u dy_u \right), \quad (11)$$

where the expectation with respect to the set of fading coefficients $\{h\}$ considers $\forall b, u, c : h_c^{b,u}, h_c^{J,u}, h_c^{b,J} \overset{\text{i.i.d.}}{\sim} \mathcal{CN}(0,1)$, $D_h^b$ is the subset of coverage area with maximal transmission rate $\mathsf{r}_{c,J}^{b,u}$ from base station $b$ given $\{h_c^{b,u}, h_c^{J,u}, h_c^{b,J}\}$, and $\forall b, b' : \mathbf{1}_c^{b,b'} = 0, \mathbf{1}_c^{b,J} = \mathbf{1}_c^b = 1$. Additionally, we note that $P_B$, $h_B$, $|\alpha|$, and $\sigma$ with arbitrary positive value will not affect the optimized jamming location.

## C. Jamming Channel Optimization

After the jammer is initialized in the true environment and have observed/listened the interference power $\mathcal{N}_c^{\text{listen}}(t-1)$ within the listening phase at time $t-1$, it will decide the subset of channels $\mathcal{C}_J(t) \subset \mathcal{C}$ to jam during jamming phase at time $t$, where $|\mathcal{C}_J(t)| = n_J(t)$ and $\mathcal{C}$ is the set of all channels $\{1, 2, \ldots, N\}$. According to (10), channels with higher $\mathcal{N}_c^{\text{listen}}$ are more likely to be better choices, but this information is not available in the jamming phase at time $t$ (since jamming is performed rather than listening). Therefore, $\mathcal{C}_J(t)$ can only be evaluated via $\mathcal{N}_c^{\text{listen}}(t-1)$ and $\mathcal{N}_c^{\text{listen}}(t+1)$. Note that $\mathcal{N}_c^{\text{listen}}(t+1)$ is not available at time $t$ and this challenge will be addressed via deep RL in the next subsection (i.e., by essentially introducing a reward to train the neural network at time $t+1$ and having that reward depend on $\mathcal{N}_c^{\text{listen}}(t-1)$ and $\mathcal{N}_c^{\text{listen}}(t+1)$. The action will depend only on observation before time $t$). In the absence of information on the requests, we assume a model in which each request arrives and is completed independently. Thus, the state of current time slot $t$ can be estimated as a linear interpolation (or a weighted average) of $\mathcal{N}_c^{\text{listen}}(t-1)$ and $\mathcal{N}_c^{\text{listen}}(t+1)$. Therefore, the optimized subset of channels to jam can be determined from

$$\mathcal{C}_J^*(t) = \underset{\mathcal{C}_J}{\operatorname{argmax}} \sum_{c \in \mathcal{C}_J} \hat{\mathcal{N}}_c^\beta(t), \tag{12}$$

where

$$\hat{\mathcal{N}}_c^\beta(t) = \frac{1}{\beta(t)+1}\left(\beta(t)\mathcal{N}_c^{\text{listen}}(t-1) + \mathcal{N}_c^{\text{listen}}(t+1)\right), \tag{13}$$

and $\beta(t)$ describes the impact from the jammer onto the victims. Typically, a request takes multiple time slots to get completed. When it is jammed, there are two possibilities. On the one hand, it may fail to meet the minimum transmission rate limit and get terminated immediately. In such a case, the next request in queue is processed, and the network slicing agent rearranges and distributes channels into a new set of slices to be allocated to different requests from different users, and thus the listened interference $\mathcal{N}_c^{\text{listen}}(t+1)$ may change dramatically. In this case, we are likely to have $\beta(t) > 1$. On the other hand, if the request under attack has lower transmission rate but it still satisfies the minimum transmission rate limit and the lifetime limit, the transmission will last longer, and the interference $\mathcal{N}_c^{\text{listen}}(t+1)$ is less likely to deviate from that at time $t$. In this case, we are likely to have $\beta(t) < 1$. Therefore, the value of $\beta(t)$ should be determined via experience:

$$\beta(t) = \max\left(\frac{2|\mathcal{T}'|\sum_c \sum_{t'' \in \mathcal{T}''} d_c^{\text{listen}}(t'')}{|\mathcal{T}''|\sum_c \sum_{t' \in \mathcal{T}'} d_c^{\text{listen}}(t')} - 1, 0\right), \tag{14}$$

where

$$d_c^{\text{listen}}(t) = \left|\mathcal{N}_c^{\text{listen}}(t+1) - \mathcal{N}_c^{\text{listen}}(t-1)\right|, \tag{15}$$

and $\mathcal{T}''$ is a set of time points in the jamming phase where each $t'' \in \mathcal{T}''$ is close to time $t$, and $\mathcal{T}'$ is a set of time points where each $t' \in \mathcal{T}'$ is in successive listening phases without jamming attack. If $T_J > 3$, $d_c^{\text{listen}}(\mathcal{T}')$ can be the set of successive listening phases in every period during training. Otherwise if $T_J \leq 3$, $d_c^{\text{listen}}(\mathcal{T}')$ has to be collected before jamming starts.

Again, it is important to note that when the jammer agent makes decisions at time $t$, $\mathcal{N}_c^{\text{listen}}(t+1)$ is not available. To address this, we propose a deep RL agent that uses the actor-critic algorithm to learn the policy.

## D. Actor-Critic Jammer Agent

Our proposed jammer agent utilizes an actor-critic deep RL algorithm to learn the policy that optimizes the output $\mathcal{C}_J(t)$ to minimize the victims' expected sum rate. The jammer works with a period $T_J$, and only switches from listening phase to jamming phase at the end of each period and uses the policy to make the decision $\mathcal{C}_J(t)$. The actor-critic algorithm [21] includes two neural networks, namely the actor and critic. The two networks have separate neurons and utilize separate back-propagation, and they may have separate hyper parameters. We then introduce the observation, action, reward, and the actor-critic update of this agent.

*1) Observation:* At each time slot, the jammer records its instant observation as a vector $O_J \in \mathbb{R}^N$. In a listening phase, $O_J^L = \{\mathcal{N}_1^{\text{listen}}, \mathcal{N}_2^{\text{listen}}, \dots, \mathcal{N}_N^{\text{listen}}\}$. Otherwise, in a jamming phase, $O_J^J = \{\mathbf{1}(1 \in \mathcal{C}_J), \mathbf{1}(2 \in \mathcal{C}_J) \dots, \mathbf{1}(N \in \mathcal{C}_J)\}$ where $\mathbf{1}$ is the indicator function. In the beginning at time slot $t$ in the jamming phase, the full observation $\mathcal{O}_J(t) = \{O_J^J(t - T_J), O_J^L(t - T_J + 1), \dots, O_J^L(t-1)\}$ is fed as the input state to the actor-critic agent.

*2) Action:* At the beginning of time slot $t$ in a jamming phase, given the input state $\mathcal{O}_J(t)$, the actor neural network outputs a vector of probabilities $\mathcal{P}_J(t) \in [0,1]^N$. From the probability vector, the decision $\mathcal{C}_J(t)$ is derived which is the subset of channels to jam, and it is described as the action $\mathcal{A}_J(t) \in \{0,1\}^N$:

$$\mathcal{C}_J(t) = \underset{\mathcal{C}_J}{\operatorname{argmax}} \sum_{c \in \mathcal{C}_J} \mathcal{P}_J(t), \tag{16}$$

$$\mathcal{A}_J(t) = \{\mathbf{1}(1 \in \mathcal{C}_J(t)), \mathbf{1}(2 \in \mathcal{C}_J(t)) \dots, \mathbf{1}(N \in \mathcal{C}_J(t))\}. \tag{17}$$

*3) Reward:* Following the jamming phase at time $t$, the reward is received to train the critic after the next listening phase at time $t+1$. This reward aims at encouraging the policy to produce an action that imitates $\hat{\mathcal{N}}_c^\beta(t)$, the linear interpolation of listened interference as in (13). Therefore, we set the reward as the negative of mean squared error:

$$R_J(t) = -\sum_c \left(\mathcal{A}_J(t)[c] - \frac{\hat{\mathcal{N}}_c^\beta(t)}{\max\left(\hat{\mathcal{N}}_c^\beta(t)\right)}\right)^2. \tag{18}$$

*4) Actor-Critic Update:* At the beginning of a jamming phase at time $t$, the actor with parameter $\theta_J$ and policy $f^{\theta_J}(\mathcal{O}_J)$ maps the input observation $\mathcal{O}_J$ to the output probability $\mathcal{P}_J$, which is similar to a Q-value generator. The critic with parameter $\phi_J$ and policy $g^{\phi_J}(\mathcal{O}_J)$ maps $\mathcal{O}_J$ to a single temporal difference (TD) error:

$$\delta_J(t) = R_J(t) + \gamma_J g^{\phi_J}(\mathcal{O}_J(t)) - g^{\phi_J}(\mathcal{O}_J(t - T_J)), \tag{19}$$

where $\gamma_J \in (0,1)$ is the discount factor. For each training sample, the critic is updated towards achieving the optimized parameter $\phi_J^*$ to minimize the least square TD:

$$\phi_J^* = \underset{\phi_J}{\operatorname{argmin}} \, (\delta_J^{g_{\phi_J}})^2. \tag{20}$$

The actor is updated towards the optimized parameter $\theta_J^*$ to minimize the policy gradient:

$$\theta_J^* = \underset{\theta_J}{\operatorname{argmax}} \nabla_{\theta_J} \sum_{c \in \mathcal{C}_J} \log f^{\theta_J}(\mathcal{O}_J)\delta_J^{g_{\phi_J}}. \tag{21}$$

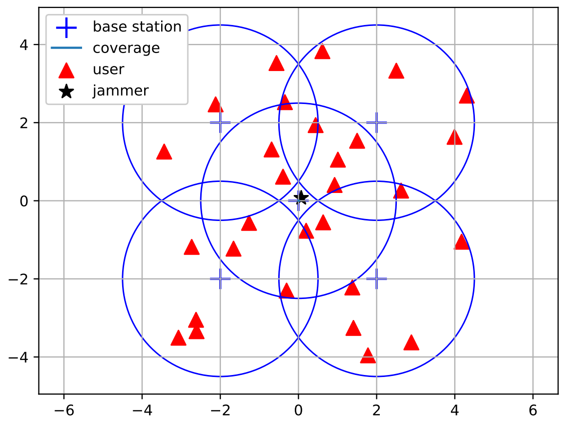

Fig. 1. Coverage map of service area with 5 base stations, 30 users, and the jammer.

Both networks are updated alternately to attain the optimal actor-critic policy. Note that during each parameter update with bootstrap method, the mini-batch of training samples (i.e., action-reward pairs) should be randomly drawn from a longer history record for faster convergence.

## IV. Numerical Results

As shown in Fig. 1, we in the experiments consider a service area with $N_B = 5$ base stations, $N_u = 30$ users, and a jammer. The theoretically optimized location of the jammer $\{x_J^*, y_J^*\}$ is determined according to (11) and it lies at the center $\{0, 0\}$, while the actual location is slightly moved away from the base station tower. There are $N = 16$ channels available, and the jammer picks at most $N_J = 8$ channels for jamming. The jamming power in each channel equals the transmission power in each channel: $P_J = P_B$. The jammer has phase switching period of $T_J = 2$ time slots.

In the experiments, the network slicing agent is initialized as the well-trained MACC agent with pointer network based actors as detailed in Section II, and time is initiated at $t = 0$. The jammer's actor and critic policies are implemented as two feed-forward neural networks (FNNs), both of them has one hidden layer with 16 hidden nodes. The jammer is initialized at $t = 100$ and begins jamming and performs online updating. During the training phase $100 \leq t < 10000$, the jammer follows an $\epsilon$-greedy policy to update the neural network parameters $\theta_J$ and $\phi_J$ with learning rate $10^{-5}$. It starts by fully exploring random actions, and the probability to choose random actions linearly decreases to 0.01, thus eventually leading the agent to mostly follow the actor policy $f^{\theta_J}(\mathcal{O}_J)$. This probability is fixed to 0.01 in the testing phase from $t = 10000$ to $t = 20000$.

In Fig. 2, we compare the performance of the proposed actor-critic jammer that approximates $\hat{\mathcal{N}}_c^{\beta}(t)$ with three other

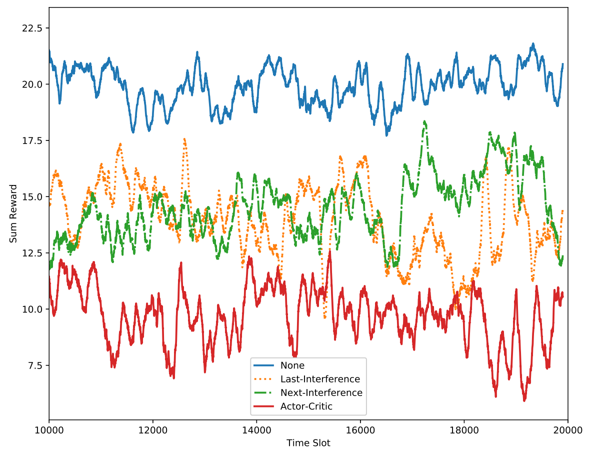

Fig. 2. Comparison of victims' sum reward in the testing phase achieved in the absence of jamming attack, and also achieved under attacks by the last-interference jammer, next-interference jammer, and the proposed actor-critic jammer.

scenarios in terms of victim sum reward in the testing phase. The first case is the setting with no jammer and hence the performance is that of the original network slicing agent in terms of the sum reward in the absence of jamming attacks. The second scenario is with a last-interference jammer agent that is positioned at the same location (i.e., the origin $\{0, 0\}$) with the same power budget. However, this jammer agent does not utilize any machine learning algorithms, and chooses the subset of channels with highest observed/listened interference power levels in the last listening phase:

$$\mathcal{C}_J^{\text{MaxIntf}}(t) = \underset{\mathcal{C}_J}{\text{argmax}} \sum_{c \in \mathcal{C}_J} \mathcal{N}_c^{\text{listen}}(t-1). \tag{22}$$

Consequently, the last-interference jammer which aims at the last time slot is equivalent to the proposed jammer when $\beta \to \infty$. The third scenario is with the next-interference jammer agent, which is also an actor-critic jammer agent but whose reward has $\beta = 0$, so it aims at the next time slot. Both of them are less efficient in suppressing the victim sum reward compared to our proposed actor-critic jammer, which aims at estimating $\hat{\mathcal{N}}_c^{\beta}(t)$ and therefore has better performance. Specifically, we observe in Fig. 2 that the proposed jammer results in the smallest sum reward values for the network slicing agents and has the most significant adversarial impact.

Furthermore, we show the numerical result of the same set of experiments in Table I, where we can see the victim under the actor-critic jamming attack completes only 73.68% of the requests, which is much less than the other cases. Additionally, we notice that the base station at coordinates $(0, 0)$ in Fig. 1, which is the closest one to the jammer's location, only completes 67.25% of the requests under the proposed jamming attack. In comparison, this base station under the other two types of attack completes about 80% of the requests. This indicates that the proposed agent is more likely to learn from the received interference.

TABLE I
PERFORMANCE COMPARISON OF DIFFERENT JAMMING ALGORITHMS DURING TEST PHASE

| jammer | average reward | complete ratio |
|---|---|---|
| None | 19.90 | 94.78% |
| Last-Interference | 13.94 | 82.36% |
| Next-Interference | 14.51 | 83.30% |
| Actor-Critic | 10.09 | 73.68% |

## V. CONCLUSION

In this paper, we considered the network slicing agents using MACC multi-agent deep RL with pointer network based actors. We developed a deep RL based jammer that aims at minimizing the network slicing agents' (i.e., victims') transmission rate and thus degrades their performance without prior knowledge of the victim policies or without receiving any direct feedback. We introduced the jamming and listening phases of the proposed jammer and addressed the jamming location optimization. We also studied the jamming channel optimization by designing an actor-critic agent that decides on which channels to jam. We have demonstrated the effectiveness of the proposed jammer via simulation results, and quantified the degradation in the performance of the network slicing agents compared to the performance achieved in the absence of any jamming attacks. We also provided comparisons among actor-critic based jammers with different assumptions on how to decide on which channels to jam (e.g., based on last or estimated next interference or linear interpolation of the two).

## REFERENCES

[1] A. Ericsson, "5G systems enabling the transformation of industry and society," *Tech. Rep.*, 2017.

[2] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[3] N. Alliance, "Description of network slicing concept," *NGMN 5G P*, vol. 1, no. 1, 2016.

[4] H. Zhang, N. Liu, X. Chu, K. Long, A.-H. Aghvami, and V. C. Leung, "Network slicing based 5G and future mobile networks: mobility, resource management, and challenges," *IEEE communications magazine*, vol. 55, no. 8, pp. 138–145, 2017.

[5] S. A. Kazmi, L. U. Khan, N. H. Tran, and C. S. Hong, *Network slicing for 5G and beyond networks*. Springer, 2019, vol. 1.

[6] R. Li, Z. Zhao, Q. Sun, I. Chih-Lin, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018.

[7] L. Zhang, J. Tan, Y.-C. Liang, G. Feng, and D. Niyato, "Deep reinforcement learning-based modulation and coding scheme selection in cognitive heterogeneous networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 6, pp. 3281–3294, 2019.

[8] Y. Shi, Y. E. Sagduyu, and T. Erpek, "Reinforcement learning for dynamic resource optimization in 5G radio access network slicing," in *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, 2020, pp. 1–6.

[9] Y. Shao, R. Li, Z. Zhao, and H. Zhang, "Graph attention network-based drl for network slicing management in dense cellular networks," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2021, pp. 1–6.

[10] F. Wang, M. C. Gursoy, and S. Velipasalar, "Multi-agent reinforcement learning with pointer networks for network slicing in cellular systems," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 1841–1846.

[11] X. Lyu, Y. Xiao, B. Daley, and C. Amato, "Contrasting centralized and decentralized critics in multi-agent reinforcement learning," *arXiv preprint arXiv:2102.04402*, 2021.

[12] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[13] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *arXiv preprint arXiv:1706.02275*, 2017.

[14] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," *arXiv preprint arXiv:1506.03134*, 2015.

[15] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[16] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9185–9193.

[17] Y. Lu, Y. Jia, J. Wang, B. Li, W. Chai, L. Carin, and S. Velipasalar, "Enhancing cross-task black-box transferability of adversarial examples with dispersion reduction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 940–949.

[18] Y. Shi, Y. E. Sagduyu, T. Erpek, K. Davaslioglu, Z. Lu, and J. H. Li, "Adversarial deep learning for cognitive radio security: Jamming attack and defense strategies," in *2018 IEEE international conference on communications workshops (ICC Workshops)*. IEEE, 2018, pp. 1–6.

[19] F. Wang, C. Zhong, M. C. Gursoy, and S. Velipasalar, "Resilient dynamic channel access via robust deep reinforcement learning," *IEEE Access*, 2021.

[20] L. Liang, J. Kim, S. C. Jha, K. Sivanesan, and G. Y. Li, "Spectrum and power allocation for vehicular communications with delayed CSI feedback," *IEEE Wireless Communications Letters*, vol. 6, no. 4, pp. 458–461, 2017.

[21] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, no. 7-9, pp. 1180–1190, 2008.