Multi-Agent Double Deep Q-Learning for Beamforming in mmWave MIMO Networks

Xueyuan Wang, M. Cenk Gursoy
Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244
Email: xwang173@syr.edu, mcgursoy@syr.edu

Abstract—Beamforming is one of the key techniques in millimeter wave (mmWave) multi-input multi-output (MIMO) communications. Designing appropriate beamforming not only improves the quality and strength of the received signal, but also can help reduce the interference, consequently enhancing the data rate. In this paper, we propose a distributed multi-agent double deep Q-learning algorithm for beamforming in mmWave MIMO networks, where multiple base stations (BSs) can automatically and dynamically adjust their beams to serve multiple highly-mobile user equipments (UEs). In the analysis, largest received power association criterion is considered for UEs, and a realistic channel model is taken into account. Simulation results demonstrate that the proposed learning-based algorithm can achieve comparable performance with respect to exhaustive search while operating at much lower complexity.

Index Terms—beamforming, deep reinforcement learning, MIMO, mmWave communications, multi-agent systems.

I. INTRODUCTION

With the rapid growth in data traffic, next-generation wireless communication systems are required to provide greater throughput to meet higher data-rate demands [1]. Under the light of this fact, millimeter wave (mmWave) multi-input multi-output (MIMO) communication has attracted much attention recently. In mmWave MIMO networks, beamforming is an effective technique that improves the quality and strength of the received signals by steering the signals generated from an array of transmit antennas to an intended angular direction [2].

Traditionally, finding the optimal beamforming solution has relied on optimization methods and iterative algorithms, which can lead to high computational complexity and increased delays and thus may not be suitable for real-time implementation [3]. Additionally, in a mobile environment, e.g., a network with mobile user equipments (UEs), UEs need to be frequently handed over from one base station (BS) to another, leading to increased control overhead and latency [4]. With advances in machine learning, there has been increased interest in deep learning and reinforcement learning algorithms to provide low-complexity solutions with low delay. Such learning-based approaches are also regarded as promising for beamforming problems.

A. Related Works

As noted above, leveraging recent advances in machine learning, optimal beamforming schemes can be determined in real time with low computational complexity using learning techniques. For instance, the authors in [4] have considered a network where a number of distributed BSs simultaneously serve one mobile UE. The UE ideally transmits one uplink training pilot sequence to all BSs equipped with omni or quasi-omni directional beam patterns, and the deep-learning model leverages the signals to train its neural network. After training, the deep-learning predicts the BS RF beamforming vectors in downlink data transmission. [5] has proposed an algorithm that combines three neural networks for performance optimization in massive MIMO beamforming. In the proposed system, one neural network is trained to generate realistic user mobility patterns, which are then used by a second neural network to produce relevant antenna diagrams. Meanwhile, a third neural network estimates the efficiency of the generated antenna diagrams and returns the corresponding reward to both networks. The authors in [6] have proposed a deep learning framework for the optimization of downlink beamforming. In particular, the solution is obtained based on convolutional neural networks and exploitation of expert knowledge, such as the uplink-downlink duality and the structure of known optimal solutions. In [7], a neural network architecture is used to jointly sense the millimeter wave channel and design hybrid precoding matrices. The neural network is first trained in a supervised manner, where a dataset of the mmWave channels and the corresponding RF beamforming/combing matrices are constructed and fed as the input and the target of the neural network, respectively. Then the trained neural network is applied online.

Reinforcement learning is also shown to be a useful tool for beamforming schemes. For instance, the authors in [8] used deep Q learning algorithm to jointly optimize the beamforming vectors and the transmit power of the BSs, and eventually to maximize the signal-to-interference-plus-noise ratio (SINR) of UEs. [9] presented a deep reinforcement learning framework to optimize MIMO broadcast beams autonomously and dynamically based on users mobility patterns or changes in user distribution, which can vary periodically. Using ray-tracing data, deep reinforcement learning engine is first trained offline, and then deployed online for real-time operation. Therefore, whenever the environment changes, the learning engine should
be re-trained offline based on the new environment.

In this paper, we consider a general setting and propose a distributed multi-agent double deep Q-learning network (DDQN) solution for beamforming in mmWave MIMO networks, where multiple BSs serve multiple mobile UEs. In this system, UEs move to different locations at each time, and may be served by different BSs according to the adopted largest received power association criterion. Each BS is a reinforcement learning agent and has its own DDQN. BSs, at each time, can only get information from the associated UEs, based on which the BSs predict the UEs mobility pattern and choose their beamforming vectors. Hence, the proposed distributed multi-agent DDQN solution adapts to UEs’ mobility. As key novel aspects compared to prior studies, we build a distributed multi-agent learning framework, and the proposed solution requires less feedback from the UEs, and the UEs’ feedback is only locally available to the serving BS. Consequently, we have novel designs and algorithms for the learning agents in a more general setting.

II. SYSTEM MODEL

In this section, we describe the considered mmWave MIMO network.

A. System Model

We consider multi-BS multi-UE MIMO networks, where BSs are simultaneously serving K mobile UEs. Each BS is equipped with $N_i$ antennas and $N_{RF}$ RF chains. We assume that the BSs apply analog-only beamforming using networks of phase shifters, and each RF chain is fully-connected with each antenna.

In this analog-only beamforming network, the BS upconverts the data stream to the carrier frequency by passing it through $N_{RF}$ RF chains. Following this, the BS uses an $N_i \times N_{RF}$ RF precoder $\mathbf{V}_{RF}$, which is implemented using analog phase shifters, i.e., with $|\mathbf{V}_{RF}(a,b)|^2 = \frac{1}{N_r}$, to construct the final transmitted signal. Note that $\mathbf{V}_{RF}(a,b) = \frac{1}{\sqrt{N_r}} e^{j\phi_{ab}}$ where $\phi_{ab}$ is the phase shift angle. For the $k$-th user, the received signal can be modeled as

$$\mathbf{y}_k = \sum_{j=1}^{J} \mathbf{H}_{kj} \mathbf{V}_{RF} \mathbf{s}_{kj} + \mathbf{n}_{kj}$$

where $\mathbf{s}_{kj}$ is the data stream vector for the $k$-th UE from the $j$-th BS. We assume that $\mathbb{E}[\mathbf{s}_{kj}\mathbf{s}_{kj}^H] = P_j \mathbf{I}_{N_r}$, and the noise factor $\mathbf{n}_{kj}$ is the channel response from the $j$-th BS to the $k$-th UE. $\mathbf{V}_{RF}$ is the RF precoder of the $j$-th BS.

B. Channel Model

We adopt a channel model with $L$ paths. $L$ is a small number for mmWave communications, and we assume $L = 1$ for line-of-sight (LOS) links. Now, the mmWave MIMO channel $\mathbf{H}$ can be expressed as

$$\mathbf{H} = \sqrt{\frac{N_i N_r}{L}} \sum_{l=1}^{L} \alpha_l^j \beta_l^j \mathbf{a}_l^j \mathbf{a}_l^H$$

where $\alpha_l$ and $\beta_l$ are the complex gain and the path loss of path $l$, respectively. $\mathbf{a}_l$ and $\mathbf{a}_l^H$ are the array response vectors at the BS and the UE sides, respectively. $\phi_l^j$ and $\phi_l^H$ are the angles of departure and arrival of the $l$-th path, respectively.

1) Array response: While the algorithms introduced in this paper can be applied to arbitrary antenna arrays, we provide the following two illustrative examples of commonly-used antenna arrays. For an $N$ element uniform linear array (ULA) on the $y$-axis, the array response vector can be written as

$$\mathbf{a}_{ULA}(\phi) = \frac{1}{\sqrt{N}} \left[ 1, e^{jkd\sin(\phi)}, ..., e^{j(N-1)kd\sin(\phi)} \right]$$

where $k = \frac{2\pi}{\lambda}$, $\lambda$ is the wavelength, and $d$ is the inter-element spacing. In the case of a uniform planar array (UPA) in the $y$-plane with $W$ and $H$ elements on the $y$ and $z$ axes respectively, the array response vector is given by

$$\mathbf{a}_{UPA}(\phi, \theta) = \frac{1}{\sqrt{N_r}} \left[ 1, ..., e^{jkd(m \sin(\phi) \cos(\theta) + n \sin(\phi) \sin(\theta))}, ..., e^{jkd(W - 1) \sin(\phi) \cos(\theta) + (H - 1) \sin(\phi) \sin(\theta)} \right]$$

where $0 < m < W$ and $0 < n < H$ are the $y$ and $z$ indices of an antenna element respectively and the antenna array size is $N = WH$. In this paper, we primarily concentrate on ULA.

2) Path Loss: Link between a UE and a BS can be either LOS or non-LOS (NLOS). The path loss model is formulated as

$$p_{l}(r) = \left\{ \begin{array}{ll} \kappa_{los} \alpha_{los}^{los}(r) & \text{with prob. } p_{los}(r) \\ \kappa_{nlos} \alpha_{nlos}^{nlos}(r) & \text{with prob. } p_{nlos}(r) = (1 - p_{los}(r)) \end{array} \right.$$  

where $r$ is the two-dimensional distance between the UE and BS. $\alpha_{los}$, $\alpha_{nlos}$ are the path loss exponents for LOS and NLOS links, respectively, $\kappa_{los}$, $\kappa_{nlos}$ are the intercepts of the LOS and NLOS path loss formulas, respectively, and $p_{los}(r)$ is the probability that the link has a LOS transmission at distance $r$.

Following the 3GPP standards described in [11], we express the probability of LOS link between the BSs and the UEs as

$$p_{los}(r) = \min(18/r, 1) \times (1 - \exp(-r/63)) + \exp(-r/63).$$

Shading is also taken into account in the channel model, and is modeled as a log-normal random variable, i.e., $10 \log v \sim \mathcal{N}(\mu_v, \sigma_v^2)$ with $\mu_v$ and $\sigma_v$ being the mean and variance of the channel power under shadowing, respectively.

C. Achievable Rate

In the multi-BS multi-UE mmWave MIMO network, besides the serving BS, other BSs inflict interference to the UEs. We can express the rate of the $k$-th UE when associated with the $j$-th BS as

$$R_k = \log_2 \left( 1 + \frac{P_j |\mathbf{H}_{kj} \mathbf{V}_{RF} \mathbf{V}_{RF}^H \mathbf{H}_{kj}^H|^2}{\sigma^2 + \sum_{l=1, l \neq j}^{J} P_l |\mathbf{H}_{kl} \mathbf{V}_{RF} \mathbf{V}_{RF}^H \mathbf{H}_{kl}^H|^2} \right)$$
where superscript $H$ and $|\cdot|$ denote the conjugate transpose and the determinant of a matrix, respectively. Then, the sum-rate of all UEs is $R_{\text{sum}} = \sum_{k=1}^{K} R_k$.

Our goal is to maximize the sum-rate of all UEs, and therefore we consider the following optimization problem:

$$\max_{V_{RF_j}, j \in J} \sum_{k=1}^{K} R_k$$

subject to $|V_{RF_j}(a, b)|^2 = \frac{1}{N_t} \forall j$, (8a)

where $V_{RF_j}$ is the analog beamforming vector of the $j$-th BS.

III. DISTRIBUTED DDQN FOR MMWAVE MIMO NETWORKS

In this section, we first describe the generalized form of DDQN, and then introduce the proposed distributed multi-agent DDQN in detail.

A. Generalized form of DDQN

In reinforcement learning, an agent dynamically interacts with an unknown environment $\mathcal{E}$, and makes sequential decisions. At each time step, the agent is in a state, $s_t \in \mathcal{S}$, selects an action, $a_t \in \mathcal{A}$, then receives an immediate scalar reward $r_t \in \mathcal{R}(s_t, a_t)$, and transitions to next state $s_{t+1}$. The cumulative discounted reward, $\mathcal{R}_t$, at time step $t$, is defined as $\mathcal{R}_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, where $\gamma \in (0, 1]$ is the reward discount factor, which balances the importance of immediate and future rewards. Q-learning is one of the most widely used algorithms for reinforcement learning. In Q-learning, the agent learns a state-action value function $Q^\pi(s, a)$, which is defined as the expected cumulative reward when the agent takes action $a_t$ in state $s_t$ following its policy $\pi$, and can be expressed as $Q^\pi(s, a) = E[\mathcal{R}_t | s, a]$. The optimal state-action value function is denoted as $Q^\ast(s, a) = \max_{\pi} Q^\pi(s, a)$, which satisfies the Bellman optimality equation [14] $Q^\ast(s, a) = E[r_t + \gamma \max_{a'} Q^\ast(s_{t+1}, a') | s, a]$. The goal is to find an optimal policy $\pi^\ast$ that maximizes the optimal action value function $Q^\ast(s, a) = \text{argmax}_{a} Q^\ast(s, a)$. The Q-learning update rule in its general form is given by

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s, a)]$$

where $\alpha$ is a scalar step size.

In problems with large state and action spaces, we can learn a parameterized value function $Q(s, a; \xi)$ [15]. A deep Q network consists of an online deep Q-learning phase and an offline deep neural network (DNN) construction phase, which is used to learn the value function $Q(s, a; \xi)$ where $\xi$ denotes the set of the parameters of the DNN. Generally, the parameter set $\xi$ can be optimized by minimizing the following loss function [16]

$$L_t(\xi_t) = E[y_t - Q(s_t, a_t; \xi_t)]^2$$

where $y_t = r_t + \gamma \max_{a} Q(s_{t+1}, a; \xi_t^-)$ is the objective function, and $\xi_t^-$ is copied every $\tau$ steps from the $\xi_t$.

The max operator in standard Q-learning and DQN uses the same values both to select and to evaluate an action, which increases the probability to select overestimated values and results in overoptimistic value estimates. In DDQN, the objective function can be written as [15]

$$y_t = r_t + \gamma Q(s_{t+1}, \text{argmax}_a Q(s_{t+1}, a; \xi_t); \xi_t^-),$$

where $\xi_t$ is used for action selection, and $\xi_t^-$ is used to evaluate the value of the policy, and $\xi_t^-$ can be updated symmetrically by switching the roles of $\xi_t$ and $\xi_t^-$. B. Distributed multi-agent DDQN

The algorithm of the distributed multi-agent DDQN for mobile UEs is described in this subsection. Before providing the definitions of states, actions, and rewards, the environment should be introduced first. In the environment $\mathcal{E}$, BSs are located with a certain distance in between. UEs move from different initial locations in different directions with different speeds. In each episode, UEs do not change their speed and direction until they move out of the coverage region of all BSs, and the episode ends. In each episode, at each time step, UEs move to new locations. We assume that at the beginning of each time step, in a very short time slot $\tau$, each BS uses an omni-directional antenna pattern, and transmits pilot signals to all UEs. Therefore, each UE receives pilot signals from all BSs, and chooses to associate with the BS providing the strongest pilot signal, and computes the immediate rate under the omni-directional pattern, and sends it to the serving BS, indicating the association condition with itself. It is assumed that each UE is equipped with a memory containing prior information (e.g., the location history, the omni-directional rate in the last $T_m$ time steps), and each UE can send this information to the associated BS as well. Subsequently, the BS performs beamforming according to our algorithm, and serves the associated UEs. Then the UEs learn the immediate rate achieved with beamforming, and feed this information back to the serving BS. The procedure in one time step is illustrated in Fig. 1.

![Fig. 1](image-url)
Algorithm 1 Distributed multi-agent DDQN

Initialization:
1: Initialize replay memory $D^j$ of DDQN, $j \in J$, to capacity $N$ for every BS.
2: Initialize online network with random parameter $\xi^j$, $j \in J$.
3: Initialize target network with parameter $\xi^{j-\tau} = \xi^j$, $j \in J$.

Learning:
4: for episode = 0; total episode do
5: Reset environment $E$.
6: Initialize $s^j_t$ for $j \in J$.
7: for $t = 1 : T$ do
8: for $j = 1 : J$ do
9: Obtain association condition and the sum-rate of all associated UEs $R^{ji}_s$ by association procedure.
10: Sample $c$ from Uniform $(0,1)$
11: if $c \leq \xi^j_t$ then
12: Select an action (beamforming vector index) randomly from the codebook $F$.
13: else
14: Select the action $a^j_t = \arg\max_a Q^j(s^j_t, a; \xi^j_t)$.
15: end if
16: Execute action $a^j_t$, i.e. apply the selected beamforming vector on the antenna arrays of $j$-th BS.
17: Observing the resulting state $s^j_{t+1}$ and the immediate sum-rate of all associated UEs $R^j_{s}$.
18: Compute the immediate reward $r^j_t$, i.e. $\frac{R^j_s}{R^j_i} \times 100\%$.
19: Store the experience tuple $(s^j_t, a^j_t, r^j_t, s^j_{t+1})$ in $D^j$.
20: Sample random minibatch of experience $(s^j_t, a^j_t, r^j_t, s^j_{t+1})$ from $D^j$.
21: Update $y^j_t = \begin{cases} r^j_t & \text{if episode terminates at step } \tau + 1 \\ r^j_t + \gamma Q(s^j_{\tau+1}, \arg\max_a Q(s^j_{\tau+1}, a; \xi^j_\tau); \xi^j_{\tau-1}) & \text{otherwise} \end{cases}$
22: Perform a gradient descent step on $(y^j_t - Q(s^j_t, a; \xi^j_t))^2$ with respect to the network parameters $\xi^j_t$.
23: For every $N_n$ steps reset target network parameter $\xi^{j-\tau} = \xi^j$.
24: end for
25: end for
26: end for

In the multi-agent DDQN model, each BS is an agent and the state, action, and reward tuple of the $j$-th BS is denoted by $(s^j_t, a^j_t, r^j_t)$. These states, actions and rewards are described in detail below:

1) State: At each time step, each BS serves multiple UEs, and is able to obtain the information in the memory of each UE. This information of all associated UEs constitute as state at time $t$, noted as $s^j_t$ for the $j$-th BS. Assume that the length of history in each memory is $T_m$, and there are $K$ UEs in this network, then the state $s^j_t = [m^j_{0-t}, m^j_{1-t}, \ldots, m^j_{T_m-t-1}, \ldots, m^j_{T_m-T_m-1}]$. It is worth noting that for UEs not associated with the $j$-th BS, $m^*_t$ is set to be $0$. In other words, the BSs only need information from the associated UEs. In this paper, we denote the rate achieved with the omni-directional antenna pattern as $m^*_t$. If the location information of each UE is also available, we denote the omni-directional pattern rate and the location information together as state $s^j_t$. In Section IV, we provide performance results with and without location information.

2) Action: At each time step, each BS chooses an analog beamforming vector $V_{RF}$. Due to the constraints on the RF hardware, such as the availability of only certain quantized angles for the RF phase shifters, the analog beamforming vectors can take only certain values. Hence, finite-size codebooks for the candidate beamforming vectors are needed. In practice, the beamforming vectors are spatial matched filters for the single-path channels [13]. Thus, they have the same form of the array response vector and can be parameterized by a simple angle. While the algorithm in this paper can be applied to arbitrary finite-size codebooks, we adopt the codebook, denoted by $F$, consisting of the steering vectors $a_i(\phi_Q)$ where $\phi_Q$ is the quantized angle. The beamforming vector index in the codebook is defined as the action $a^j_t$.

3) Reward: As noted before, at the beginning of each time step, each BS learns the immediate rate, $R^j_{i}$, provided to all associated UEs with the omni-directional antenna radiation pattern. After performing beamforming, each BS also learns the immediate rate $R^j_s$ achieved with beamforming. We regard the ratio of two rates, $\frac{R^j_s}{R^j_i}$ as the immediate reward $r^j_t$ of the $j$-th BS when this BS takes action $a^j_t$ in state $s^j_t$.

Setting the analog beamforming vector codebook $F$, the optimization problem can be reformulated as

$$\max_{V_{RF}, t, j \in J} \sum_{k=1}^{K} R_k[t]$$

subject to $V_{RF, j}[t] \in F$, $\forall j$. (12a)

where $V_{RF, j}[t]$ is the analog beamforming vector of the $j$-th BS at time $t$.

IV. Performance Evaluation

In this section, we provide simulation results and evaluate the performance of the proposed multi-agent DDQN for mmWave MIMO beamforming.
A. Environment Setting

The considered environment is illustrated in Fig. 2. As shown in the figure, we have two intersecting streets. The BSs are located along the streets, while the UEs are moving from the beginning of either street in either direction. UEs move at random speeds. The length of each street is set as 100m, starting from -50m to 50m on each axis. The width of the road is set as 8m, i.e., (-4m, 4m). The speed of the UEs can be between 2 to 5 m/s. The locations of BSs are set at the coordinates of [(5,-5),(-25,-5)]. In addition, the number of antennas at the base stations and the UEs are set, respectively, as $N_t = 16$, $N_r = 1$. Finally, the number of RF chains is $N_{RF} = 2$.

B. Hyperparameters

In our experiment, we construct the DDQN via three-layered neural networks using Adam optimizer to evaluate the gradient descent of the evaluated and target networks. Our input size depends on the number of UEs in the environment, i.e., the input size is $KT_m$, when $K$ is the number of UEs and $T_m$ is the length of historic/prior information at of each UE, and is set to be 8 in the simulation. The output size should be the size of the action notebook $\mathcal{F}$. The number of neurons of the three layers are $12K$, $8K$ and 8, respectively. The discount factor is 0.95, batch size is 32, and the learning rate for two BSs are 0.0001 and 0.005, respectively. We also use $\epsilon$-greedy policy and the maximum value of $\epsilon$ is 0.9 and the minimum is 0.1.

C. Experiment results

Fig. 3 plots the sum-rate as a function of number of UEs for different number of BSs. Results from exhaustive search among all possible beamforming directions and also random selection are provided as two benchmark results. Note that the exhaustive search requires perfect channel state information (CSI), and in random selection the BSs randomly choose actions from the codebook. Figs. 3(a) and 3(b) (in which we consider the cases of a single BS and two BSs, respectively) show that DDQN can achieve better performance than random selection, and comparable results with exhaustive search which has high computational complexity and incurs potentially large delays especially in mobile scenarios. In addition, in both figures, we provide the performance curves when the location information is available and unavailable. The performance with location information given is only slightly better, indicating that such information is not critical for our algorithm. Furthermore, when the number of UEs increases, the sum-rate grows almost linearly, demonstrating that the proposed algorithm can handle multiple UEs without much decrease in the data rate experienced at each UE. On the
other hand, when we compare Figs. 3(a) and 3(b), we notice that for the same number of UEs, two BSs can provide higher sum-rate than in the case of a single BS. Even though there is interference when there are multiple BSs, two BSs can provide larger coverage, leading to higher SINR levels and rates with effective beamforming. We also notice that random selection performs rather poorly especially in the presence of two BSs.

Fig. 4 displays the sum-rate performances during testing, when the DDQN is trained for the exact number of UEs used in the tests and also when DDQN is trained considering 6 UEs regardless of how many UEs we have in the test period. When the DDQN is trained for 6 UEs, if the real number of UEs is less than 6 in the testing, we need to do zero padding in the input of the DDQN; if the actual number of UEs is more than 6, we just randomly choose 6 UEs and make a decision. From Fig. 4 we observe that when the DDQN is trained for the actual number of UEs, the testing performance is slightly better than that of the other case, indicating that the pre-trained model does not need to be restricted to a certain number of UEs and can be applied to scenarios in which the number of UEs in the test is different from that in the training period.

V. Conclusion

In this paper, we have proposed a multi-agent DDQN algorithm for beamforming in mmWave MIMO networks. Largest received power association criterion has been considered for BS association of the UEs. BSs act as reinforcement learning agents, and according to the limited information obtained from the associated UEs, they automatically and dynamically adjust their beams to improve the received power of the associated UEs. Via simulations, we have demonstrated that the proposed algorithm can achieve comparable network performance with respect to exhaustive search, and better performance than the random selection, which leads to especially poor performance when there are multiple BSs. We have noted that location information is not critical in our algorithm. In addition, the pre-trained model is not restricted to the same number of UEs as in the testing phase and can be applied to multiple testing scenarios with different number of UEs. 3D beamforming in mmWave MIMO networks remains as an interesting future research direction in which DDQN can also be employed.

REFERENCES

[1] H. Huang, W. Xia, J. Xiong, J. Yang, G. Zheng, and X. Zhu, “Unsupervised learning-based fast beamforming design for downlink MIMO,” IEEE Access, vol. 7, pp. 7599–7605, 2019.

[2] I. Ahmed, H. Khammari, A. Shahid, A. Musa, K. S. Kim, E. De Poorter, and I. Moerman, “A survey on hybrid beamforming techniques in 5G: Architecture and system model perspectives,” IEEE Communications Surveys Tutorials, vol. 20, no. 4, pp. 3060–3097, Fourthquarter 2018.

[3] W. Xia, G. Zheng, Y. Zhu, J. Zhang, J. Wang, and A. P. Petropulu, “A deep learning framework for optimization of MISO downlink beamforming,” CoRR, vol. abs/1901.00354, 2019. [Online]. Available: http://arxiv.org/abs/1901.00354

[4] A. Alkhateeb, S. Alex, P. Varkey, Y. Li, Q. Qu, and D. Tjukovic, “Deep learning coordinated beamforming for highly-mobile millimeter wave systems,” IEEE Access, vol. 6, pp. 37328–37348, 2018.

[5] T. Maksymyuk, J. Gazda, O. Yaremko, and D. Nevinskiy, “Deep learning based massive MIMO beamforming for 5G mobile network,” in 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), September 2018, pp. 241–244.

[6] W. Xia, G. Zheng, Y. Zhu, J. Zhang, J. Wang, and A. P. Petropulu, “A deep learning framework for optimization of MISO downlink beamforming,” CoRR, vol. abs/1901.00354, 2019. [Online]. Available: http://arxiv.org/abs/1901.00354

[7] X. Li and A. Alkhateeb, “Deep learning for direct hybrid precoding in millimeter wave massive MIMO systems,” CoRR, vol. abs/1905.13212, 2019. [Online]. Available: http://arxiv.org/abs/1905.13212

[8] F. B. Mismar, B. L. Evans, and A. Alkhateeb, “Deep reinforcement learning for 5G networks: Joint beamforming, power control, and interference coordination,” arXiv preprint arXiv:1907.00123, 2019.

[9] R. Shafin, H. Chen, Y. H. Nam, S. Hur, J. Park, J. Reed, L. Liu et al., “Self-tuning sectorization: Deep reinforcement learning meets broadcast beam optimization,” arXiv preprint arXiv:1906.06021, 2019.

[10] O. E. Ayach, S. Rajagopal, S. Abu-Surra, Z. Pi, and R. W. Heath, “Spatially sparse precoding in millimeter wave MIMO systems,” IEEE Transactions on Wireless Communications, vol. 13, no. 3, pp. 1499–1513, March 2014.

[11] 3GPP, “3GPP TR 36.828 (v11.0.0): Further enhancements to LTE time division duplex (TDD) for downlink-uplink (DL-UL) interference management and traffic adaptation,” 2012.

[12] J. Jiang and D. Kong, “Joint user scheduling and MU-MIMO hybrid beamforming algorithm for mmwave FDMA massive MIMO system,” International Journal of Antennas and Propagation, vol. 2016, 2016.

[13] A. Alkhateeb, G. Leus, and R. W. Heath, “Limited feedback hybrid precoding for multi-user millimeter wave systems,” IEEE Transactions on Wireless Communications, vol. 14, no. 11, pp. 6481–6494, November 2015.

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

[15] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double q-learning,” in Thirtieth AAAI conference on artificial intelligence, 2016.

[16] P. Lv, X. Wang, Y. Cheng, and Z. Duan, “Stochastic double deep q-network,” IEEE Access, vol. 7, pp. 79446–79454, 2019.