Distributed Multi-Agent Deep Q-Learning for Load Balancing User Association in Dense Networks

Byungju Lim, *Member, IEEE*, and Mai Vu, *Senior Member, IEEE*

***Abstract*—Distributed learning can lead to effective user association with low overhead, but faces significant challenges in incorporating load balancing at all base stations (BS) because of coupling constraints. In this letter, we propose a distributed multi-agent deep Q-learning model for user association to satisfy the load balancing constraint at every BS. Specifically, we design a deep Q-network (DQN) with target Q-network and experience replay buffer at each user as an agent. We also propose a multi-agent matching policy to control the number of users connected to each BS for load balancing. The policy enhances network throughput by implementing a novel updating rule for the preference list at each BS. The proposed multi-agent DQN model operates in a fully distributed manner, where each agent only uses local information and requires no information exchange between agents. Simulation results demonstrate that our proposed algorithm outperforms a conventional distributed load balancing algorithm and approaches a centralized scheme performance, while exhibiting fast convergence and high adaptability to channel changes.**

***Index Terms*—Association, load balancing, deep Q-learning, multi-agent, DQN, distributed learning.**

I. Introduction

TO MEET the ever-increasing demand for high data rates, future cellular networks will expand to higher frequency band including millimeter wave and sub-terahertz frequencies. As the main challenge at higher frequencies is the high path loss and short coverage, dense networks have emerged as a viable solution to provide seamless coverage and high throughput. In such a dense environment, even with beamforming transmission, interference can be a major factor impeding performance. User association plays a key role in reducing interference to other users, maximizing the network throughput, and maintaining load balancing among all base stations.

Deep reinforcement learning (DRL) has recently attracted attention as an effective method for user association. A centralized DRL algorithm can lead to high throughput for user association [1], but requires global channel state information which causes significant backhaul overhead, and has high computational complexity due to the large number of base stations. To overcome these issues, distributed DRL algorithms have been considered [2]; however, existing work requires explicit message passing among the base stations.

Furthermore, in a dense network, base stations are prone to be overloaded in crowded/congested areas or times. Overloading a base station can degrade network performance in terms of throughput and fairness. Avoiding an overload scenario necessitates load balancing user association. Load balancing has recently been considered in reinforcement learning schemes for user association, either explicitly as in [3] or indirectly via a collision cost as in [4]. Explicit load balancing constraints introduce dependency among different agents' actions and require some form of actions coordination. For example, each user performs the association action and receives a reward in a distributed manner, but a central coordinator is needed to make load balancing association decisions and compute the sum rate from all users [3]. Such a central coordinator increases the signaling overhead and can limit the algorithm scalability. Hence, load balancing user association is important in practice but is challenging to design in a fully distributed manner without a central coordinator.

In this letter, we propose a distributed multi-agent deep Q-network (MADQN) for user association under load balancing constraints. In particular, we design a deep Q-network (DQN) at each agent and propose a matching game based multi-agent policy to satisfy load balancing, where each BS maintains a preference list to make association decisions. To the best of our knowledge, this is the first attempt to perform user association using a fully distributed MADQN under load balancing. The contributions can be summarized as follows.

- A novel distributed MADRL framework is proposed for load balancing user association in a dense network. Each UE agent employs a private deep Q-network which only exploits local information and does not require information exchange between BSs and/or agents.
- To accomplish fully distributed training and execution, we propose a matching-game based multi-agent policy which guarantees load balancing via a mechanism of users sending association requests and BSs responding with ACK/NACK signals.
- The algorithm performance is verified via extensive simulation, showing network throughput approaching that of the near-optimal centralized worst connection swapping (WCS) algorithm, with fast convergence and high adaptability to channel changes.

II. System Model and Problem Formulation

A. System Model

Consider a multi-cell network where B BSs equipped with N antenna arrays and K single-antenna users are deployed. Let

Manuscript received 23 November 2022; revised 16 January 2023; accepted 21 February 2023. Date of publication 28 February 2023; date of current version 11 July 2023. This work was supported in part by the National Science Foundation through the CNS Core Program under Grant 1908552 and the CNS/SWIFT Program under Grant 2127648. The associate editor coordinating the review of this article and approving it for publication was S. Bi. *(Corresponding author: Byungju Lim.)*

Byungju Lim was with the Department of Electrical and Computer Engineering, Tufts University, Medford, MA 02155 USA. He is now with the Department of Electronic Engineering, Pukyong National University, Busan 48513, South Korea (e-mail: limbj@pknu.ac.kr).

Mai Vu is with the Department of Electrical and Computer Engineering, Tufts University, Medford, MA 02155 USA (e-mail: mai.vu@tufts.edu).

Digital Object Identifier 10.1109/LWC.2023.3250492

$\mathcal{B} = \{1, 2, \ldots, B\}$ and $\mathcal{K} = \{1, 2, \ldots, K\}$ denote the set of BSs and users, respectively. Note that for mmWave communication, hybrid beamforming structure is desirable to reduce the energy consumption and hardware complexity. In the hybrid structure, S_b RF chains are implemented, which is less than the number of transmit antennas. Due to the limitations of RF chains, each BS can transmit up to S_b data streams simultaneously and the maximum number of users served by each BS is S_b. Thus, the number of associated users at each BS cannot exceed its number of RF chains.

Let $\rho_{b,k} \in \{0, 1\}$ be the association variable between BS b and UE k. The received signal at the k-th user served by the b-th BS can be written as

$$y_k = \mathbf{h}_{b,k}^* \mathbf{w}_{b,k} x_k + I_{\text{Intra}} + I_{\text{Inter}} + n_{b,k}, \tag{1}$$

where $I_{\text{Intra}} = \sum_{\substack{k' \in \mathcal{K} \\ k' \neq k}} \rho_{b,k'} \mathbf{h}_{b,k}^* \mathbf{w}_{b,k'} x_{k'}$, $I_{\text{Inter}} = \sum_{\substack{b' \in \mathcal{B} \\ b' \neq b}} \sum_{k' \in \mathcal{K}} \rho_{b',k'} \mathbf{h}_{b',k}^H \mathbf{w}_{b',k'} x_{k'}$, $\mathbf{h}_{b,k} \in \mathbb{C}^{N \times 1}$ denote the channel between the b-th BS and the k-th user, and $\mathbf{w}_{b,k} \in \mathbb{C}^{N \times 1}$ is the transmit beamforming vector for the k-th user at the b-th BS. For beamforming vector, we employ a MRT based beamforming $\mathbf{w}_{b,k} = \sqrt{P_{b,k}} \frac{\mathbf{h}_{b,k}^*}{\|\mathbf{h}_{b,k}\|}$ where $P_{b,k}$ is transmit power from the b-th BS to the k-th user.

For mmWave communication, we adopt cluster based mmWave channel model as follows:

$$\mathbf{h} = \frac{1}{\sqrt{L}} \sum_{l=0}^{L} \alpha_l \mathbf{a}(\phi_l, \psi_l), \tag{2}$$

where L is the number of NLoS path with $l = 0$ denoting a LoS path, and α_l is a complex gain of the l-th path. In (2), ϕ_l and ψ_l are the azimuth and elevation angle of departure, respectively, and $\mathbf{a}(\phi, \psi)$ denotes the array response vector for a uniform planar array (UPA) to perform 3D beamforming.

B. Problem Formulation

Considering the received signal in (1), signal-to-interference plus noise ratio (SINR) can be expressed as

$$\gamma_{b,k} = \frac{|\mathbf{h}_{b,k} \mathbf{w}_{b,k}|^2}{\sum_{\substack{k' \in \mathcal{K} \\ k' \neq k}} \rho_{b,k'} |\mathbf{h}_{b,k} \mathbf{w}_{b,k'}|^2 + \sum_{\substack{b' \in \mathcal{B} \\ b' \neq b}} \sum_{k' \in \mathcal{K}} \rho_{b',k'} |\mathbf{h}_{b',k} \mathbf{w}_{b',k'}|^2 + \sigma^2}, \tag{3}$$

where σ^2 is noise power. We stress that the interference is significantly dependent on user association, and thus it should be carefully designed to improve system performance. The user association problem for load balancing in a dense network to maximize the sum rate can be formulated as

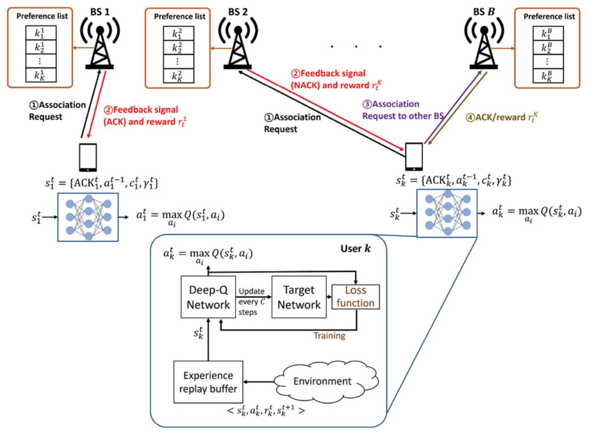
$$\max_{\rho_{b,k}} \ \Sigma_{k=1}^{K} \Sigma_{b=1}^{B} \rho_{b,k} R_{b,k} \tag{4a}$$

$$\text{s.t} \ \Sigma_{b=1}^{B} \rho_{b,k} \leq 1, \ \forall \ k \in \mathcal{K}, \tag{4b}$$

$$\Sigma_{k=1}^{K} \rho_{b,k} \leq S_b, \ \forall \ b \in \mathcal{B}, \tag{4c}$$

$$\rho_{b,k} \in \{0, 1\}, \tag{4d}$$

where $R_{b,k} = \log_2(1+\gamma_{b,k})$. In (4), (4b) represents the unique association for each user and (4c) denotes the load balancing constraint. By restricting the number of maximum users at each BS, it can balance the loads among all BSs in a dense wireless network [5].



Preference list
BS 1
k_1^1
k_2^1
k_K^1
Preference list
BS 2
k_1^2
k_2^2
k_K^2
BS B
Preference list
k_1^B
k_2^B
k_K^B
②Feedback signal (NACK) and reward r_t^K
①Association Request
②Feedback signal (ACK) and reward r_t^1
①Association Request
③Association Request to other BS
④ACK/reward r_t^K
$s_1^t = \{\text{ACK}_1^t, a_1^{t-1}, c_1^t, \gamma_1^t\}$
$s_k^t = \{\text{ACK}_k^t, a_k^{t-1}, c_k^t, \gamma_k^t\}$
s_1^t
$a_1^t = \max_{a_i} Q(s_1^t, a_i)$
s_k^t
$a_k^t = \max_{a_i} Q(s_k^t, a_i)$
User k
$a_k^t = \max_{a_i} Q(s_k^t, a_i)$
Deep-Q Network
Update every C steps
Target Network
Loss function
Training
s_k^t
Experience replay buffer
Environment
$< s_k^t, a_k^t, r_k^t, s_k^{t+1} >$

Fig. 1. Structure of the proposed distributed MADQN system; each user has its own DQN where one is shown in detail.

Centralized algorithms have been proposed to solve this problem [3], [5] but requires a central controller, which increases the communication signal overhead. To address this issue, we propose a distributed MADQN algorithm in which each user independently determines an action, that is to decide the value of $\rho_{b,k}$, with only locally observed information.

III. Multi-Agent Deep Q-Learning for Load-Balancing User Association

In this section, we examine how user association can be performed with DRL using only local information in a fully distributed manner while satisfying the load balancing constraint. We first describe a deep Q-network (DQN) model for each user as an agent and propose a distributed multi-agent DRL system consisting of multiple agents each running its own DQN model, and a multi-agent policy that governs the interaction among the agents and how their actions are determined (see Fig. 1).

The learning is performed in episodes, where each episode can correspond to a frame and contains multiple learning time steps. The channels are assumed to be fixed during an episode and can change from episode to episode depending on the coherence time and network dynamics. During each episode, the DQN is learning and updating every time step, and association for each user is performed once per episode. To the best of our knowledge, this is the first work that uses multi-agent DRL for user association, while ensuring load balancing in a fully distributed fashion.

A. DQN Design

For multi-agent deep Q-learning, each user acts as an agent and has an independent DQN without a central coordinator. In this letter, we consider the DQN architecture proposed in [6], which contains two key ideas to improve the stability and convergence of deep Q-learning. First is the use of an experience replay buffer to randomly sample from past experience for the neural network training, and second is the use of a target network which is updated periodically from the learning network to reduce the correlation in Q-values. Next, we introduce the state, action, reward function, output, and a distributed training procedure for the DQN.

1) State: Each agent k obtains local information from environment and defines its state at time t as

$$s_k^t = \{\mathrm{ACK}_k^t, a_k^{t-1}, c_k^t, \gamma_k^t\}, \tag{5}$$

where $\mathrm{ACK}_k^t \in \{0, 1\}$ is a feedback from BS, action a_k^{t-1} is the index of the BS which the k-th user requested to associate at time $t-1$, $c_k^t = \{c_{b,k}^t\}$ is a rejected BS list consisting of indices of BSs which the k-th user has applied and been rejected, and $\gamma_k^t = \{\gamma_{b,k}^t\}_{b\in\mathcal{B}}$ is SINR information from all BSs. Note that Eq. (3) is only used for analysis but not for computing the SINR at each user. In practice, the SINR at each user can be obtained directly by locally measuring the reference signal received quality (RSRQ) from the reference signal at each BS without requiring explicit channel state information or computation [7]. This local RSRQ measurement makes it possible to perform our multi-agent system in a fully distributed manner. Each agent gets an ACK/NACK signal determined by the requested BS to satisfy the load balancing constraint and constructs a rejected BS list based on the NACK feedback. It is worth noting that the state only includes local information and there is no information exchange between agents to obtain the state. Hence, actions can be taken at each agent in a fully distributed manner using only local state information.

2) Action: The overall action space for each agent is defined as $\mathcal{A} = \{1, 2, \ldots, B\}$. Due to constraint (4b), each user only chooses one BS at each time t. That is, initially, the k-th agent selects an action, $a_k^t \in \mathcal{A}$, and requests association to that BS. To avoid repeatedly choosing a rejected BS at each agent, we adopt an *action masking technique* that masks out invalid actions and selects an action only from a valid action space [8]. Based on the current state information, each agent masks a rejected action, for which its Q-value maps into $-\infty$. The masked out actions are those in the list c_k^t in the k-th agent's state. The agent will only choose to take an action from *the valid action space*, which is updated at each time step as $\mathcal{A}_k^t = \mathcal{A} \backslash c_k^t$. This separation between invalid and valid action spaces helps to significantly speed up the learning process and convergence.

3) Reward: To evaluate the action of each agent, an instantaneous rate is given as immediate reward, r_k^t, at time t. However, if the request of an agent is rejected from BS, it would get a negative reward because it violates the load balancing constraint. As such, a reward for each agent at each time t can be defined as

$$r_k^t = \begin{cases} -1, & \text{NACK received} \\ \log_2\left(1 + \gamma_{a_t^k, k}\right), & \text{ACK received} \end{cases} \tag{6}$$

4) Output: DQN is to estimate Q-value for each action and the size of output layer equals to the number of possible action space, $|\mathcal{A}|$. Accordingly, DQN for each agent returns Q-value as $Q_k(s_k^t, a_k^t)$ for each action given its state information and each agent takes an action corresponding to a maximum value from the obtained Q-values.

B. DQN Training and Updating

For multi-agent DRL, we employ the DQN architecture in [6] at each agent to estimate a Q-value, $Q(s, a; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the weights of the learning deep neural network (DNN). Each agent trains its own DQN model to minimize the loss function

$$\mathcal{L}_k(\boldsymbol{\theta}_k) = \mathbb{E}\left[\left(y_k^{DQN} - Q_k(s_k^t, a_k^t; \boldsymbol{\theta}_k)\right)^2\right], \tag{7}$$

where $y_k^{DQN} = R_k^t + \gamma \max_{a_k' \in \mathcal{A}} Q(s_k^{t+1}, a_k'; \boldsymbol{\theta}_k^-)$ and the expectation is taken over a mini batch sampled from the experience replay buffer. Here, $\boldsymbol{\theta}_k^-$ denotes the weights of the target network, which is updated from the learning Q-network every C steps where C is a constant. Specifically, each agent uses stochastic gradient descent (SGD) to update the parameter of its DQN as follows [9]:

$$\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_k + \eta\left(y_k^{DQN} - Q_k(s_k^t, a_k^t; \boldsymbol{\theta}_k)\right)\nabla Q_k(s_k^t, a_k^t; \boldsymbol{\theta}_k), \tag{8}$$

where η is the learning rate.

Furthermore, each agent adds the current experience $(s_k^t, a_k^t, r_k^t, s_k^{t+1})$ into its experience replay buffer and randomly samples a mini batch from this buffer at each time step to train its DQN. To learn more efficiency using the experience replay, we use *prioritized experience relay (PER)* which samples a mini batch based on a set of priority [10]. Specifically, the sampling probability of sample i is defined as $P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$ where $p_i = |\delta_i| + \epsilon$ is the priority of sample i, δ_i is a temporal difference (TD) error obtained as $y_k^{DQN} - Q_k(s_k^t, a_k^t; \boldsymbol{\theta}_k)$, ϵ is a positive offset, and $\alpha \in [0, 1]$ is a prioritization parameter.

C. Multi-Agent Matching Policy

The main challenge in designing a distributed multi-agent DRL is to satisfy the load balancing constraint in (4c), which cannot be controlled at the user side. To address this problem, we propose a multi-agent policy based on a matching game [11] to allow each user to independently learn and take the best action at each time in a distributed manner, while the load balancing constraint is ensured every time step.

1) Multi-Agent Matching Policy Per Episode: The proposed multi-agent policy is based on the matching game and consists of four steps as follows.

(i) Each agent determines the index of the best BS from its valid action set based on its own DQN output and sends a request to that BS together with the SINR-value for that BS, to be used by the BS to update its preference list.

(ii) Each BS maintains a waiting list equal to its load balancing quota. After receiving association requests from users, each BS ranks all requests based on its own criterion such as sum rate (discussed in more details below), and uses the ranking to decide acceptance or rejection of requests for ensuring the load balancing constraint. If a user is accepted, the BS updates and inserts that user into its waiting list, but no actual BS-UE connection is made until the episode terminates. Then, each BS sends a feedback signal to each requested user containing ACK or NACK information.[1]

[1]For a centralized scheme, global SINR values are sent to a central coordinator, which requires *SBK* bits where *S* is the number of quantization bits for an SINR value. On the other hand, an ACK/NACK signal requires only 1 bit information, and thus significantly reduces the signaling overhead.

(iii) If a user receives a NACK, that user updates its state including a rejected BS list, then comes back to Step (i).

(iv) When all users receive an ACK signal from a requested BS or have applied to all BSs, the episode terminates and the final waiting list at each BS determines its user association.

Note that a rejected user does not request the same BS again because it has a lower preference at that BS than the accepted users. At the next time step, the rejected user still has a low preference, thus it would be rejected and need not request again.

This multi-agent policy is based on the fundamentals of matching theory which enable our proposed algorithm operate in a distributed manner. Even though the action is taken at each agent independently, the feedback signal from each BS ensures load balancing at that BS. To determine the acceptance of requesting users, each BS ranks these users and continuously updates a preference list as follows.

2) Update of BS Preference List: We propose an updating rule for the preference list at each BS which has a strong impact on performance. Each BS maintains its own preference list for all users in episode p at time step t as $\{v_k^{p,t}\}$. These values are first initialized as $v_k^{0,0} = 0, \ \forall k$. Each BS then follows these steps to update its preference list in episode p.

(i) When association requests arrive at time step $t+1$ together with their SINR-values, the BS updates the ranking value of requested user k as

$$v_k^{p,t+1} = v_k^{p,t} + \log_2(1+\gamma_{b,k}^{p,t+1}) - \log_2(1+\gamma_{b,k}^{p,t}). \quad (9)$$

The ranking value for a non-requested user is retained as $v_k^{p,t+1} = v_k^{p,t}$.

(ii) After updating the ranking values in each time step, the BS sorts these values for all users in descending order to build its preference list. The BS then sends an ACK signal to the top S_b users in its updated preference list.

(iii) After the episode is terminated, each user estimates its SINR value from its associated/connected BS and sends this information to the BS. Then, each BS computes a sum rate from its connected users as $v_{s,b}^p = \sum_{k\in\mathcal{K}} \rho_{b,k}^p R_{b,k}$ where $\rho_{b,k}^p$ is the association result of episode p. The BS b then updates its initial ranking value in the next episode for user k as $v_k^{p+1,0} = \max(v_{s,b}^p, v_k^{p,T})$, where $v_k^{p,T}$ is the latest updated ranking value for user k in episode p. This update helps maintain the highest ranking value for each user over episodes.

The metric used for ranking the BS's preference is very important. Instead of using the sum rate as above, the SINR-value or Q-value sent by each agent can be mapped directly into the ranking value, $v_k^{p,t}$, to simplify the building and updating of a BS preference list. Furthermore, if we build a preference list by ranking the minimum rate of requested users instead of the sum rates, it can improve fairness performance by maximizing the minimum user rate. The construction and update of the preference list at the BS have a strong impact on the end system performance. We will show the effects of BS preference list in Section IV.

The overall distributed multi-agent DQN system is described in Algorithm 1.

Algorithm 1 Distributed MADQN for Load Balancing User Association

```
1: Initialize the replay buffer and the weights of DQN.
2: Initialize the weights of target network with $\boldsymbol{\theta}_k^- \leftarrow \boldsymbol{\theta}_k$.
3: for episode $= 1, 2, \ldots, I$ do
4:     Each agent observes its own initial state, $s_k^1$.
5:     for time step $= 1, 2, \ldots, T$ do
6:         Each agent chooses a BS index $a_k^t$ at state $s_k^t$
            using an $\epsilon$-greedy policy based on its DQN output
            and send the association request to that BS.
7:         Each BS updates its preference list, and responds
            based on its load balancing constraint.
8:         Each agent adds the experience $(s_k^t, a_k^t, r_k^t, s_k^{t+1})$
            into its replay buffer.
9:         Each agent samples a mini batch from its replay
            buffer and computes the loss function (7), then
            updates the weights of its DQN as in (8).
10:        if ACK$_k^{t+1}$=1 $\forall$ $k$ or no available BS quota then
11:            break
12:        end if
13:    end for
14:    Every $C$ steps $\boldsymbol{\theta}_k^- \leftarrow \boldsymbol{\theta}_k$
15: end for
```

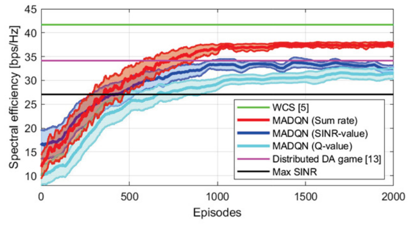
IV. Simulation Results

In this section, we simulate the performance of distributed MADQN for load balancing user association. In our simulations, we consider 5 BSs, each with a $N = 100$ element antenna array, located at (0, 0), (125, 125), (125, −125), (−125, −125), and (−125, 125), in a rectangular area of $500 \times 500m^2$, and each BS has a load limit of $S_b = 4$. The carrier frequency and bandwidth are 28GHz and 400MHz, respectively, and the path loss model for 28GHz in [12] is adopted. The transmit power for each BS is 30dBm. The DQN structure for each agent is constructed with three hidden layers with (64, 32, 32) hidden nodes, respectively. Rectified linear unit (ReLU) activation function is applied in each hidden layer. We also use a learning rate of $\eta = 10^{-5}$ and a discount factor of $\gamma = 0.9$. In each time step, the weights of DQN are updated using an Adam optimizer with a mini batch of size 64. Furthermore, we use the following benchmarks for comparison.

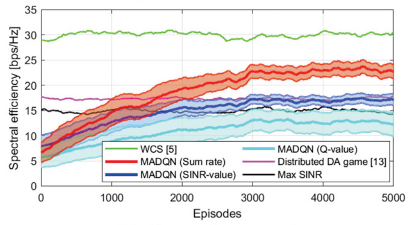
- *Worst connection swapping (WCS) [5]:* Successively swapping the worst BS-UE connection with another one to provide the UE a stronger link and improve the sum rate.
- *Distributed deferred acceptance (DA) game [13]:* A non-learning scheme based on the matching game where preference lists are built based on users' instantaneous rate.
- *Max SINR:* Association by selecting at each user the index of BS providing the maximum SINR.

Fig. 2 represents the performance of the proposed algorithm in both a static and a dynamic network scenario. In addition, Jakes model is adopted for the dynamic mobile network using a channel correlation between consecutive episodes as $\rho = 0.9$ ($V = 4$*km/h* and $T_s = 1ms$). As seen in Fig. 2, our proposed MADQN algorithms converge to a local optimum for the static network even as each agent trains its own DQN independently. Furthermore, the proposed algorithms show a similar and stable convergence behavior in the dynamic network even though the channels change from one episode to the next.

Fig. 2 also show that the proposed MADQN scheme with sum-rate based preference list significantly enhances the



(a) Static network



(b) Dynamic network

Fig. 2. Spectral efficiency obtained by distributed MADQN for load balancing user association with $K = 20$ users, starting from initial rest, where shaded areas indicate standard deviations. In the dynamic network, the channels are changed in each episode with a time correlation factor of 0.9.

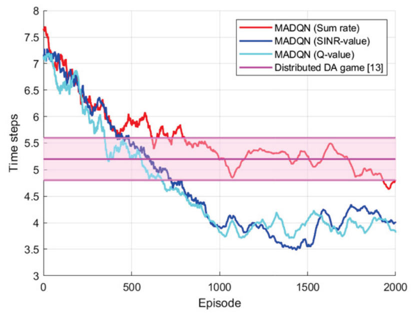


Fig. 3. Convergence behavior: The number of required time steps for user association to converge in each episode ($K = 20$).

network spectral efficiency, outperforms other schemes with preference list built on the SINR or Q-values as well as the distributed DA game, while reaching closely the performance of centralized WCS scheme. Thus, the choice of preference list strongly affects network performance. This result verifies the effectiveness of the proposed MADQN and multi-agent matching policy with sum rate based preference list.

Fig. 3 shows the required time steps in each episode for load balancing user association to converge. Note that this time step convergence indicates association delay. We confirm that the required number of time steps for convergence is monotonically reduced during the training procedure. The SINR-value and Q-value based preference list have the lowest association delay. The proposed MADQN with sum rate based preference list achieves a comparable association delay with the distributed DA game. This result validates that the proposed MADQN is fast and efficient, striking a good balance between the performance and delay.

Fig. 4 compares the spectral efficiency performance with different number of users, where $K = 20$ is critical load and $K > 20$ represents overloaded networks. Under the same BS load limits, the spectral efficiency increases as the number of users increases since the system can exploit multi-user diversity. It should be emphasized that our proposed algorithm enhances the performance even in overload scenarios. These observation

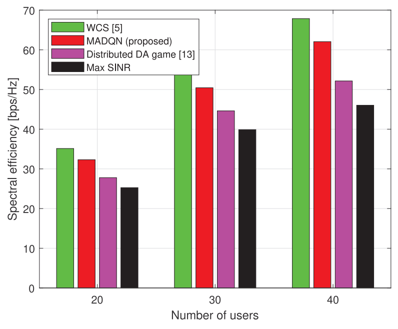


Fig. 4. Sum rate performance comparison for critical load and overload scenarios versus the number of users K, where $B = 5$ and $S_b = 4$.

confirms that the proposed distributed MADQN provides comparable performance with the centralized algorithm even though it performs in a fully distributed manner.

V. Conclusion

We proposed a distributed multi-agent DQN for load balancing user association in dense networks. To achieve fully distributed training and execution, we introduced a matching game based multi-agent policy that enhances the performance of the distributed algorithm by utilizing a sum rate based preference list at the BSs. Numerical results showed that the proposed algorithm provides comparable performance with a centralized algorithm, while outperforming conventional distributed algorithms. Furthermore, our algorithm exhibits fast convergence and high adaptivity to channel changes.

References

[1] Q. Zhang, Y.-C. Liang, and H. V. Poor, "Intelligent user association for symbiotic radio networks using deep reinforcement learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4535–4548, Jul. 2020.

[2] N. Zhao, Y.-C. Liang, D. Niyato, Y. Pei, M. Wu, and Y. Jiang, "Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 11, pp. 5141–5152, Nov. 2019.

[3] A. Alizadeh and M. Vu, "Reinforcement learning for user association and handover in mmWave-enabled networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 11, pp. 9712–9728, Nov. 2022.

[4] M. Sana, A. De Domenico, W. Yu, Y. Lostanlen, and E. C. Strinati, "Multi-agent reinforcement learning for adaptive user association in dynamic mmWave networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6520–6534, Oct. 2020.

[5] A. Alizadeh and M. Vu, "Load balancing user association in millimeter wave MIMO networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 6, pp. 2932–2945, Jun. 2019.

[6] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[7] "5G NR Physical layer measurements," 3GPP, Sophia Antipolis, France, 3GPP Rep. TS 38.215 version 15.2.0 Release 15, 2018.

[8] O. Vinyals et al., "Starcraft ii: A new challenge for reinforcement learning," 2017, *arXiv:1708.04782*.

[9] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, 2016, pp. 2094–2100.

[10] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–21.

[11] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *Amer. Math. Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[12] T. S. Rappaport, G. R. MacCartney, M. K. Samimi, and S. Sun, "Wideband Millimeter-wave propagation measurements and channel models for future wireless communication system design," *IEEE Trans. Commun.*, vol. 63, no. 9, pp. 3029–3056, Sep. 2015.

[13] A. Alizadeh and M. Vu, "Distributed user association in B5G networks using early acceptance matching game," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2428–2441, Apr. 2021.