Multi-Agent Deep Reinforcement Learning for Handover Management in Massive Industrial Internet of Things Networks

Naufan Raharya 1[1], Muhammad Suryanegara[1]
[1]Department of Electrical Engineering Universitas Indonesia
[1]Depok, West Java, Indonesia
email: naufanraharya@ui.ac.id, suryanegara@gmail.com

ARTICLE INFORMATION

Received 02 August 2023
Revised 14 September 2023
Accepted 06 November 2023

Keywords :
Collision error probability
Decoding error probability
Handover
Industrial IoT
Mult-agent deep reinforcement learning

ABSTRACT

The industrial Internet of Things (IoT) is considered one of the applications in the fifth generation (5G) networks. In this application, users' high mobility in a typical industrial scenario needs high reliability. The increased mobility creates frequent handover, creating extra control signaling to a new base station (BS). The users' congestion to the new BSs can lead to an outage. In this paper, we investigate how to manage the handover of users to improve reliability in a high-mobility scenario using deep learning. We first use an offline centralized algorithm to create labels for user association to a BS, which is done without adding a handover coefficient. Then, we train the neural network and use the introduced parameter to make the multi-agent deep reinforcement learning (RL) learn better. This is done to avoid long iterative methods in reinforcement learning. The results show that our method can outperform the offline centralized algorithm by 40% when the handover coefficient increases.

1. Introduction

Industrial Internet-of-Things (IIoT) is one of the typical application scenarios in the fifth generation (5G) cellular networks. One of the applications in the IIoT is a connected factory, where all industrial machines, sensors, and other tools are connected to a network (T. M. Fernández-Caramés & P. Fraga-Lamas, 2018). In this application, any item in the network can talk to each other. For instance, an automated guided vehicle can deliver items controlled through wireless systems, or a human worker can monitor and collect data in the IioT devices through wireless systems. With stringent latency and reliability requirements, supporting many IIoT devices in wireless networks remains an open problem. As a user moves, it may associate with different BSs to maintain its quality-of-service (QoS). The basic idea is to let users associate with a BS with a high signal-to-noise ratio (SNR) and low congestion status.On the other hand, handover causes extra communication overheads, which come from the additional control signaling to the new BS. Besides, if the new BS has a high traffic load, the newly arrived devices may not be served, resulting in a service outage. Therefore, user association (UA) policy plays a critical role in IIoT but remains a challenging task, especially when the density of devices is large (A. Gupta & R. K. Jha, 2015; D. Liu et al., 2016; Q. Ye et al., 2013).

DOI: 10.17933/bpostel.v21i1.374

Handover policies have been investigated for over 30 years (S. Tekinay & B. Jabbari, 1991). The bottleneck lies in the trade-off between computational complexity for optimizing the handover policy and the QoS experienced by users. Recently, handover policies were studied in different communication systems, such as wireless local area networks (Chen et al., 2020), ultra-dense networks with coordinate multipoint communications (W. Sun et al., 2021), satellite communications (H. Xu et al., 2020), and 5G systems (V. Yajnanarayana et al., 2020). Their results imply that with recent breakthroughs in deep learning, it is possible to achieve good QoS with low computational complexity. To get labeled training samples for supervised deep learning, a centralized optimization algorithm was proposed by (N. Raharya et al., 2021) to maximize the reliability of the worst-case user. After the offline training, the deep neural network (DNN) is broadcast to all the devices for distributed execution. Given the local state of a machine, it can select a BS according to the output of the DNN.

Notably, the centralized optimization algorithm proposed by (N. Raharya et al., 2021) maximized the reliability without taking handover overhead (HO) into account. Since HO depends on the mobility of devices, the problem turns out to be a sequential decision-making problem with high complexity, especially when the state or action space is ample. To solve such kinds of issues, deep Q-learning (DQN) has been applied in (N. Zhao et al., 2019) and (D. Guo et al., 2020). Specifically, the DQN algorithm learns from the feedback of the environment by trial and error, where the parameters of the DQN are initialized with random parameters. Improving the training efficiency of DQN remains an open problem and is crucial for real-time implementation in wireless networks.

We propose an offline initialization and online training framework for DQN to address the above issue. In the offline initialization, we utilize the centralized optimization algorithm proposed by (N. Raharya et al., 2021). to train the DQN without taking HO into account. In this way, we can obtain an excellent initial DQN without random exploration. After the offline initialization, we apply online training to train the DQN in the scenario with HO, where the central server collects the experience of different devices and fine-tunes the DQN centrally. Meanwhile, the updated DQN is broadcast to all the contrivances for distributed execution.

This paper comprises five chapters: introduction, method, result and discussion, and conclusion. The method part explains the system model and the multi-agent deep reinforcement learning, and the development and discussion part presents the simulation setup and results and the analysis of the results.

2. Method

This section will explain the system model and formulation used in the paper and our multi-agent deep reinforcement learning algorithm. The system model and formulation contain the communications system model on the reliability and the objective function. Then, in the following subsection, we will explain how to implement multi-agent reinforcement learning in the system model.

2.1. System Model and Formulation

We established uplink transmissions within a cellular network, with N individual users, each using a single antenna to transmit packets to L base stations equipped with multiple antennas. All BSs are connected to a central server. Each base station (BS) has various subchannels, adhering to the 5G New Radio (NR) standard. Additionally, we presume that distinct frequency bands are allocated to individual base stations to prevent interference from adjacent channels between neighboring BSs.

2.1.1. System Model

In our model, we divide time into discrete slots, each with a duration of Ts. During each space, users can transmit a packet to a base station (BS) or refrain from doing so. Conforming to the 5G New Radio (NR) standard, Ts represents the transmission time interval and is shorter than the channel coherence time. We assume that the large-scale channel fading from the n-th BS to the n-th user remains constant within each frame but can vary due to user mobility. Figure 1 illustrates the relationship between slots and structures.

The large-scale channel fading and the distance from the *n*-th user to the *l*-th BS in the t_f-th frame are represented as $\alpha_{n,l}(t_f)$ and $d_{n,l}(t_f)$ (in meters), respectively. Their relationship is expressed as follows (Third Generation Partnership Project (3GPP), 2014)

$$10\log_{10}\alpha_{n,l}(t_f) = -34.5 - 35\log_{10}[d_{n,l}(t_f)]. \quad (1)$$

At the start of t_f-th frame, *n*-th user selects a BS based on $\alpha_{n,l}(t_f)$. We represent the user association (UA) of user *n* to BS *l* in t_f-th frame as $x_{n,l}(t_f)$, where n ranges from 1 to *N* and l ranges from 1 to *L*, mathematically expressed as follows

$$x_{n,l}(t_f) = \begin{cases} 1, & \text{if } n\text{-th user is connected to} \\ & l\text{-th BS during } t_f\text{-th frame} \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

The UA of *n*-th user to *l*-th BS can be represented by a vector, $\mathbf{x}_n(t_f) = [x_{n,1}(t_f), ..., x_{n,L}(t_f)]^T$. We consider a constraint where each user is limited to associating with only one BS, and this can be expressed as follows,

$$\sum_{l=1}^{L} x_{n,l}(t_f) \le 1, n = 1, ..., N. \quad (3)$$

1. Collision Probability

   We implement a multi-channel slotted ALOHA protocol at the MAC layer to eliminate the need for the request-and-grant procedure during uplink transmission (B. Singh et al., 2018). λ (packets/slot) represents the average packet arrival rate of the n-th user, which is assumed to be much smaller than one. During each time slot, user n chooses a subchannel and transmits a packet with a probability of λ. 5G NR employs repetition as a reliability enhancement technique, as it does not rely on acknowledgments but sends K duplicates of packets across K consecutive slots (Third Generation Partnership Project (3GPP), n.d.). In this grant-free (GF) random access (RA) process, packet collisions occur when multiple users select the same subchannel in the same slot. A packet is deemed lost if all K repetitions collide with other packets.

   The number of users associated with a BS can be given as follows

$$C_l(t_f) = \sum_{n=1}^{N} x_{n,l}(t_f). \quad (4)$$

We consider (4) as the bottleneck or congestion status of a BS. (4) shows that there is a high chance of collisions when the number of associated users is large. Then, the collision probability experienced by a user with a BS is given as follows,

$$\rho_l(t_f) = 1 - \left(\frac{e^{-K\lambda} + M_s^K - 1}{M_s^K}\right)^{C_l(t_f)-1}, \text{ .... } (5)$$

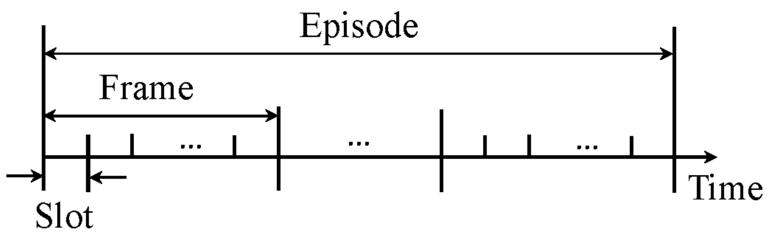

Figure 1: Slots, Frames, Episodes during the time observation.

where M_s represents the quantity of subchannels available in a BS. When user n of t_f -th frame is associated with BS l, the collision probability in (5) can be written as follows,

$$\beta_n(t_f) = \sum_{l=1}^{L} x_{n,l}(t_f)\rho_l(t_f). \text{ .... } (6)$$

2. Decoding Error Probability

Sending a short packet quickly necessitates a short block length, meaning a decoding error arises when the received packet is decoded incorrectly. We denote *Bw*, *T*s, *bp*, as the bandwidth of each subchannel, the transmission duration of each packet, and the packet size (bits), respectively. The decoding error probability user *n* of *tf*-th frame to BS *l* is given as follows (C. She et al., 2018, 2021).

$$\xi_{n,l}(t_f) = \mathrm{E}_{g_{n,l}}\left\{ f_Q\left\{ \sqrt{T_s B_w}\left[\ln\left(1 + \frac{\alpha_{n,l}(t_f) g_{n,l} P_t}{\varphi N_0 B_w}\right) - \frac{b_p \ln 2}{T_s B_w}\right]\right\}\right\}. \text{ .... } (7)$$

When user n of t_f -th frame is associated with BS l, the decoding probability in (7) is expressed as follows,

$$\epsilon_n(t_f) = \sum_{l=1}^{L} \xi_{n,l}(t_f) x_{n,l}(t_f). \text{ .... } (8)$$

3. Handover

As a user moves, it may associate with another BS, i.e., handover (HO), which boosts signal strength. However, a frequent HO may cause extra signaling that increases communications latency. We then denote *wn*(*tf*) as a variable indicating whether the associated BS of user *n* changes at frame *tf*, given as follows

$$w_n(t_f) = \sum_{l=1}^{L} \left| x_{n,l}(t_f) - x_{n,l}(t_f - 1) \right|. \text{ .... } (9)$$

Thus, if user n changes its association at frame t_f, the value of (9) is two. Otherwise, the value of (9) is zero.

4. Problem Formulation

Packet losses can result from collisions occurring at the MAC layer as well as decoding errors in the physical layer. The packet loss probability in t_f-th frame of user n is expressed as follows

$$P_n^{\text{Loss}}(t_f) = 1-[1-\grave{o}_n(t_f)][1-\beta_n(t_f)] \approx \grave{o}_n(t_f)+\beta_n(t_f), \quad (10)$$

where the approximation holds true since $\grave{o}_n(t_f)\beta_n(t_f) \square \min\{\grave{o}_n(t_f), \beta_n(t_f)\}$ when both $\grave{o}_n(t_f)$ and $\beta_n(t_f)$ are small in magnitude.

We represent the utility function of user n at t_f-th frame by $U_n(t_f)$, which is defined as follow,

$$U_n(t_f) \square -\log_{10}[\grave{o}_n(t_f)+\beta_n(t_f)] - \nu w_n(t_f), \quad (11)$$

where ν is an HO coefficient whose value is $0 \le \nu \le 1$. In this article, we want to maximize (11) for the worst-case user by solving the following problem,

$$\max_{x_{n,l}(t_f)} \quad \min\{U_1(t_f),...,U_N(t_f)\} \quad (12)$$
$$\text{s.t.} \quad (2) \text{ and } (3).$$

Problem (12) is a non-convex integer optimization problem. To solve (12), we need to consider the handover frequency of users, in which, according to Figure 1, a handover occurs when a user changes its BS connection in the subsequent frame. Here, handover management is essential in (12) when it is non-zero to prevent a frequent handover, adding latency to a user's uplink connection and resulting in a new link to a BS. In the following section, we develop a distributed and low-complexity solution based on the deep reinforcement learning algorithm.

2.2. Model-Assisted Multi-Agent Deep Reinforcement Learning

In this section, we develop a model-assisted Multi-Agent Deep Reinforcement Learning (MADRL) algorithm to solve (12). First, we explain how to formulate a MADRL solution based on Deep Q-Network (DQN). Then, we apply the formulated DQN into \eqref{objective}, divided into offline initialization and online learning. In the offline initialization, we use an offline label algorithm to initialize the deep neural network (DNN) in the DQN. Then, we use the initialized DQN as a model-assisted tool in online learning.

2.2.1. Deep Q-Network Overview

In this subsection, we formulate DQN for our problem optimization in (12). We first define the agent, environment, state, action, and reward. An agent is the decision maker that determines the UA selection at time t_f and is designated to n-th user. The environment is defined as the place where $x_{n,l}$ configurations are evaluated. Here, the environment is the objective function in (12) and it is located at the central server. The action of n-th user is to select a BS at t_f, i.e. $a_n(t_f)$, and $a_n(t_f) = 1, \cdots, L$. The selected action is also equivalent to set $x_{n,a_n(t_f)}(t_f) = 1$. We then denote $\mathbf{s}_n(t_f) \in \square^{1\times(L+1)}$ as the state of n-th agent at time t_f. The first L elements in $\mathbf{s}_n(t_f)$ consist of $\tilde{\xi}_{n,l}(t_f) \in \{\tilde{\xi}_{n,1}(t_f), \cdots, \tilde{\xi}_{n,L}(t_f)\}$ and $\tilde{\rho}_l(t_f) \in \{\tilde{\rho}_1(t_f), \cdots, \tilde{\rho}_L(t_f)\}$ that represent the normalized value of $\xi_{n,l}(t_f)$ and $\rho_l(t_f)$ at time t_f respectively. $\tilde{\xi}_{n,l}(t_f)$ and $\tilde{\rho}_{n,l}(t_f)$ are given as follows,

$$\tilde{\xi}_{n,l}(t_f) = -\log_{10}[\xi_{n,l}(t_f)]. \qquad (13)$$

$$\tilde{\rho}_l(t_f) = -\log_{10}[\rho_{n,l}(t_f - 1)]. \qquad (14)$$

The last element in $\mathbf{s}_n(t_f)$ is $\tilde{x}_n(t_f - 1) = 1, \cdots, L$ that represents the selects an action in the previous frame, i.e. $t_f - 1$. We then denote $r_n(t_f)$ as the reward of *n*-th user at time t_f, given as follows,

$$r_n(t_f) = \min\{U_1(t_f), ..., U_N(t_f)\}. \qquad (15)$$

Note that, based on (15), reward for all users is the same. Another element in DQN is experience replay D. Here, at time t_f, *n*-th user stores a transition from its interaction with the environment into D. The transition consists of state $\mathbf{s}_n(t_f)$, action $a_n(t_f)$, reward $r_n(t_f)$, and next state $\mathbf{s}_n(t_f + 1)$ and can be represented as a tuple of $e = \{s_n(t_f), a_n(t_f), r(t_f), s_n(t_f + 1)\}$.

In DQN, a deep neural network (DNN) is used to approximate the Q-value. Here, the Q-value is a function of $\mathbf{s}_n(t_f)$, $a(t_f)$, and θ, i.e. $Q(\mathbf{s}_n(t_f), a_n(t_f - 1) \mid \theta)$, where θ is the training parameter of DQN. The Q-network updates θ by minimizing the loss function given as follows,

$$loss(\theta) = \frac{1}{N_t} \sum_{e \in \mathrm{D}} \left(y_n - Q(\mathbf{s}_n(t_f), a_n(t_f - 1) \mid \theta) \right)^2, \qquad (16)$$

where N_t is the size of minibatch samples and y_n is a target of *n*-th user, given as follows,

$$y_n = r_n(t_f) + \sigma \max_a Q(\mathbf{s}_n(t_f + 1), a_n(t_f) \mid \theta_{target}), \qquad (17)$$

where σ is the discount factor and θ_{target} is the target parameter of DQN, which is updated every $\tilde{T}_f$ frames. During the learning process, a DQN can be trained by randomly sampling N_t minibatches of D to reduce the correlation among training samples, which in turn increases the stability of the algorithm (Mnih et al., 2015). Then, by using Adam optimizer as proposed by (Kingma & Ba, 2015). with learning rate 0.001, the parameters of the DQN are optimized to minimize (16). The general framework for DQN is given in Figure 2.

2.2.2. Offline Initialization and Online Learning

In this subsection, we apply the DQN algorithm to our objective function in (12). We classify the solution into offline initialization and online learning. In offline initialization, we execute offline training to initialize the DQN at the central server with the UA solution from the label training samples obtained from an offline label algorithm. The offline label algorithm is summarized as follows. When a user is positioned at the center of a cell, and far away from other cells, it should be linked with the nearest BS to minimize the probability of decoding errors. On the other hand, if a user is on the edge of the cell and its large-scale channel gain to two or more BSs is comparable, then the user should be associated with the BS with a lower traffic load, subject to the congestion status of the BS. Based on these two ideas, the offline label algorithm finds each user's highest large-scale channel gain, puts them in a list, and then sorts them in descending order. In this way, the users in the cell centre are at the top of the list, and those in the edge cell are at the bottom. Then, the algorithm will solve the association for all users according to (12) sequentially, starting from the user at the top of the list. The algorithm will select the BS with the lowest packet loss probability in each sequential step according to (11). After completing the sequential process, users located at the cell edge are redistributed among various BSs to evenly distribute the traffic load. The offline label algorithm then results on associated BS for *n*-th user at t_f represented by $\tilde{x}_n(t_f)$. The detailed explanation on the offline algorithm can be found in (N. Raharya et al., 2021). Next, *n*-th user sets $a_n(t_f) = \tilde{x}_n(t_f)$ and gets $r_n(t)$ by substituting $x_{n,a_n(t_f)}(t_f) = 1$ into (3), from which (11) and (15) are obtained. From here, *n*-th user can set $e = \{\mathbf{s}_n(t_f), a_n(t_f), r(t_f), \mathbf{s}_n(t_f + 1)\}$, which is stored in D. The training is then done once per $\tilde{T}_f$ frames to minimize loss function in (16). After several episodes, the DQN offline training is completed and the DQN parameter, θ, is updated. Note that ν in the offline initialization is set to zero to reduce the complexity from the sequential process in the offline label algorithm. Figure 3 describes the offline initialization process.

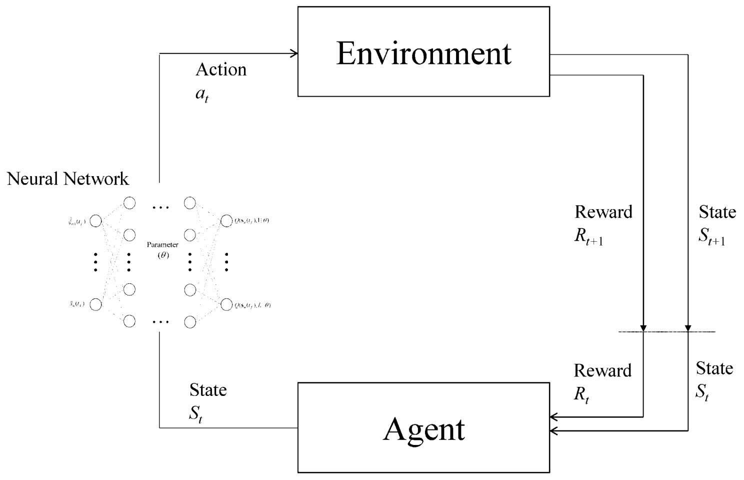

Figure 2: DQN framework.

In the online learning, we tune the initialized DQN with handover coefficient, i.e. $\nu \neq 0$. Here, the UA decision is done distributively at each user by taking an action according to τ-greedy policy, which is given as follows

$$a_n(t_f) = \begin{cases} \text{argmax} Q(\mathbf{s}_n(t_f), a_n(t_f - 1) \mid \theta_{target}), \\ \text{with probability } 1 - \tau \\ a_n(t_f - 1), \text{ with probability } \tau. \end{cases} \quad \text{.....(18)}$$

The selected action translates into $x_{n,a_n(t_f)}(t_f)=1$, from which the environment gives reward $r_n(t_f)$. In addition, the environment also gives next state feedback, $\mathbf{s}_n(t_f+1)$, to each user and sends $e=\{\mathbf{s}_n(t_f), a_n(t_f), r(t_f), \mathbf{s}_n(t_f+1)\}$ to $\mathcal{D}$. Then, the central server randomly samples N_t minibatch of samples, trains the DQN parameter θ, and updates target DQN parameter, θ_{target}, i.e. $\theta_{target}=\theta$ every $\tilde{T}_f$ frames. θ_{target} is shared among users to execute an action. The online algorithm process is shown in Figure 4.

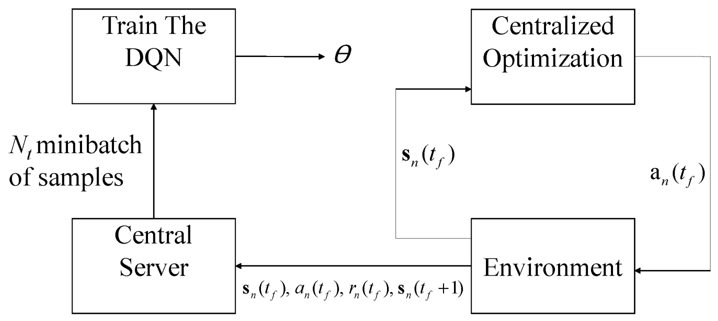

Figure 3: Offline Initialization.

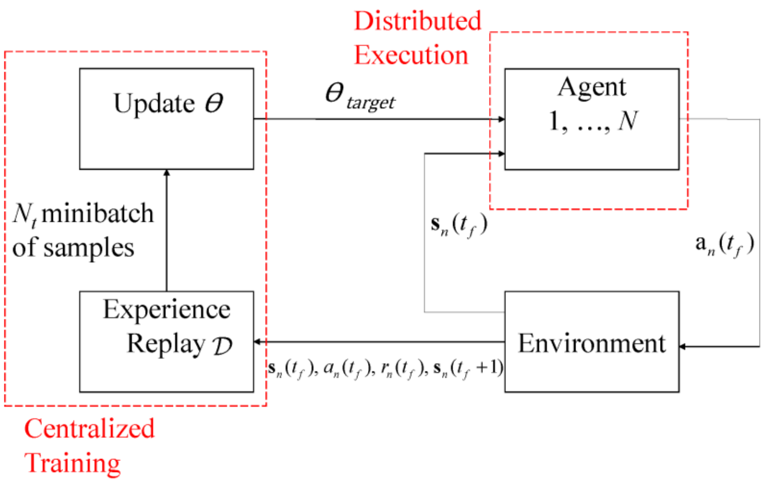

Figure 4: Online Learning.

2.2.3. Computational Complexity and Communication Overheads

In the offline label algorithm of the initialization, we need to evaluate $U_n(t_f)$, NL times in each frame. Consequently, the computational complexity can be defined as $\mathrm{O}(NL)$. To implement the offline label algorithm, Both BSs and users need to transmit $\alpha_{n,l}, n=1,\dots,N, l=1,\dots,L$ to the central server. After gaining the solution, the central server dispatches $x_{n,l}, n=1,\dots,N, l=1,\dots,L,$ to the BSs and users. Hence, the communication overheads are corresponding to NL. The complete explanation on this algorithm can be found in (N. Raharya et al., 2021). In the online learning, the centralized server sends the DQN parameter, θ_{target}, to all users. Then, each user takes an action according to the τ-greedy given θ_{target}. Subsequently, within each frame, the BSs transmit information regarding their congestion status to the users. Consequently, the communication overhead in each frame remains consistent at *L*.

In terms of computational complexity, the training stage in the offline initialization and online learning is much higher than the offline label algorithm. However, the training phase is done offline at the central server in the offline initialization stage, meaning that it is not required to execute training in each frame. The training stage in the online learning is also done at the central server and the shared parameter, θ_{target}, is only updated once per $\tilde{T}_f$ frames. Thus, the online learning does not need to update the θ_{target} in each frame. In addition, the online learning is distributively executed. We can conclude that the proposed distributed DQN has low complexity.

2.2.4. Structure of the DQN

The DQN has *C* layers, including (*C*-2) hidden layers, one input layer, and one output layer. The number of neurons in the *c*-th layer is denoted by J_c. The DQN parameter is defined as $\theta = \{\mathbf{W}^{[c]}, \mathbf{b}^{[c]}, c = 1, \cdots, C\}$, where $\mathbf{W}^{[c]} \in \mathbb{R}^{J_c \times J_{c-1}}$ and $\mathbf{b}^{[c]} \in \mathbb{R}^{J_c \times 1}$ are the weights and biases in *c*-th layer, respectively. The relationship between the input, $\mathbf{v}^{[c]} \in \mathbb{R}^{J_{c-1} \times 1}$, and the output, $\mathbf{u}^{[c]} \in \mathbb{R}^{J_c \times 1}$, of the *c*-th layer is given as follows,

$$\mathbf{u}^{[c]} = f_\delta^{[c]}(\mathbf{W}^{[c]}\mathbf{v}^{[c]} + \mathbf{b}^{[c]}), \quad (19)$$

where $f_\delta^{[c]}(\cdot)$ is the Rectified Linear Unit (ReLU) activation function of the *v*-th layer, i.e.,

$$f_\delta^{[c]}(\mathbf{y}) = \max(0, \mathbf{y}). \quad (20)$$

The input layer of the DQN is $\mathbf{s}_n(t_f)$ and the output layer is a *Q*-value vector, i.e. $\{Q(\mathbf{s}_n(t_f), 1 | \theta), \cdots, Q(\mathbf{s}_n(t_f), L | \theta)\}$. *n*-th user will associate to $\tilde{l}$-th BS, where

$$\tilde{l} \triangleq \operatorname{argmax}_l \{Q(\mathbf{s}_n(t_f), 1 | \theta), \cdots, Q(\mathbf{s}_n(t_f), L | \theta)\}. \quad (21)$$

Figure 5 shows our DQN structure.

3. Result and Discussion

In this section, we assess the performance of the proposed DQN algorithm and create a performance comparison between it and several methods, which are the highest signal-to-noise ratio (SNR) policy, offline label algorithm, and distributed Deep Supervised Learning (DSL). The distributed DSL is a scheme in (N. Raharya et al. 2021).

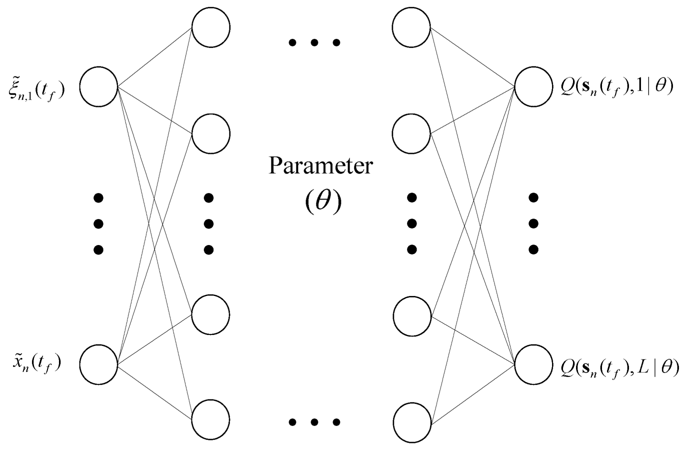
$\tilde{\xi}_{n,1}(t_f)$
$\tilde{x}_n(t_f)$
Parameter
(θ)
$Q(\mathbf{s}_n(t_f),1\,|\,\theta)$
$Q(\mathbf{s}_n(t_f),L\,|\,\theta)$

Figure 5: Structure of the DQN

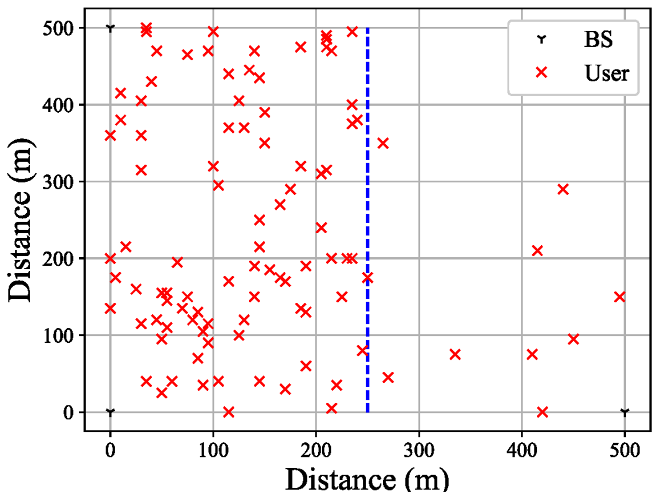

Figure 6: Locations of BSs and users.

3.1. Simulation Setup

We examine a square area measuring 500 meters on each side, where four base stations (BSs) are positioned at the corners, as depicted in Figure 6. This figure illustrates the division of the area into a left-hand side and a right-hand side region. We apply different user densities to these two regions. Specifically, 90% of the users are uniformly distributed in the left-hand side region, while the remaining 10% are in the right-hand side region. Each frame corresponds to one second. Users remain stationary within each frame but change their positions randomly in subsequent frames at a velocity of 15 meters per frame. Consequently, the UA configuration may or may not change with each frame. We assume the square area wraps around the edges to mitigate boundary effects.

We specify two parameters in the simulation; the radio environment parameter in Table 1, which defines the setting for wireless communication, and the DQN hyperparameter in Table 2, which defines the setting for the DQN algorithm. The number of episodes and frames in both offline initialization and online learning is 8 episodes and 100 frames. In the online learning, the value of τ is updated in each frame of an episode and then it is reset in the new episode. τ in the next frame is updated as $\tau(t_f+1)=\tau(t_f)\times\tau_{decay}$$. The value of τ stops updated until $\tau=0.001$. By updating τ, we can balance exploration and exploitation.

To create a realistic scenario, we create a numerical computation and users' moving dynamics in codes run on a programming language such as Python. The users can move randomly in one of the four directions, north, south, west, and east, within a specified number of frames and episodes. Then, the large-scale channel gain of each user can be obtained. Next, we can compute (12) and solve it using a DQN method.

Table 1. Radio Environment Parameters.

Parameters	Value
T_S	0.25 ms
B_W	180 kHz
Pt	0.2 Watt
N_0	-174 dBm/Hz
Bp	32×8 bits
Φ	1.1
M_S	20
N	100
L	4

Table 2. DQN Hyperparameter.

Parameter	Value
Minibatch Size N_t (Samples)	100
Discount Factor σ	0.95
Experience Replay D size	20000
Minimum value of τ	0.001
τ_{decay}	0.995
Initial value of τ	1
Number of hidden layers	2
Number of neurons at a hidden layer	128
T_f (frames)	10

3.2. Performance Evaluation

To evaluate the reliability of online learning in the DQN scheme, we compare it with three other baselines. The first one is the highest SNR policy, which is widely used in the existing wireless networks. The second one is the offline label algorithm that is used to obtain labeled training samples. The third one is distributed Deep Supervised Learning (DSL), which uses the UA results from the offline label algorithm as labels to train the DNN. We set the input and output dimensions of the DNN in the distributed DSL to be similar to the input and output of the DQN. The cumulative distribution functions (CDFs) of packet loss probability experienced by the worst-case user are provided in Figures 6, 7, and 8.

The results in Figure 7 show that when there is no penalty for doing a handover, i.e., the CDF achieved by the DQN is similar to the distributed DSL. In addition, both are very close to the Offline Label Algorithm, with a gap of around 10%. This means that distributed DSL and DQN can accurately approximate the offline label algorithm. On the other hand, the highest SNR policy is the worst since it only considers the closest BS, which results in heavy congestion in a particular BS.

The results in Figure 8 show that the packet loss probability of the DQN algorithm is approximately 28% lower than that of the distributed DSL. This happens because the labels used to train the distributed DSL from the offline label algorithm are less optimal; as we can see in the figure, the packet loss probability of the DQN algorithm is 5% lower than that of the offline label algorithm. The figure also shows that the highest SNR policy has the worst packet loss probability since the congestion in BSs is not balanced, and the loss from handover is not well addressed.

Figure 9 shows that the packet loss probability of all schemes generally increases compared to the previous results because there is a higher penalty from handover. The figure also shows that the DQN algorithm performs far better than other schemes, as shown by its large packet loss probability gap compared to others. In detail, the gap between the offline label algorithm and the DQN algorithm increases to 40% from the previous result. This means that the online learning capability in the DQN learning can adapt more to the dynamic environment.

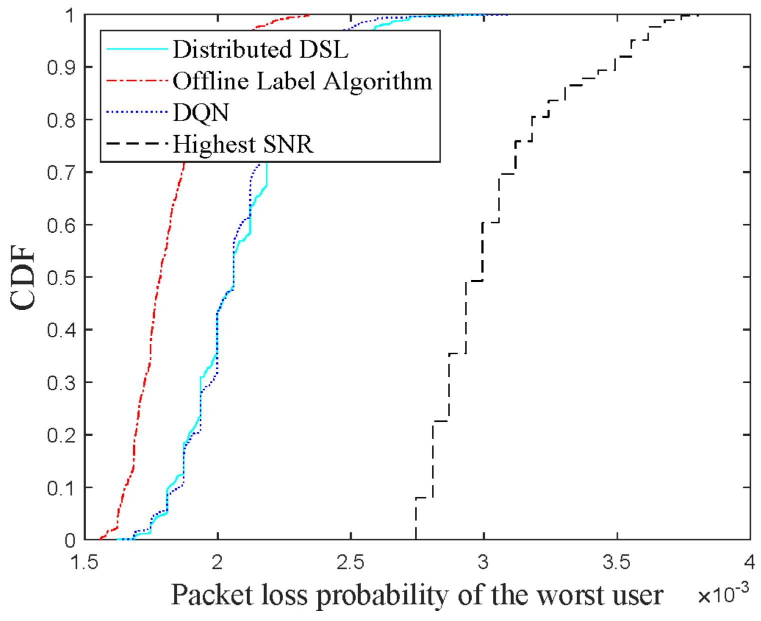

Figure 7: CDF of packet loss probability of the worst-case user, where $\nu = 0$.

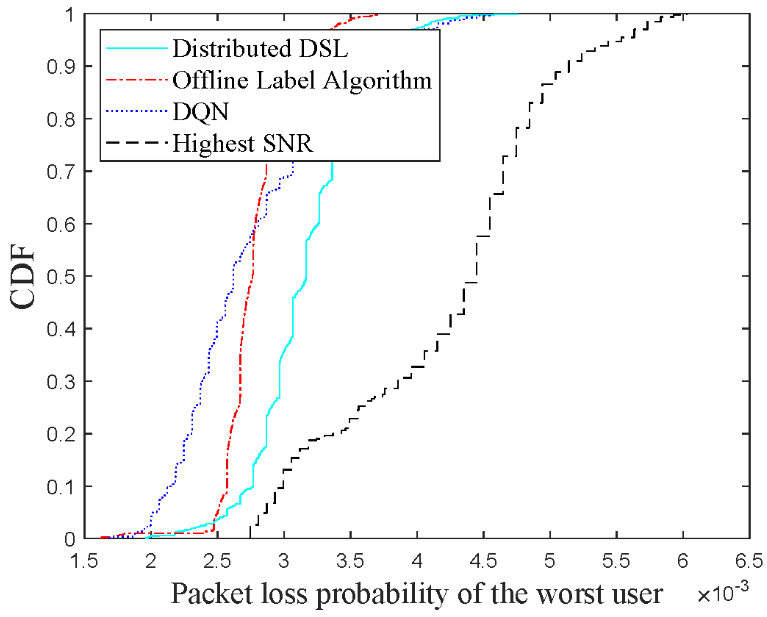

Figure 8. CDF of packet loss probability of the worst-case user, where $\nu = 0.1$.

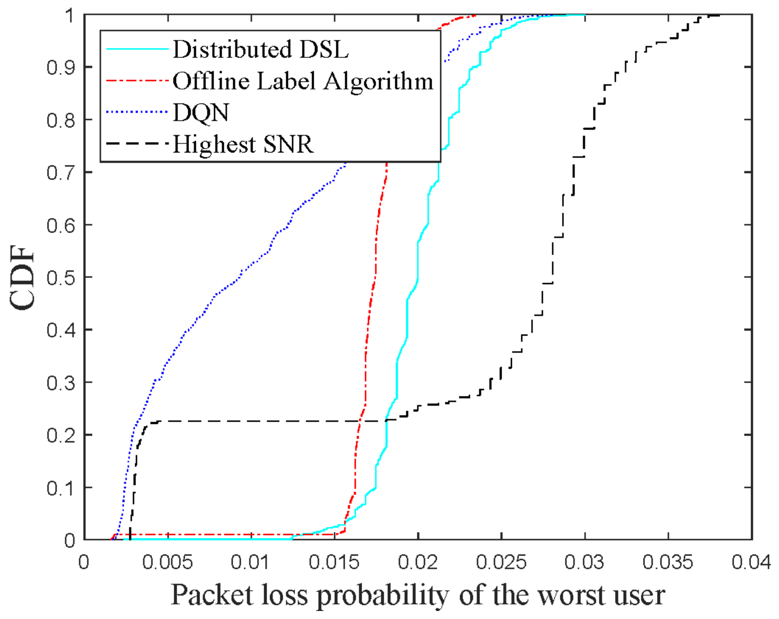

Figure 9. CDF of packet loss probability of the worst-case user, where $\nu = 0.5$

To summarize the results in Figure 7-9, we create a comparison table on the average packet loss probability between existing methods and the proposed method as shown in Table 3.

Table 3: Average Packet Loss Probability Comparison between Existing Methods and The Proposed DQN Method.

	$\nu = 0$	$\nu = 0.1$	$\nu = 0.5$
Distributed DSL	2×10^{-3}	3×10^{-3}	0.02
Offline Label Algorithm	1.5×10^{-3}	2.6×10^{-3}	0.016
DQN	1.5×10^{-3}	2.5×10^{-3}	0.009
Highest SNR	3×10^{-3}	4.5×10^{-3}	0.027

4. Conclusion

This paper investigates how to manage a handover in massive IIoT networks to improve packet transmission reliability by optimizing a UA policy. We propose a DQN-based solution to determine the associated BS for users in a dynamic environment. The DQN algorithm comprises an offline initialization and online learning. The offline initialization is utilized to train the DQN without the HO scenario, and it is done to obtain an excellent initial DQN without random exploration. Then, in the online learning, we fine-tune the DQN in a centralized manner and let each user execute the UA policy. The results show we can achieve a low packet loss probability when the HO coefficient is high compared to the offline label algorithm, distributed DSL, and highest SNR policy. Furthermore, the research in this area can be expanded into the joint resource allocation and user association problem.

5. Acknowledgements

This paper is based on the original doctoral thesis work by the first author (Raharya, 2021). Part of this journal was presented in the 2021 IEEE Wireless Communications and Networking Conference (WCNC).

References

A. Gupta & R. K. Jha. (2015). A Survey of 5G Network: Architecture and Emerging Technologies. *IEEE Access*, *3*, 1206–1232. https://doi.org/10.1109/ACCESS.2015.2461602

B. Singh, O. Tirkkonen, Z. Li, & M. A. Uusitalo. (2018). Contention-Based Access for Ultra-Reliable Low Latency Uplink Transmissions. *IEEE Wireless Communications Letters*, *7*(2), 182–185. https://doi.org/10.1109/LWC.2017.2763594

C. She, C. Sun, Z. Gu, Y. Li, C. Yang, H. V. Poor, & B. Vucetic. (2021). A Tutorial on Ultrareliable and Low-Latency Communications in 6G: Integrating Domain Knowledge Into Deep Learning. *Proceedings of the IEEE*, *109*(3), 204–246. https://doi.org/10.1109/JPROC.2021.3053601

C. She, C. Yang, & T. Q. S. Quek. (2018). Cross-Layer Optimization for Ultra-Reliable and Low-Latency Radio Access Networks. *IEEE Transactions on Wireless Communications*, *17*(1), 127–141. https://doi.org/10.1109/TWC.2017.2762684

Chen, Z., Luo, Z., Duan, X., & Zhang, L. (2020). Terminal handover in software-defined WLANs. *EURASIP Journal on Wireless Communications and Networking*, *2020*(1), 68. https://doi.org/10.1186/s13638-020-01681-w

D. Guo, L. Tang, X. Zhang, & Y. -C. Liang. (2020). Joint Optimization of Handover Control and Power Allocation Based on Multi-Agent Deep Reinforcement Learning. *IEEE Transactions on Vehicular Technology*, *69*(11), 13124–13138. https://doi.org/10.1109/TVT.2020.3020400

D. Liu, L. Wang, Y. Chen, M. Elkashlan, K. -K. Wong, R. Schober, & L. Hanzo. (2016). User Association in 5G Networks: A Survey and an Outlook. *IEEE Communications Surveys & Tutorials*, *18*(2), 1018–1044. https://doi.org/10.1109/COMST.2016.2516538

H. Xu, D. Li, M. Liu, G. Han, W. Huang, & C. Xu. (2020). QoE-Driven Intelligent Handover for User-Centric Mobile Satellite Networks. *IEEE Transactions on Vehicular Technology*, *69*(9), 10127–10139. https://doi.org/10.1109/TVT.2020.3000908

Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. http://arxiv.org/abs/1412.6980

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

N. Raharya, C. She, W. Hardjawana, & B. Vucetic. (2021). Deep Learning for Distributed User Association in Massive Industrial IoT Networks. *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, 1–6. https://doi.org/10.1109/WCNC49053.2021.9417303

N. Zhao, Y. -C. Liang, D. Niyato, Y. Pei, M. Wu, & Y. Jiang. (2019). Deep Reinforcement Learning for User Association and Resource Allocation in Heterogeneous Cellular Networks. *IEEE Transactions on Wireless Communications*, *18*(11), 5141–5152. https://doi.org/10.1109/TWC.2019.2933417

Q. Ye, B. Rong, Y. Chen, M. Al-Shalash, C. Caramanis, & J. G. Andrews. (2013). User Association for Load Balancing in Heterogeneous Cellular Networks. *IEEE Transactions on Wireless Communications*, *12*(6), 2706–2716. https://doi.org/10.1109/TWC.2013.040413.120676

Raharya, N. (2021). Machine Learning for Massive Connections in Wireless Networks. In *Machine Learning for Massive Connections in Wireless Networks*. https://hdl.handle.net/2123/25863

S. Tekinay & B. Jabbari. (1991). Handover and channel assignment in mobile cellular networks. *IEEE Communications Magazine*, *29*(11), 42–46. https://doi.org/10.1109/35.109664

T. M. Fernández-Caramés & P. Fraga-Lamas. (2018). A Review on Human-Centered IoT-Connected Smart Labels for the Industry 4.0. *IEEE Access*, *6*, 25939–25957. https://doi.org/10.1109/ACCESS.2018.2833501

Third Generation Partnership Project (3GPP). (n.d.). *Study on new radio (NR) access technology; physical layer aspects (release 14) TR 38.802 V.2.0.0.* (Standard TR 38.802 V.2.0.0.).

Third Generation Partnership Project (3GPP). (2014). *Spatial channel model for Multiple Input Multiple Output (MIMO) simulations TR 25.996 V.12.0.0* [Standard].

V. Yajnanarayana, H. Rydén, & L. Hévizi. (2020). 5G Handover using Reinforcement Learning. *2020 IEEE 3rd 5G World Forum (5GWF)*, 349–354. https://doi.org/10.1109/5GWF49715.2020.9221072

W. Sun, L. Wang, J. Liu, N. Kato, & Y. Zhang. (2021). Movement Aware CoMP Handover in Heterogeneous Ultra-Dense Networks. *IEEE Transactions on Communications*, *69*(1), 340–352. https://doi.org/10.1109/TCOMM.2020.3019388