Optimizing Wireless Resource Management and Synchronization in Digital Twin Networks

Hanzhi Yu, Yuchen Liu, *Member IEEE*, Zhaohui Yang, *Member IEEE*, Haijian Sun, *Member IEEE*, and Mingzhe Chen, *Senior Member IEEE*

***Abstract*—In this paper, we investigate an accurate synchronization between a physical network and its digital network twin (DNT), which serves as a virtual representation of the physical network. The considered network includes a set of base stations (BSs) that must allocate its limited spectrum resources to serve a set of users while also transmitting its partially observed physical network information to a cloud server to generate the DNT. Since the DNT can predict the physical network status based on its historical status, the BSs may not need to send their physical network information at each time slot, allowing them to conserve spectrum resources to serve the users. However, if the DNT does not receive the physical network information of the BSs over a large time period, the DNT's accuracy in representing the physical network may degrade. To this end, each BS must decide when to send the physical network information to the cloud server to update the DNT, while also determining the spectrum resource allocation policy for both DNT synchronization and serving the users. We formulate this resource allocation task as an optimization problem, aiming to maximize the total data rate of all users while minimizing the asynchronization between the physical network and the DNT. The formulated problem is challenging to solve by traditional optimization methods, as each BS can only observe a partial physical network, making it difficult to find an optimal spectrum allocation strategy for the entire network. To address this problem, we propose a method based on the gated recurrent units (GRUs) and the value decomposition network (VDN). The GRU component allows the DNT to predict future status using the historical data, effectively updating itself when the BSs do not transmit the physical network information. The VDN algorithm enables each BS to learn the relationship between its local observation and the team reward of all BSs, allowing it to collaborate with others in determining whether to transmit physical network information and optimizing spectrum allocation. Simulation results show that our GRU and VDN based algorithm improves the weighted sum of data rates and the similarity between the status of the DNT and the physical network by up to 28.96%, compared to a baseline method combining GRU with the independent Q learning (IQL).**

***Index Terms*—Resources allocation, digital network twin, gate recurrent units, value decomposition network.**

Hanzhi Yu and Mingzhe Chen are with the Department of Electrical and Computer Engineering and Frost Institute for Data Science and Computing, University of Miami, Coral Gables, FL 33146 USA (Emails: hanzhiyu@miami.edu; mingzhe.chen@miami.edu).

Yuchen Liu is with the Department of Computer Science, North Carolina State University, Raleigh, NC 27695 USA (Email: yuchen.liu@ncsu.edu).

Zhaohui Yang is with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China (Email: yang_zhaohui@zju.edu.cn).

Haijian Sun is with the School of Electrical and Computer Engineering, University of Georgia, Athens, GA 30602 USA (Email: hsun@uga.edu).

This work was supported by the U.S. National Science Foundation under Grants CNS-2350076, CNS-2312139, CNS–2312138, and SaTC–2350075.

I. Introduction

Digital twin (DT) is a virtual representation of a physical product or process, used to understand and predict the physical counterpart's performance characteristics [1]–[3]. Different from those traditional simulation tools, which use computer-based models or mathematical concepts to test systems, processes, and the effects of various variables, a DT utilizes real-time data from its associated physical object for simulations, analysis, and online control. The two-way information flow improves the performance of the predictive analytical model [4]. Based on the definition of the DT, the concept of the DT network (DTN), or the digital network twin (DNT), is generated to describe a network that constructed by multiple one-to-one DTs. A DNT uses advanced communication technologies to realize information sharing not only between each physical object and its twin, but also among different physical objects and among different twins [1], [5]. According to previous works, the creation of a DNT presents several challenges. First, constructing a DNT requires mapping not only physical objects but also several unique networking factors (i.e., network protocols, wireless channel dynamics, and the network performance metrics). Hence, it is impractical to directly map all network features for DNT generation and one must select appropriate network features for DNT creations [6]. Second, there is no standardized metrics exist for evaluating the DNT synchronization [7]. Third, privacy protection in DTNs is a crucial issue, as they are vulnerable to data breaches and malicious attacks during information exchange, necessitating robust security measures to protect users' private information. [1], [8].

A number of existing works [8]–[11] have studied the generation and creations of DNTs. In particular, the authors in [8] presented several use cases, the design standardization, and an implementation example of the DNT. In [9], the authors designed a deep neural network (DNN) based method to create a DNT for the approximation of the optimal user association scheme in a mobile edge computing network. The work in [10] created a DNT using a Bayesian model. In [11], the authors proposed a DT empowered framework for optimizing network resource management, where DTs collect real-time data from users to predict and dynamically allocate network resources, in order to enhance efficiency of resource utilization and reduce reconfiguration costs. However, most of these works [8]–[11] did not consider the similarity and synchronization between the DNT and a physical network, and assumed that the DNT is already synchronized with a physical network without

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2025.3543126

considering how the DNT collects physical network data to achieve synchronization.

To address this issue, a number of existing studies [12]–[15] have focused on the DT synchronization optimization. In particular, the authors in [12] proposed a continual learning framework to build a synchronized DNT of a autonomous vehicle network so as to make vehicle driving decisions. In [13], the authors proposed a DT-empowered framework for resource allocation in UAV-assisted edge mobile networks, leveraging a deviation model to address discrepancies between the DT and physical states. In [14], the authors aimed to optimize data synchronization in vehicular DT networks by developing a deep learning model that predicts relay waiting times. In [15], the authors designed a dynamic hierarchical synchronization framework for IoT-assisted DTs in the Metaverse, and the framework optimizes synchronization intensities through a multi-level game-theoretic approach. However, these works [12]–[15] did not consider how the generation of DNTs affect the physical network performance since DNT generations require the transmission of a large amount of data, which will also introduce significant communication overhead.

The main contribution of this work is to design a novel DNT framework that jointly optimizes the performance of a physical network and the synchronization between the DNT and the physical network. Our key contributions include:

- We consider a DNT enabled network that consists of a physical network with a set of base stations (BSs), several users, and a DNT. The DNT is a virtual representation of the physical network and can predict physical network dynamics. Each BS must use limited spectrum resources to serve the users and transmit the information of the physical network to a cloud server to generate the DNT. Since the DNT can predict the physical network status, the BSs may not need to transmit the information to the server at every time slot, and thus conserving spectrum resources to better serve the users. To this end, each BS in the physical network needs to determine whether to transmit the physical network information to the cloud server for updating the DNT, and optimize spectrum resource allocation for the users and physical network information transmission. We formulate this problem as an optimization problem aiming to maximize the data rates of all users while minimizing the gap between the states of the physical network and the DNT.
- The formulated problem is challenging to solve since each BS cannot fully observe the entire status of the physical network. To solve this problem, we proposed a machine learning (ML) method that integrates the gate recurrent units (GRUs) [16], [17] and the value-decomposition based reinforcement learning (VD-RL) method [18]. The GRUs allow the DNT to predict its future status using historical data and to update its status when the DNT cannot receive the information of the physical network. The VD-RL method can learn from the GRU prediction accuracy to enable each BS to decide its associated users, whether to send the physical network information to the cloud server, and RB allocation, optimizing the data rate of all users in the physical network and ensuring an accurate synchronization between the physical network and the DNT. Compared to other RL methods [19], the VD-RL method allows each BS to use its partial observation, specifically, the locations of the users in its coverage, to collaboratively find a globally optimal solution for both physical network and DNT.

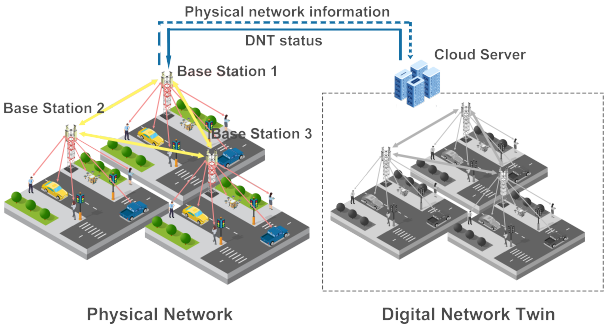

Fig. 1. The considered DNT enabled network.

Simulation results show that our proposed method improves the weighted sum of data rates and the the similarity between the status of the DNT and the physical network by up to 28.96%, compared to a baseline method combining GRU with the independent Q learning (IQL).

The rest of this paper is organized as follows. The system model and problem formulation are introduced in Section II. The design of the GRU and VD-RL integrated method will be introduced in Section III. The analysis of the complexity, the convergence, and the implementation of the designed method are studied in Section IV. Simulation results are presented and discussed in Section V. Finally, conclusions are drawn in Section VI.

II. System Model and Problem Formulation

We consider a DNT enabled network which consists of: 1) a physical network including a set $\mathcal{U}$ of U mobile users and a set $\mathcal{M}$ of M BSs, and 2) a DNT that is generated and controlled by a cloud server, as shown in Fig. 1. Each BS can provide communication service to the users that are located in its coverage. Meanwhile, each BS must transmit the physical network information collected from its associated users to the cloud server for the generation of the DNT. We first introduce the mobility model of each user. Next, we introduce the transmission model. Then, we describe the DNT model. Finally, we present our considered optimization problem. Table I provides a summary of the notations used hereinafter.

A. Mobility Model

For simplicity, we use a random walk model to describe the mobility of each user [20]. At each time slot t, each user u can choose from five possible movements: 1) stay at the current location, 2) move forward, 3) move backward, 4) move left, and 5) move right. The probability of each user u choosing

TABLE I
LIST OF NOTATIONS

Notation	Description	Notation	Description
N	Number of RBs of each BS	U	Number of users
$\mathcal{N}$	Set of RBs	$\mathcal{U}$	Set of users
$\boldsymbol{p}_u$	Probability vector of user u chooses each possible movement	$\boldsymbol{l}^{\mathrm{U}}_{u,t}$	The position of user u at time slot t
c_{umt}	Data rate of BS m transmitting data to user u at time slot t	Δl	Distance a user can move in one time slot
d_{umt}	Distance between user u and BS m at time slot t	$\boldsymbol{x}_{umt}, \boldsymbol{y}_{mt}$	RB allocation vector
$I^{\mathrm{U}}_{umt,n}, I^{\mathrm{C}}_{mt,n}$	Interference caused by other BSs that use RB n	B	Bandwidth of each RB
c^{C}_{mt}	Data rate of BS m transmitting data to the cloud server	P	Transmission power
$h_m\left(\boldsymbol{l}^{\mathrm{U}}_{u,t}\right)$	Channel gain between user u and BS m at time slot t	o_u	Rayleygh fading parameter
$h_m\left(\boldsymbol{l}^{\mathrm{C}}\right)$	Channel gain between BS m and the cloud server	$\boldsymbol{l}^{\mathrm{C}}$	The position of the cloud server
D_m	The size of data needed to be transmitted to the cloud server	$\boldsymbol{l}^{\mathrm{M}}_m$	The position of BS m
T_{mt}	Transmission delay of BS m transmitting data to the cloud server	$\boldsymbol{s}_t$	Physical network status at time slot t
$\boldsymbol{s}^m_t$	Partial physical network status observed by BS m	$\overline{\boldsymbol{s}}^m_t$	A mapping of $\boldsymbol{s}^m_t$
$\hat{\boldsymbol{s}}^m_t$	The status of BS m estimated by the DNT	ϵ	Weight parameter
$\boldsymbol{h}_t$	Hidden states of the GRU at time slot t	α	Threshold of the transmission delay
$\boldsymbol{W}^{\mathrm{r}}, \boldsymbol{W}^{\mathrm{h}}, \boldsymbol{W}^{\mathrm{z}}, \boldsymbol{W}^{\mathrm{o}}$	Weight matrix of the GRU	$\boldsymbol{U}^{\mathrm{r}}, \boldsymbol{U}^{\mathrm{h}}, \boldsymbol{U}^{z}$	Weight matrix of the GRU
N^{h}	Number of units in the hidden layer of the GRU	$\boldsymbol{r}^{\mathrm{G}}_t$	The reset gate of the GRU
K	Length of the GRU input sequence	$\tilde{\boldsymbol{h}}_\tau$	Candidate hidden state
$\boldsymbol{z}_{mt}$	Association users vector of BS m at time slot t	$\boldsymbol{z}^{\mathrm{G}}_\tau$	The update date of the GRU
λ_G	Learning rate of the GRU	$\boldsymbol{a}^m_t$	The action of BS m at time slot t
$\boldsymbol{a}_t$	The joint action of all BSs at time slot t	$\boldsymbol{\pi}^m\left(\boldsymbol{a}^m_t \vert \boldsymbol{s}^m_t\right)$	The policy of BS m
ξ_{ut}	Number of BSs that serve user u at time slot t	$r\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)$	The team reward
ρ	Penalty for one user obtaining RBs from multiple BSs	$\mathcal{G}^m$	Bipartite graph of BS m
$\mathcal{U}^m$	Set of vertices represent the associated users of BS m	$\mathcal{N}^m$	Set of vertices represent the RBs of BS m
$\mathcal{E}^m$	Set of edges that connect vertices from $\mathcal{U}^m$ and $\mathcal{N}^m$	e^m_{un}	The edge in $\mathcal{E}^m$
$\mathcal{X}^m_t$	A subset of $\mathcal{E}^m$	ψ^m_{un}	The weight of edge e^m_{un}
$\boldsymbol{\theta}_m$	The parameters of BS m's Q network	$\tilde{\boldsymbol{\theta}}_m$	The parameters of BS m's target network
$\mathcal{D}_g$	Set of transitions used to train the Q networks in epoch g	G	Number of the Q networks training epoch
D	Transitions collected in one training epoch	λ_Q	Learning rate of the Q networks
θ^{h}	Size of the hiddent states of the Q networks		

each possible movement is $\boldsymbol{p}_u = [p_{u,1}, p_{u,2}, p_{u,3}, p_{u,4}, p_{u,5}]$. We assume that the position of user u at time slot t is $\boldsymbol{l}^{\mathrm{U}}_{u,t} = \left[l^{\mathrm{U}}_{u,t,1}, l^{\mathrm{U}}_{u,t,2}\right]$. Thus, the position of user u at time slot $t+1$ is

$$\boldsymbol{l}^{\mathrm{U}}_{u,t+1} = \begin{cases} \left[l^{\mathrm{U}}_{u,t,1}, l^{\mathrm{U}}_{u,t,2}\right] & \text{with probability } p_{u,1}, \\ \left[l^{\mathrm{U}}_{u,t,1}, l^{\mathrm{U}}_{u,t,2} + \Delta l\right] & \text{with probability } p_{u,2}, \\ \left[l^{\mathrm{U}}_{u,t,1}, l^{\mathrm{U}}_{u,t,2} - \Delta l\right] & \text{with probability } p_{u,3}, \\ \left[l^{\mathrm{U}}_{u,t,1} - \Delta l, l^{\mathrm{U}}_{u,t,2}\right] & \text{with probability } p_{u,4}, \\ \left[l^{\mathrm{U}}_{u,t,1} + \Delta l, l^{\mathrm{U}}_{u,t,2}\right] & \text{with probability } p_{u,5}, \end{cases} \quad (1)$$

with Δl being the distance each user can move in one time slot.

B. Transmission Model

In our considered network, each BS m utilizes an orthogonal frequency division multiple access (OFDMA) technique to serve its associated users and transmit the information of the physical network over a set $\mathcal{N}$ of N resource blocks (RBs). Each user can only be served by one BS and each BS will only allocate one RB to a user at each time slot. The transmission rate of BS m transmitting data to user u at time slot t is

$$\begin{aligned} &c_{umt}\left(\boldsymbol{x}_{umt}, \boldsymbol{X}_{(-m)t}, \boldsymbol{Y}_{(-m)t}\right) \\ &\quad = \sum_{n=1}^{N} x_{umt,n} B \log_2\left(1 + \frac{P h_m\left(\boldsymbol{l}^{\mathrm{U}}_{u,t}\right)}{I^{\mathrm{U}}_{umt,n} + BN_0}\right), \end{aligned} \quad (2)$$

where $\boldsymbol{X}_{(-m)t} = \left[\boldsymbol{x}_{ujt}\right]_{u\in\mathcal{U}, j\in\mathcal{M}, j\neq m}$, $\boldsymbol{x}_{umt} = [x_{umt,1}, ..., x_{umt,N}]$ is an RB allocation vector, with $x_{umt,n} \in \{0,1\}$ indicating whether RB n of the BS m is allocated to user u at time slot t; $\boldsymbol{Y}_{(-m)t} = \left[\boldsymbol{y}_{1t}, \ldots, \boldsymbol{y}_{(m-1)t}, \boldsymbol{y}_{(m+1)t}, \ldots, \boldsymbol{y}_{Mt}\right]$ with $\boldsymbol{y}_{mt} = [y_{mt,1}, ..., y_{mt,N}]$ being the RB allocation vector of BS m for the physical network information transmission, with $y_{mt,n} \in \{0,1\}$ indicating whether RB n of BS m is allocated for the physical network information transmission; B is the bandwidth of each RB; P is the transmit power; $h_m\left(\boldsymbol{l}^{\mathrm{U}}_{u,t}\right) = o_u d^{-2}_{umt}$ is the channel gain between user u and BS m, with o_u being the Rayleigh fading parameter, $d_{umt} = \sqrt{\|\boldsymbol{l}^{\mathrm{U}}_{u,t} - \boldsymbol{l}^{\mathrm{M}}_m\|_2}$ being the distance between user u and BS m at time slot t, $\boldsymbol{l}^{\mathrm{M}}_m$ being the position of BS m; N_0 is the noise power spectral density; $h_j\left(\boldsymbol{l}^{\mathrm{C}}\right)$ is the channel gain between BS j and the cloud server, with $\boldsymbol{l}^{\mathrm{C}}$ being the position of the cloud server; and $I^{\mathrm{U}}_{umt,n}$ is the interference caused by other BSs that use RB n for serving users and physical network information transmissions, which is given by

$$\begin{aligned} I^{\mathrm{U}}_{umt,n} = \sum_{i\in\mathcal{U}, i\neq u} \sum_{j\in\mathcal{M}, j\neq m} &\left(x_{ijt,n} P h_j\left(\boldsymbol{l}^{\mathrm{U}}_{u,t}\right)\right. \\ &\left. + y_{jt,n} P h_j\left(\boldsymbol{l}^{\mathrm{U}}_{u,t}\right)\right). \end{aligned} \quad (3)$$

Similarly, the transmission rate of BS m transmitting physical network information to the cloud server at time slot t is

$$c_{mt}^{\mathrm{C}}\left(\boldsymbol{y}_{mt}, \boldsymbol{X}_{(-m)t}, \boldsymbol{Y}_{(-m)t}\right) = \sum_{n=1}^{N} y_{mt,n} B \log_2\left(1+\frac{Ph_m\left(\boldsymbol{l}^{\mathrm{C}}\right)}{I_{mt,n}^{\mathrm{C}}+BN_0}\right), \tag{4}$$

where $I_{mt,n}^{\mathrm{C}}$ is the interference caused by other BSs that use RB n for serving users and physical network information transmission, which is given by

$$I_{mt,n}^{\mathrm{C}} = \sum_{i\in\mathcal{U}}\sum_{j\in\mathcal{M},j\neq m}\left(x_{ijt,n}Ph_j\left(\boldsymbol{l}^{\mathrm{C}}\right)+y_{jt,n}Ph_j\left(\boldsymbol{l}^{\mathrm{C}}\right)\right). \tag{5}$$

We assume that the data size of the physical network information that is needed to be transmitted to the cloud server is D_m. Given the data rate c_{mt}^{C}, the transmission delay of BS m transmitting physical network information to the cloud server at time slot t can be represented as

$$T_{mt} = \frac{D_m}{c_{mt}^{\mathrm{C}}\left(\boldsymbol{y}_{mt}, \boldsymbol{X}_{(-m)t}, \boldsymbol{Y}_{(-m)t}\right)}. \tag{6}$$

C. Digital Network Twin Model

As a virtual representation of the physical network, the DNT should have the same status and the network management strategy with the physical network, even when some BSs do not transmit the physical network information. We first define a vector $\boldsymbol{s}_t$ to represent the status of the physical network at time slot t. In our considered network, the positions of all U users are used to describe the status of the physical network. Since each BS has its service area, it can only serve a subset $\mathcal{U}_t^m$ of users that are located in its coverage. Thus, the entire physical network status $\boldsymbol{s}_t$ is partially observed by each BS m. We use $\boldsymbol{s}_t^m = \left[\boldsymbol{l}_{u,t}^{\mathrm{U}}\right]_{u\in\mathcal{U}_t^m}$ to denote physical network status observed by BS m at time slot t. Then, we have $\boldsymbol{s}_t = \left[\boldsymbol{s}_t^1, ..., \boldsymbol{s}_t^M\right]$. Accordingly, the DNT status at time slot t can be defined as $\overline{\boldsymbol{s}}_t = \left[\overline{\boldsymbol{s}}_t^1, ..., \overline{\boldsymbol{s}}_t^M\right]$, with $\overline{\boldsymbol{s}}_t^m$ being a mapping of the physical network status $\boldsymbol{s}_t^m$. Given $\boldsymbol{y}_{mt}$ and $\boldsymbol{s}_t^m$, $\overline{\boldsymbol{s}}_t^m$ is given by

$$\overline{\boldsymbol{s}}_t^m\left(\boldsymbol{y}_{mt}\right) = \begin{cases} \boldsymbol{s}_t^m, & \text{if } \sum_{n=1}^{N} y_{mt,n} = 1, \\ \hat{\boldsymbol{s}}_t^m & \text{if } \sum_{n=1}^{N} y_{mt,n} = 0, \end{cases} \tag{7}$$

where $\hat{\boldsymbol{s}}_t^m = \left[\hat{\boldsymbol{l}}_{u,t}^{\mathrm{U}}\right]_{u\in\mathcal{U}_t^m}$ is the status estimated by the DNT at time slot t. From (7), we can see that if each BS m allocates an RB to transmit its partially observed physical network information to the cloud server (i.e., $\sum_{n=1}^{N} y_{mt,n} = 1$), the status of the DNT and the status of the physical network is similar, since the DNT can obtain the physical network status from BS m directly. Otherwise, if BS m does not allocate any RBs to the cloud server, the DNT must estimate the physical network status of BS m. When the physical network status and the DNT status are identical (i.e., $\boldsymbol{s}_t = \overline{\boldsymbol{s}}_t$), we consider the DNT synchronizes with the physical network.

D. Problem Formulation

Given the our designed system model, we next describe our considered optimization problem. Our goal is to maximize the sum of the data rates of all U users, while guaranteeing the synchronization between the DNT and the physical network over a set $\mathcal{T}$ of T time slots. The optimization problem includes optimizing the RB allocation matrix $\boldsymbol{X}_t$ and physical network information transmission matrix $\boldsymbol{y}_t$. The optimization problem can be formulated as

$$\max_{\{\boldsymbol{X}_t,\boldsymbol{Y}_t\}_{t\in\mathcal{T}}} \sum_{t=1}^{T}\left(-\frac{1-\epsilon}{U}\|\boldsymbol{s}_t-\overline{\boldsymbol{s}}_t\|_2^2 + \epsilon\sum_{m=1}^{M}\sum_{u=1}^{U} c_{umt}\left(\boldsymbol{x}_{umt}, \boldsymbol{X}_{(-m)t}, \boldsymbol{Y}_{(-m)t}\right)\right), \tag{8}$$

$$\text{s.t. } x_{umt,n}\in\{0,1\}, m\in\mathcal{M}, n\in\mathcal{N}, u\in\mathcal{U}, t\in\mathcal{T}, \tag{8a}$$

$$\sum_{m=1}^{M}\sum_{n=1}^{N} x_{umt,n}\leq 1, u\in\mathcal{U}, t\in\mathcal{T}, \tag{8b}$$

$$\sum_{u=1}^{U} x_{umt,n}\leq 1, m\in\mathcal{M}, n\in\mathcal{N}, t\in\mathcal{T}, \tag{8c}$$

$$y_{mt,n}\in\{0,1\}, m\in\mathcal{M}, n\in\mathcal{N}, t\in\mathcal{T}, \tag{8d}$$

$$\sum_{n=1}^{N} y_{mt,n}\leq 1, m\in\mathcal{M}, t\in\mathcal{T}, \tag{8e}$$

$$\sum_{u=1}^{U} x_{umt,n}+y_{mt,n}\leq 1, m\in\mathcal{M}, n\in\mathcal{N}, t\in\mathcal{T}, \tag{8f}$$

$$T_{mt}\leq\alpha, m\in\mathcal{M}, t\in\mathcal{T}, \tag{8g}$$

where $\epsilon\in(0,1)$ is a weight parameter, and α is a threshold of the transmission delay of each BS transmitting the physical network information to the cloud server. In problem (8), constraints (8a) and (8b) imply that each user can be served by only one BS and can occupy only one RB of this BS. Constraints (8a) and (8c) implies that each RB can be occupied by only one user. Constraints (8d) and (8e) imply that each BS m can allocate only one RB for physical network information transmission. (8f) implies that each RB n of BS m can be used either for serving a user or for physical network information transmission. Constraint (8g) is a constraint of the delay of each BS m transmitting physical network information to the cloud server.

Problem (8) is challenging to solve by conventional optimization methods due to the following reasons. First, each BS m cannot fully observe the entire physical network status $\boldsymbol{s}_t$, making it difficult to find the RB allocation policy which optimizes the data rates of all users and the DNT synchronization using conventional optimization methods [21]. Second, each BS cannot know the future DNT status as it depends on the observations of the physical network, the decisions of physical network information transmission by other

BSs, as well as the estimation accuracy of the DNT. Third, our considered system is dynamic due to user movements. Hence, the status of the physical network and the DNT will change over time. This makes it complicated to solve problem (8) using conventional optimization methods because we need to resolve the problem when the physical network statuses change. To solve problem (8), we propose a value decomposition based MARL method which enables each BS to find its optimal RB allocation policy and synchronization policy, while considering the RB allocation and synchronization policy of others, thus optimizing the data rate of all users in the physical network and keeping an accurate synchronization between the physical network and the DNT.

III. Solution of Problem

To solve problem (8), we next introduce our proposed ML based method that combines GRUs with the value decomposition network (VDN). In our designed method, the GRU is implemented by the DNT to estimate the status of the physical network. The VDN is used by each BS to determine its associated users, as well as allocating RBs to serve the associated users and to transmit physical network information to the cloud server. Meanwhile, the use of GRUs enables the designed method to use historical user mobility data for future user movement prediction without the reliance on user mobility models. Compared to traditional recurrent neural networks (RNNs) methods for user mobility prediction, the GRUs can effectively fix the gradient vanishing problem thus improving prediction accuracy and efficiency. Compared to traditional multi-agent reinforcement learning (MARL) methods (i.e., IQL [19]), the VDN allows each BS to decide its associated users and whether to transmit the physical network information based on its partial observation, but optimize the performance of the entire system (i.e., the sum of data rates of all users, and the synchronization between the DNT and the physical network). Next, we first introduce the GRU for physical network status estimation. Then, we introduce the components of the VDN based MARL framework. Finally, we explain the procedure of using our proposed method to solve problem (8).

A. GRUs for Physical Network Status Estimation

We first introduce the GRUs for physical network status estimation. The GRU-based estimation model is deployed at the cloud server, and its output is an estimated status of the physical network of all BSs. Accurate status estimation enables the DNT to simulate the physical network, even when the BSs do not transmit their partially observed physical network information to the cloud server. The GRU-based estimation model consists of three components: 1) input, 2) output, and 3) the GRU model, which are introduced as follows.

1) Input: The input of the GRU-based estimation model consists of the most recent K statuses of the entire DNT, which can be represented as $\overline{\boldsymbol{S}}_t = [\overline{\boldsymbol{s}}_{t-K+1}, ..., \overline{\boldsymbol{s}}_t]$.

2) Output: The output of the GRU-based estimation model is the estimated status $\hat{\boldsymbol{s}}_{t+1}$.

3) The GRU Model: The GRU model is used to approximate the relationship between the input $\overline{\boldsymbol{S}}_t$ and the output $\hat{\boldsymbol{s}}_{t+1}$. A GRU model consists of an input layer, a hidden layer, and an output layer. The hidden states $\boldsymbol{h}_t$ of the units in the hidden layer at time slot t are used to store the information related to the previous states from time slots $t-K+1$ to t. At each time slot τ from $t-K+1$ to t, the hidden states $\boldsymbol{h}_\tau$ is determined by the reset gate $\boldsymbol{r}_\tau^{\mathrm{G}}$ and the update gate $\boldsymbol{z}_\tau^{\mathrm{G}}$. The reset gate $\boldsymbol{r}_\tau^{\mathrm{G}}$ determines how much of the previous hidden states $\boldsymbol{h}_{\tau-1}$ should influence the new candidate state, effectively allowing the model to retain relevant portions of past information. The reset gate at time slot τ is

$$\boldsymbol{r}_\tau^{\mathrm{G}} = \phi\left(\boldsymbol{W}^{\mathrm{r}}\overline{\boldsymbol{s}}_\tau + \boldsymbol{U}^{\mathrm{r}}\boldsymbol{h}_{\tau-1}\right), \tag{9}$$

where $\phi(\cdot)$ is the sigmoid function, and $\boldsymbol{W}^{\mathrm{r}} \in \mathbb{R}^{N^{\mathrm{h}}\times 2U}$ and $\boldsymbol{U}^{\mathrm{r}} \in \mathbb{R}^{N^{\mathrm{h}}\times N^{\mathrm{h}}}$ are the weight matrices of the reset gate with N^{h} being the number of the units in the hidden layer. Given the reset gate $\boldsymbol{r}_\tau^{\mathrm{G}}$, the new candidate hidden state $\tilde{\boldsymbol{h}}_\tau$ is

$$\tilde{\boldsymbol{h}}_\tau = \tanh\left(\boldsymbol{W}^{\tilde{h}}\overline{\boldsymbol{s}}_\tau + \boldsymbol{U}^{\tilde{h}}\left(\boldsymbol{h}_{\tau-1}\odot \boldsymbol{r}_\tau^{\mathrm{G}}\right)\right), \tag{10}$$

where $\tanh(\cdot)$ is the hyperbolic tangent function, $\boldsymbol{W}^{\tilde{h}} \in \mathbb{R}^{N^{\mathrm{h}}\times 2U}$ and $\boldsymbol{U}^{\mathrm{r}} \in \mathbb{R}^{N^{\mathrm{h}}\times N^{\mathrm{h}}}$ are the weight matrices of the hidden states, and $\odot$ is an element-wise multiplication.
The update gate $\boldsymbol{z}_\tau^{\mathrm{G}}$ controls how much of the past information is retained and how much of the new input is incorporated into the current hidden states. The update of $\boldsymbol{z}_\tau^{\mathrm{G}}$ is given by

$$\boldsymbol{z}_\tau^{\mathrm{G}} = \phi\left(\boldsymbol{W}^{\mathrm{z}}\overline{\boldsymbol{s}}_\tau + \boldsymbol{U}^{\mathrm{z}}\boldsymbol{h}_{\tau-1}\right), \tag{11}$$

where $\boldsymbol{W}^{\mathrm{z}} \in \mathbb{R}^{N^{\mathrm{h}}\times 2U}$ and $\boldsymbol{U}^{\mathrm{z}} \in \mathbb{R}^{N^{\mathrm{h}}\times N^{\mathrm{h}}}$ are the weight matrices of the update gate. Given (9), (10), and (11), the hidden state $\boldsymbol{h}_\tau$ can be updated by

$$\boldsymbol{h}_\tau = \left(\boldsymbol{1} - \boldsymbol{z}_\tau^{\mathrm{G}}\right)\odot \tilde{\boldsymbol{h}}_\tau + \boldsymbol{z}_\tau^{\mathrm{G}}\odot \boldsymbol{h}_{\tau-1}. \tag{12}$$

By Given the hidden state $\boldsymbol{h}_t$, the output of the GRU model estimates the state of the physical network at time slot $t+1$ as follows:

$$\hat{\boldsymbol{s}}_{t+1} = \boldsymbol{W}^{\mathrm{o}}\boldsymbol{h}_t, \tag{13}$$

where $\boldsymbol{W}^{\mathrm{o}}$ is the output weight matrix.

B. GRU Training

The GRU model is used to approximate the relationship between the input $\overline{\boldsymbol{S}}_t$ and the output $\hat{\boldsymbol{s}}_{t+1}$. Hence, the loss function of the GRU-based estimation model is given by

$$J_G = \frac{1}{2U}\|\hat{\boldsymbol{s}}_{t+1} - \boldsymbol{s}_{t+1}\|^2. \tag{14}$$

To train our proposed GRU-based estimation model, we use a mini-batch stochastic gradient decent (SGD) method to minimize (14). The update rule of the parameter matrices in the GRU model is defined as

$$\begin{aligned}\boldsymbol{W}^i &\leftarrow \boldsymbol{W}^i - \lambda_G\nabla_{\boldsymbol{W}^i}J_G,\\ \boldsymbol{U}^j &\leftarrow \boldsymbol{U}^j - \lambda_G\nabla_{\boldsymbol{U}^j}J_G,\end{aligned} \tag{15}$$

where $i \in \{\mathrm{r}, \mathrm{z}, \tilde{h}, \mathrm{o}\}$, $j \in \{\mathrm{r}, \mathrm{z}, \tilde{h}\}$, λ_G is the learning rate, and $\nabla_{\boldsymbol{W}^i}J_G$ and $\nabla_{\boldsymbol{U}^j}J_G$ are the gradient of the loss function with respect to $\boldsymbol{W}^i$ and $\boldsymbol{U}^j$. [22]

C. Components of the VDN Algorithm

Given the status predicted by the DNT, we use a VDN based method to solve problem (8). Next, we first introduce the components of the VDN algorithm. The VDN algorithm consists of seven components: 1) agents, 2) states, 3) actions, 4) policy, 5) reward function, 6) local Q function, and 7) global Q function, which are introduced as follows [23]:

1) Agents: The agents are M BSs. Each BS implements an unique deep Q network (DQN), observes a part of the physical network, and made decisions for user-BS association and whether to transmit the partially observed physical network information to the cloud server. The DQNs of BSs are not controlled by any central nodes.

2) States: The local state of each BS describes the positions of users within its service area, while the global state of all BSs describes the positions of all users in the physical network. Hence, we use $\boldsymbol{s}_t^m$ to denote the local state of BS m at time slot t, and $\boldsymbol{s}_t$ to denote the global state.

3) Actions: For each BS m, the action describes: 1) whether BS m transmits its partially observed physical network information to the cloud server at time slot t and which RB will be used for the transmission, and 2) the subset of users that BS m serves. Here, 1) is expressed by $|\boldsymbol{y}_{mt}|$, and 2) can be expressed by a vector $\boldsymbol{z}_{mt} = [z_{mt,1}, ..., z_{mt,U}]$ with $z_{mt,u} \in \{0, 1\}$ indicating whether BS m serves user u. Hence, the action of BS m at time slot t is $\boldsymbol{a}_t^m = [|\boldsymbol{y}_{mt}|, \boldsymbol{z}_{mt}]$, $\boldsymbol{a}_t^m \in \mathcal{A}$, where $\mathcal{A}$ is the action space. The joint action of all BSs at time slot t is $\boldsymbol{a}_t = \left[\boldsymbol{a}_t^1, ..., \boldsymbol{a}_t^M\right]$.

4) Policy: The policy $\boldsymbol{\pi}^m\left(\boldsymbol{a}_t^m | \boldsymbol{s}_t^m\right)$ of each BS m is the conditional probability that BS m chooses action $\boldsymbol{a}_t^m$ given its current state $\boldsymbol{s}_t^m$.

5) Reward Function: The team reward $r\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)$ captures the total reward of all BSs taking a joint action $\boldsymbol{a}_t$ under a global state $\boldsymbol{s}_t$. The team reward function of all BSs at each time slot t is the weighted sum of: 1) the sum of data rates of all users, and 2) the similarity between the physical network status $\boldsymbol{s}_t$ and the DNT status $\overline{\boldsymbol{s}}_t$, which can be expressed by:

$$r\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right) = \begin{cases} -\dfrac{1-\epsilon}{U}\|\boldsymbol{s}_t - \overline{\boldsymbol{s}}_t\|_2^2 + \displaystyle\sum_{m=1}^{M}\sum_{u=1}^{U} \epsilon c_{umt}, \\ \qquad\qquad\qquad\qquad \text{if } \forall u, \xi_{ut} = 1, \\ \displaystyle\sum_{u=1}^{U} \mathbb{1}_{\{\xi_{ut}>1\}} \xi_{ut} \rho, \qquad \text{otherwise.} \end{cases} \tag{16}$$

where $\xi_{ut} = \sum_{m=1}^{M} z_{mt,u}$ is the total number of BSs that serve user u at time slot t, and $\rho < 0$ is a penalty for one user obtaining RBs from multiple BSs. From (16), we see that the sum of data rates of all users depends on the physical network information transmissions. The sum of data rates of all users cannot directly obtained using the joint action $\boldsymbol{a}_t$ since $\boldsymbol{a}_t^m$ only determines the users connect to BS m (i.e., $\boldsymbol{z}_{mt}$) but does not consider how to allocate RBs to these users (i.e., $\boldsymbol{x}_{umt}$). Next, we introduce how to calculate the team reward of all BSs given an action $\boldsymbol{a}_t$. In particular, given an action $\boldsymbol{a}_t$, problem (8) can be divided into M suboptimization problems that can be solved by each BS iteratively. Hence, the suboptimization of BS m is given by

$$\max_{\substack{\{\boldsymbol{x}_{ijt}\}_{(i,j)\in\mathcal{Z}_t^m, t\in\mathcal{T}} \\ \{\boldsymbol{y}_{kt}\}_{k\in\{1,\ldots,m\}}}} \sum_{t=1}^{T} \Bigg(-\frac{1-\epsilon}{U}\|\boldsymbol{s}_t^m - \overline{\boldsymbol{s}}_t^m\|_2^2 + \sum_{u\in\mathcal{U}_t^m} \epsilon c_{umt}\left(\boldsymbol{x}_{umt}, [\boldsymbol{x}_{ijt}], [\boldsymbol{y}_{kt}]\right) \Bigg), \tag{17}$$

$$\text{s.t. } x_{umt,n} \in \{0,1\}, n \in \mathcal{N}, t \in \mathcal{T}, \tag{17a}$$

$$\sum_{n=1}^{N} x_{umt,n} \le 1, u \in \mathcal{U}_t^m, t \in \mathcal{T}, \tag{17b}$$

$$\sum_{u\in\mathcal{U}_t^m} x_{umt,n} \le 1, t \in \mathcal{T}, \tag{17c}$$

$$\sum_{u\in\mathcal{U}_t^m} x_{umt,n} + y_{mt,n} \le 1, n \in \mathcal{N}, t \in \mathcal{T}. \tag{17d}$$

where $\mathcal{Z}_t^m = \{(i,j) \mid z_{jt,i} = 1, j < m\}$. Since the associated users $\boldsymbol{z}_{mt}$ of BS m have determined, only the RB allocation vectors $\boldsymbol{x}_{umt}$ need to be optimized. In (17), since the objective function (17) is linear, the constraints are non-linear, and the optimization variables are integers, we can use an iterative Hungarian algorithm [24] to solve the suboptimization problems for all BSs. Compared to standard convex optimization algorithms, using the Hungarian algorithm to solve problem (17) does not require to calculate gradients of each variables nor dynamically adjusting the step size for convergence. To implement Hungarian algorithm for finding the optimal RB allocation for all BSs, each BS must transform its suboptimization problem (17) into a bipartite graph matching problem. Hence, a bipartite graph $\mathcal{G}^m = \langle \mathcal{U}^m \times \mathcal{N}^m, \mathcal{E}^m \rangle$ must be constructed for each BS m, where $\mathcal{U}^m$ is the set of vertices represent the users associated with BS m, $\mathcal{N}^m$ is the set of vertices represent the RBs that are not occupied by physical network information transmission such that can be allocated to each associated user of BS m, and $\mathcal{E}^m$ is the set of edges that connect vertices from $\mathcal{U}^m$ and $\mathcal{N}^m$. Let $e_{un}^m \in \mathcal{E}^m$ be the edge connecting vertex u in $\mathcal{U}^m$ and vertex n in $\mathcal{N}^m$ with $e_{un}^m \in \{0,1\}$, where $e_{un}^m = 1$ indicates that BS m allocates RB n to user u (i.e., $x_{umt,n} = 1$), and $e_{un}^m = 0$ otherwise. Let $\mathcal{X}_t^m$ be a subset of $\mathcal{E}^m$ at time slot t, in which two edges can neither share a common vertex in $\mathcal{N}^m$, nor in $\mathcal{U}^m$. Therefore, in $\mathcal{X}_t^m$, each RB n can only be allocated to one associated user (constraint (8c) and (8f) are satisfied), and each user can only occupy one RB (constraint (17b) is satisfied). The weight of edge e_{un}^m is

$$\psi_{un}^m = \begin{cases} c_{umt}, & \text{if } e_{ut}^m = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

Given the formulated bipartite matching problem of BS m, we can find an optimal RB allocation scheme for BS m when the RB allocation schemes of other BSs are given. Hence, to optimize the RB allocation schemes of all BSs, one must iteratively update the RB allocation scheme of each BS

Algorithm 1 The Process of Using the Hungarian algorithm to find RB allocation schemes for all BSs

for each environment step t **do**
  if $m = 1$ **then**
    BS 1:
    Use its Q network to determine $\boldsymbol{a}_t^1$.
    Based on $\boldsymbol{a}_t^1$, use the Hungarian algorithm to determine the optimal $\mathcal{X}_t^{1*}$.
    Use $\mathcal{X}_t^{1*}$ to find the optimal RB allocation policy $[\boldsymbol{x}_{u1t}]_{u\in\mathcal{U}_t^1}$.
    Send $[\boldsymbol{x}_{u1t}]_{u\in\mathcal{U}_t^1}$ to BS 2.
  else
    for $m = 2 \to M$ **do**
      BS m**:**
      Use its Q network to determine $\boldsymbol{a}_t^m$.
      Receive $[\boldsymbol{x}_{ijt}]_{i,j\in\mathcal{Z}^m}$ from BS $m-1$.
      Based on $\boldsymbol{a}_t^m$ and $[\boldsymbol{x}_{ijt}]_{i,j\in\mathcal{Z}_t^m}$, use the Hungarian algorithm to determine the optimal $\mathcal{X}_t^{m*}$.
      Use $\mathcal{X}_t^{m*}$ to find the optimal RB allocation policy $[\boldsymbol{x}_{umt}]_{u\in\mathcal{U}_t^m}$.
      Send $[\boldsymbol{x}_{ijt}]_{i,j\in\mathcal{Z}_t^{m+1}}$ to BS $m+1$.
    end for
  end if
end for

using the Hungarian algorithm [22]. Algorithm 1 summarizes the procedure of using the Hungarian algorithm to find RB allocation schemes for all BSs.

6) Local Q Function: The local Q function $Q_m\left(\boldsymbol{s}_t^m, \boldsymbol{a}_t^m\right)$ of BS m is used to estimate the expected reward of the BS taking action $\boldsymbol{a}_t^m$ under a local state $\boldsymbol{s}_t^m$. Each BS uses a GRU, referred to as the Q network, parameterized by $\boldsymbol{\theta}_m$, to approximate its Q function. The output of the Q network represents the Q values under a given state and different actions. Therefore, the local Q function approximated by the Q network with parameters $\boldsymbol{\theta}_m$ is expressed as $Q_{\boldsymbol{\theta}_m}\left(\boldsymbol{s}_t^m, \boldsymbol{a}_t^m\right)$.

7) Global Q Function: The global Q function $Q\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)$ is defined as a Q function that estimates the reward of all BSs taking action $\boldsymbol{a}_t$ under a global state $\boldsymbol{s}_t$. In our algorithm, we cannot obtain the global Q function since we do not have a centralized DQN to approximate the global Q function. However, if we want the BSs to maximize the team reward, we need to use global Q value to update the DQN model of each BS. During the training process, we will introduce an approximate a global Q function $Q_{tot}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)$ that is estimated using local Q values.

D. Training of the VDN

Next, we introduce the training process of the VDN algorithm. Since we consider a MARL and each BS will not share its reward, actions, and states with other BSs, we cannot obtain the values of the global Q values. However, we need the global Q function to search for polices that optimize $r\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)$. To obtain the global Q values, we assume that a global Q value can be additively decomposed into local Q values of all BSs, which is given by

$$Q_{tot}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right) = \sum_{m=1}^{M} Q_{\boldsymbol{\theta}_m}\left(\boldsymbol{s}_t^m, \boldsymbol{a}_t^m\right). \tag{19}$$

Using (19), each BS can update their Q networks in a distributed manner since the BSs do not need to know $Q\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)$, we introduce two key techniques to improve the training process of the designed MARL method. First, we use a neural network $\tilde{\boldsymbol{\theta}}_m$ whose structure is similar to that of Q network $\boldsymbol{\theta}_m$, to approximate the target Q function $\tilde{Q}_{\tilde{\boldsymbol{\theta}}_m}\left(\boldsymbol{s}_t^m, \boldsymbol{a}_t^m\right)$ so as to define the loss function. Different from $\boldsymbol{\theta}_m$, the target Q network $\tilde{\boldsymbol{\theta}}_m$ updates and hence, the update speed of $\tilde{\boldsymbol{\theta}}_m$ is much slower compared to that of $\boldsymbol{\theta}_m$. For instance, the Q network $\boldsymbol{\theta}_m$ is updated at each training epoch while $\tilde{\boldsymbol{\theta}}_m$ is updated every several training epochs. The target value is expressed as

$$\tilde{y}_m = r^m\left(\boldsymbol{s}_t^m, \boldsymbol{a}_t^m\right) + \gamma \max_{\boldsymbol{a}_{t+1}^m} \tilde{Q}_{\tilde{\boldsymbol{\theta}}_m}\left(\boldsymbol{s}_{t+1}^m, \boldsymbol{a}_{t+1}^m\right), \tag{20}$$

where γ is the discount factor. Similar to $Q\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)$, the target value of the global Q function is

$$\tilde{y} = r\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right) + \gamma \max_{\boldsymbol{a}_{t+1}} \sum_{m=1}^{M} \tilde{Q}_{\tilde{\boldsymbol{\theta}}_m}\left(\boldsymbol{s}_{t+1}^m, \boldsymbol{a}_{t+1}^m\right). \tag{21}$$

Here, $\tilde{y}$ will be used to calculate the training loss of our designed VDN algorithm [25]. Second, we use experience replay technique to store past states, actions, and rewards in a buffer [26]. The experience replay technique allows the model to learn from a diverse set of states, actions, and rewards sampled from the buffer, improving training stability and the performance of the model by breaking the correlation between consecutive states. To introduce the experience replay technique, we first define a transition as $\left(\boldsymbol{s}_t, \boldsymbol{a}_t, r\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right), \boldsymbol{s}_{t+1}\right)$ and assume that each BS will collect D transitions per training epoch g. These transitions are then stored in a replay memory, which also contains transitions collected from previous training epochs. In each training epoch g, a set $\mathcal{D}_g$ of transitions is sampled from the replay memory to update the local Q functions. Accordingly, the loss function of the VDN algorithm is defined as [25]

$$\begin{aligned} J_Q &= \mathbb{E}_{\left(\boldsymbol{s}_t, \boldsymbol{a}_t, r, \boldsymbol{s}_{t+1}\right)\sim\mathcal{D}_g}\Bigg[\Bigg(\Bigg(r\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right) \\ &\quad + \gamma \max_{\boldsymbol{a}_{t+1}} \sum_{m=1}^{M} \tilde{Q}_{\tilde{\boldsymbol{\theta}}_m}\left(\boldsymbol{s}_{t+1}^m, \boldsymbol{a}_{t+1}^m\right)\Bigg) - Q_{tot}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\Bigg)^2\Bigg] \\ &= \mathbb{E}_{\left(\boldsymbol{s}_t, \boldsymbol{a}_t, r, \boldsymbol{s}_{t+1}\right)\sim\mathcal{D}_g}\left[\left(\tilde{y} - Q_{tot}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right)^2\right]. \end{aligned} \tag{22}$$

Given the loss function, the parameters $\boldsymbol{\theta}_m$ of each BS m is updated by a SGD method as follows:

$$\boldsymbol{\theta}_m \leftarrow \boldsymbol{\theta}_m - \lambda_Q \nabla_{\boldsymbol{\theta}_m} J_Q, \tag{23}$$

where λ_Q is the learning rate, and $\nabla_{\boldsymbol{\theta}_m} J_Q$ is the gradient of J_Q with respect to $\boldsymbol{\theta}_m$, which is given by

$$\begin{aligned} &\nabla_{\boldsymbol{\theta}_m} J_Q \\ &= \mathbb{E}_{\left(\boldsymbol{s}_t, \boldsymbol{a}_t, r, \boldsymbol{s}_{t+1}\right)\sim\mathcal{D}_g}\left[\left(\tilde{y} - Q_{tot}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right)\nabla_{\boldsymbol{\theta}_m} Q_{tot}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right] \\ &= \mathbb{E}_{\left(\boldsymbol{s}_t, \boldsymbol{a}_t, r, \boldsymbol{s}_{t+1}\right)\sim\mathcal{D}_g}\left[\left(\tilde{y} - Q_{tot}\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)\right)\nabla_{\boldsymbol{\theta}_m} Q_{\boldsymbol{\theta}_m}\left(\boldsymbol{s}_t^m, \boldsymbol{a}_t^m\right)\right]. \end{aligned} \tag{24}$$

Algorithm 2 The Training Procedure if the VDN Algorithm

Input: Discount factor γ; learning rate λ_Q; training epoch G; environment steps per training epoch D; minibatch size $|\mathcal{D}_g|$.
Init: Q network parameters $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_M$ generated randomly.
for $g = 1 \rightarrow G$ **do**
  Data collection stage:
  for each time step t **do**
    The DNT estimates $\hat{\boldsymbol{s}}_t$ based on (9) - (13).
    for each BS $m \in \mathcal{M}$ **do**
      Record local state $\boldsymbol{s}_t^m$.
      Choose action $\boldsymbol{a}_t^m$ based on the current policy $\boldsymbol{\pi}^m\left(\boldsymbol{a}_t^m|\boldsymbol{s}_t^m\right)$.
      Determine the RB allocation scheme by using the Hungarian algorithm.
      if $|\boldsymbol{y}_{mt}| = 1$ **then**
        Synchronize $\boldsymbol{s}_t^m$ with the DNT.
      end if
    end for
    The DNT calculate $\overline{\boldsymbol{s}}_t$ based on (7).
    Each BS m receives the team reward $r\left(\boldsymbol{s}_t, \boldsymbol{a}_t\right)$ based on (16).
    Each BS m stores the transition into the reply memory.
  end for
  Training stage:
  for each BS $m \in \mathcal{M}$ in parallel **do**
    Sample a set $\mathcal{D}_g$ of transitions from the reply memory.
    for $d = 1 \rightarrow |\mathcal{D}_g|$ **do**
      Calculate the approximated global Q value $Q_{tot}\left(\boldsymbol{s}_d, \boldsymbol{a}_d\right)$ based on (19).
    end for
    Update $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_M$ based on (23).
  end for
end for

The specific training procedure of the VDN algorithm is summarized in Algorithm 2.

IV. Complexity and Convergence of the Proposed Algorithm

In this section, we analyze the complexity and the convergence of the proposed method for DNT synchronization and user data rate optimization in the physical network.

A. Complexity Analysis

We first analyze the complexity of our proposed method, which includes: 1) the DNT state predictions, 2) each BS determining its associated users and whether to send its partial observed physical network state to the DNT, and 3) RB allocation schemes. Thus, the complexity of the proposed method is determined by: 1) the complexity of the GRU-based predictive model, 2) the complexity of the Q networks of the VDN algorithm, and 3) the complexity of the Hungarian algorithm. Next, we will introduce the complexity of these components.

Using the result from [27], [28], we know that the computational complexity of the GRU-based predictive model is $\mathcal{O}\left(KN^{\mathrm{h}}\left(U+N^{\mathrm{h}}\right)\right)$, and the computational complexity of each Q network at a BS is $\mathcal{O}\left(\theta^{\mathrm{h}}\left(U+\theta^{\mathrm{h}}+|\mathcal{A}|\right)\right)$, where θ^{h} is the size of the hidden states of the Q network. Hence, the complexity of all BS's Q networks is $\mathcal{O}\left(M\theta^{\mathrm{h}}\left(U+\theta^{\mathrm{h}}+|\mathcal{A}|\right)\right)$. Since the GRU-based predictive model is trained at the cloud server, and the Q networks are trained at BSs, both possessing sufficient computational power, the training overhead of these models can be ignored [28].

We now analyze the computational complexity for each BS m using the Hungarian algorithm to determine its RB allocation scheme. First, each BS m requires $|\mathcal{U}^m|\,N$ iterations to calculate the data rate of each associated user over each RB. Then, the Hungarian algorithm updates ψ_{un}^m to find the optimal matching set $\mathcal{X}^{m*}$. The worst-case complexity of each BS m is $\mathcal{O}\left(|\mathcal{U}^m|^2\,N\right)$, and the worst-case complexity of all BSs is $\mathcal{O}\left(\sum_{m=1}^{M}|\mathcal{U}^m|^2\,N\right)$. In contrast, the best-case complexity of each BS m is $\mathcal{O}\left(|\mathcal{U}^m|\,N\right)$, leading to a best-case complexity of $\mathcal{O}\left(\sum_{m=1}^{M}|\mathcal{U}^m|\,N\right)=\mathcal{O}\left(UN\right)$ for all BSs [22].

Finally, we compare the complexity of the proposed method with the standard VDN algorithm that directly optimizes user association and RB allocation without using the Hungarian algorithm. The maximum complexity of our proposed method is

$$\mathcal{O}\left(KN^{\mathrm{h}}\left(U+N^{\mathrm{h}}\right)+M\theta^{\mathrm{h}}\left(U+\theta^{\mathrm{h}}+|\mathcal{A}|\right)+\sum_{m=1}^{M}|\mathcal{U}^m|^2\,N\right). \tag{25}$$

In contrast, the complexity of the standard VDN method is

$$\mathcal{O}\Big(KN^{\mathrm{h}}\left(U+N^{\mathrm{h}}\right)+M\theta^{\mathrm{h}}\left(U+\theta^{\mathrm{h}}+|\mathcal{A}_b|\right)\Big), \tag{26}$$

where $\mathcal{A}_b$ is the action space of the standard VDN. In our proposed method, since the Q network of each BS is only used to determine the associated users and whether to send information to the DNT, we have $|\mathcal{A}| = 2^{U+1}$. However, in the standard VDN, each BS must consider to allocate N RBs to U users and the physical network information transmission link, resulting in an action space size of $|\mathcal{A}_b| = N^{U+1}$. Hence, our proposed method can significantly reduce the complexity of the standard method, especially in large networks with a big number U of users.

B. Convergence Analysis

Next, we analyze the convergence of the multi-agent VDN algorithm. We first define the gap between the actual global Q function $Q^{\pi}\left(\boldsymbol{s}, \boldsymbol{a}\right)$ and the global Q function $Q_{tot}^{\pi}\left(\boldsymbol{s}, \boldsymbol{a}\right)$ approximated by Q networks under a policy π as

$$\varepsilon^{\pi}\left(\boldsymbol{s}, \boldsymbol{a}\right) = Q^{\pi}\left(\boldsymbol{s}, \boldsymbol{a}\right) - Q_{tot}^{\pi}\left(\boldsymbol{s}, \boldsymbol{a}\right), \tag{27}$$

where

$$Q^{\pi}\left(\boldsymbol{s}, \boldsymbol{a}\right) = \sum_{\boldsymbol{s}'} P\left(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}\right)\left[r\left(\boldsymbol{s}, \boldsymbol{a}\right) + \gamma \max_{\boldsymbol{a}'} Q^{\pi}\left(\boldsymbol{s}', \boldsymbol{a}'\right)\right] \tag{28}$$

with $\boldsymbol{s}', \boldsymbol{a}'$ being the global state and the joint action of all BSs at the next time step, and $P\left(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}\right)$ being the transition probability from the current global state $\boldsymbol{s}$ to the next global state $\boldsymbol{s}'$ given the joint action $\boldsymbol{a}$.

Then, the convergence of VDN is presented in the following lemma.

Lemma 1. If 1) $\varepsilon \to 0$, or $\gamma \to 1$, and 2) $|\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a})| \le \varepsilon$ for any π_1, π_2, our proposed VDN method can converge to the optimal Q^*_{tot}.

Proof. To analyze the convergence of the VDN, we first define the Bellman operator H^{V} of our proposed VDN method for updating the approximated global Q function Q_{tot}. Since $Q^{\pi}_{tot}(\boldsymbol{s},\boldsymbol{a}) = Q^{\pi}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi}(\boldsymbol{s},\boldsymbol{a})$ and the standard Bellman operator is $(HQ)(\boldsymbol{s},\boldsymbol{a}) = \sum_{\boldsymbol{s}'} P(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})\left[r(\boldsymbol{s},\boldsymbol{a}) + \gamma \max_{\boldsymbol{a}'} Q(\boldsymbol{s}',\boldsymbol{a}')\right]$, the Bellman operator of our proposed VDN method is [29], [30]

$$\left(H^{\mathrm{V}}Q^{\pi}_{tot}\right)(\boldsymbol{s},\boldsymbol{a}) = \sum_{\boldsymbol{s}} P(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})\Big[r(\boldsymbol{s},\boldsymbol{a}) + \gamma\Big(\max_{\boldsymbol{a}'} Q^{\pi}_{tot}(\boldsymbol{s}',\boldsymbol{a}') + \varepsilon^{\pi}(\boldsymbol{s}',\boldsymbol{a}')\Big)\Big] - \varepsilon^{\pi}(\boldsymbol{s},\boldsymbol{a}). \quad (29)$$

Given (29), we use the Banach fixed point theorem [31] to prove the convergence of our proposed VDN method. In particular, according to the Banach fixed point theorem, to prove the convergence of our proposed VDN method, we only need to prove that H^{V} satisfies the following condition:

$$\|H^{\mathrm{V}}Q^{\pi_1}_{tot} - H^{\mathrm{V}}Q^{\pi_2}_{tot}\|_\infty \le \gamma\|Q^{\pi_1}_{tot} - Q^{\pi_2}_{tot}\|_\infty, \quad \text{for any } \pi_1, \pi_2, \quad (30)$$

where $Q^{\pi_1}_{tot}(\boldsymbol{s},\boldsymbol{a})$ and $Q^{\pi_2}_{tot}(\boldsymbol{s},\boldsymbol{a})$ are the global Q function approximated by Q networks under policies π_1 and π_2, we have

$$\begin{aligned}
&\|H^{\mathrm{V}}Q^{\pi_1}_{tot} - H^{\mathrm{V}}Q^{\pi_2}_{tot}\|_\infty \\
&= \max_{\boldsymbol{s},\boldsymbol{a}} \left|H^{\mathrm{V}}Q^{\pi_1}_{tot} - H^{\mathrm{V}}Q^{\pi_2}_{tot}\right| \\
&= \max_{\boldsymbol{s},\boldsymbol{a}} \Bigg| \sum_{\boldsymbol{s}'} P(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})\,\gamma \left[\max_{\boldsymbol{a}'_1}\left(Q^{\pi_1}_{tot}(\boldsymbol{s}',\boldsymbol{a}'_1) + \varepsilon^{\pi_1}(\boldsymbol{s}',\boldsymbol{a}'_1)\right)\right. \\
&\qquad \left. - \max_{\boldsymbol{a}'_2}\left(Q^{\pi_2}_{tot}(\boldsymbol{s}',\boldsymbol{a}'_2) + \varepsilon^{\pi_2}(\boldsymbol{s}',\boldsymbol{a}'_2)\right)\right] \\
&\qquad - \left(\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right)\Bigg| \\
&= \max_{\boldsymbol{s},\boldsymbol{a}} \Bigg| \sum_{\boldsymbol{s}'} P(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})\gamma\left(\max_{\boldsymbol{a}'_1} Q^{\pi_1}(\boldsymbol{s}',\boldsymbol{a}'_1) - \max_{\boldsymbol{a}'_2} Q^{\pi_2}(\boldsymbol{s}',\boldsymbol{a}'_2)\right) \\
&\qquad - \left(\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right)\Bigg|.
\end{aligned} \quad (31)$$

Since $\max_{\boldsymbol{a}'_1} Q^{\pi_1}(\boldsymbol{s}',\boldsymbol{a}'_1) - \max_{\boldsymbol{a}'_2} Q^{\pi_2}(\boldsymbol{s}',\boldsymbol{a}'_2) \le \max_{\boldsymbol{a}'}\left(Q^{\pi_1}(\boldsymbol{s}',\boldsymbol{a}') - Q^{\pi_2}(\boldsymbol{s}',\boldsymbol{a}')\right)$, we have

$$\begin{aligned}
&\|H^{\mathrm{V}}Q^{\pi_1}_{tot} - H^{\mathrm{V}}Q^{\pi_2}_{tot}\|_\infty \\
&\le \max_{\boldsymbol{s},\boldsymbol{a}} \Bigg| \sum_{\boldsymbol{s}'} P(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})\gamma \max_{\boldsymbol{a}'}\left(Q^{\pi_1}(\boldsymbol{s}',\boldsymbol{a}') - Q^{\pi_2}(\boldsymbol{s}',\boldsymbol{a}')\right) \\
&\qquad - \left(\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right)\Bigg|.
\end{aligned} \quad (32)$$

Since $\sum_{\boldsymbol{s}'} P(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})\gamma \max_{\boldsymbol{a}'}\left(Q^{\pi_1}(\boldsymbol{s}',\boldsymbol{a}') - Q^{\pi_2}(\boldsymbol{s}',\boldsymbol{a}')\right)$ is the expected value of $\gamma \max_{\boldsymbol{a}'}\left(Q^{\pi_1}(\boldsymbol{s}',\boldsymbol{a}') - Q^{\pi_2}(\boldsymbol{s}',\boldsymbol{a}')\right)$ with respect to $\boldsymbol{s}'$, it is less than or equal to $\max_{\boldsymbol{s}',\boldsymbol{a}'}\left(Q^{\pi_1}(\boldsymbol{s}',\boldsymbol{a}') - Q^{\pi_2}(\boldsymbol{s}',\boldsymbol{a}')\right)$. Hence, we have

$$\begin{aligned}
&\|H^{\mathrm{V}}Q^{\pi_1}_{tot} - H^{\mathrm{V}}Q^{\pi_2}_{tot}\|_\infty \\
&\le \max_{\boldsymbol{s},\boldsymbol{a}} \Bigg| \gamma \max_{\boldsymbol{s}',\boldsymbol{a}'}\left(Q^{\pi_1}(\boldsymbol{s}',\boldsymbol{a}') - Q^{\pi_2}(\boldsymbol{s}',\boldsymbol{a}')\right) \\
&\qquad - \left(\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right)\Bigg|.
\end{aligned} \quad (33)$$

In (33), since $\max_{\boldsymbol{s}',\boldsymbol{a}'}\left(Q^{\pi_1}(\boldsymbol{s}',\boldsymbol{a}') - Q^{\pi_2}(\boldsymbol{s}',\boldsymbol{a}')\right) = \max_{\boldsymbol{s},\boldsymbol{a}}\left(Q^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - Q^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right)$, and $\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a}) \ge \min_{\boldsymbol{s},\boldsymbol{a}}\left(\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right)$, we have

$$\begin{aligned}
&\gamma \max_{\boldsymbol{s}',\boldsymbol{a}'}\left(Q^{\pi_1}(\boldsymbol{s}',\boldsymbol{a}') - Q^{\pi_2}(\boldsymbol{s}',\boldsymbol{a}')\right) - \left(\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right) \\
&\le \max_{\boldsymbol{s},\boldsymbol{a}} \gamma\left(Q^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - Q^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right) - \min_{\boldsymbol{s},\boldsymbol{a}}\left(\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right) \\
&= \max_{\boldsymbol{s},\boldsymbol{a}} \left[\gamma\left(Q^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - Q^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right) - \left(\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right)\right].
\end{aligned} \quad (34)$$

Based on (34), (33) can be further simplified as

$$\begin{aligned}
&\|H^{\mathrm{V}}Q^{\pi_1}_{tot} - H^{\mathrm{V}}Q^{\pi_2}_{tot}\|_\infty \\
&\le \left| \max_{\boldsymbol{s},\boldsymbol{a}} \left[\gamma\left(Q^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - Q^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right)\right.\right. \\
&\qquad \left.\left. - \left(\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right)\right] \right| \\
&\le \max_{\boldsymbol{s},\boldsymbol{a}} \Bigg| \gamma\left(Q^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - Q^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right) \\
&\qquad - \left(\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right)\Bigg| \\
&= \max_{\boldsymbol{s},\boldsymbol{a}} \Bigg| \gamma\left(Q^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - Q^{\pi_2}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) + \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right) \\
&\qquad - (1-\gamma)\left(\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right)\Bigg| \\
&= \max_{\boldsymbol{s},\boldsymbol{a}} \Bigg| \gamma\left(Q^{\pi_1}_{tot}(\boldsymbol{s},\boldsymbol{a}) - Q^{\pi_2}_{tot}(\boldsymbol{s},\boldsymbol{a})\right) \\
&\qquad - (1-\gamma)\left(\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a})\right)\Bigg|.
\end{aligned} \quad (35)$$

Since $|\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - \varepsilon^{\pi_2}(\boldsymbol{s},\boldsymbol{a})| \le \varepsilon$, we have $\varepsilon^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) -$

TABLE II
SYSTEM PARAMETERS [32], [33]

Parameter	Value	Parameter	Value
U	10	M	3
G	75	B	1
P	1	N_0	1×10^{-5}
θ^{h}	128	λ_G	1×10^{-3}
N^{h}	128	K	5
γ	0.2	λ_Q	1×10^{-4}

$\varepsilon^{\pi_2}(\boldsymbol{s}, \boldsymbol{a}) \geq -\varepsilon$. Hence, (35) can be rewritten as

$$\begin{aligned}
&\|H^{\mathrm{V}} Q_{tot}^{\pi_1} - H^{\mathrm{V}} Q_{tot}^{\pi_2}\|_\infty \\
&\leq \max_{\boldsymbol{s},\boldsymbol{a}} \left| \gamma \left(Q_{tot}^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - Q_{tot}^{\pi_2}(\boldsymbol{s},\boldsymbol{a}) \right) - (1-\gamma)\,\varepsilon \right| \\
&= \gamma \max_{\boldsymbol{s},\boldsymbol{a}} \left| Q_{tot}^{\pi_1}(\boldsymbol{s},\boldsymbol{a}) - Q_{tot}^{\pi_2}(\boldsymbol{s},\boldsymbol{a}) \right| + (1-\gamma)\,\varepsilon \\
&= \gamma \|Q_{tot}^{\pi_1} - Q_{tot}^{\pi_2}\|_\infty + (1-\gamma)\,\varepsilon. \qquad (36)
\end{aligned}$$

From (36), we see that when $\varepsilon \to 0$ or $\gamma \to 1$, the VDN Bellman operator H^{V} satisfies (30). Based on the Banach fixed-point theorem, the VDN in our proposed method will converge to Q_{tot}^*. This completes the proof. □

V. SIMULATION RESULTS

In this section, we perform extensive simulations to evaluate the performance of our designed GRU and VDN based method. We first introduce the setup of the simulations. Then, we analyze the simulation results of our designed method.

A. System Setup

For simulations, we consider a 300×100 network area. 3 BSs are located at the coordinates $[-100, 0]$, $[0, 0]$, and $[100, 0]$, respectively. Each BS has a coverage radius of 60. The BSs in the system serve $U = 12$ users. The cloud server that generates and controls the DNT is located at $[0, 50]$. The GRU model used to predict the DNT states is implemented on the cloud server. The cloud server will collect a dataset of $2,000$ trajectories from all U users to train the GRU model, with each trajectory having 30 steps. For comparison purposes, we consider an independent Q learning method in which the Q network of each BS m is trained by its local Q values $Q_m(\boldsymbol{s}^m, \boldsymbol{a}^m)$ rather than the global Q values $Q_{tot}(\boldsymbol{s}, \boldsymbol{a})$ as follows:

$$\boldsymbol{\theta}_m \leftarrow \boldsymbol{\theta}_m - \lambda_Q \nabla_{\boldsymbol{\theta}_m} J_Q^m, \qquad (37)$$

where $J_Q^m = \mathbb{E}_{(\boldsymbol{s}_t^m, \boldsymbol{a}_t^m, r^m, \boldsymbol{s}_{t+1}^m) \sim \mathcal{D}_g} \Big[\Big(\Big(r^m(\boldsymbol{s}_t^m, \boldsymbol{a}_t^m) + \gamma \max_{\boldsymbol{a}_{t+1}^m} \tilde{Q}_{\tilde{\boldsymbol{\theta}}_m}(\boldsymbol{s}_{t+1}^m, \boldsymbol{a}_{t+1}^m) \Big) - Q_{\boldsymbol{\theta}_m}(\boldsymbol{s}_t^m, \boldsymbol{a}_t^m) \Big)^2 \Big]$, with $r^m(\boldsymbol{s}_t^m, \boldsymbol{a}_t^m)$ being the local reward of BS m. The baseline model parameters are similar to that of the designed method. Other parameters used in the simulations are listed in Table II.

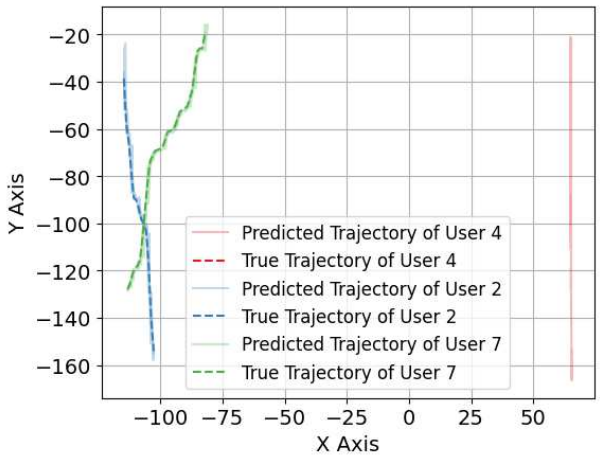

Fig. 2. The prediction of the user movement trajectories via the GRU model.

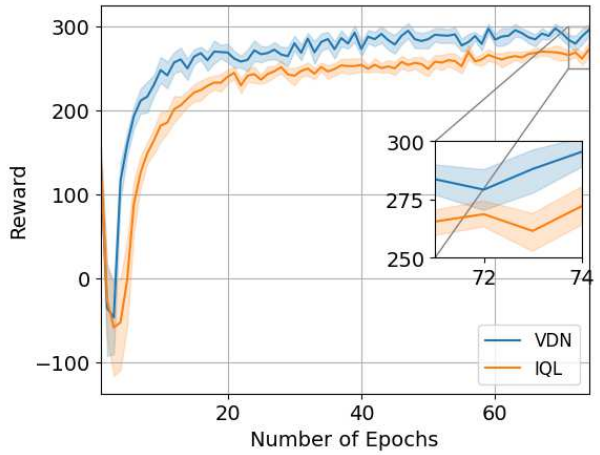

Fig. 3. Convergence of VDN and IQL ($\epsilon = 0.3, N = 12$).

B. Simulation Results and Analysis

Fig. 2 shows the user movement trajectories predicted by the GRU model. The users in Fig. 2 are randomly selected from all users. From Fig. 2, we see that the positioning mean square errors of user 2, user 4, and user 7 are respectively 0.188, 0.075 and 0.320. This is because the GRU effectively captures dependencies of the historical user movement through its gating mechanisms. From this figure, we also see that the mobility prediction mean square errors of users 2 and 7 are higher compared to that of user 4. This is because the movement dynamics of users 2 and 7 are higher compared to that of user 4.

In Fig. 3, we show the convergence of both the proposed method and the independent Q based method. Fig. 3 shows that, as the number of training epochs increases, the average rewards of both considered algorithms increase. This is because the policy of determining the associated users and the physical network information transmission is optimized by the considered RL algorithms. We also see that our designed

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2025.3543126

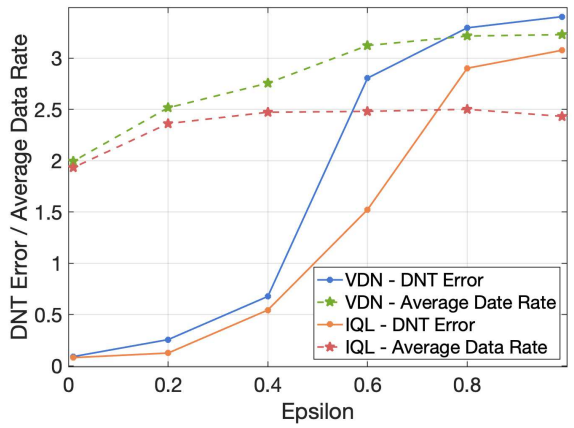

Fig. 4. The average DNT error and the data rate as ϵ varies ($N = 9$).

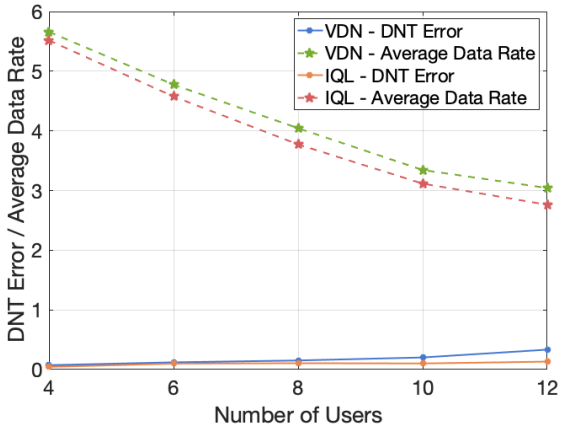

Fig. 5. The average DNT error and the data rate as the number of users varies ($\epsilon = 0.25, N = 12$).

algorithm can improve the weighted sum of data rates and the synchronization accuracy by up to 28.96% compared to the independent Q. This gain stems from the fact that the proposed method allows each BS to use its local state to collaboratively find a globally optimal policy that maximize total data rates of all users, while minimize the asynchronization between the physical network and the DNT.

Fig. 4 shows how the average DNT error and the average data rate of our proposed method and the independent Q method change as ϵ varies. Fig. 4 shows that our proposed method improves the average data rate by up to 31.79% compared to the independent Q method when $\epsilon = 0.8$, while the average DNT error of the proposed method is higher compared to the independent Q method. This is because our proposed method optimizes the weighted sum of the data rates of all users and the gap between the DNT and the physical network. Fig. 4 also shows that the average data rate of the independent Q method reduces as ϵ increases from 0.8 to 0.99. This is because in the independent Q method, as the value of the epsilon increases, each BS prioritizes the data rate of its associated users, which may consequently increase the interference between BSs.

In Fig. 5, we show how the average DNT error and the average data rate of our proposed method and the independent Q method change as the number of users varies. From Fig. 5, we see that the DNT errors of both our considered algorithm and the independent Q method increase as the number of users increases. This is because as the network serves more users, the BSs may not have enough RBs to maintain DNT synchronization. Fig. 5 also shows that the average data rates of both methods decrease with the increase of the number of users. This stems from the fact that the BSs have limited RBs to serve users, such that several users are served by the RBs with large interference. Furthermore, Fig. 5 shows that when the number of users is 12, the data rate of our proposed method is 10.15% higher than that of the independent Q method but the average DNT error of the proposed method is 0.2 higher compared to the independent Q method. This is because our proposed method adapts the RB allocation policy of each BS

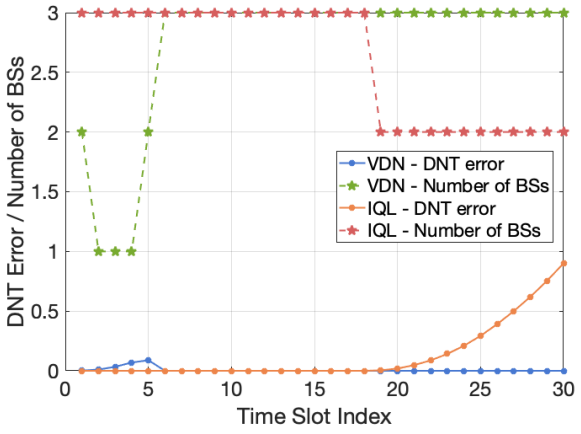

Fig. 6. The number of BSs updating the DNT and the DNT errors at each step within a single episode ($\epsilon = 0.25, N = 12$).

to accommodate the growing number of users to maintain a balance between serving the users in the physical network and the DNT synchronization.

Fig. 6 shows how the DNT errors of our proposed method and the independent Q method change as the number of BSs that updates the DNT at each step. In Fig. 6, we show that our proposed method can achieve a small DNT error compared to the independent Q method. This is due to the fact that our method enables better BS collaborations, allowing them to optimize RB allocation by considering the joint impact of their decisions on the overall system performance.

VI. Conclusion

In this paper, we have proposed a DNT enabled network consisting of a physical network and its DNT. In this network, a set of BSs in the physical network must use their limited spectrum resources to serve a set of users while periodically transmitting the partial observed physical network information to a cloud server to update the DNT. We have formulated this

resource allocation task as an optimization problem aimed at maximizing the sum of data rates for all users while minimizing the asynchronization between the physical network and the DNT. To address this problem, we have introduced a method based on GRUs and the VDN. The GRU component enables the DNT to predict its future state and maintain updates even when physical network information is not transmitted. The VDN component allows each BS to learn the relationship between its local observation and the team reward of all BSs, allowing it to collaborate with others in determining whether to transmit physical network information and optimizing spectrum allocation. Simulation results have demonstrated that, compared to a baseline approach utilizing GRU and IQL, our proposed method significantly improves the weighted sum of user data rates and the asynchronization between the physical network and the DNT.

REFERENCES

[1] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13789–13804, September 2021.

[2] L. Hui, M. Wang, L. Zhang, L. Lu, and Y. Cui, "Digital twin for networking: A data-driven performance modeling perspective," *IEEE Network*, vol. 37, no. 3, pp. 202–209, May/June 2023.

[3] Y. Liu, Z. Peng, Z. Zhang, H. Yu, and M. Chen, "Digital network twins for next-generation wireless: Creation, optimization, and challenges," *arXiv preprint arXiv:2410.18002*, October 2024.

[4] L. U. Khan, Z. Han, W. Saad, E. Hossain, M. Guizani, and C. S. Hong, "Digital twin of wireless systems: Overview, taxonomy, challenges, and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2230–2254, August 2022.

[5] Q. Guo, F. Tang, T. K. Rodrigues, and N. Kato, "Five disruptive technologies in 6G to support digital twin networks," *IEEE Wireless Communications*, vol. 31, no. 1, pp. 149–155, February 2024.

[6] Z. Zhang, M. Chen, Z. Yang, and Y. Liu, "Mapping wireless networks into digital reality through joint vertical and horizontal learning," *arXiv preprint arXiv:2404.14497*, April 2024.

[7] A. Sharma, E. Kosasih, J. Zhang, A. Brintrup, and A. Calinescu, "Digital twins: State of the art theory and practice, challenges, and open research questions," *Journal of Industrial Information Integration*, vol. 30, pp. 100383, September 2022.

[8] X. Lin, L. Kundu, C. Dick, E. Obiodu, T. Mostak, and M. Flaxman, "6G digital twin networks: From theory to practice," *IEEE Communications Magazine*, vol. 61, no. 11, pp. 72–78, June 2023.

[9] R. Dong, C. She, W. Hardjawana, Y. Li, and B. Vucetic, "Deep learning for hybrid 5G services in mobile edge computing systems: Learn from a digital twin," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4692–4707, July 2019.

[10] C. Ruah, O. Simeone, and B. M. Al-Hashimi, "A Bayesian framework for digital twin-based control, monitoring, and data collection in wireless systems," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3146–3160, August 2023.

[11] C. Zhou, J. Gao, M. Li, X. S. Shen, and W. Zhuang, "Digital twin-empowered network planning for multi-tier computing," *Journal of Communications and Information Networks*, vol. 7, no. 3, pp. 221–238, September 2022.

[12] O. Hashash, C. Chaccour, and W. Saad, "Edge continual learning for dynamic digital twins over wireless networks," *arXiv preprint arXiv:2204.04795*, April 2022.

[13] Q. Guo, F. Tang, and N. Kato, "Resource allocation for aerial assisted digital twin edge mobile network," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3070–3079, October 2023.

[14] J. Zheng, T. H. Luan, Y. Zhang, R. Li, Y. Hui, L. Gao, and M. Dong, "Data synchronization in vehicular digital twin network: A game theoretic approach," *IEEE Transactions on Wireless Communications*, vol. 22, no. 11, pp. 7635–7647, November 2023.

[15] Y. Han, D. Niyato, C. Leung, D. I. Kim, K. Zhu, S. Feng, X. Shen, and C. Miao, "A dynamic hierarchical framework for IoT-assisted digital twin synchronization in the metaverse," *IEEE Internet of Things Journal*, vol. 10, no. 1, pp. 268–284, January 2023.

[16] K. Cho, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, June 2014.

[17] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, December 2014.

[18] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, et al., "Value-decomposition networks for cooperative multi-agent learning," *arXiv preprint arXiv:1706.05296*, June 2017.

[19] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. International Conference on Machine Learning*, Amherst, MA, USA, June 1993, pp. 330–337.

[20] H. Tabassum, M. Salehi, and E. Hossain, "Fundamentals of mobility-aware performance characterization of cellular networks: A tutorial," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2288–2308, March 2019.

[21] William S Lovejoy, "A survey of algorithmic methods for partially observed Markov decision processes," *Annals of Operations Research*, vol. 28, no. 1, pp. 47–65, 1991.

[22] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 269–283, January 2021.

[23] Y. Hu, X. Wang, and W. Saad, "Distributed and distribution-robust meta reinforcement learning (D^2-RMRL) for data pre-storage and routing in cube satellite networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 17, no. 1, pp. 128–141, January 2023.

[24] R. Jonker and T. Volgenant, "Improving the Hungarian assignment algorithm," *Operations Research Letters*, vol. 5, no. 4, pp. 171–175, October 1986.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, February 2015.

[26] L. Lin, Reinforcement learning for robots using neural networks, Carnegie Mellon University, 1992.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Conference on Neural Information Processing Systems*, Long Beach, CA, USA, December 2017, vol. 30.

[28] Y. Wang, M. Chen, Z. Yang, T. Luo, and W. Saad, "Deep learning for optimal deployment of UAVs with visible light communications," *IEEE Transactions on Wireless Communications*, vol. 19, no. 11, pp. 7049–7063, November 2020.

[29] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep Q-learning," in *Proc. Learning for Dynamics and Control*, Feburary 2020, pp. 486–489.

[30] S. Wang, M. Chen, Z. Yang, C. Yin, W. Saad, S. Cui, and H. V. Poor, "Distributed reinforcement learning for age of information minimization in real-time IoT systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 3, pp. 501–515, April 2022.

[31] J. Abounadi, D. P. Bertsekas, and V. Borkar, "Stochastic approximation for nonexpansive maps: Application to Q-learning algorithms," *SIAM Journal on Control and Optimization*, vol. 41, no. 1, pp. 1–22, March 2002.

[32] W. Zhang, Y. Wang, M. Chen, T. Luo, and D. Niyato, "Optimization of image transmission in cooperative semantic communication networks," *IEEE Transactions on Wireless Communications*, vol. 23, no. 2, pp. 861–873, 2023.

[33] J. Hu, S. Jiang, S. A. Harding, H. Wu, and S. Liao, "Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning," *arXiv preprint arXiv:2102.03479*, Feburary 2021.