Task Offloading for Edge Metaverse: A Joint BSUM and Reinforcement Learning Approach

Latif U. Khan, *Member, IEEE*, Maher Guizani, *Member, IEEE*, Sami Muhaidat, *Senior Member, IEEE*, Asad Masood Khattak, *Senior Member, IEEE*, Adel Khelifi, *Senior Member, IEEE*, and Zhu Han, *Fellow, IEEE*

***Abstract*—A metaverse can bring many benefits (i.e., self-sustaining and proactive analytics (e.g., analysis before user requests)) to wireless applications; however, its deployment is very challenging due to simultaneous quality of service (QoS) and quality of physical experience (QoE) constraints. Furthermore, the computing and communication resources of end-nodes are limited. Therefore, this paper proposes a novel task offloading framework for metaverse-empowered wireless systems. Our formulated problem aims at minimizing the cost of task offloading in the metaverse while considering both QoS and QoE constraints by optimizing the task offloading, resource allocation, and transmit power allocation variables. To optimize the formulated problem, we use a decomposition-based scheme that further uses modified block-successive upper-bound minimization (BSUM), convex optimization, and multi-agent reinforcement learning (MARL) for transmit power allocation, resource allocation, and task offloading, respectively. Our solution of using convex optimization-assisted MARL for joint resource allocation and task offloading significantly improves the performance of learning in terms of reward. Furthermore, BSUM significantly improves transmit power allocation when used in conjunction with a convex optimizer and MARL. Other than that, we also use dueling to further improve the performance of MARL. Our analyses show that convex optimization, BSUM, and dueling help in significantly improving the performance of MARL. Compared to traditional MARL, our proposal results in significant improvement in terms of reward and cost, as illustrated by the results.**

***Index Terms*—Metaverse, Internet of Things, convex optimization, digital twins, deep reinforcement learning.**

I. Introduction

AN IMMERSIVE trend of migrating from traditional systems (i.e., considering quality of service (QoS) metrics) toward novel wireless systems (i.e., considering both QoS and quality of experience metrics (QoE)) has been witnessed [1], [2]. The main purpose of this trend is to successfully enable the diverse requirements (i.e., to fulfill the user-defined QoE metrics, e.g., quality of physical experience for haptics applications) of the emerging applications (e.g., digital healthcare and haptics-based systems) [1]. To effectively enable wireless systems, the concept of metaverse has been introduced in many works [3], [4], [5], [6]. The key features of a metaverse for enabling emerging wireless applications are self-sustaining (i.e., the ability to operate the wireless system with the least external intervention) and proactive analysis abilities (i.e, analysis of a system before deployment and user requests) [3]. These abilities are very important, especially when the number of users is massive, and there is a need to meet both traditional QoS metrics (e.g., latency) and user-defined QoE metrics (e.g., sense of touch in haptics-based applications).

A typical metaverse framework has two main entities: (a) the physical world (i.e., physical space) and (b) the virtual metaverse world (i.e., meta space). These spaces interact with each other to enable various applications.[1] Although the metaverse for wireless systems will enable many benefits, many challenges need to be addressed. These challenges are efficient and effective signaling of the metaverse operation, sensing of physical world states and sharing them with the meta space, and task offloading (e.g., offloading of inference and rendering tasks by end users to the meta space), among others. Sensing is performed to synchronize the physical world with a metaverse virtual world. Metaverse signaling involves signaling for overall physical system management. On the other hand, there is a wide variety of tasks (e.g., sensing of 3D images by a virtual reality (VR) shooting sensing device, rendering tasks for AR/VR applications, sensing for violence detection, collision avoidance in autonomous cars, and lane change assistance) to be performed by the physical space entities. However, there are limitations on the availability of computing power in the physical space. Therefore, the physical space nodes should offload their tasks to the meta space.

In this paper, our focus is on task offloading for various metaverse entities. Note that there are three main entities, such as service-requesting devices, learning devices, and sensing entities, in the physical space of a metaverse [3], [4], [5], [6]. Here, we provide example scenarios showing the

Received 5 May 2025; revised 18 October 2025; accepted 1 December 2025. Date of publication 24 December 2025; date of current version 7 May 2026. Recommended for acceptance by W. Gong. *(Corresponding author: Latif U. Khan.)*

Latif U. Khan and Adel Khelifi are with the Department of Computer Science and Information Technology, Abu Dhabi University, Abu Dhabi, UAE (e-mail: latif.u.khan2@gmail.com, latif.khan@adu.ac.ae).

Maher Guizani is with the Computer Science and Engineering Department, University of Texas at Arlington, Arlington, TX 76019 USA.

Sami Muhaidat is with 6G Research Center, Khalifa University, Abu Dhabi, UAE.

Asad Masood Khattak is with the College of Technological Innovation, Zayed University, Dubai 19282, UAE.

Zhu Han is with the Electrical and Computer Engineering Department, University of Houston, Houston, TX 77004 USA, also with the Computer Science Department, University of Houston, Houston, TX 77004 USA, and also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul 02447, South Korea.

Digital Object Identifier 10.1109/TMC.2025.3648196

[1]For more details, please refer to [3]

importance of task offloading in the metaverse for various entities.

- *Sensing units:* Consider a scenario of industrial manufacturing based on a metaverse. The end technicians/workers will use AR headsets to interact with the meta space and get guidance about real-time assembly, production processes, and fault correction. Additionally, the data shared by physical space with the meta space will be used in the training of a meta space model for run-time resource management. The AR users with headsets will have multiple tasks to perform. These multiple tasks (e.g., rendering task and 3D shooting task) need to be performed either by the AR headset or the meta space. Similarly, one can use depth-sensing cameras not only to capture but to sense the depth of the actual scenario in the physical space [7]. Meanwhile, there will be AR users who will request some of the services (e.g., rendering tasks) from the meta space [8], [9]. All of the tasks are difficult to perform by the end-devices. End-devices will compute some tasks and offload the rest of the tasks to the meta space.
- *Service-requesting devices:* Consider a user in a vehicular metaverse who is wearing AR/VR headset and wants to get information on traffic for network optimization and planning. Such a user can initiate a voice command (i.e., request for a task to compute the best route) *"Show me the areas with the most frequent traffic congestion and suggest improvements to reduce congestion."* and share with the meta space deployed at the edge or cloud. The reason for task offloading to the meta space is due to the limited computing power of its AR device. The meta space will use pre-trained large language models to analyze the user query and then generate the response accordingly to guide the end-users with an optimum route.
- *Learning devices:* Consider split federated learning (SFL) for the training of metaverse models. In SFL, due to resource constraints, the end-devices learn a part of their local models and offload the remaining learning task with the edge server (i.e., a meta space can perform this partial task computation in our model) for partial computation. The computed model updates are then shared with the end-devices to update their local models. Next, the local models are aggregated to yield a global model. This kind of distributed learning is very promising for scenarios where we have limited computing power at end-devices. The exchange of learning updates between the end-devices and the edge servers takes place iteratively.

Based on the above observations, it is evident that we should consider task offloading in the metaverse over wireless networks. First, we discuss various recent works and then we differentiate our contributions from them.

A. Related Work

1) Metaverse and Wireless: Some of the existing works considered metaverse and wireless systems [3], [4], [5], [6], [8], [9], [10], [11], [12]. The work in [3] presented a vision of using the metaverse to effectively and efficiently realize wireless systems. A general architecture that consists of two spaces was proposed. These spaces are physical space that contain all the physical entities and the logical meta space. All the entities in the physical space will be controlled by the meta space. The authors also presented a case study for performing resource optimization in implementing their proposed metaverse architecture. Finally, open challenges were represented. In another work [4], a network virtualization-empowered metaverse was considered. The authors formulated a problem and then presented a hierarchical matching and convex optimization-based solution. Finally, they provided extensive numerical results to show the validity of their proposal. The work in [5] presented an edge intelligence-empowered metaverse. They proposed an edge intelligence-based framework for the metaverse and highlighted the role of edge intelligence in enabling the metaverse. On the other hand, the works in [8] and [9] considered the task offloading in the metaverse. The work in [8] considered a cost minimization problem for AR applications in the context of the metaverse. Another work [10] proposed an efficient FL scheme for metaverse by considering resource allocation, dynamic user selection (i.e., different compared to traditional FL, which considers static nodes), and gradient quantization. The work in [11] considered a continual reinforcement learning framework for resource allocation in virtual reality streaming. The work in [12] considered resource optimization for enhancing the QoE in metaverse. Different from the works in [3], [4], [5], [6], [8], [9], [10], [11], [12], we consider task offloading for various services in the metaverse. We consider both latency and energy along with novel QoS and QoE constraints. In contrast to [12], our QoE metric considers both packet error rate and immersive experience. Furthermore, we formulate a novel optimization and present a novel dueling DDQN-based solution.

2) Resource Optimization in Metaverse: The works in [12], [13], [14], [15], [16] considered resource optimization in metaverse. The work in [13] proposed a meta slicing framework for resource allocation in metaverse. The work in [14] studied resource allocation in the AR-based metaverse. Specifically, they defined a system utility that considered gross profit and energy consumption. Moreover, they optimized the computing resources of AR vehicles, the transmit power, the size of the computing model, and the metaverse service computing resources. The work in [12] presented a solution based on evolutionary reinforcement learning for metaverse experience maximization. They presented a metaverse experience metric and then used reinforcement learning to maximize their defined metric. Additionally, [15] considered an attention-aware resource allocation in the metaverse. In contrast to traditional URLLC, which focuses on QoS, they evolved the traditional URLLC to reflect QoE in the metaverse. Moreover, they designed an optimal contract for modeling the interaction between the metaverse service providers and the infrastructure providers while optimizing their meta-immersive metric-based utility. The work in [16] analyzes non-fungible tokens, blockchain, and the metaverse while addressing resource and privacy issues. In order to maximize connection, resource allocation, and trust-cost balance for 6G edge computing, it presents its solution based on optimization, semi-definite relaxation (SDR), the Hungarian method, and a novel fractional programming technique. Their

efficiency in developing NFT-driven blockchain-Metaverse applications is validated by simulations. Different from the works in [12], [13], [14], [15], [16], we consider task offloading and optimize task offloading, resource allocation, power allocation, and metaverse end-devices computing resource allocation.

3) Task Offloading in Wireless Networks: Many works considered task offloading in wireless networks [8], [9], [17], [18]. Mostly, the works studied task offloading in the context of general wireless networks instead of specifically considering the metaverse. The works in [17] and [18] considered tasks offloading for edge networks. Chen et al. in [17] considered task offloading problem in software-defined ultra-dense networks. They formulated a problem to minimize the overall latency by optimizing task placement and resource allocation. The other work [18] focused on the use of DRL for task offloading in edge networks. Specifically, they considered non-divisible tasks that are delay-sensitive. They formulated task offloading problem and maximized a long-term reward using a DRL scheme based on long short-term memory (LSTM), deep Q-network (DQN), and double-DQN techniques. Motivated by the above, we consider task offloading and minimize its cost by optimizing task offloading variable, resource allocation, power allocation, and metaverse end-devices computing resource allocation.

B. Our Contributions

Different from [3], [4], [5], [6], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], our work focuses on a metaverse-empowered wireless system.[2] Specifically, we consider task offloading for sensing, service-requesting users, and learning devices. Our contributions are summarized as follows:

- We propose a framework for task offloading in a metaverse-empowered wireless system. A problem is formulated to minimize the cost function that jointly considers latency and energy via optimization of end-devices computing resource optimization, task offloading, transmit power allocation, and resource optimization.
- To solve the formulated problem, a solution based on decomposition is proposed. For task offloading, multi-agent deep reinforcement learning (MARL) is used, whereas for time interval allocation in time division multiplexing (TDMA), convex optimization is used. For transmit power allocation, block successive upper-bound minimization (BSUM) is used. For MARL, we consider double deep Q-networks (DDQN) with dueling. The introduction of dueling in DDQN results in a performance improvement compared to standalone DDQN. Meanwhile, the performance of DDQN with dueling improves with the optimization of transmit power allocation using BSUM. Furthermore, the addition of convex optimization to BSUM and dueling empowered DDQN further improves the performance.
- Finally, validation is carried out using extensive simulation results. Our analysis shows that the proposed scheme using dueling DDQN with BSUM and convex optimizer significantly outperforms the traditional DDQN. Additionally, with the addition of different optimizers (e.g., BSUM and convex optimizer), the performance of dueling DDQN shows improvement.

The rest of the paper is organized as follows: Section II presents the system model and problem formulation. Section III presents a proposed solution approach using BSUM, MARL, and convex optimization. Simulation results are discussed in Section IV, and the paper is concluded in Section V.

II. System Model and Problem Formulation

A. System Model

Consider a metaverse-enabled wireless system, as shown in Fig. 1. The system model has two spaces: a meta space using a virtual model and a physical space containing actual physical entities. The physical and meta spaces communicate with each other using wireless interfaces. The meta space/s is/are implemented using edge/cloud servers. Moreover, the meta space uses a virtual model of the physical world along with mathematical optimization, machine learning, and game theory, among others, to perform efficient resource management for metaverse-based applications. In the metaverse, meta space/s can be deployed either at the network edge, cloud, or both edge-cloud, depending on computing and latency requirements. In our system model, we deploy meta spaces at the network edge for low-latency operation. On the other hand, in the physical space, there are mainly three entities: (a) sensing units (i.e., used to sense the state of static physical entities), (b) learning devices (i.e., used to learn local models and then share the learning data with a meta space), and (c) requesting users/devices. Let the set $\mathcal{S}$ of S entities and $\mathcal{A}$ of A devices denote the sensing entities and learning devices, respectively. Additionally, the service-requesting devices will be represented by the set $\mathcal{R}$ of R devices/end-users. The set of all physical space entities in a metaverse can be given by $\mathcal{N} = \{\mathcal{S} \cup \mathcal{A} \cup \mathcal{R}\}$. In our system, there is a set $\mathcal{M}$ of M distributed meta spaces running on the edge computing-enabled small cell base stations (SBSs). Next, we present the metaverse computing model.

B. Metaverse Computing Model

Consider a task $\mathcal{B}_n(b_n, v_n, z_n)$ to be performed at various entities in the physical space. Where b_n, v_n, and z_n denote the computing points, the computing power needed to process a single computing point, and the computing deadline, respectively. All the entities in the metaverse have multiple tasks to be performed. Meanwhile, there is a limited computing power (i.e., CPU-cycle/sec) available for a particular task. Therefore, the metaverse entities will compute the partial task locally, whereas they will offload the remaining task to the meta spaces. The local computing time for task $\mathcal{B}_n$ can be given by

$$\ell^n_{local}(\boldsymbol{\kappa}) = \begin{cases} \frac{b_n v_n}{\kappa_n}, & \forall n \in \mathcal{A}, \\ \frac{b_n v_n}{\kappa_n}, & \forall n \in \mathcal{R}, \\ \frac{I_n e \eta(x_1,x_2,x_3) v_n}{\kappa_n}, & \forall n \in \mathcal{S}, \end{cases} \tag{1}$$

where κ_n denotes the operating frequency of unit n. In (1), the local computing time depends on operating frequency (i.e., κ_n),

[2]For more information on the architecture of metaverse-enabled wireless system, please refer to [3].

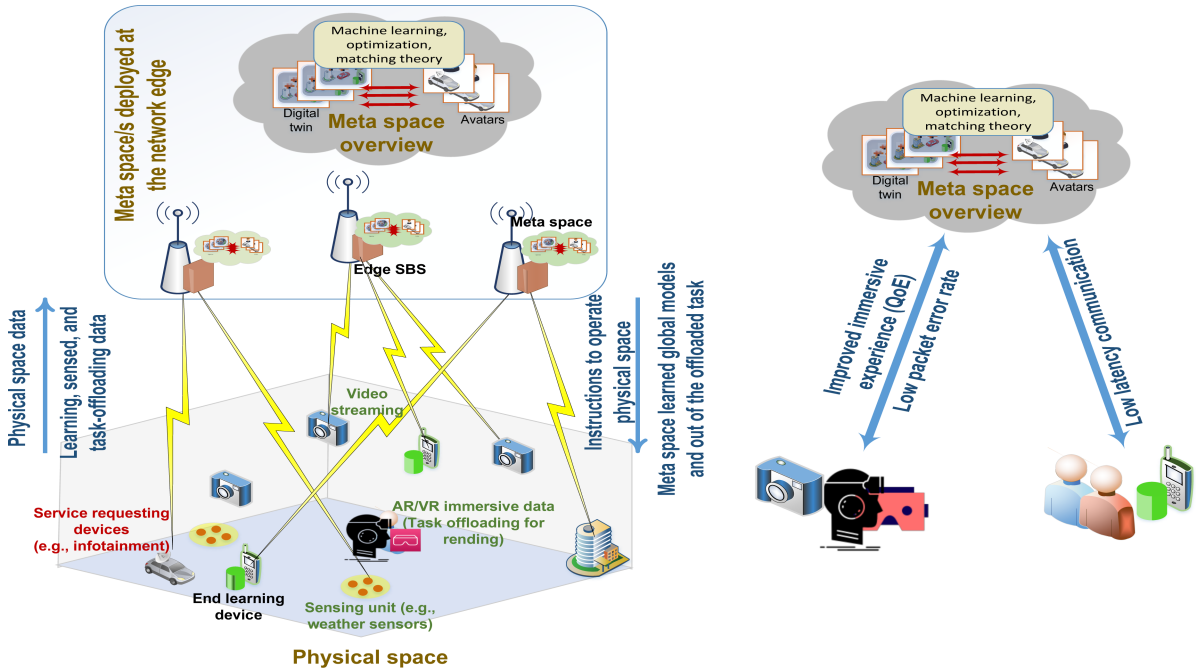


Fig. 1. Proposed task offloading-enabled metaverse system model.

computing points (i.e., b_n), and the computing power (i.e., v_n) required to process a single computing point. On the other hand, a massive number of sensors will be deployed, similar to the work in [19]. Then, the sensors are divided into discrete sensing units for modeling computing and communication with a low complexity instead of modeling for a massive number of sensors. In (1), for sensing units, it is clear that the local computing time depends on the number of data points that can be represented using the data flow (i.e., $\epsilon\eta(x_1, x_2, x_3)$ bits) concept for sensing units due to the fact that there is a massive number of sensors in the whole sensing region. I_n is the sensing interval. More details about sensing and the term $\epsilon\eta(x_1, x_2, x_3)$ will be given in Section II. C. Additionally, the local computing latency depends on the processing power needed to compute the single data point of the sensed data. It is evident that an increase in the computing power (i.e., that has limited availability) will minimize the local computing delay for various entities of the metaverse. On the other hand, the local energy consumption is proportional to the square of the operating frequency. The local compute energy can be given by

$$\chi^n_{local}(\boldsymbol{\kappa}) = \begin{cases} t_n v_n \kappa_n^2, & \forall n \in \mathcal{A}, \\ t_n v_n \kappa_n^2, & \forall n \in \mathcal{R}, \\ I_n \epsilon\eta(x_1, x_2, x_3) v_n \kappa_n^2, & \forall n \in \mathcal{S}. \end{cases} \tag{2}$$

Based on the aforementioned observations, there is a need for a limit on the maximum utilization of the operating frequencies for all metaverse entities.

$$\sum_{n\in\mathcal{A}} \sum_{n\in\mathcal{R}} \sum_{n\in\mathcal{S}} \kappa_n \leq \kappa_{\text{MAX}}. \tag{3}$$

Meanwhile, the computing power assigned to metaverse entities must be within limits.

$$\kappa_{\min} \leq \kappa_n \leq \kappa_{\max}, \forall n \in \{\mathcal{A} \cup \mathcal{R} \in \mathcal{S}\}. \tag{4}$$

On the other hand, the computing delay of tasks at metaverse entities in the physical space must be within the maximum allowed threshold. For instance, if the task is learning a local model at the metaverse entity (e.g., smart sensor) and then sharing the model with the meta space implemented at the network edge, the meta space will wait for a certain maximum duration before combining the local models received from the various physical space entities. Then, the meta space will combine (i.e., aggregate) all the local models to produce a global metaverse model. The entities of the physical space might not have sufficient power and they offload the task to the meta space. Here, for simplicity, we assume that the meta space has sufficient power and significantly less latency. Therefore, we ignore the edge latency. To reflect such kind of situations, there must be a latency constraint as follows:

$$(\ell^n_{local} + \ell^n_{trans}) \leq z_n, \forall n \in \{\mathcal{A} \cup \mathcal{R} \cup \mathcal{S}\}, \tag{5}$$

where z_n denotes the maximum delay a certain task can endure in our system. The values of z_n depend on the nature of the application. For instance, for live streaming of videos, the value of z_n should be chosen small compared to climate monitoring. Therefore, we should wisely select the values of z_n for various applications. Next, we discuss the sensing channel model for the metaverse.

C. Metaverse Sensing and Channel Model

In our system model, there is a variety of entities (e.g., moving objects and static sensing units) in the physical space that will communicate with the meta space. For effective sensing, a massive number of sensors are deployed. For instance, if we consider intelligent transportation systems, there will be a massive number of sensing entities (e.g., roadside units and cars LIDARS). Similarly, for other applications like smart industries, there will be a massive number of sensors. Specifically, for a metaverse-based wireless system, there is a need for efficient and effective synchronization of the physical space with the meta space for an updated operation, as evident from the literature in [4], [19]. To do so, we need massive sensors in the physical space. Such a massive distribution will be represented by a continuous distribution function $f(x_1, x_2, x_3)$, where x_1, x_2, and x_3 denote the longitude, latitude, and height, respectively.

Modeling various wireless network parameters (e.g., latency and transmission energy consumption) for a massive number of sensing devices is very challenging due to their massive number. Therefore, it is necessary to address this challenge. To do so, one can divide the massive number of sensing devices into sensing units (i.e., each sensing unit can accommodate multiple sensors). Let the set $\mathcal{S}$ consists of S sensing units and will accomodate all sensors. To compute the transmission latency and the transmission energy for such kinds of sensing units, we use the notion of data flow from sensing units. The concept of using unit volumetric density $\eta(x_1, x_2, x_3)(bps/m^3)$ was used in [20]. For fixed parameters (e.g., bandwidth allocated to a single sensing unit), the transmission latency and transmission energy depend on the data generated from the sensing units. We consider TDMA for the communication between the meta space and the physical space. The throughput can be given by

$$\zeta_n(t) = \tau_n W_j(t) \mathbb{E}_{h^n_{a,m}} \times \left(\log_2 \left(1 + \frac{\phi_n(t) h^n_{a,m}(t)}{\sum_{c\in\mathcal{C}} p_c(t) h_{c,m}(t) + N_o^2} \right) \right), \quad (6)$$

where τ_n denotes the time interval of TDMA. $W_j(t)$ and $(\frac{\phi_n(t)h^n_{a,m}(t)}{\sum_{c\in\mathcal{C}} p_c(t)h_{c,m}(t)+N_o^2})$ are the bandwidth of the resource block j and the signal-to-interference-plus-noise-ratio (SINR), respectively. $\phi_n, \forall n \in \mathcal{N}$ denotes the transmit power. $\mathbb{E}_{h^n_{d,m}}$ denotes the expectation with respect to $h^n_{d,a}(t)$. $\sum_{c\in\mathcal{C}}(p_c(t)h_{c,m}(t) + N_o^2)$ is interference due to cellular users. Now, we will write expressions for the transmission latency and transmission energy for various metaverse entities (i.e., learning devices, sensing entities, and requesting devices/users).

The expression for the transmission latency can be written in (7), shown at the bottom of this page. In (7) $y^{(n\to m)}(t)$ and $o_n(t)$ denote the task offloading variable and offloading decision variable at time slot t, respectively. U_a denotes the transmission overhead. $y^{(n\to m)}(t) = 1$, if the entity n task is offloaded to the meta space m and $y^{(n\to m)}(t) = 0$, otherwise. $o_n(t) = 1$ if the unit n wants to offload its task to the meta space m and $o_n(t) = 0$, otherwise. $\epsilon\eta(x_1, x_2, x_3)(t)$ denotes the data generated by the sensing unit at time step t. $I_t(t)$ denotes the time interval of sensing at time t. Note that the data generated by the sensing unit depends on the sensing interval and $\epsilon\eta(x_1, x_2, x_3)(t)$.

On the other hand, we must consider transmission energy in our model. The transmission energy can be given by equation (8), shown at the bottom of the next page. From (8), it is clear that the transmission energy depends on the task overhead, SINR, and the bandwidth allocated during time slot t. Additionally, the transmit power of the learning device significantly affects the transmission energy. For sensing units, the transmission energy depends mainly on the transmission overhead of the sensing units and transmit power. As we are using TDMA, there should be constraints on the time slot allocation to various entities.

$$\tau_{\min} \le \tau_n \le \tau_{\max}, \forall n \in \{\mathcal{A} \cup \mathcal{R} \cup \mathcal{S}\}. \quad (9)$$

In our TDMA-based system, we should assign time slots to all devices in a way that the summation of all interval units allocation should not exceed the maximum limit, i.e., τ_{MAX}.

$$\left(\sum_{a=1}^{A} \tau_a + \sum_{r=1}^{R} \tau_r + \sum_{s=1}^{S} \tau_s \right) \le \tau_{\text{MAX}}. \quad (10)$$

On the other hand, there must be constraints while performing associations between the physical space entities with the meta spaces for various tasks (i.e., task offloading and requesting a service). To do so, one must perform a unique association between the physical space entity and the meta space.

$$\sum_{m\in\mathcal{M}} y^{(n\to m)}(t) \le 1, \forall n \in \{\mathcal{A} \cup \mathcal{R} \cup \mathcal{S}\}. \quad (11)$$

On the other hand, every meta space has a certain capacity to serve physical space entities. For instance, a meta space needs computing resources, storage, and wireless resources to serve the physical space entities. Computing resources are used for various purposes, such as partial/complete task computation and aggregation of local models in the case of distributed training of metaverse models. Furthermore, storage resources are needed to store the temporary data or trained metaverse models for future use. Communication resources are used by the meta space

$$\ell^n_{\text{trans}}(\boldsymbol{y}, \boldsymbol{\tau}, \boldsymbol{\phi}) = \begin{cases} \frac{(1+\theta_n) y^{(n\to m)}(t) o_n(t) U_a(t)}{\tau_n W_j(t) \mathbb{E}_{h^n_{d,m}} \left(\log_2 \left(1 + \frac{\phi_n(t) h^n_{a,m}(t)}{\sum_{c\in\mathcal{C}} p_c(t) h_{c,m}(t) + N_o^2} \right) \right)}, & \forall n \in \mathcal{A}, \\ \frac{y^{(n\to m)}(t) o_n(t) U_r(t)}{\tau_n W_j(t) \mathbb{E}_{h^n_{d,m}} \left(\log_2 \left(1 + \frac{\phi_n(t) h^n_{a,m}(t)}{\sum_{c\in\mathcal{C}} p_c(t) h_{c,m}(t) + N_o^2} \right) \right)}, & \forall n \in \mathcal{R}, \\ \frac{y^{(n\to m)}(t) o_n(t) I_t(t) \epsilon\eta(x_1,x_2,x_3)(t)}{\tau_n W_r \mathbb{E}_{h^n_{d,m}} \left(\log_2 \left(1 + \frac{p^n_d h^n_{d,m}}{\sum_{c\in\mathcal{C}} p_c h_{c,m} + N_o^2} \right) \right)}, & \forall n \in \mathcal{S}, \end{cases} \quad (7)$$

for communication with the physical space entities. All of the aforementioned resources are limited. Therefore, the number of physical space entities associated with the meta space should be within the limit.

$$\sum_{n\in\mathcal{A}}\sum_{n\in\mathcal{R}}\sum_{n\in\mathcal{S}} y^{(n\to m)}(t) \le y_m^{\text{MAX}}, \forall m \in \mathcal{M}, \tag{12}$$

where y_m^{MAX} denotes the maximum threshold of physical space entities that can be served by the meta space m. On the other hand, there is a limited availability of transmit power. Therefore, the transmit power should be within limits:

$$\phi_{\min} \le \phi_n(t) \le \phi_{\max}, \forall n \in \mathcal{N}, \tag{13}$$

where $\phi_{\min}$ and $\phi_{\max}$ are the lower and maximum limit of transmit power a certain entity can take. Meanwhile, there are limitations on the availability of transmit power for all the metaverse entities as indicated by the following constraint.

$$\sum_{n=1}^{N} \phi_n(t) \le \phi_{\text{MAX}}, \tag{14}$$

where ϕ_{MAX} denotes the maximum transmit power that can be allocated to all devices. It is desirable to have such kind of constraint due to the fact that in every system we have energy limitations. Therefore, we must have a constraint to put limitations on the overall assignment of transmit power to all devices. On the other hand, in the metaverse, we need both QoS and QoE metrics. Therefore, in the next section, we present both QoE and QoS constraints.

D. QoE and QoS Constraints

In the case of the metaverse, the nature of traffic/data is generally different and involves mainly visual data for various applications. For instance, if we consider an AR/VR headset, then there is a need for immersive transmission (i.e., data, visuals, audio, and interactive experiences) from end-devices to the meta spaces deployed at edge servers. For such kind of immersive transmission, there is a need for novel performance metrics. Therefore, we should take care of traditional QoS metrics in addition to QoE metrics.

For QoS, we can apply various metrics, i.e., latency and reliability. On the other hand, we should also consider the user experience in terms of QoE metrics. First, we write a QoS metric in terms of reliability as follows:

$$\Pr\left[\left(\sum_{n=1}^{N}\zeta_n\right) \le \mathcal{E}_n\mathcal{J}_n\right] \le \Theta_{\text{MAX}},$$

$$\forall n \in \mathcal{N} \in \{\mathcal{A}\cup\mathcal{S}\cup\mathcal{R}\}. \tag{15}$$

Note here that the value of the product, i.e., $\mathcal{E}_n\mathcal{J}_n$, determines the reliability of communication in the metaverse. A small value will lead to higher reliability and vice versa. However, it is very hard to meet a very high reliability. To meet a very high reliability, we will need better communication conditions and more communication resources. Meanwhile, communication resources are limited, and the channel has uncertainties as well. Therefore, we should make a trade-off between reliability and communication cost. Similar to (15), we can write the QoS in terms of latency as:

$$\alpha\left(\Pr\left[\left(\sum_{n=1}^{N}\zeta_n\right) \le \mathcal{E}_n\mathcal{J}_n\right]\right) + (1-\alpha)\,\ell_{\text{trans}}^n \le \Xi_n,$$
$$\forall n \in \mathcal{N}, \mathcal{N} \in \{\mathcal{A}\cup\mathcal{S}\cup\mathcal{R}\}. \tag{16}$$

where Ξ_n denotes the maximum threshold beyond which the summation of latency and reliability term should not exceed. In (16), α is a constant that scales between reliability and latency while computing the QoS constraint. For instance, for some applications (e.g., telemedicine and pharmaceutical company production), we need more reliability compared to latency. In contrast, some applications (e.g., AR/VR and live video streaming) give more priority to latency compared to reliability. Therefore, one can scale the proportion of latency and reliability while computing the QoS constraint in (16).

In a typical metaverse-based system, we want our communication to fulfill the QoE constraints in addition to QoS constraints. For our case, we consider immersive experience and packet error rate as follows:

$$\underbrace{\beta\left(\frac{1}{1-\exp\left(\frac{-\vartheta\left(\sum_{y\in\mathcal{Y}_r} h_y^r P_y^r+\sigma^2\right)}{\phi_n h_{n,m}^r}\right)}\right)}_{\text{Packet error rate}} + \underbrace{(1-\beta)\sum_{n=1}^{N} T_n ln\left(\frac{\zeta_n}{\zeta_{\min}}\right)}_{\text{Subjective feeling}} \ge \Psi_n,\ \forall n \in \mathcal{N} \in \{\mathcal{S}\cup\mathcal{R}\}, \tag{17}$$

where $\sum_{n=1}^{N} T_n ln(\frac{\zeta_n}{\zeta_{\min}})$ denotes the immersive experience term. We want to maximize the immersive experience term for a better realization of a physical world using virtual experience. In contrast, we want to minimize the packet error rate, i.e.,

$$\varkappa_{\text{trans}}^n(\boldsymbol{y},\boldsymbol{\tau},\boldsymbol{\phi}) = \begin{cases} \dfrac{(1+\theta_n)y^{(n\to m)}(t)o_n(t)U_a(t)\phi_n}{\tau_n W_j(t)\mathbb{E}_{h_{d,m}^n}\left(\log_2\left(1+\frac{\phi_n(t)h_{a,m}^n(t)}{\sum_{c\in\mathcal{C}}p_c(t)h_{c,m}(t)+N_o^2}\right)\right)}, & \forall n\in\mathcal{A}, \\ \dfrac{y^{(n\to m)}(t)o_n(t)U_r(t)\phi_n}{\tau_n W_j(t)\mathbb{E}_{h_{d,m}^n}\left(\log_2\left(1+\frac{\phi_n(t)h_{a,m}^n(t)}{\sum_{c\in\mathcal{C}}p_c(t)h_{c,m}(t)+N_o^2}\right)\right)}, & \forall n\in\mathcal{R}, \\ \dfrac{y^{(n\to m)}(t)o_n(t)\phi_n I_t(t)\epsilon\eta(x_1,x_2,x_3)(t)}{\tau_n W_r\mathbb{E}_{h_{d,m}^n}\left(\log_2\left(1+\frac{p_d^n h_{d,m}^n}{\sum_{c\in\mathcal{C}}p_c h_{c,m}+N_o^2}\right)\right)}, & \forall n\in\mathcal{S}, \end{cases} \tag{8}$$

$\left(1-\exp\left(\frac{-\vartheta(\sum_{y\in\mathcal{Y}_r} h_y^r P_y^r+\sigma^2)}{\phi_n h_{n,m}^r}\right)\right)$ [21]. Both packet error rate and immersive experience terms are conflicting. Therefore, we use the term $\left(\frac{1}{1-\exp\left(\frac{-\vartheta(\sum_{y\in\mathcal{Y}_r} h_y^r P_y^r+\sigma^2)}{\phi_n h_{n,m}^r}\right)}\right)$ in (17). Ψ_n is the threshold such that the scaled summation of the packet error rate term and the immersive experience term should be more than or equal to it. T_n denotes the n^{th} metaverse user attention for the environment. The immersive experience engages the user's senses and attention to enable a better feeling of virtual worlds similar to that of the physical world. The time of attention determines the importance of that object. Specifically, we can use AR/VR devices to create immersive experiences in a metaverse. In the case of rendering sources (AR/VR headsets), more attention is given to the objects when we stare more and vice versa. In (17), we want to maximize the QoE for a better realization of the virtual world. Meanwhile, the scaling constant β enables us to scale between the immersive experience and the packet error rate. Next, we present our problem formulation.

E. Problem Formulation

In this section, we write a cost function that will consider both latency and energy. Meanwhile, we consider both QoE and QoS constraints. First, we write an expression for the overall metaverse latency, i.e., transmission latency for service-requesting devices, learning devices, sensing units, and local computing latency for learning devices.

$$\ell_{\text{Overall}}(\boldsymbol{y},\boldsymbol{\tau},\boldsymbol{\phi},\boldsymbol{\kappa}) = \underbrace{\sum_{r=1}^{R}\ell_{trans}^{r}+\sum_{a=1}^{A}\ell_{trans}^{a}}_{\text{Trans. latency for learning devices and requesting users}} + \underbrace{\left(\iiint_V \ell_{trans}^{s} f(x_1,x_2,x_3)\,dx_1\,dx_2\,dx_3\right)}_{\text{Trans. latency for sensing}} + \underbrace{\sum_{n\in(\mathcal{A}\cup\mathcal{R})}\ell_{local}^{n}+\iiint_V \ell_{local}^{s} f(x_1,x_2,x_3)\,dx_1\,dx_2\,dx_3,}_{\text{Local computing latency}} \tag{18}$$

where $f(x_1,x_2,x_3)$ is a distribution function for metaverse sensors in the physical space. As we consider the distribution of sensors in a continuous fashion similar to other works [19], therefore, we use the distribution function to model the sensors while computing latency in (18). Furthermore, the learning latency at end-devices is also important for distributed training of meta space models. Therefore, we consider the local computing latency of learning devices, sensing latency, and service requesting devices latency while computing latency cost in (18). Similar to (18), one can write the energy cost in terms of sensing transmission energy, learning transmission energy, service-requesting transmission energy, and local meta space model computing energy, as follows.

$$\chi_{\text{Overall}}(\boldsymbol{y},\boldsymbol{\tau},\boldsymbol{\phi},\boldsymbol{\kappa}) = \underbrace{\sum_{r=1}^{R}\varkappa_{trans}^{r}+\sum_{a=1}^{A}\varkappa_{trans}^{a}}_{\text{Trans. energy for learning devices and requesting users}} + \underbrace{\left(\iiint_V \varkappa_{trans}^{s} f(x_1,x_2,x_3)\,dx_1\,dx_2\,dx_3\right)}_{\text{Trans. energy for sensing}} + \underbrace{\sum_{n\in(\mathcal{A}\cup\mathcal{R})}\chi_{local}^{n}+\iiint_V \chi_{local}^{s} f(x_1,x_2,x_3)\,dx_1\,dx_2\,dx_3\,.}_{\text{Local computing energy}} \tag{19}$$

Similar to (18), we also consider the continuous distribution of sensors and thus we use the distribution function while computing the sensing transmission energy. Now, we write the total cost that considers both energy and latency as follows.

$$\mathcal{C}(\boldsymbol{y},\boldsymbol{\tau},\boldsymbol{\phi},\boldsymbol{\kappa}) = \iota\ell_{\text{Overall}}(\boldsymbol{y},\boldsymbol{\tau},\boldsymbol{\phi},\boldsymbol{\kappa}) + (1-\iota)\chi_{\text{Overall}}(\boldsymbol{y},\boldsymbol{\tau},\boldsymbol{\phi},\boldsymbol{\kappa}), \tag{20}$$

where $\iota\in(0,1)$ is a scaling constant that scales the proportion of the latency and energy in computing the cost for the proposed system. Our aim is to minimize the cost of task offloading in the metaverse while fulfilling the QoS and QoE constraints. Now, we formulate a problem to minimize the cost as follows.

$$\mathbf{P}: \quad \underset{\boldsymbol{y},\boldsymbol{\tau},\boldsymbol{\phi},\boldsymbol{\kappa}}{\text{minimize}} \ \mathcal{C}(\boldsymbol{y},\boldsymbol{\tau},\boldsymbol{\phi},\boldsymbol{\kappa}) \quad \text{subject to:} \quad (3)-(5),(9)-(14),(16),(17). \tag{21}$$

The formulated problem **P** is a MINLP and non-convex. Constraint (3) sets a limit on the maximum local computing resources for all learning devices. Constraint (4) sets the upper and lower limits of the computing resources of the end devices. Constraint (5) restricts local computing and transmission latency should not exceed the maximum limit. Constraint (9) sets limitations on the minimum and maximum limit of time interval allocations for TDMA. Another constraint (10) sets a limit on the maximum time allocation for all entities. Constraint (11) is about the task offloading for a physical space entity with a single meta space. Constraint (12) is about the maximum limit of physical space entities connected with the meta spaces. Constraint (13) refers to the maximum and lower limits of the transmit power. Constraint (14) is about the maximum limit of the transmit power assigned to all devices. Constraint (16) ensures the QoS in terms of latency and reliability. Constraint (17) ensures QoE for metaverse task offloading. Solving **P** is very difficult to solve due to the presence of a probabilistic constraint (i.e., reliability) in computing QoS. Additionally, the problem is non-convex and has both binary and continuous variables. Therefore, we will propose a solution that is based on decomposition in the next section.

III. BSUM, Convex Optimization, and Dueling DDQN-Enabled Solution

In this section, we present our proposed solution. The formulated problem **P** is a non-convex problem. Furthermore, it is hard to solve due to its MINLP nature as well as the probabilistic constraint, i.e., constraint (15). Therefore, there is a need to transform this probabilistic constraint into a linear constraint.

$$\Pr\left[\left(\sum_{n=1}^{N}\zeta_n\right)\leq \mathcal{E}_n\mathcal{J}_n\right]\leq \Theta_{\text{MAX}} \quad \forall n\in\mathcal{N}\in\{\mathcal{A}\cup\mathcal{S}\cup\mathcal{R}\}. \tag{22}$$

In (22), there is a need for simplifying the probabilistic constraint, $\Pr[(\sum_{n=1}^{N}\zeta_n)\leq \mathcal{E}_n\mathcal{J}_n]\leq\Theta_{\text{MAX}}$ into a linear constraint. First, we compute the complementary event, i.e., $\Pr[(\sum_{n=1}^{N}\zeta_n)> \mathcal{E}_n\mathcal{J}_n]$ as follows:

$$\Pr\left[\left(\sum_{n=1}^{N}\zeta_n\right)> \mathcal{E}_n\mathcal{J}_n\right]=1-\Pr\left[\left(\sum_{n=1}^{N}\zeta_n\right)\leq \mathcal{E}_n\mathcal{J}_n\right]. \tag{23}$$

Using the fact in (23), one can write (22) as.

$$\Pr\left[\left(\sum_{n=1}^{N}\zeta_n\right)> \mathcal{E}_n\mathcal{J}_n\right]\geq 1-\Theta^{n}_{\text{MAX}}. \tag{24}$$

Using Markov inequality and expectation, one can write:

$$\Pr\left(\sum_{n=1}^{N}\zeta_n\leq \mathcal{E}_n\mathcal{J}_n\right)=\Pr\left(\mathcal{E}_n\mathcal{J}_n\geq\sum_{n=1}^{N}\zeta_n\right)\leq\frac{\mathbb{E}[\mathcal{E}_n\mathcal{J}_n]}{\sum_{n=1}^{N}\zeta_n}. \tag{25}$$

To enforce $\Pr[(\sum_{n=1}^{N}\zeta_n)\leq \mathcal{E}_n\mathcal{J}_n]\leq\Theta_{\text{MAX}}$ holds, one can write as:

$$\frac{\mathbb{E}[\mathcal{E}_n\mathcal{J}_n]}{\sum_{n=1}^{N}\zeta_n}\leq\Theta^{n}_{\text{MAX}}. \tag{26}$$

Rearranging, we can write the probabilistic constraint into a linear form as:

$$\sum_{n=1}^{N}\zeta_n\geq\frac{\mathbb{E}[\mathcal{E}_n\mathcal{J}_n]}{\Theta^{n}_{\text{MAX}}}. \tag{27}$$

Now, one can write the constraint (22) as:

$$\sum_{n=1}^{N}\zeta_n\geq\frac{\mathbb{E}[\mathcal{E}_n\mathcal{J}_n]}{\Theta^{n}_{\text{MAX}}},\ \forall n\in\mathcal{N}\in\{\mathcal{A}\cup\mathcal{S}\cup\mathcal{R}\}. \tag{28}$$

We will use the linear constraint of (28) in our problem solution. From now on, for QoS, we will consider the linear form of the simplified reliability constraint (28) and the transmission latency constraint, separately. If we look at the problem, it is still hard to solve due to the presence of many variables, i.e., continuous and binary variables. Therefore, we use a decomposition-based scheme, as shown in Fig. 2. First, we solve the local computing power allocation problem.

A. Local Computing Resource Allocation

In our system, the set of learning devices learn local models and then share them with the meta space. Furthermore, other devices also perform local tasks. All the devices use local computing resources which are limited in nature and thus, we need to allocate them wisely. The local computing resource allocation problem can be written as:

$$\mathbf{P}-\mathbf{C}:\ \underset{\boldsymbol{\kappa}}{\text{minimize}}\ \mathcal{C}(\boldsymbol{\kappa}) \tag{29}$$

subject to:

$$\kappa_{\min}\leq\kappa_n\leq\kappa_{\max},\forall n\in\{\mathcal{A}\cup\mathcal{R}\in\mathcal{S}\}, \tag{29a}$$

$$\sum_{n\in\mathcal{A}}\sum_{n\in\mathcal{R}}\sum_{n\in\mathcal{S}}\kappa_n\leq\kappa_{\text{MAX}}, \tag{29b}$$

$$(\ell^{n}_{local}+\ell^{n}_{trans})\leq z_n,\forall n\in\{\mathcal{A}\cup\mathcal{R}\cup\mathcal{S}\}. \tag{29c}$$

Problem **P-C** is a convex optimization problem whose convexity can be proved by checking the objective function as well as constraints. If we take the twice derivative of the objective function, i.e., $\mathcal{C}(\boldsymbol{\kappa})$ in (29). The value of twice derivative of the cost for learning devices, service-requesting devices, and sensing units are greater than or equal to zero. Meanwhile, the constraints are linear inequality constraints. Therefore, we can say that the problem **P-C** is a convex optimization problem and one can use a convex optimizer (e.g., CVX optimizer in python) for its solution. Next, we consider the transmit power allocation problem.

B. Transmit Power Allocation Using BSUM

In this section, we keep task offloading and resource allocation fixed and then write the expression for transmit power allocation. For transmit power allocation, one can write the problem as:

$$\mathbf{P}-\phi:\ \underset{\boldsymbol{\phi}}{\text{minimize}}\ \mathcal{C}(\boldsymbol{\phi}) \tag{30}$$

subject to:

$$\phi_{\min}\leq\phi_n(t)\leq\phi_{\max},\forall n\in\mathcal{N}, \tag{30a}$$

$$(\ell^{n}_{local}+\ell^{n}_{trans})\leq z_n,\forall n\in\{\mathcal{A}\cup\mathcal{R}\cup\mathcal{S}\}, \tag{30b}$$

$$\sum_{n=1}^{N}\zeta_n\geq\frac{\mathbb{E}[\mathcal{E}_n\mathcal{J}_n]}{\Theta^{n}_{\text{MAX}}}, \tag{30c}$$

$$\ell^{n}_{\text{trans}}\leq\Delta_n,\ \forall n\in\mathcal{N}\in\{\mathcal{A}\cup\mathcal{S}\cup\mathcal{R}\}, \tag{30d}$$

$$\beta\left(\frac{1}{1-\exp\left(\frac{-\vartheta(\sum_{y\in\mathcal{Y}_r}h^{r}_{y}P^{r}_{y}+\sigma^2)}{\phi_n h^{r}_{n,m}}\right)}\right)+(1-\beta)\sum_{n=1}^{N}T_n ln\left(\frac{\zeta_n}{\zeta_{\min}}\right)\geq\Psi_n, \tag{30e}$$

$$\sum_{n=1}^{N}\phi_n(t)\leq\Theta_{\text{MAX}}. \tag{30f}$$

Problem **P-**ϕ has a non-convex nature and has a continuous power allocation variable, i.e., ϕ. Therefore, one can not use

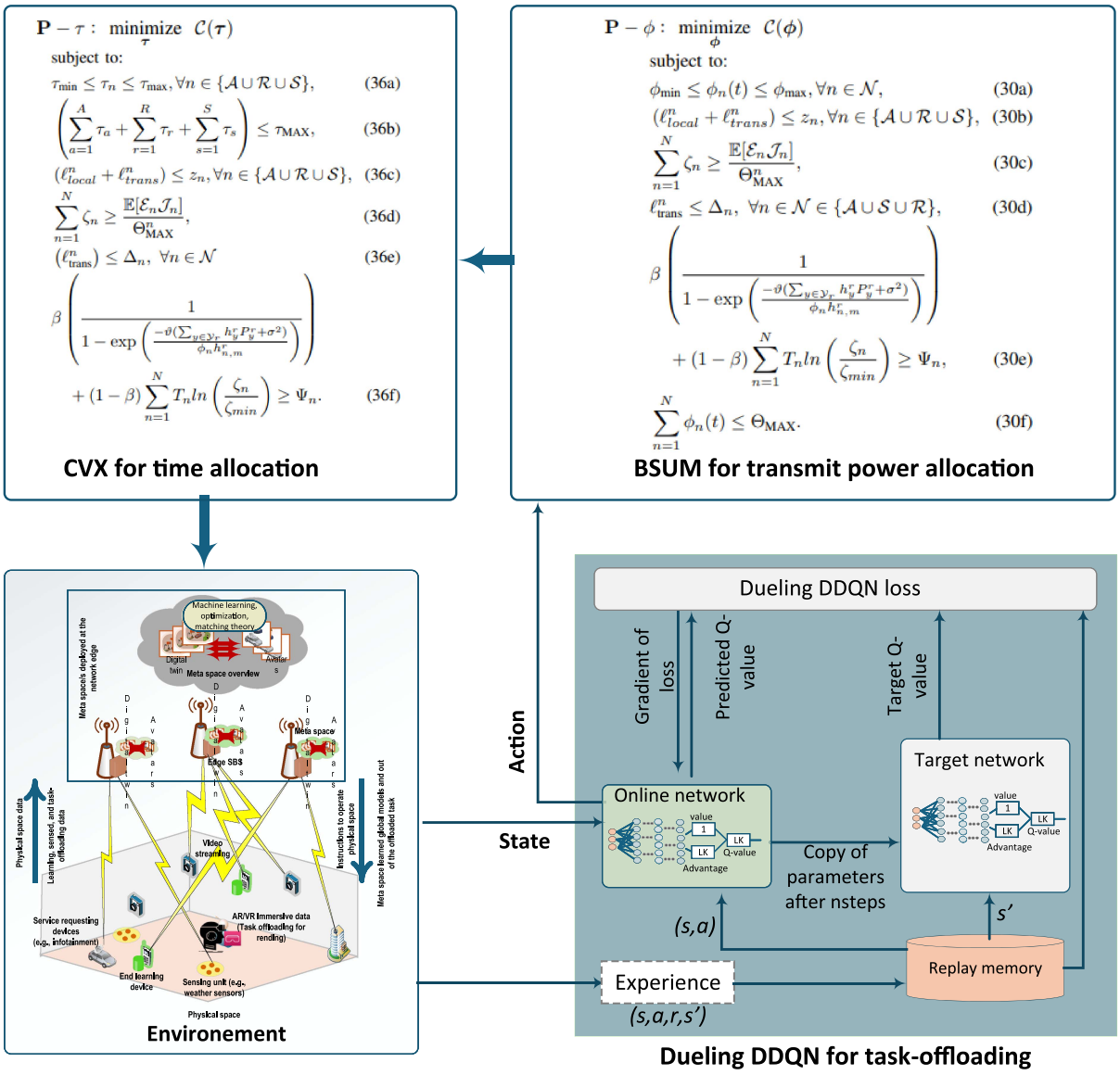


Fig. 2. Proposed solution approach.

convex optimization. To solve problem **P-**ϕ, we use BSUM that is based on the addition of a proximal term of the objective function, i.e., $\mathcal{C}(\phi)$ [22], [23]. Instead of minimizing $\mathcal{C}(\phi)$, we will minimize the $\mathcal{C}_{\text{BSUM}}(\phi)$. We discuss here the preliminaries for BSUM as follows.

$$\min_{\gamma} g(\gamma_1, \gamma_2, \ldots, \gamma_J),$$
$$\text{s.t.} \quad \gamma_J \in \mathbf{\Gamma_j}, \forall j \in \mathcal{J}, j = 1, 2, 3, \ldots, J. \tag{31}$$

where $\mathbf{\Gamma_j} := \mathbf{\Gamma_1} \times \mathbf{\Gamma_2} \times \ldots, \mathbf{\Gamma_J}$. Furthermore, $g(\cdot)$ is continuous in nature. $\boldsymbol{\gamma_j}$ denotes the block of variables. $\mathbf{\Gamma_j}$ is a closed convex set. To solve a block of variables, we can use block-coordinated descent (BCD) [24], [25].

$$\mathbf{\Gamma}_j^t \in \underset{\gamma_j \in \mathbf{\Gamma}_j}{\text{argmin}}\, g(\gamma_j, \gamma_{-j}^{t-1}), \tag{32}$$

where $\gamma_{-j}^{t-1} := (\gamma_1^{t-1}, \ldots, \gamma_{j-1}^{t-1}, \gamma_{j+1}^{t-1}, \ldots, \gamma_j^{t-1})$, $\gamma_k^t = \gamma_k^{t-1}$ for $j \neq k$. Solving the power allocation problem using BCD is very challenging due to the non-convex nature of the problem. In contrast to BCD, which is based on minimizing the objective function, BSUM uses the minimization of the upper-bound function $b(\gamma_o, \boldsymbol{\lambda})$ of $g(\gamma_o, \boldsymbol{\lambda}_{-j})$. Note here that the reason for adding a proximal term is to make the non-convex objective function convex. To do so, different upper-bound functions (e.g., Jensen's upper-bound and quadratic upper-bound) can be used. We use a quadratic upper-bound function. For applying BSUM, we make the following assumptions:

Assumptions:

1) $b(\gamma_j, \boldsymbol{\lambda}) = g(\boldsymbol{\lambda})$,

2) $b(\gamma_j, \boldsymbol{\lambda}) > g(\gamma_j, \boldsymbol{\lambda}_{-j})$,

3) $b'(\gamma_j, \boldsymbol{\lambda}; \boldsymbol{q}_j)|_{\gamma_j = \boldsymbol{\lambda}_j} = \nabla(\boldsymbol{\lambda}; \boldsymbol{q}), \boldsymbol{\lambda}_j + \boldsymbol{q}_j \in \Gamma_j$.

The upper-bound nature of b is ensured by the assumptions (1) and (2). The upper-bound function steps in BSUM are in the direction of q and negative of the objective function's gradient. On the other hand, for a penalty, we use a quadratic one as follows:

$$b(\boldsymbol{\gamma}_j, \boldsymbol{\lambda}) = g(\boldsymbol{\gamma}_j, \boldsymbol{\lambda}_{-j}) + \frac{\mu}{2} \parallel \boldsymbol{\gamma}_j - \boldsymbol{\lambda}_j \parallel^2 . \tag{33}$$

In (33), $\mu > 0$ denotes the positive scaling term. Furthermore, the upper-bound function can be solved as follows.

$$\begin{cases} \boldsymbol{\gamma}_j^t \in \min\limits_{\boldsymbol{\gamma}_j \in \Gamma_j} b(\boldsymbol{\gamma}_j, \boldsymbol{\gamma}_j^{t-1}), \forall j \in \mathcal{J}, \\ \boldsymbol{\gamma}_j^t = \boldsymbol{\gamma}_k^{t-1}, \forall k \notin \mathcal{J}. \end{cases} \tag{34}$$

In a typical BSUM, successive updating of blocks of variables is performed with the aim of minimizing the upper-bound function. A variables block should reach a minimal point in BSUM, i.e., $\boldsymbol{\gamma}^* = \boldsymbol{\gamma}_j^{t+1}$. For BSUM, we can write our problem as:

$$\min_{\phi \in \boldsymbol{\phi}} \mathcal{C}_{\text{BSUM}}(\boldsymbol{\phi}), \tag{35}$$

where $\mathcal{C}_{\text{BSUM}}(\boldsymbol{\phi})$ is the upper-bound function of the cost function $\mathcal{C}$. Meanwhile, the feasible set can be written as:

$$\begin{aligned} \boldsymbol{\phi} \triangleq \{\phi : \phi_{\min} \leq \phi_n \leq \phi_{max}, \forall n \in \mathcal{N}, \sum_{n=1}^{N} \phi_n \leq \phi_{\text{MAX}}, \\ \left(\sum_{n=1}^{N} \zeta_n \geq \frac{\mathbb{E}[\mathcal{E}_n \mathcal{J}_n]}{\Theta_{\text{MAX}}^n} \right), (\ell_{\text{trans}}^n) \leq \Delta_n, \forall n \in \mathcal{N}, \\ \beta \left(\frac{1}{1 - \exp\left(\frac{-\vartheta(\sum_{y \in \mathcal{Y}_r} h_y^r P_y^r + \sigma^2)}{\phi_n h_{n,m}^r} \right)} \right) \\ + (1 - \beta) \sum_{n=1}^{N} T_n ln \left(\frac{\zeta_n}{\zeta_{\min}} \right) \geq \Psi_n \}. \end{aligned}$$

In our problem, we are solving only transmit power allocation using BSUM. Therefore, one can solve (35) while using the aforementioned feasible set.

C. Wireless Resource Optimization

In this section, we keep the task offloading variable and transmit power allocation variables fixed and then, write the TDMA time interval optimization sub-problem as follows:

$$\mathbf{P} - \tau : \quad \underset{\boldsymbol{\tau}}{\text{minimize}} \ \mathcal{C}(\boldsymbol{\tau}) \tag{36}$$

subject to:

$$\tau_{\min} \leq \tau_n \leq \tau_{\max}, \forall n \in \{\mathcal{A} \cup \mathcal{R} \cup \mathcal{S}\}, \tag{36a}$$

$$\left(\sum_{a=1}^{A} \tau_a + \sum_{r=1}^{R} \tau_r + \sum_{s=1}^{S} \tau_s \right) \leq \tau_{\text{MAX}}, \tag{36b}$$

$$(\ell_{local}^n + \ell_{trans}^n) \leq z_n, \forall n \in \{\mathcal{A} \cup \mathcal{R} \cup \mathcal{S}\}, \tag{36c}$$

$$\sum_{n=1}^{N} \zeta_n \geq \frac{\mathbb{E}[\mathcal{E}_n \mathcal{J}_n]}{\Theta_{\text{MAX}}^n}, \tag{36d}$$

Algorithm 1. Dueling DDQN for Task Offloading in Metaverse.

1: Initialize target replacement frequency N^-, DDQN network parameters θ, and replay memory $\mathcal{D}$.
2: Online network initialization $Q_n(g, z_n; \theta)$ with weights θ.
3: Target network initialization $Q_n(g', z_n; \theta^-)$ with weights $\theta^- = \theta$.
4: **for** each episode $e = 1$ to EP **do**
5: Initialize network state g through message passing.
6: **for** each step $t = 1$ to T **do**
7: ε-greedy policy from $Q_n(g, z_n; \theta)$ is used by every metaverse entity to select an action z_n at state g.
8: An immediate reward $u_i(g, z_i)$ is obtained using selected action z_i.
9: Every metaverse entity obtains next network state g' by message passing and makes $g \leftarrow g'$.
10: The transition $(g, z_n, u_n(g, z_n), g')$ in $\mathcal{D}$ is stored by all metaverse entities.
11: From $\mathcal{D}$, every metaverse entity randomly samples mini-batch of transitions $(g, z_n, u_n(g, z_n), g')$.
12: Eq. (48) is used by every metaverse entity to set target y_n^{DDQN} with dueling.
13: Each metaverse entity performs a gradient descent step to minimize the loss function.
14: **if** $t \bmod N^- = 0$ **then**
15: The target parameters $\theta^- \leftarrow \theta$ by every metaverse entity.
16: **end if**
17: **end for**
18: **end for**

$$(\ell_{\text{trans}}^n) \leq \Delta_n, \ \forall n \in \mathcal{N} \tag{36e}$$

$$\begin{aligned} \beta \left(\frac{1}{1 - \exp\left(\frac{-\vartheta(\sum_{y \in \mathcal{Y}_r} h_y^r P_y^r + \sigma^2)}{\phi_n h_{n,m}^r} \right)} \right) \\ + (1 - \beta) \sum_{n=1}^{N} T_n ln \left(\frac{\zeta_n}{\zeta_{\min}} \right) \geq \Psi_n. \end{aligned} \tag{36f}$$

In problem **P-**τ, constraint (36a) denotes the upper and lower limit of the time interval. Constraint (36b) limits the total time intervals allocated to all entities to not exceed the overall time interval. Constraint (36c) limits the maximum latency in communication. Constraints (36d) and (36e) fulfill the QoS in terms of reliability and latency. Finally, constraint (36f) ensures the QoE in terms of PER and immersive experience. Problem **P-**τ is a convex optimization problem whose complexity can be proved by checking the objective function as well as constraints. Take the second derivative of the objective function, i.e., $\mathcal{C}(\boldsymbol{\tau})$ in (36). The value of twice derivative of the cost for learning devices, service-requesting devices, and sensing units are greater than or equal to zero. Meanwhile, the constraints are linear inequality constraints. Therefore, we can say that the problem **P-**τ is a convex optimization problem and one can use a convex

optimizer (e.g., CVX optimizer in python) for its solution. Next, we consider task-offloading problem.

D. Task Offloading Optimization Using MARL

In this section, for a fixed transmit power allocation and time interval allocation, we consider task offloading problem $\boldsymbol{P}-\boldsymbol{y}$ (i.e., cost function of (20) along with constraints (5), (11), (12), linear form of (16), and (17)). Problem $\boldsymbol{P}-\boldsymbol{y}$ has a combinatorial, non-convex nature. This makes it very difficult to solve it. Therefore, similar to other works [26], [27], [28], [29], we consider a MARL scheme. Maximizing the long-term reward is MARL's aim. The objective of problem **P-$\boldsymbol{y}$** is to minimize the cost function. The goal of MARL, on the other hand, is to maximize the long-term reward. The reward function is thus expressed as $\mathcal{R}_n = \frac{1}{\mathcal{C}}$. The total of the immediate rewards from several devices is the long-term reward in MARL.

$$\widehat{\Delta}_n = \sum_{t=0}^{T-1} \gamma^t \Delta_n(t) \tag{37}$$

where the discount rate, γ^t, indicates the weight of the future reward. We are clearly only taking into account a present benefit, with no consideration of future gains, for $\gamma^t = 0$. Regarding $\gamma^t \leq 1$, it is clear that future rewards are less significant than past rewards for calculating the long-term reward, and vice versa. Before applying MARL, it should be formulated as a stochastic game. The goal of our system is to optimize task offloading so that all users can maximize their long-term reward. At a given time, t, the reward is based on the agent's current state as well as the states of other agents. The following is a possible formulation for the stochastic game $\langle \mathcal{N}, \mathcal{G}, \mathcal{Z}, \mathcal{P}, \Delta \rangle$ [30]).

- $\mathcal{N}$ is the set of N metaverse physical space entities.
- $\mathcal{G}$ is the state space.
- $\mathcal{Z}_n$ is entity n action. The action vector for task offloading is given by $\overrightarrow{z} = (z_1, z_2, z_3, \ldots, z_N)^T \in \times_n \mathcal{Z}_n$.
- $\mathcal{P}$ is the state transition probability.
- Δ_n denotes the immediate reward of entity n.

In our system, we use MARL for task offloading only. Therefore, one can write the action space as:

$$z_n(t) = \{y^{n\to m}(t)\}. \tag{38}$$

Where $y^{n\to m}(t) \in \{0, 1\}$, and $y^{n\to m}(t)$ belongs to the set $\{y^{n\to 0}(t), y^{n\to 1}(t), \ldots, y^{n\to(M-1)}(t)\}$. In our solution, the total number of states is M. The actions of other devices are represented by $\mathcal{Z}-n$ at a specific time t for a given device n. The set of these actions is given by $\boldsymbol{\mathcal{Z}}-n(t) = z_0(t), \ldots, z_{n-1}(t), z_{n+1}(t), \ldots, z_{N-1}(t)$. The immediate reward of each device n, denoted as $\Delta_n(g, \mathcal{Z}n, \boldsymbol{\mathcal{Z}}-n)$, depends on the actions of other devices $\boldsymbol{\mathcal{Z}}-n$ as well as its own action $\mathcal{Z}n$ at a given time t. Moreover, Nash Equilibrium (NE) can be used to determine whether the stochastic game is stable. The following conditions are established to reach a NE.

$$\Delta_n(g, z_n^*, \boldsymbol{z}_{-n}^*) \geq \Delta_n(g, z_n, \boldsymbol{z}_{-n}^*), \ \forall z_n \in \mathcal{Z}_n. \tag{39}$$

We use a finite state Markov decision process (MDP) to model our stochastic game. The device reward is determined by the states of other devices and its current state. This shows that our process complies with the Markov property. $\langle \mathcal{N}, \mathcal{G}, \mathcal{Z}, \mathcal{P}, \Delta \rangle$ in our system define MDP. There are two ways to design a reward function. Selecting the throughput is one approach, whereas considering the cost function's reciprocal is another approach. As with many reinforcement learning works, the $\frac{1}{\mathcal{C}}$ can be used as a reward. Our goal is to learn a policy $\pi_n^* : \mathcal{G} \to \mathcal{Z}_n$ in a typical stochastic environment. This $\pi_n^* : \mathcal{G} \to \mathcal{Z}_n$ policy aims to maximize agents long-term rewards. Devices in MARL communicate with each other by sending messages over direct wireless links or backhaul communication links. The optimal policy π_i^* is obtained by maximizing the value-state function

TABLE I
SUMMARY OF THE KEY NOTATIONS

Notation	Description
	General notations
$\mathcal{N}$	Set of all metaverse entities
$\mathcal{S}$	Set of sensing entities
$\mathcal{A}$	Set of learning devices
$\mathcal{R}$	Set of service-requesting devices
$\mathcal{B}_n$	Computing task of n^{th} metaverse entity
κ_n	Operating frequency of n^{th} metaverse entity
χ_{local}^n	Local computation energy of n^{th} metaverse entity
ζ_n	Throughput of n^{th} metaverse entity
ϕ_n	Transmit power n^{th} metaverse entity
$\eta(x_1, x_2, x_3)$	Unit volumetric density located at x_1, x_2, and x_3
$\mathcal{M}$	Set of meta spaces at the network edge
τ_n	Time interval allocated to metaverse entity n for communication
I_n	Sensing interval of n^{th} sensing entity
$y^{(n\to m)}$	Task offloading variable for n^{th} metaverse entity
ℓ_{trans}^n	Transmission latency of n^{th} metaverse entity
χ_{trans}^n	Transmission energy of n^{th} metaverse entity
Ξ_n	Threshold for the QoS constraint
Ψ_n	Threshold for the QoE constraint
ι	Scaling constant for cost function
β	Scaling constant for QoE constraint
	Notations used in MARL
Δ_n	Reward of entity n
$\mathcal{G}$	States space
$\mathcal{Z}_n$	Action of entity n
$\mathcal{P}$	State transition probability
V_n	Value function
δ	Learning rate
γ	Discount rate

TABLE II
SIMULATION PARAMETERS

Parameter	Value
Channel bandwidth	180 kHz
Transmit power range	30 dBm – 50 dBm
Noise	-174 dBm/Hz
Learning rate	0.01, 0.001, 0.0001
Agents	50, 150, 200
Maximum exploration rate	0.9
Minimum exploration rate	0.0
Steps within episode	20
Discount factor	0.9
Target network update frequency	50
Activation function	ReLU
Optimizer	RMSprop
Replay memory size	500
Mini batch size	8
ϵ-greedy increment	0.003

$V_i(g, \pi_i, \pi_{-i})$ of each metaverse entity in each state:

$$V_n(g, \pi_n, \pi_{-n}) = \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t \Delta_n(g(t), \pi_n(t), \pi_{-n}(t)) \,\middle|\, g(0) = g\right], \quad (40)$$

where the joint policies π_n and π_{-n} yield a set of trajectories. $\mathbb{E}$ denotes the expectation operator. The other agents' strategies than n are indicated by $\pi_{-n} = (\pi_1, \pi_2, .., \pi_{n-1}, \pi_{n+1}, \ldots, \pi_N)$. Given the Markov property, the value function is as follows:

$$V_n(g, \pi_n, \pi_{-n}) = u_n(g, \pi_n, \pi_{-n}) + \gamma \sum_{g' \in \mathcal{G}} \mathcal{P}_{gg'}(\pi_n, \pi_{-n}) V_n(g', \pi_n, \pi_{-n}), \quad (41)$$

where $u_n(g_t, \pi_n, \pi_{-n}) = \mathbb{E}[\Delta_n(g, \pi_n, \pi_{-n})]$ and $\mathcal{P}_{gg'}(\pi_n, \pi_{-n})$ is the state transition probability. If NE exists for π_n, then the following requirement should be satisfied:

$$V_n(g, \pi_n^*, \pi_{-n}^*) \geq V_n(g, \pi_n, \pi_{-n}^*), \quad \forall g \in \mathcal{G}. \quad (42)$$

There is a NE for mixed-strategy equilibrium-based finite games that satisfies the subsequent Bellman optimality:

$$V_n^*(g, \pi_n, \pi_{-n}) = \max_{z_i' \in \mathcal{Z}_i} [u_n(g, z_n, \pi_{-n}^*) + \gamma \sum_{g' \in \mathcal{G}} \mathcal{P}_{gg'}(z_n, \pi_{-n}^*) V_n(g', \pi_n^*, \pi_{-n}^*)]. \quad (43)$$

We can use Q-learning to solve the aforementioned MDP.

$$Q_n^*(g, z_n) = u_n(g, z_n, \pi_{-n}^*) + \gamma \sum_{g' \in \mathcal{G}} \mathcal{P}_{gg'}(z_n, \pi_{-n}^*) V_n(g', \pi_n^*, \pi_{-n}^*), \quad (44)$$

where $V_n(g', \pi_n^*, \pi_{-n}^*) = \max_{z_n \in \mathcal{Z}_n} Q_n^*(g, z_n)$. Rewrite (43) in the following way:

$$Q_n(g, z_n, \pi_{-n}) = u_n(g, z_n, \pi_{-n}^*) + \gamma \sum_{g' \in \mathcal{G}} \mathcal{P}_{gg'}(z_n, \pi_{-n}^*) \max_{z_n' \in \mathcal{Z}_n} Q_n^*(g', z_n'). \quad (45)$$

It is difficult to get the information regarding the transition probability, i.e., $\mathcal{P}_{gg'}$. To address this challenge, the Q-value function $Q_n^*(g, z_n)$ is calculated as follows:

$$Q_n(g, z_n) = Q_n(g, z_n) + \delta\{u_n(g, z_n, \pi_{-n}) + \gamma \max_{z_n' \in \mathcal{Z}_n} Q_n(g', z_n') - Q_n(g, z_n)\}, \quad (46)$$

where the δ symbol for the learning rate helps determine the Q value. Carefully selecting the value of δ is necessary for rapid convergence. Additionally, a trade-off needs to be considered when selecting an activity. This work uses ϵ greedy technique. The chance of selecting the action with the largest reward is $1 - \epsilon$, while the likelihood of selecting the random action is ϵ. The aforementioned Q-learning method is more appropriate for small state and action spaces. Q-learning does not work well in large action spaces. One solution to this problem is to use Deep Q-network (DQN). Two neural networks, i.e., the target network and the online network, are the foundation of DQN. The training procedure involves regular updates to an online network. The online network may become unstable as a result of such frequent updates. DQN employs a target network that is updated less frequently to increase stability. The Q-network is updated using the loss function as follows.

$$L_n(\theta) = \mathbb{E}_{g, z_n, u_n(g, z_n), g'}[(y_n^{DQN} - Q_n(g, z_n; \theta))^2], \quad (47)$$

where $y_n^{DQN} = u_n(g, z_n) + \gamma \max_{z_n' \in \mathcal{Z}_n} Q_n(g' z_n'; \theta^-)$. The weights of the target network are indicated by θ^-. Using the online network $Q_n(g, z_n; \theta)$, an action z_n is chosen. Regular updates are made to the online network's weights. Performance instability could be the result of the weights being updated so frequently. The target network can be updated on a regular basis to address this and increase stability. In order to achieve better results and faster convergence, previous values are also used in the computation in addition to current data. However, overestimation bias is a challenging issue in DQN. The idea of double DQN (DDQN) can be used to address this challenge. DDQN is based on the following.

$$y_n^{DDQN} = u_n(g, z_n) + \gamma Q_n\left(n', \max_{z_n' \in \mathcal{Z}_n} Q_n(g' z_n'; \theta); \theta^-\right). \quad (48)$$

DDQN uses the target network and the online network to get the value $Q_n(g, z_n; \theta)$. The y_n^{DDQN} is then calculated using the current reward and the discount factor. Despite the potential for performance improvement, DDQN may suffer from stability issues. The summary of the proposed solution scheme is given in

Fig. 2. Even though DDQN produces better results, the concept of dueling (i.e., summary is given in Algorithm I) can be used to further enhance DDQN performance [30]. The last layer of the DDQN network can be divided into two networks in order to produce the advantage function and value function. The Q-value is calculated using the value and advantage functions. On the other hand, replay memory is used to store the previous experience in a tuple, which contains the current state, action, next state, and reward. The purpose of replay memory is to improve learning efficiency and to further stabilize the performance of MARL.

E. Complexity Analysis

In this section, we analyze the complexity of the proposed solution. For local computing resource allocation, we use a convex optimizer whose complexity depends mainly on the number of iterations of the CVX optimizer. Generally, CVX optimizer converges fast within a few iterations and thus optimizing local computing resources using CVX optimizer will have a reasonable complexity [31]. On the other hand, we analyze the time interval allocation, task offloading, and transmit power allocation. For time interval allocation, we use a CVX optimizer which has a low complexity as we already discussed. For transmit power allocation, we use BSUM whose complexity can be given by the maximum number of iterations, i.e., $\mathcal{O}(\log(1/\epsilon))$. It is evident that BSUM has a reasonable complexity. Now, finally, we discuss the complexity of DDQN with dueling. In DDQN with dueling, we have two networks: (a) target network and (b) online network. Complexity in DDQN mainly depends on the neural network complexity which in turn is determined by the number of hidden layers, inputs, and outputs. Therefore, overall we can say that DDQN with dueling has a reasonable complexity. Overall, our proposed scheme has a reasonable complexity. Next, we extensively study the performance of the proposed scheme.

IV. Performance Evaluations

A. Simulation Settings

In our simulation settings, we randomly generate metaverse physical space entities. Moreover, the locations of edge meta spaces are kept fixed during a certain run. For computing cost, we consider an average of many runs (e.g., 30). The transmit power range is considered between 30 dBm and 50 dBm. Moreover, a free space path loss model is considered. The value of noise power density considered is -174 dBm/Hz [32]. For analysis, we use learning rates of 0.01, 0.001, and 0.0001. For the DDQN agent, we consider a neural network that consists of $nn.Linear(64, 32)$ and $nn.Linear(32,\ Num.\ of\ BSs\ *\ Resource\ blocks)$, i.e., input layer, two hidden layers, and an output layer [30]. On the other hand, for dueling DDQN, there is a need for splitting the last layer into two parts, i.e., value function and advantage function. Later, the value function and advantage function are used to compute the Q value. Furthermore, for a fair comparison, we consider the following baselines and proposed scheme:

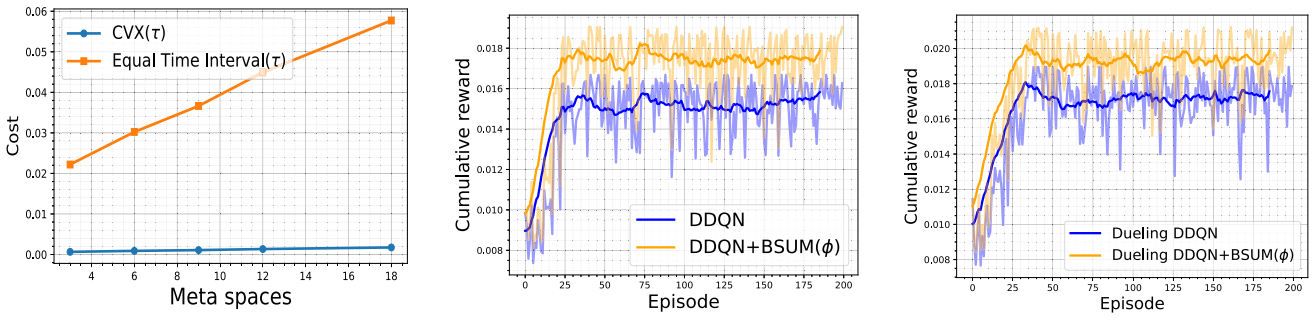
- *DDQN:* In this case, DDQN is used for task offloading, whereas equal time interval and equal transmit power allocation are considered.
- *DDQN+CVX(τ):* In this case, DDQN and CVX optimizer are used for task offloading and time interval allocation. Furthermore, equal transmit power allocation is used.
- *DDQN+CVX(τ)+BSUM(ϕ):* In this case, DDQN and CVX optimizer are used for task offloading and time interval allocation. Furthermore, BSUM is used for transmit power allocation.
- *Dueling DDQN+CVX(τ)+BSUM(ϕ):* In this case, dueling DDQN and CVX optimizer are used for task offloading and time interval allocation. Furthermore, BSUM is used for transmit power allocation.
- *Equal time interval:* In this case, equal time intervals are considered for communication of all metaverse entities.
- *Equal transmit power allocation:* In this case, equal transmit power allocation is considered for all metaverse entities.

B. Simulation Results

Now, we present extensive simulation results. We study the effect of CVX optimization and BSUM on time interval allocation and transmit power allocation, respectively. Fig. 3(a) shows the cumulative reward vs. episodes using 20 time steps for DDQN and DDQN+CVX(τ). It is evident from Fig. 3(a) that DDQN+CVX(τ) outperforms DDQN. Using equal time intervals in the case of DDQN will not enable the DDQN network to perform well. Meanwhile, for DDQN+CVX(τ), it is evident that CVX-based time interval allocation causes the DDQN to get higher rewards. This shows that our proposal of combining CVX optimizer-based time interval allocation with DDQN can be used for other wireless network problems that uses TDMA along with other tasks (e.g., transmit power allocation). In Fig. 3(a), DDQN-based task offloading along with CVX optimizer for time interval allocation is considered, which shows a reasonable performance in terms of reward. However, the DDQN might not be able to capture the network dynamics and thus might have a loss in reward during training of the DDQN. To address this challenge, dueling can be one of the promising solutions that is based on partitioning the last layer of the agent neural network into two parts, i.e., value function and advantage function, to compute the Q-value. This will enable the DDQN to more effectively counter the network dynamics and thus more improved performance. We plot the reward vs. episodes using 20 steps for dueling DDQN and dueling DDQN+CVX(τ) in Fig. 3(b). Fig. 3(b) clearly illustrates that dueling improves performance. Meanwhile, similar to Fig. 3(a), Fig. 3(b) also confirms the importance of using CVX optimizer for time interval allocation. For dueling DDQN+CVX(τ), the convergence is fast compared to dueling DDQN using equal time interval allocation. Moreover, the reward is high for dueling DDQN+CVX(τ) compared to dueling DDQN. Both Fig. 3(a) and (b) show that CVX optimizer-based time interval allocation is beneficial for both DDQN and dueling DDQN. This suggests

(a) (b) (c)

Fig. 3. (a) Cumulative reward vs. episodes for DDQN and DDQN+CVX(τ) using learning rate $= 0.001$ and 20 steps, (b) cumulative reward vs. episodes for dueling DDQN and dueling DDQN+CVX(τ) using learning rate $= 0.001$ and 20 steps, and (c) cost vs. agents for equal time resource allocation and the proposed CVX (τ) for 5 meta spaces.

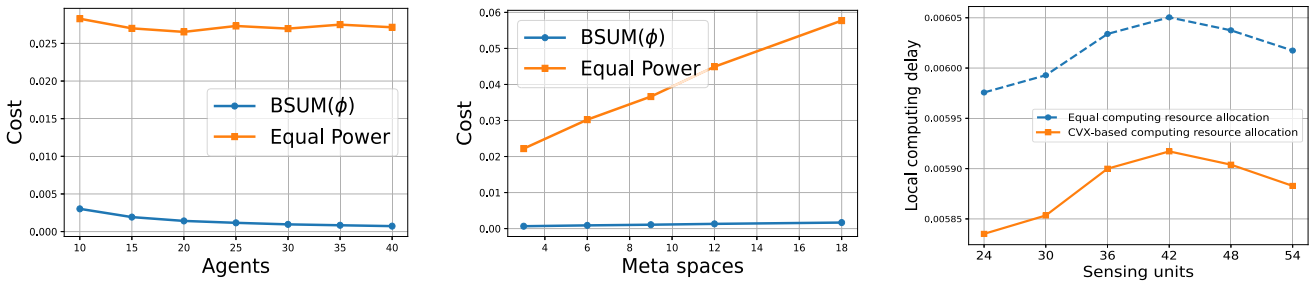


(a) (b) (c)

Fig. 4. (a) Cost vs. meta spaces for equal time resource allocation and the proposed CVX (τ) for 36 agents, (b) cumulative reward vs. episodes for DDQN and DDQN+BSUM(ϕ) using learning rate $= 0.001$ and 20 steps, and (c) cumulative reward vs. episodes for dueling DDQN and dueling DDQN+BSUM(ϕ) using learning rate $= 0.001$ and 20 steps.

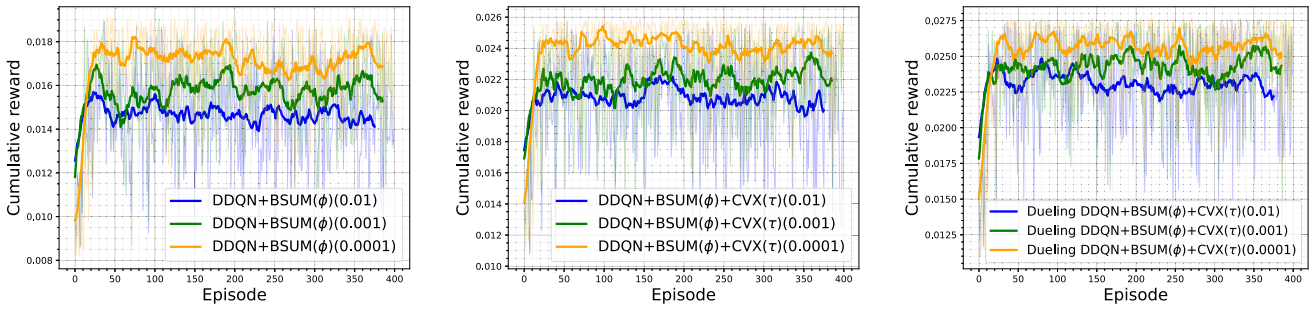
the promising use of CVX optimizer-based time interval allocation with other reinforcement learning schemes. Now, we study the effect of CVX optimizer-based time interval allocation on the cost. Fig. 3(c) illustrates average cost vs. agents for both CVX optimizer-based time interval allocation and equal time interval allocation. For various numbers of agents, it is evident that CVX optimizer-based time interval allocation significantly improves the performance. This trend of performance in Fig. 3(c) shows the importance of CVX optimization-based time interval allocation. Fig. 4(a) shows the cost vs. meta spaces for both CVX optimizer-based time interval allocation and equal time interval allocation for TDMA. Similar to Fig. 3(c), Fig. 4(a) confirms the order of performance and reveals the importance of CVX optimizer in allocating time intervals for TDMA.

On the other hand, Fig. 4(b) and (c) show the effect of BSUM-based power allocation on the performance of both DDQN and dueling DDQN. For both DDQN and dueling DDQN, BSUM(ϕ) significantly improves the performance compared to equal transmit power allocation. If we consider Fig. 4(b), it is evident that BSUM(ϕ) based power allocation results in significant performance improvement compared to equal power allocation. For DDQN+BSUM(ϕ), the performance is better compared to DDQN alone. The cumulative reward is high for DDQN+BSUM(ϕ) compared to DDQN with equal transmit power allocation. Similar to Fig. 4(b), Fig. 4(c) also confirms the order of performance for dueling DDQN+BSUM(ϕ) and dueling DDQN. Now, we study the effect of BSUM(ϕ) on the performance of cost for different settings in Fig. 5(a) and (b). It is evident from both Fig. 5(a) and (b) that the proposed scheme using BSUM for power allocation results in less cost for various numbers of agents and meta spaces compared to equal transmit power allocation. Although BSUM is based on proximal upper bound minimization function, it still outperforms the equal transmit power allocation. This shows the importance of BSUM for solving non-convex optimization problems. Meanwhile, the complexity of BSUM is reasonable, which makes it more attractive for use in solving non-convex problems. On the other hand, Fig. 6(a), (b), and (c) evaluate the performance of various schemes using different learning rates of 0.01, 0.001, and 0.0001. For all the learning rates, the performance of all proposed schemes shows a reasonable performance in terms of reward, which shows their stable nature. Among various learning rates, the reward is highest for the learning rate 0.0001. For all three Fig. 6(a), (b), and (c), the trend of performance is the same. However, if we look carefully at Fig. 6(a), (b), and (c), the



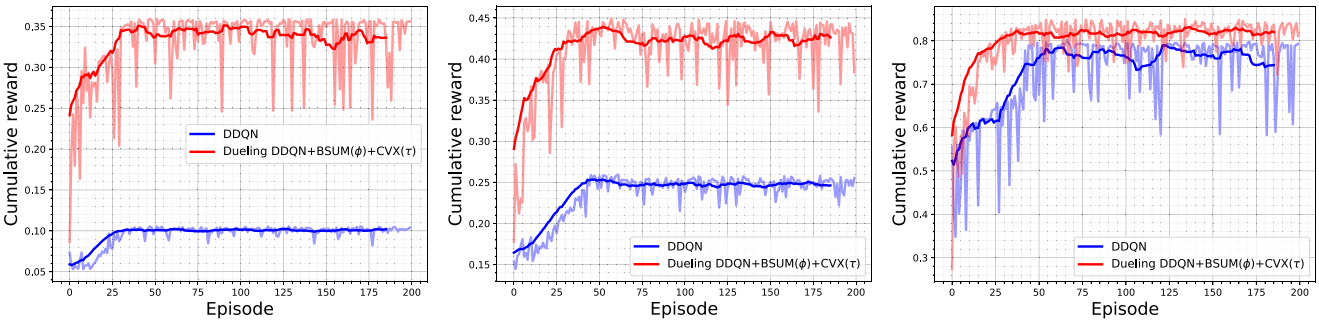
(a) (b) (c)

Fig. 5. (a) Cost vs. agents for equal power allocation and the proposed BSUM (ϕ) for 36 agents, (b) cost vs. meta spaces for equal power allocation and the proposed BSUM (ϕ) using learning rate $= 0.0001$ and 20 steps, and (c) local computing latency for the proposed CVX-based computing resource allocation and the equal computing resource allocation scheme.



(a) (b) (c)

Fig. 6. (a) Cumulative reward vs. episodes for DDQN+BSUM(ϕ) for various learning rates, (b) cumulative reward vs. episodes for DDQN+BSUM(ϕ)+CVX(τ) for various learning rates, (c) cumulative reward vs. episodes for dueling DDQN+BSUM(ϕ)+CVX(τ) for various learning rates.

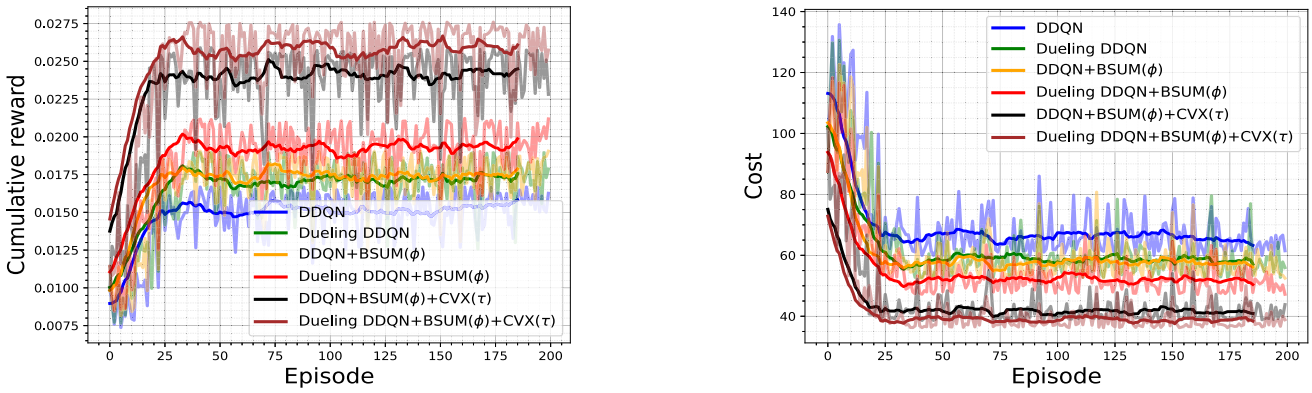


(a) (b) (c)

Fig. 7. (a) Cumulative reward vs. episodes for DDQN and dueling DDQN+BSUM(ϕ)+CVX(τ) using learning rate $= 0.001$ and 20 steps for 50 agents, (b) cumulative reward vs. episodes for DDQN and dueling DDQN+BSUM(ϕ)+CVX(τ) using learning rate $= 0.001$ and 20 steps for 150 agents, and (c) cumulative reward vs. episodes for DDQN and dueling DDQN+BSUM(ϕ)+CVX(τ) using learning rate $= 0.001$ and 20 steps for 200 agents.

reward is highest for the learning rate 0.0001 for dueling DDQN+CVX(τ)+BSUM(ϕ). On the other hand, we illustrate reward vs. episodes for various numbers of agents in Fig. 7 to check the scalability of our proposed solution. It is evident from Fig. 7 that the proposed solution outperforms the traditional DDQN scheme for various numbers of agents. Meanwhile, for all cases, the proposed scheme converges within a reasonable number of episodes. This shows the scalable nature of the proposed scheme.

Now, we compare the performance in terms of reward vs. episodes for various baselines in Fig. 8(a). The performance order in terms of reward from worst to best is DDQN, dueling DDQN, DDQN+BSUM(ϕ), dueling DDQN+BSUM(ϕ), DDQN+BSUM(ϕ)+CVX(τ), and dueling DDQN+BSUM(ϕ)+CVX(τ). From Fig. 8(a), it is evident that dueling significantly improves the performance of DDQN. The reason for this improvement lies in the effective handling of the dynamic nature of the network during the training of the



(a) (b)

Fig. 8. (a) Cumulative reward vs. episodes for various baselines using 20 steps per episode and 20 agents and (b) cost vs. episodes for various baselines using 20 steps per episode and 20 agents.

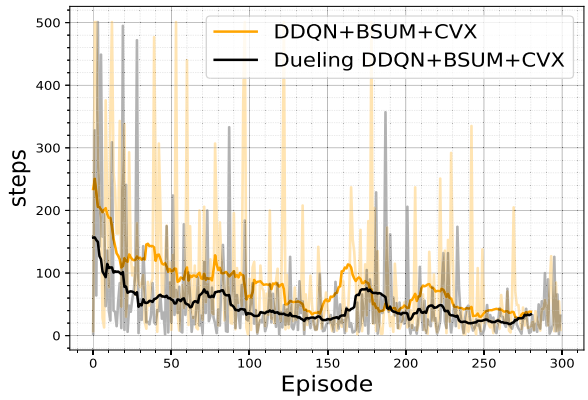


Fig. 9. Steps needed for all nodes to have cost less than or equal to the QoS threshold (i.e., fulfill both latency and reliability) for various schemes.

agents. Other than dueling, CVX optimizer for time interval optimization also improves the performance in the case of both DDQN and dueling DDQN. Furthermore, BSUM also improves the performance of both DDQN and dueling DDQN. Therefore, we can conclude that the proposed scheme of using dueling DDQN along with BSUM for transmit power allocation and CVX optimizer for time interval optimization is very effective and outperforms the traditional DDQN scheme. Another Fig. 8(b) shows the cost vs. episodes for various schemes. Similar to Fig. 8(a), it is evident that Fig. 8(b) also follows the same trend of performance, but in terms of cost. The lowest cost among DDQN, dueling DDQN, DDQN+BSUM(ϕ), dueling DDQN+BSUM(ϕ), DDQN+BSUM(ϕ)+CVX(τ), and dueling DDQN+BSUM(ϕ)+CVX(τ) is for dueling DDQN+BSUM(ϕ)+CVX(τ). This is because of the reason that dueling DDQN+BSUM(ϕ)+CVX(τ) considers optimization of transmit power allocation, time interval allocation, and dueling. On the other hand, Fig. 9 shows the number of steps needed to fulfill QoS metric. Fig. 9 makes it clear that the proposed scheme dueling DDQN+BSUM(ϕ)+CVX(τ) typically needs less number of steps within every episode to reach that QoS.

V. Conclusion

In this paper, we have considered a QoS and QoE aware task offloading in the metaverse. For QoS, we consider both latency and reliability, whereas for QoE, we consider packet error rate and immersive experience. Our formulated problem considered latency and energy while computing the cost function. The cost function is minimized using local computing resource optimization, task offloading optimization, time interval optimization, and transmit power optimization. For time interval optimization, we used CVX optimizer, which significantly helped in getting a high reward. Moreover, it showed that we can extend CVX along with DDQN for optimization problems. Other than time interval optimization, we used BSUM for transmit power allocation, which also showed a significant improvement in the performance of DDQN. Meanwhile, dueling also improved the performance of DDQN. Collectively, we concluded that CVX optimizer for time interval allocation, BSUM for transmit power allocation, dueling DDN for task offloading, and local computing resource optimization using CVX optimizer significantly improved the performance of task offloading in the metaverse. Furthermore, we concluded that the use of standalone DDQN with dueling might not result in very good performance for complex wireless systems. Instead, if we add other blocks (e.g., convex optimizer and BSUM), it will result in better performance.

References

[1] L. U. Khan, A. Elhagry, M. Guizani, and A. El Saddik, "Edge intelligence empowered vehicular metaverse: Key design aspects and future directions," *IEEE Internet Things Mag.*, vol. 7, no. 1, pp. 120–126, Jan. 2024.

[2] W. Saad et al., "Artificial general intelligence (AGI)-native wireless systems: A journey beyond 6G," in *Proc. IEEE*, vol. 113, no. 9, pp. 849–887, 2024, doi: 10.1109/JPROC.2025.3526887.

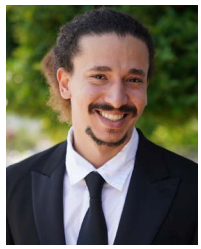
[3] L. U. Khan, Z. Han, D. Niyato, M. Guizani, and C. S. Hong, "Metaverse for wireless systems: Vision, enablers, architecture, and future directions," *IEEE Wireless Commun. Mag.*, vol. 31, no. 4, pp. 245–251, Aug. 2024.

[4] L. U. Khan, M. Guizani, I. Yaqoob, A. Al-Fuqaha, A. Erbad, and Z. Han, "Network virtualization empowered metaverse: A hierarchical matching approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 13, pp. 5403–5416, 2026, doi: 10.1109/TNSE.2025.3588451.

[5] W. Y. B. Lim et al., "Realizing the metaverse with edge intelligence: A match made in heaven," *IEEE Wireless Commun.*, vol. 30, no. 4, pp. 64–71, Aug. 2023.

[6] H. Zhang, S. Mao, D. Niyato, and Z. Han, "Location-dependent augmented reality services in wireless edge-enabled metaverse systems," *IEEE Open J. Commun. Soc.*, vol. 4, pp. 171–183, 2023.
[7] A. D. Wilson, "Depth-sensing video cameras for 3D tangible tabletop interaction," in *Proc. 2nd Annu. IEEE Int. Workshop Horizontal Interactive Hum.- Comput. Syst.*, 2007, pp. 201–204.
[8] I. Aliyu et al., "Toward a dynamic tasks offloading and resource allocation for the metaverse in in-network computing," in *Proc. 14th Int. Conf. Ubiquitous Future Netw.*, 2023, pp. 798–803.
[9] Z. Xu et al., "Learning-driven algorithms for responsive AR offloading with non-deterministic rewards in metaverse-enabled MEC," *IEEE/ACM Trans. Netw.*, vol. 32, no. 2, pp. 1556–1572, Apr. 2024.
[10] X. Hou, J. Wang, C. Jiang, Z. Meng, J. Chen, and Y. Ren, "Efficient federated learning for metaverse via dynamic user selection, gradient quantization and resource allocation," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 4, pp. 850–866, Apr. 2024.
[11] J. Yu, A. Y. Alhilal, T. Zhou, P. Hui, and D. H. Tsang, "Attention-based QoE-aware digital twin empowered edge computing for immersive virtual reality," *IEEE Trans. Wireless Commun.*, vol. 23, no. 9, pp. 11276–11290, Sep. 2024.
[12] L. Feng et al., "Resource allocation for metaverse experience optimization: A multi-objective multi-agent evolutionary reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 24, no. 4, pp. 3473–3488, Apr. 2025.
[13] N. H. Chu et al., "Metaslicing: A novel resource allocation framework for metaverse," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4145–4162, May 2024.
[14] J. Feng and J. Zhao, "Resource allocation for augmented reality empowered vehicular edge metaverse," *IEEE Trans. Commun.*, vol. 73, no. 3, pp. 1987–2001, Mar. 2025.
[15] H. Du et al., "Attention-aware resource allocation and QoE analysis for metaverse xURLLC services," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 7, pp. 2158–2175, Jul. 2023.
[16] L. Qian, C. Liu, and J. Zhao, "User connection and resource allocation optimization in blockchain empowered metaverse over 6G wireless communications," *IEEE Trans. Wireless Commun.*, vol. 24, no. 1, pp. 19–34, Jan. 2025.
[17] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.
[18] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile Edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, Jun. 2022.
[19] O. Hashash, C. Chaccour, W. Saad, K. Sakaguchi, and T. Yu, "Towards a decentralized metaverse: Synchronized orchestration of digital twins and sub-metaverses," in *Proc. IEEE Int. Conf. Commun.*, 2023, pp. 1905–1910.
[20] S. Toumpis and L. Tassiulas, "Optimal deployment of large wireless sensor networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 7, pp. 2935–2953, Jul. 2006.
[21] 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 Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, Jan. 2021.
[22] M. Hong, T.-H. Chang, X. Wang, M. Razaviyayn, S. Ma, and Z.-Q. Luo, "A block successive upper-bound minimization method of multipliers for linearly constrained convex optimization," *Math. Operations Res.*, vol. 45, no. 3, pp. 833–861, 2020.
[23] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM J. Optim.*, vol. 23, no. 2, pp. 1126–1153, 2013.
[24] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods," *SIAM J. Optim.*, vol. 23, no. 4, pp. 2037–2060, 2013.
[25] J. Zeng, T. T. -K. Lau, S. Lin, and Y. Yao, "Global convergence of block coordinate descent in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7313–7323.
[26] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Commun. Surv. Tut.*, vol. 23, no. 2, pp. 1226–1252, Secondquarter 2021.
[27] O. S. Oubbati, A. Lakas, and M. Guizani, "Multiagent deep reinforcement learning for wireless-powered UAV networks," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 16044–16059, Sep. 2022.
[28] Y. Yu, S. C. Liew, and T. Wang, "Multi-agent deep reinforcement learning multiple access for heterogeneous wireless networks with imperfect channels," *IEEE Trans. Mobile Comput.*, vol. 21, no. 10, pp. 3718–3730, Oct. 2022.
[29] M. Chafii, S. Naoumi, R. Alami, E. Almazrouei, M. Bennis, and M. Debbah, "Emergent communication in multi-agent reinforcement learning for future wireless networks," *IEEE Internet Things Mag.*, vol. 6, no. 4, pp. 18–24, Dec. 2023.
[30] N. Zhao, Y.-C. Liang, D. Niyato, Y. Pei, M. Wu, and Y. Jiang, "Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 11, pp. 5141–5152, Nov. 2019.
[31] S. P. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge, U.K.: Cambridge Univ. Press, 2004.
[32] S. A. Kazmi et al., "Mode selection and resource allocation in device-to-device communications: A matching game approach," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3126–3141, Nov. 2017.

Latif U. Khan (Member, IEEE) received the MS (with distinction) degree in electrical engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2017, and the PhD degree in computer engineering from Kyung Hee University (KHU), South Korea. He is author of two books such as *Network Slicing for 5G and Beyond* and *Federated Learning for Wireless Networks*. He has reviewed more than 200 times for the top ISI- Indexed journals and conferences. He has authored many most popular articles in the leading journals (such as, *IEEE Communications Surveys and Tutorials*) and magazines (*IEEE Communication Magazine*, *IEEE Network*, and *IEEE Wireless Communications Magazine*). His research interests include analytical techniques of optimization and game theory to edge computing, end-to-end network slicing, wireless federated learning, and digital twins. He was the recipient of KHU Best Thesis Award.



Maher Guizani (Member, IEEE) received the master's degree in electrical engineering from Purdue University, in 2016. He is currently working toward the PhD degree in computer engineering from the University of Texas at Arlington. He was with General Motors as a calibration engineer from 2016 to 2018. He was with Embedded Engineering Department, PACCAR, from 2018 to 2019 and also with Electrified Semi Truck Department, Meritor, from 2020 to 2022. He has experience working in the automotive industry.

Sami Muhaidat (Senior Member, IEEE) received the PhD degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2006. From 2007 to 2008, he was a postdoctoral fellow with the Department of Electrical and Computer Engineering, University of Toronto, ON, Canada. From 2008 to 2012, he was an assistant professor with the School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada. He was also a visiting reader with the Faculty of Engineering, University of Surrey, Guildford, U.K. He is currently a professor with Khalifa University, Abu Dhabi, UAE. He was the recipient of several scholarships during his undergraduate and graduate studies and the winner of 2006 Post-Doctoral Fellowship Competition. He was the senior editor of *IEEE Communications Letters* and associate editor for *IEEE Transactions on Communications*, *IEEE Communications Letters*, and *IEEE Transactions on Vehicular Technology*. He is also the area editor of *IEEE Transactions on Communications*.

Asad Masood Khattak (Senior Member, IEEE) is currently a graduate program coordinator with the College of Technological Innovation, Zayed University, where he was an assistant chair. He has more than 17 years of industry and academia experience in various capacities, such as webmaster, developer, team leader, principal investigator, assistant chair, graduate program coordinator, assistant, associate, and full professor. He has multicultural teaching experience and has taught in Pakistan, China, South Korea, and UAE. He has authored more than 180 publications (book, book chapters, journal publications, and conference publications). He has a US Patents registered and Korean Patent. His research interests include data curation and representation, data mining, semantic web, machine learning, deep learning, computational intelligence, social media analysis, e & m-healthcare, smart cities, and secure computing. Taking bold, decisive, and definitive actions to solve problems, building strategic alliances, and working in collaborative teams are his strengths. He has experience of leading research teams of more than 40 members in more than four countries. He is open to explore more and learn more that has mutual benefits for him, his partners, and his institution. He has delivered various keynote speeches, invited talks, invited lectures, and short courses. He is appointed as an ACM distinguish speaker.

Adel Khelifi (Senior Member, IEEE) received the PhD degree from the Engineering School of High Technology, Canada, in 2005. He was the dean of computer information technology with American University in the Emirates, Dubai, United Arab Emirates. He was a lecturer with the Engineering School of Technology, Canada; United Nations MSF, Canada; Ministry of Citizenship and Immigration, Canada, and Ministry of Finance, Tunisia. He is currently an associate professor with Abu Dhabi University. He has extensive knowledge and experience. He is driving the open-source software paradigm in the region. He is a Canadian ISO member of software engineering. He is an ABET PEV. His research focuses on archaeology.

Zhu Han (Fellow, IEEE) received the BS degree in electronic engineering from Tsinghua University, in 1997, and the MS and PhD degrees in electrical and computer engineering from the University of Maryland, College Park, in 1999 and 2003, respectively. From 2000 to 2002, he was an R&D engineer with JDSU, Germantown, MD, USA. From 2003 to 2006, he was a research associate with the University of Maryland. From 2006 to 2008, he was an assistant professor with Boise State University, Idaho. He is currently a John and Rebecca Moores professor with the Electrical and Computer Engineering Department and also with Computer Science Department, University of Houston, Texas. Dr. Han's research focuses on the novel game-theory related concepts critical to enabling efficient and distributive use of wireless networks with limited resources. He was an IEEE Communications Society distinguished lecturer from 2015 to 2018, AAAS fellow since 2019, and ACM distinguished member since 2019. Dr. Han is a 1% highly cited researcher since 2017 according to Web of Science. He is also the winner of 2021 IEEE Kiyo Tomiyasu Award (an IEEE Technical Field Award), for outstanding early to mid-career contributions to technologies holding the promise of innovative applications, with the following citation: "for contributions to game theory and distributed management of autonomous communication networks".