# Dynamic Resource Allocation Using Deep Reinforcement Learning for 6G Metaverse

Haesik Kim

VTT Technical Research Centre of Finland
P.O. Box 1100, FI-90570 Oulu, Finland
haesik.kim@vtt.fi

***Abstract*— The 6th Generation (6G) networks are now under development. The 6G will revolutionize the cellular networks more intelligently. In 6G era, we expect much higher network requirements including massive network traffics, huge numbers of devices, extremely low latency, low energy consumption and so on. A metaverse is one of key applications in 6G. Wireless techniques are directly related to performance of a metaverse application. The metaverse application requires both a high throughput and a low latency. The condition is more challenging than 5G. In this paper, we investigate dynamic resource allocation for 6G metaverse. We formulate the resource allocation problem for metaverse as the Markov decision processes (MDP) and solve the resource allocation problem using deep reinforcement learning (DRL). The main contributions of this paper are summarized as follows: We optimize the resource allocation for both a high throughput and a low latency. We adopt a sparse reward function of the reinforcement learning in the system model. It is more realistic because we can check whether or not the resource allocation scheme satisfies the requirements after completing the packet transmission.**

***Index Terms*— 6G, Metaverse, Resource allocation, Machine learning, Deep reinforcement learning, etc.**

## I. Introduction

THE 6TH Generation (6G) networks are now under development. The 6G will revolutionize the cellular networks more intelligently. In 6G era, we expect much higher network requirements including massive network traffics, huge numbers of devices, extremely low latency, low energy consumption and so on. They will be characterized by connectivity solution with ultra-low latency, ultra-high speed, hyper-connectivity, and super-intelligence. In order to satisfy the high-level network requirements, 6G networks will accommodate new technologies such as artificial intelligence, open RAN, Terahertz communication, Integrated Sensing and Communication, Reconfigurable Intelligent Surface, and so on. A new application will be possible to operate in 6G networks. A metaverse as one of key 6G applications can be defined as 3-dimensional (3D) virtual space that can be implemented by multiple technologies. In the 3D virtual space, we will be able to collaborate, learn, or play. The key techniques to implement a metaverse are virtual reality, augmented reality, digital twin, artificial intelligence, Internet of Things (IoT), blockchain, brain-computer interfaces, and so on. Among them, wireless techniques are highly related to quality of service (QoS) and quality of experience (QoE). They are directly affecting the performance of an interactive experience system. They are related to the data exchange and model accuracy. In order to connect the virtual world and the real world and provide users with immersive experience, we should maintain a reliable and accurate connectivity while saving computing and communicating costs. For doing this, it is a key research challenge to optimize computing and radio resource in wireless networks. The resource optimization allows us to employ computing and communicating resources maximizing/minimizing the objective functions (Ex. latency, energy efficiency, complexity, or others for the given metaverse model) under the given constraints.

When deploying 6G networks, we should consider distribution and allocation of radio and computing resources. In particular, with the growing needs for high throughput applications and the limited radio resource to satisfy them, it is a key research challenge of 6G systems to manage network traffic and find optimal radio resource allocation while alleviating the network traffic congestion, satisfying the latency sensitive requirement, and supporting a better QoS. As machine learning techniques are widely adopted in wireless systems, there are many attempts to adopt them to resource allocation and scheduling and improve the performance. Among machine learning techniques, the reinforcement learning (RL) solves sequential decision-making problem formulated in a Markov decision process (MDP). The elements of RL are policy, reward, value, model. Their role can be simply explained as Policy (what to do), Reward (what is good or bad), Value (what is good because it predicts reward), and Model (what follows what, Optional element) [1]. The RL can learn from the errors and correct by iteration. Thus, it works very well to solve complex sequential decision-making problems. However, it requires a huge training data and high computational complexity. The deep reinforcement learning (DRL) combines reinforcement learning and deep learning. In many sequential decision-making problems, the states are expressed in high dimensional components. Thus, it is not easy to solve and the complexity is very high. In DRL, the agent embedded neural network approximates a state-value function or policy function in a Q learning framework. It allows us to solve the complex sequential decision-making problem easily. The RL can be regarded as one of promising methods to optimize resource allocation block in the orthogonal frequency-division multiple access (OFDMA) based wireless systems.

There are a lot of research works to optimize the resource allocation blocks or adopt machine learning techniques to

resource allocation problems. In [2], the authors predict incoming traffics and show us the tradeoff relationship between enhanced mobile broadband (eMBB) data rate and ultra reliable low latency communications (URLLC) latency. In [3], the authors proposed a learning mechanism to improve the coordination in the open radio access network (O-RAN) and show us an optimized resource allocation in the radio access network. In [4], the authors proposed a flexible resource allocation for the slicing in order to reduce the number of active metro nodes. In [5], the authors proposed a learning-based resource allocation to share the bandwidth between different network slices. In [6], the authors proposed an online learning method for resource allocation and improve computational efficiency in cloud networks. In [7], the authors formulated a stochastic mixed integer nonlinear programming to jointly optimize task offloading, resource allocation and scheduling. In [8, 9], the RL is adopted to solve resource allocation problem in wireless systems. The Q learning based RL is used to find an optimal policy by interaction with the environment. The Q learning converges slowly and occurs the explosion of action state space.

The main contributions of this paper can be summarized as follows:

(1) The resource allocation problem for 6G metaverse applications targeting both a high throughput and a low latency is formulated and solved.

(2) The sparse reward as more practical condition is considered in the model of deep reinforcement learning.

(3) The performances of the proposed method are evaluated.

The remaining parts of this paper are as follows: In section II, system model is defined. In section III, the resource allocation for 6G metaverse application is formulated in MDP. The proposed method is described. In section IV, numerical analysis is included. Section V contains the conclusion and summary.

## II. System Model

We consider OFDMA based downlink single cell model as shown in figure 1. A gNodeB (gNB) covers a set $U$ of mobile users (or User Equipment (UE)) $u \in U$ that are stationary. They are uniformly deployed in the range of the gNB. The set $U$ is composed of metaverse application users requiring both a high throughput and a low latency. The 3GPP 5G NR [10] supports the numerology allowing us to have flexibility and adaptivity of the transmissions in terms of different use case scenarios. It enables us to adopt different frequency bands, symbol duration, latency and so on. A resource element (RE) is the smallest element of the resource grid. A resource block (RB) is defined as 12 consecutive subcarriers. A resource block group (RBG) consists of a set of consecutive resource blocks. The lengths of frame and subframe are 10 ms and 1 ms, respectively. The number of slots in one subframe varies in terms of numerology. The number of OFDM symbols in one slot is 14 OFDM symbols for normal cyclic prefix. One resource grid is assigned for one antenna port and numerology. We define the total number of resource blocks $N_{RB}^{DL}$ for downlink transmission.

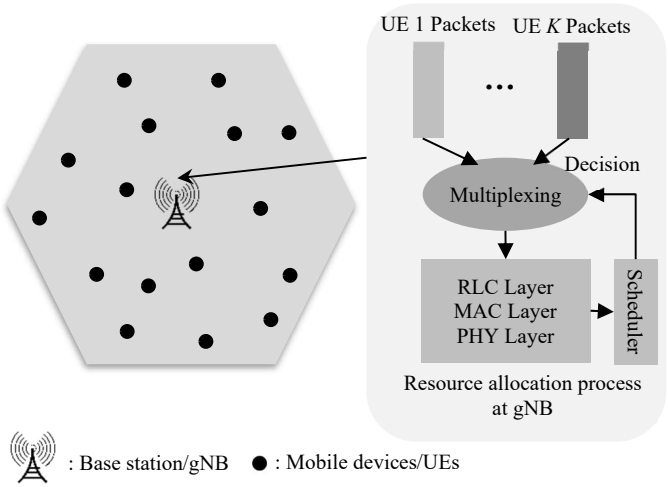


Figure 1 System model.

Since a mobile user requires a metaverse application, connectivity with both a high throughput and a low latency transmission is required. The 3GPP adopted adaptive modulation and coding to achieve a high throughput and define an optimal combination of modulation and coding scheme (MCS) in terms of channel quality [10]. It enables us to transmit data packets adaptively and guarantee reliability. In this system model, we assume that the gNB obtains channel quality indicator (CQI) via uplink control channel and uses a link adaptation. The data traffics of UEs are queued in the transmission buffer at the gNB. The scheduler of the gNB performs a resource allocation in the transmission buffer. We assume that the traffic model is constant and the queuing delay is trivial.

## III. Deep Reinforcement Learning and Proposed Method

### *A. Problem Formulation and Elements Definition of Deep Reinforcement Learning*

The purpose of the dynamic resource allocation for metaverse applications is to determine a radio resource block allocation of mobile users that maximizes the probability of achieving the total data volume $V_G$ and the probability of achieving the end-to-end (E2E) latency at the time horizon $T$. The dynamic resource allocation strategy is to find the optimal solution of the following multi-period and multi functions optimization problem:

$$\max_{a(t),t<T} P(V(T) \geq V_G) \tag{1}$$

and

$$\max_{t<T} P(D(T) \leq D_G) \tag{2}$$

where $V(T)$ is the final data volume, $D(T)$ is the final E2E latency, $D_G$ is the target total latency, and $a(t)$ are the possible actions and resource allocations at time $t$. We will use DRL to solve the problem (1) and (2). The purpose of DRL is to train an agent to complete a task within an environment. They should be modelled by the Markov decision processes. The MDP is a stochastic decision-making tool and allows us to

describe an environment of reinforcement learning. It is useful for modelling decision making problems. Since the system model has sequential actions stochastically, we can formulate our problem as the MDP. In the system model, the gNB adopts the deep reinforcement learning. In order to solve the problem using DRL, you can define the elements of DRL in the MDP as follows: The states are data volume at the time period *t*. The actions are resource block weights in terms of modulation order for dynamic transmission strategy. The rewards are the feedbacks about how well the action contributes to the task to achieve the goal. The agent is the component to choose the actions and train for completing the task. The DQN agent is used. The environment is the evolution model simulates the next observation after deciding an action and calculates its reward. The environment allows the agent to have a state. The agent selects an action. Depending on the action, the agent receives a positive or negative reward from the environment along with the new state. These steps are repeated until objective function is satisfied. Namely, the agent learns the optimal behaviour through a trial-and-error approach under the given environment without human interaction. What we find it in unknown environment is the main advantage of the RL. The main purpose of the RL is to find an optimal policy $\pi$ that has a maximum cumulative reward. The policy function is mapping each state to the bast actions based on the observations from the environment. In order to train a policy function, we need to evaluate the policy to take actions, apply them to environments, and then obtain the observation and rewards. The learning algorithm iteratively performs this process and updates the parameters of the policy. The learning algorithm enables us to find an optimal policy. Figure 2 illustrates the DRL process.

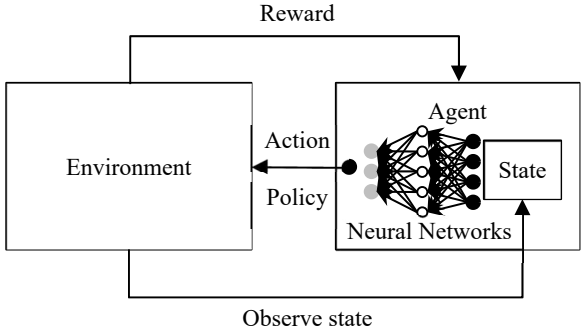


Figure 2 Elements of Deep Reinforcement Learning.

We explain the components of DRL in details as follows:

**Definition of states** *s(t)*: The states represent the data volume *V(t)* at the time *t* and have two elements as *s(t)* = [*V(t)*, *t*]. We can create a grid of data volume value for all time periods. The grid is composed of time *t* on the columns and data volume value *V* on the rows. We can represent the data volume evolution. The state is represented by a node on the grid. The state transition is modelled by

$$V(t+1) = V(t) f(m, P_m) \tag{3}$$

where the function *f* () represents the relationship between throughput and error probability in terms of $m$ and $P_m$ where $m$ is modulation order and $P_m$ is error probability. The function *f*() is expressed as follows:

$$f(m, P_m) = \alpha R_m Z(m, P_m) \tag{4}$$

where $\alpha$, $R_m$ and *Z* represent adjusting factor, throughput, and channel impairment, respectively. In cellular networks, throughput $R_m$ (in Mbps) can be calculated as follows [11]:

$$R_m = 10^{-6} \sum_{j=1}^{J} \left( v_L^j Q_n^j g_n^j \gamma_{mx} \frac{N_{PRB}^{BWj.\mu} 12}{T_s^{\mu}} (1 - OH^j) \right) \tag{5}$$

where *J* is the number of carrier, $v_L^j$ is the number of layer, $Q_n^j$ is maximum modulation order, $g_n^j$ is scaling factor at higher layers, $\gamma_{mx}$ is maximum code rate, $N_{PRB}^{BWj.\mu}$ is maximum resource block allocation in bandwidth BW with numerology $\mu$, $T_s^{\mu}$ is average OFDM symbol duration, and $OH^j$ is overhead. In order to simplify, we put all variables as constant except $Q_n^j \approx m$. Thus, the throughput $R_m$ (in Mbps) is a function with only one variable $m$. The values of the modulation order $m$ can be as follows: $m$ = 1 for BSPK , $m$=2 for QPSK , $m$=4 for 16 QAM, $m$=6 for 64 QAM, $m$= 8 for 256 QAM and so on. When we have MPSK and MQAM modulation, the relationship between symbol error rate $p_m$ and the bit energy $\frac{E_b}{N_o}$ can be simply expressed as follows [12]:

$$p_{mPSK} = erfc\left( \sqrt{\log_2 m \frac{E_b}{N_o}} \sin\left(\frac{\pi}{m}\right) \right) \tag{6}$$

and

$$p_{mQAM} \tag{7}$$

$$= 2\left(1 - \frac{1}{\sqrt{m}}\right) erfc\left( \sqrt{\frac{3}{2(m-1)} \frac{\log_2 m\, E_b}{N_o}} \right)$$

$$- \left(1 - \frac{2}{\sqrt{m}} + \frac{1}{m}\right) erfc^2\left( \sqrt{\frac{3}{2(m-1)} \frac{\log_2 m\, E_b}{N_o}} \right)$$

where the complementary error function *erfc*(*z*) is defined as

$$erfc(z) = 1 - erf\ (z) \tag{8}$$

and

$$erf(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt. \tag{9}$$

In cellular network, the transmit power is limited. we fix the signal power in this model. In cellular network environment, fading effects are caused by multiple copies of the transmitted signals at the receiver and signal variation with different variables such as time, frequency, and geographical location. They are common channel impairments in that the signal strength fluctuates over time and distance. The main components of fading are path loss, Doppler effect, reflection

and diffractions of signals and so on. The fading effect significantly affects to the data transmission. The Rayleigh fading probability density function (pdf) is expressed as follows:

$$p(r) = \begin{cases} \frac{r}{\sigma^2} exp\left(-\frac{r^2}{2\sigma^2}\right), & r \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

where $r$ is the envelope amplitude of received signal and $2\sigma^2$ is the mean power of the signal that depends on the distance between transmitter and the receiver. The channel impairment in this system model is composed of symbol error probability and fading effect as follows:

$$Z(m, P_m) = \beta p(r) p_m \quad (11)$$

where $\beta$ is adjusting factor. The state transition function (3) depends on many variables and is complex with many variables. We can create a table to simplify it as a function of $m$ and $P_m$.

**Definition of actions** *a(s(t))*: We assume that the distribution of the actions is time homogeneous and define the action as an element in the time index *t*. The set of actions is the policy that is a function of the state space and also updates to maximize the rewards. We create the discrete action space for the environment. In this system model, the actions are to choose the modulation order for the next period. As increasing the modulation order, we can send more data to a receiver and the throughput (bits/symbol) increases. However, the error probability increases. The action is represented as follows:

$$a(s(t)) = [m, P_m]. \quad (12)$$

We can set 15 modulation type from $m = 1$ to $m = 15$ and their error probability at BER $10^{-6}$ and can create a table expressing their relationship.

**Definition of reward** *r(s(t),a(t))*: Typically, the agent takes an action and receives a reward at each state. However, in this model, the rewards are only given at the final time *T* because the OFDMA symbols are transmitted continuously. After transmitting the packet that is composed of OFDMA symbols, we can determine whether or not the packet is successfully received. The goal of this problem is to achieve the target data volume while minimizing the transmission time slot. We use a sparse reward function if the goal is achieved at the end of the transmission period as follows:

$$R(V(t), t) = \begin{cases} 0, & \text{when } t < T \text{ or } V(t) < V_G \\ 1, & \text{when } t = T \text{ or } V(t) \geq V_G \end{cases} \quad (13)$$

As we can observe (13), the agent should explore the environment from the initial state and receive the reward at the final state. In many real-world scenarios, an agent faces the challenge of sparse extrinsic reward. Therefore, this is a realistic condition.

**Definition of agent**: Since we have a sparse reward function, the agent receives the reward value 1 only upon reaching the goal state at the final time *T*. A large amount of the states does not have a reward as the episodes become long. We train the agent to find the optimal series of actions that maximizing the possibility of transmitting the total data volume while minimizing the transmission time. We adopt a Deep Q Network (DQN) to train the agent. The DQN method is based on a model free, online, and off policy. One disadvantage of Q learning is to be infeasible or require a huge memory to store Q values when we have large states, and the Q table grows exponentially. Thus, the DQN combine Q learning and deep learning. The neural network replaces the Q table and acts as the Q value approximator. It maps the input state to the pair of action and Q value.

*B. Proposed method for solving the MDP.*

In this section, we describe the proposed dynamic resource allocation. In this system model, a policy $\pi(s)$ is defined as a mapping that selects actions in the set of actions. The optimal policy $\pi^*(s)$ represents the policy maximizing the total reward with the probability of the final data volume $V(T)$ greater than $V_G$. Solving the MDP means finding the optimal policy. There are three approaches to solve the MDP: Model based algorithms, Model-free algorithms, and Function approximators based algorithms. We take the third approach. We define the policy as a function approximator with tunable parameters. The function approximator approximates the state space, generalizes to unseen states, and finds the value of state or action. The function approximator can be a linear function, a neural network, or a decision tree. The neural network as the function approximator is useful when the number of states or action is large. The deep Q learning as a variant of Q learning is popular and uses a neural network to represent the Q function. The Q learning is a model free off-policy reinforcement learning. We can define action-value function (or $Q$ function) $Q_\pi(s, a)$ as the expected return starting state *s*, taking acting $a$, and following policy $\pi$. The action-value function specifies how good it is to take a particular action from a certain state. The update rule of Q learning can be described as follows:

$$Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \max_a Q_\pi(s_{t+1}, a_{t+1}) - Q_\pi(s_t, a_t)\right). \quad (14)$$

where $Q_\pi(s_t, a_t)$ is the current action, $Q_\pi(s_{t+1}, a_{t+1})$ is the estimate of the optimal next action, $r_{t+1}$ is an immediate reward, $\alpha$ is a learning rate and $\gamma$ is a discount factor for future rewards. As we can observe (14), the term $r_{t+1} + \gamma \max_a Q_\pi(s_{t+1}, a_{t+1})$ is the learned value and the term $Q_\pi(s_t, a_t)$ is the current value. The difference between them becomes the update value. We always update it using maximum value of Q value available from the next state. In the deep Q learning, a neural network is used to approximate the state action value function $Q_\pi(s, a)$ and trained to create the ground-true values for Q by replacing a Q value table. The Deep Q Network (DQN) algorithm was developed by Google DeepMind in 2015 [13, 14]. The DQN uses two neural networks to stabilize the learning. The main neural network denoted by the weigh vector $\theta$ is used to estimate the Q values

for the current state and action $Q(s_t, a_t; \theta)$. All learning processes take place in the main neural network. The target neural network denoted by the weight vector $\theta'$ has same structure as the main neural network but is used to estimate the Q values for the next state and action $Q(s_{t+1}, a_{t+1}; \theta')$. The agent updates the weight vector to improve the stability. The weight vectors of the main neural networks are sent to the target neural network. Both neural networks are implemented by function approximators. The difference between Q learning and Deep Q learning is illustrated in figure 3.

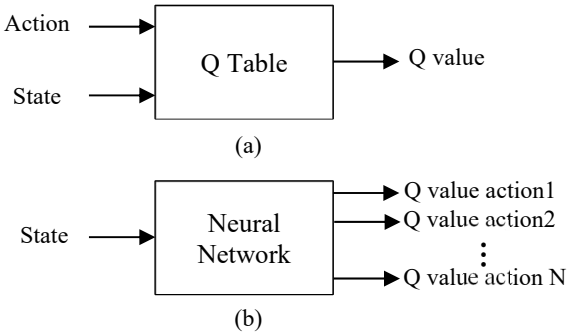


Figure 3 Q learning (a) and Deep Q learning (b).

The pseudo code of the proposed dynamic resource allocation is as follows:

**Procedure** Dynamic resource allocation process
- Model a resource allocation problem as MDP
- Define the Q function mapping to states and actions in the MDP.
- Initialize $Q(s_t, a_{t,k}; \theta) = 0, 0 \le t \le T, 0 \le k \le K$ where $t$, $k$ and $\theta_t$ are time index, resource block weight index, and random parameter, respectively. (Note. the weight factor $\theta'_t$ of the target neural network is same as the one of the main neural network.)
- Define an action $a_{t,k}$ as a pair of modulation and error probability $[m, P_m]$.
- Initialize $\alpha$ and $\gamma$.
- Initialize the data volume $V(0)$ to a constant at $t = 0$.

**For** episode = *1, M* **do**

**For** $t = 1, T$ **do**

- Select an action $a_{t,k}$ with probability $\epsilon$. Otherwise select $a_{t,k} = \max_a Q_\pi(s_{t+1}, a_{t+1,k}; \theta')$ // Epsilon greedy approach arbitrarily chooses a random strategy with the probability $\epsilon$ to balance exploration and exploitation.
- Execute action $a_{t,k}$ .
- Observe the reward $r_t$ and the next state $s_{t+1}$.
- Store observation $(s_t, a_t, r_t, s_{t+1})$ in memory buffer
- Sample a random mini-batch of experiences $(s_j, a_j, r_j, s_{j+1})$ from memory buffer.
- For all experiences in the mini-batch, set the value function target

$$y_j = \begin{cases} r_j, & \text{for a terminal state } s_{j+1} \\ r_j + \gamma \max_a Q_\pi(s_{j+1}, a_{j+1,k}; \theta'), & \text{for a nonterminal state } s_{j+1} \end{cases}$$

// Select the action maximizing the Q function maintained by the target neural network

- Update the weight factor by one-step minimization of the loss $L$

$$L = \frac{1}{2M} \sum_{j=1}^{M} \left( y_j - Q_\pi(s_j, a_{j,k}; \theta) \right)^2$$

- Update the target weight factor $\theta'$ periodically.
- Update the probability threshold $\epsilon$.

**End for**

**End for** $M$ episodes are reached.

**Return** $\pi^*(s)$

## IV. Numerical Analysis

Numerical analysis is presented in this section to evaluate the performance of the proposed method. We set a target throughput and latency and evaluate how much the proposed scheme satisfies the requirement. In order to simplify the evaluation model, we consider the fixed number of users and constant user traffic. In addition, we create the table for the state transition. The Matlab Reinforcement Learning and Deep learning toolboxes [16] are used. The key simulation configurations are summarized as follows: Single cell model. The number of users is 5. The target data volume is 2 Kbyte. The target latency is 10 OFDM symbols period. The 3GPP resource allocation structure is considered. The channel model is the Rayleigh fading channel. The reinforcement learning type is DQN. The reward type is sparse reward. The max episode is 3000. The number of the test data is 1000.

Firstly, we train the DQN agent. We obtain the data volume observation and rewards at the end episode. The reward represents whether or not the target data volume for a metaverse application is achieved. We compute the success rate for the policy by the DQN agent. Figure 4 illustrates reward values of episodes when DQN agent is trained. As we can observe figure 4, the agent receives 1 or 0 reward after the transmission as expressed in sky blue. Dark blue represents the average reward values.

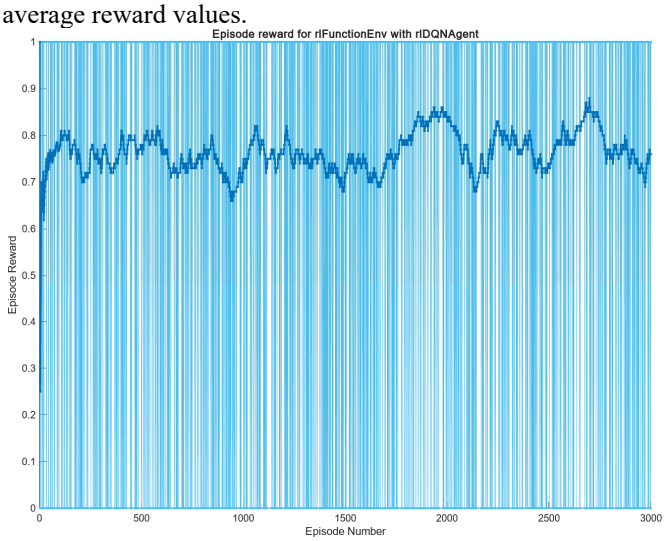


Figure 4 Training of the model.

Now, we perform simulation with 1000 data packet transmission and check how much data packet satisfy the target data volume and latency. Figure 5 illustrates an example of data transmissions satisfying the requirements or not.

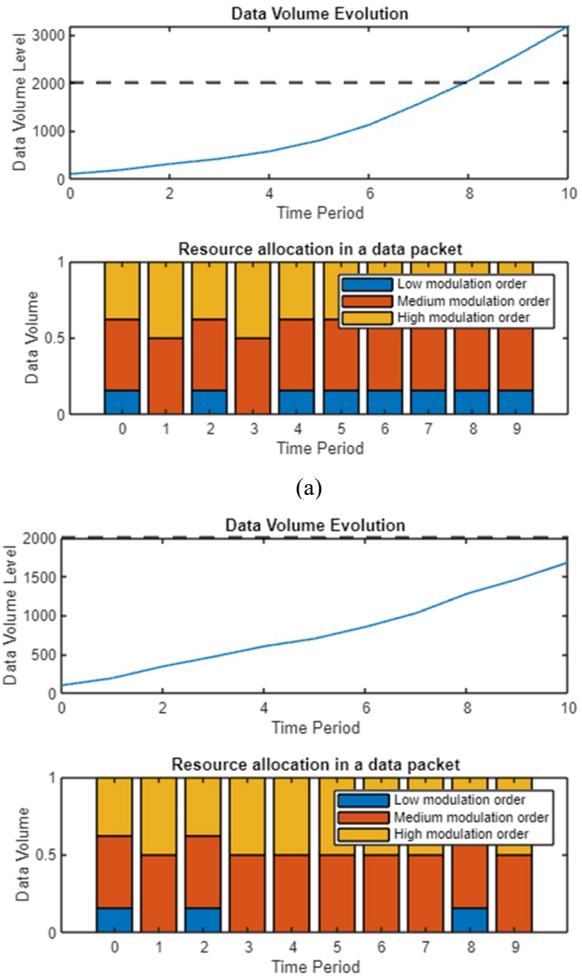


Figure 5 An example of data transmission achieving the target data volume (a) and not satisfying the target data volume (b).

In figure 5, data volume evolution represents how much data are successfully transmitted. The dash line is the target data volume. Resource allocation in a data packet represents how much ratio of modulation type is assigned in the OFDMA packet. The successful transmission means that the target data volume 2Kbyte is transmitted in the time slot 10 (Namely, 10 OFDM symbols). In this simulation configuration, the success rate to achieve the target data volume transmission in the give time frame is 78.90% that is better than normal transmission (Static resource allocation) success rate 65%.

## V. Conclusion and Further Works

Metaverse application requires both time sensitive transmission and high throughput transmission. In order to satisfy this requirement, the dynamic resource allocation scheme is investigated. Using deep reinforcement learning, we achieve a better success rate of transmission under the given simulation configuration. This paper includes the initial results of dynamic resource allocation research for 6G metaverse. There are many further works: Traffic model as Possion model, Comparison study with other cellular resource allocation schemes, Training model study for practical system, and so on.

## Acknowledgement

This work was supported by 6GBridge-EMETA (Enabling Metaverse, www.emeta.fi) project.

## References

[1] Haesik Kim, *Artificial Intelligence for 6G*, Springer, 2022.

[2] C. Bektas, D. Overbeck, and C. Wietfeld, "SAMUS: Slice-Aware Machine Learning-based Ultra-Reliable Scheduling," *ICC 2021 – IEEE International Conference on Communications*. Jun 2021, pp.1–6.

[3] H. Zhang, H. Zhou, and M. Erol-Kantarci, "Team learning-based resource allocation for open radio access network (o-ran)," *IEEE International Conference on Communications (ICC) 2022*.

[4] H. Yu, F. Musumeci, J. Zhang, M. Tornatore, and Y. Ji, "Isolation-Aware 5G RAN Slice Mapping over WDM Metro-Aggregation Networks," *Journal of Lightwave Technology*, vol. 38, no. 6, pp. 1125–1137, 2020.

[5] Y. Sun, Y. Wang, H. Yu, B. Guo, and X. Zhang, "A learning-based bandwidth resource allocation method in sliced 5G C-RAN," *IEEE Globecom Workshops, GC workshop 2019*, 2019.

[6] Q. Han, S. Yang, X. Ren, C. Zhao, X. Zhang, and X. Yang, "OL4EL: online learning for edge-cloud collaborative learning on heterogeneous edges with resource constraints," *IEEE Communications Magazine*, vol. 58, no. 5, pp. 49–55, 2020.

[7] Q. Zhang, L. Gui, F. Hou, J. Chen, F. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobileedge computing in dense cloud RAN," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3282–3299, 2020.

[8] Y. Wei, F. R. Yu, M. Song and Z. Han, "User Scheduling and Resource Allocation in HetNets With Hybrid Energy Supply: An Actor-Critic Reinforcement Learning Approach," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 680-692, Jan. 2018

[9] A. Asheralieva, "Bayesian reinforcement learning-based coalition formation for distributed resource sharing by device-to-device users in heterogeneous cellular networks", *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 5016-5032, Aug. 2017.

[10] 3GPP, "NR; Physical layer procedures for data," Technical Specification 38.214, 3rd Generation Partnership Project (3GPP), 7 2018. Version 15.2.0.

[11] 3GPP TS 38.306 V16.3.0, 3rd Generation Partnership Project; Technical Specification Group Radio Access Network;

[12] NR; User Equipment (UE) radio access capabilities, (Release 16), 2020-12.

[13] Haesik Kim, *Wireless Communications Systems Design*, John Wiley & Sons, ISBN:9781118610152, August 2015.

[14] https://deepmind.google/

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, "Playing Atari with deep reinforcement learning," *Technical report Deepmind Technologies*, 2013. Retrieved from https://arxiv .org/abs/1312.5602

[16] https://se.mathworks.com/products/matlab.html