# *Play to Earn* in the Metaverse over Wireless Networks with Deep Reinforcement Learning

Terence Jie Chua
Graduate College
Nanyang Technological University
terencej001@e.ntu.edu.sg

Wenhan Yu
Graduate College
Nanyang Technological University
wenhan002@e.ntu.edu.sg

Jun Zhao
School of Computer Science & Engineering
Nanyang Technological University
junzhao@ntu.edu.sg

*Abstract*—**The Metaverse Play-to-Earn games on mobile have been gaining popularity as it enables players to earn in-game tokens which can be translated to real-world profits. With the advancements in augmented reality (AR) technologies, users are able to join the digital society and play AR games in the Metaverse. However, these high-resolution games are deeply interactive and compute intensive, and in-game graphical scenes needs to be offloaded from mobile devices to an edge server for computation. In our work, we consider a realistic asymmetric dual-joint optimization problem where the Metaverse Service Providers (MSP)'s objective is to reduce in-game graphics down-link transmission latency, up-link transmission latency, and users' (UE) device energy consumption, while maximizing UE resolution-influenced in-game earning potential through optimizing the down-link user-Metaverse Service Provider Cell Station (UE-MSPCS) assignment (discrete-case allocation) and the up-link transmission power selection (continuous-case power selection). The data downlink and uplink transmissions are then executed asynchronously. We proposed a novel multi-agent, loss-sharing (MALS) deep reinforcement learning model to tackle the above-mentioned asynchronous and asymmetric problem. We then compare our proposed multi-agent model with other baseline models and show its superiority over other methods.**

## I. Introduction

**Background.** The introduction and rapid expansion of the Metaverse has brought a myriad of opportunities [1], and *Play-to-Earn* is one of the most iconic features. Traditionally, and prior to the introduction of the Metaverse, players' do not have complete ownership over their in-game earnings, assets and tokens as they ultimately belonged to the game developers and creators. Yet now, the existence of the Metaverse allows players to gain ownership over their in-game token earnings and possessions through block-chain contracts [2]. These contracts certify the legitimacy and ownership of in-game items thus enabling its real-world value.

**Compute-intensive Mobile Augmented Reality (MAR).** Many of the Play-to-Earn games are developed for mobile devices [3], enabling people to play them whenever and wherever possible to earn in-game currency and items. With rapid development within the gaming industry, the general trend of modern Play-to-Earn games such as *Polkacity* [4], and *Reality Clash* [5] are also moving in the direction of delivering high-quality, Mobile Augmented Reality (MAR) graphics, rendering these Metaverse game graphics to be much more compute-intensive. Despite rapid development in AR technologies, current mobile devices do not have sufficient computing power to keep up with the rapidly increasing graphical demands of augmented reality (AR) game-play. Yet in the context of AR play-to-earn games, the fluidity of gaming experience should be highly regarded as it influences player's game play. An alternative, yet feasible method to powering AR game-play is to off-load computation to an edge computing server [6], [7].

**Motivation.** In the context of the play-to-earn Metaverse games, an example of in-game graphics offloading pipeline can be described as such: In a single iteration of transmissions, high-resolution in-game graphics can first be downloaded from the Metaverse Service Provider Cell Station (MSPCSs) to the players (UEs), in which players will view in-game scenes and take actions, altering the in-game scenes. These altered in-game scenes are then offloaded back to the MSPCSs for computation. In the above-mentioned computation offloading process, there is often concerns pertaining to long transmission delays, poor in-game graphic resolution, and expensive energy consumption required to power the transmission process.

**Problem.** We formulated the problem in such a way that the objective of the Metaverse Service Providers (MSP) is to minimize the (i) graphical data down-link transmission delay, (ii) up-link transmission delay, (iii) and UE device battery consumption, while maximizing (iv) players' in-game earning potential influenced by graphic resolution, by optimizing the UE-MSPCS allocation, and UE up-link energy consumption. This is to ensure player retention by improving their overall in-game fluidity, experience, profitability and UE device battery life for long periods of play. This problem is non-trivial as there is a clear trade-off between UE device battery consumption for up-link transmission and up-link transmission latency.

### A. Our Approach and Rationale

To tackle these above-mentioned problems, we introduce a novel Multi-Agent Loss-Sharing (MALS) deep reinforcement learning (RL)-based orchestrator which houses a down-link (DL) and an up-link (UL) agent which aims to jointly maximize the utility functions of the MSP. Specifically, we designed the DL agent to be a discrete-action space RL agent which chooses the UE-MSPCS allocation, and a continuous-action space RL agent which selects UE device uplink power to maximize the utility function. However, instead of providing the agents an overarching objective function, the agents are assigned to tackle sub-objectives which are within their control. It is intuitive to have differing sub-objectives in each of the DL and UL stages as the data transmitting party in each stage, and hence, variables in which they control, differ. This is to avoid "confusing" the agents while providing the agents with clearer goals. Note that the agents are *asymmetric*, in that they have separate sub-objectives and have different action space type. The UL and DL agents are also executed *asynchronously* and the DL agent will first select the UE-MSPCS allocation, conduct the DL data transfer, followed by the UL transmission and UL power selection by the UL agent. While it may seem that the UL and DL transmission processes are two separate processes: *asymmetric* and *asynchronous*, it is important to consider the concurrent optimization of both stages. We do not employ the traditional Multi-Agent Reinforcement Learning (MARL) [8] as our proposed problem is asynchronous. In our proposed scenario, the down-link and up-link agent will execute actions in an alternate fashion. However, the traditional MARL considers all agents' actions and each agent executes actions simultaneously in each time-step, rendering the traditional MARL inefficient in our proposed scenario.

## B. Related Works and Our Work's Novelty.

**Computation offloading of Metaverse applications.** Since the Metaverse is still relatively new, limited studies consider the computation offloading of Metaverse applications. Chua *et al.* [9] introduced an AR socialization over 6G wireless networks within the Metaverse scenario and proposed a deep RL approach to tackle it.

**Deep Reinforcement Learning for Task-offloading.** There are several works utilizing deep reinforcement learning for the purpose of optimizing task-offloading [10]–[12]. However, these works consider the system as a single entity. In practicality, the DL and UL process may involve different parties in the transmission process and may have different sub-objectives.

In contrast to these abovementioned works, there are also some excellent works focused on solving joint optimization problems with Multi-Agent RL. Guo et al. [13] solved the handover control and power allocation joint problem using MAPPO, under the traditional Centralized Training Decentralized Execution (CTDE) paradigm. He et al. [14] utilized Multi-Agent RL to solve a joint power allocation and channel assignment problem. However, these works do not address a multi-stage, asynchronous transmission scenario, like the one we consider in our work.

**Contributions.** Our contributions are as follows:

- ***Dual Joint Optimization:*** We formulated a novel Metaverse Play-to-Earn-enabled mobile edge computing (MEC), dual joint optimization problem.
- ***Multi-Agent-Loss-Sharing:*** We proposed a novel *asymmetric* and *asynchronous* Multi-Agent (Discrete-Continuous) Loss-Sharing reinforcement learning-based orchestrator to tackle a novel Asynchronous and Asymmetric wireless communication problem.
- ***Comparison of our proposed methods against baseline:*** We compare our (i) proposed method MALS with baseline methods such as (ii) independent dual-agent (IDA) and (iii) CTDE (MARL) [8] reinforcement learning algorithms and show the superiority of our proposed method in handling *Asynchronous* executions.

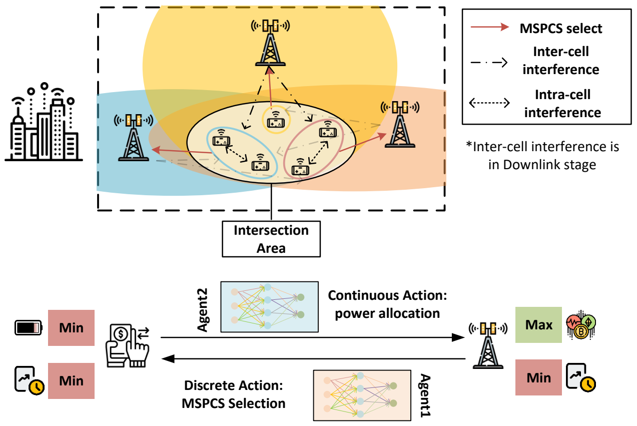

MSPCS select
Inter-cell interference
Intra-cell interference
*Inter-cell interference is in Downlink stage
Intersection Area
Agent2
Continuous Action: power allocation
Min
Min
Max
Min
Discrete Action: MSPCS Selection
Agent1

Fig. 1. System Model illustrating the interaction of the UEs with the MSCPS, Agents' sub-objectives and variables-of-control.

## II. System Model

**Problem Scenario.** Consider the real-time down-link (DL) and up-link (UL) transmission of $\mathcal{N} = \{1, 2, ..., N\}$ players (UE) moving about. In each complete transmission iteration, each UE $i \in \mathcal{N}$ begins downloading Metaverse in-game scenes from an MSPCS. We consider both intra-cell interference and inter-cell interference in the DL transmission. After the DL of in-game graphical data, we consider the UL transmission of the same set of $\mathcal{N} = \{1, 2, ..., N\}$ players' (UE) changes to in-game graphical scenes, to its allocated MSPCS from a set of $\mathcal{M} = \{1, 2, ..., M\}$. Each UE $i \in \mathcal{N}$ will upload their Metaverse in-game graphical scenes to their allocated MSPCS. We consider intra-cell interference in the up-link transmission. For the sake of simplifying an already complex scenario, we consider the scenario in which users only make movement after each successful down- and up-link data transmissions.

### A. Down-link Communication

Our communication scenario is based on wireless cellular network. Each MSPCS $\mathcal{M} = \{0, 1, 2, ..., M\}$ will have its down-link channel assigned to the UEs $\mathcal{N} = \{1, 2, ..., N\}$. $D^t = \{D_1^t, D_2^t, ..., D_N^t\}$ is the size of the in-game graphics to be downloaded from the MSPCSs to the UEs, where $D_i^t$ denotes the size of data to be downloaded by player (UE) $i \in \mathcal{N}$ at transmission iteration $t$. In addition, we have channel allocation $\mathbf{c}^t = (c_1^t, ..., c_N^t)$, where $c_i^t = v(i \in \mathcal{N}, v \in \mathcal{M})$ denotes UE $i$ is allocated to MSPCS $v$ at iteration $t$. We can derive the *signal to interference plus noise ratio* (SINR) of UE $i$ at iteration-step $t$ as:

$$\Gamma_i^t = \frac{g_{v,i}^t p_{v,i}^t}{g_{v,i}^t \sum\limits_{n\in\mathcal{N}\setminus\{i\}:c_n^t=c_i^t} p_{v,n}^t + \sum\limits_{j\in\mathcal{M}\setminus\{v\}} g_{j,i}^t \sum\limits_{k\in\mathcal{N}:c_k^t=c_i^t} p_{j,k}^t + w\sigma^2}. \tag{1}$$

where $p_{v,i}^t$ is the transmit power of MSPCS $v$ for communication with UE $i$ at iteration step $t$, $g_{v,i}^t$ is the channel gain between MSPCS $v$ and UE $i$ at iteration step $t$. $p_{v,n}^t$ is the power allocated by CS $v$ to user $n \neq i$ at iteration step $t$, $g_{j,i}^t$ is the channel gain between other MSPCS $j$ and UE $i$, and $p_{j,k}^t$ is the transmit power of MSPCS $j$ to other UE $k$. $w$ is the bandwidth of each MSPCS and $\sigma^2$ denotes the additive white gaussian background noise. Note that from here, we may discuss variables and terms without index $t$ for convenience and simplicity of explanation.

Essentially, $g_{v,i} \cdot p_{v,i}$ is the transmission signal from MSPCS $v$ to UE $i$, $\left[g_{v,i} \cdot \sum\limits_{n\in\mathcal{N}\setminus\{i\}:c_n^t=c_i^t} p_{v,n}\right]$ is the intra-cell interference caused by the interaction between MSPCS $v$ and other UE $n \neq i$ to UE $i$, and $\left[\sum\limits_{j\in\mathcal{M}\setminus\{v\}} g_{j,i} \sum\limits_{k\in\mathcal{N}:c_k^t=c_i^t} p_{j,k}\right]$ is the inter-cell interference caused by the signals of all other MSPCSs $j \neq v$ to UE $i$.

For each iteration step $t$, the down-link data transfer rate $r_i^{d,t}$ from the MSPCS to UE $i$ is influenced by the SINR as such, based on the Shannon-Hartley theorem [15]:

$$r_i^{d,t} = w \cdot \log_2\left(1 + \Gamma_i^t\right), \tag{2}$$

For a fixed data size to be transmitted, a higher data transfer rate results in a shorter down-link transmission delay at iteration step $t$ as shown:

$$\ell_i^{d,t} = \frac{D_i^t}{r_i^{d,t}} \tag{3}$$

where $D_i^t$ is the size of down-link in-game data from MSPCS to UE $i$ at iteration step $t$ and the resolution-impacted earning potential function $\omega(r_i^{d,t})$ is influenced by the UE $i$'s SINR. Intuitively, a more efficient UE to MSPCS allocation results in a higher UE $i$ SINR, UE $i$ down-link transmission rate (data transmitted per unit time $r_i^{d,t}$), and lower total down-link transmission delay $\ell_i^{d,t}$, at iteration $t$, for all UE $i$. A higher data transmitted per unit time is translated to improved graphic resolution per unit time, resulting in an enhancement in players' in-game visuals, which translates to better in-game performance and hence, better resolution-influenced earning potential.

### *B. Up-link Communication*

In the up-link leg of the communication model, each player (UE) allocated to an MSPCS will upload its graphical changes to the previously downloaded Metaverse in-game scene data, to its assigned MSPCS. These graphical changes to the scenes are denoted $zD^t = \{zD_1^t, zD_2^t, ..., zD_N^t\}$ where $z$ represents a scaling factor, relating the UL and DL data size. We consider the intra-MSPCS interference and derive the *signal to interference plus noise ratio* (SINR) of MSPCS $v$ at iteration step $t$ as:

$$\beta_i^t = \frac{g_{v,i}^t \cdot p_{i,v}^t}{\sum\limits_{n\in\mathcal{N}\setminus\{i\}:c_n^t=c_i^t} p_{n,v}^t \cdot g_{v,n}^t + w\sigma^2} \tag{4}$$

where $p_{i,v}^t$ is the power transmitted from UE $i$ to MSPCS $v$ at iteration $t$, $p_{n,v}^t$ is the power transmitted from other UE $n$ allocated to MSPCS $v$, to the MSPCS $v$, at iteration $t$, and $g_{v,n}^t$ is the channel gain between other UE $n$ and MSPCS $v$ at iteration $t$. The up-link data rate $r_i^{u,t}$can similarly be represented by $r_i^{u,t} = \vartheta \cdot \log_2(1 + \beta_i^t)$, where $\vartheta$ is the bandwidth of the transmitting UE devices. Similarly, as the number of UEs allocated to the same MSPCS increases, the SINR and hence rate of data transfer, decreases.

The up-link transmission latency of UE $i$ at iteration step $t$ is defined as such:

$$\ell_i^{u,t} = \frac{zD_i^t}{r_i^{u,t}} \tag{5}$$

where $zD_i^t$ is the up-link in-game graphics data size of UE $i$ at iteration step $t$, $r_i^{u,t}$ is the up-link data transfer rate of UE $i$ at iteration step $t$. The energy consumed by UE $i$ to upload the in-game graphics data at iteration step $t$ is defined as:

$$E_i^t = p_{i,v}^t \cdot \ell_i^{u,t} \tag{6}$$

In each round (iteration or step $t$) of in-game data up-link, the battery-life remaining in UE $i$ is:

$$Bat_i^t = Bat_i^{t-1} - E_i^t \tag{7}$$

where $Bat_i^0$ is the battery capacity of UE $i$ at full charge. Finally, we represent the total battery charge consumed as of iteration step $t$ as a percentage of battery size at full charge, for each UE as such: $Q_i^t = \frac{Bat_i^0 - Bat_i^t}{Bat_i^0} \cdot 100$, where $Q_i^t$ is the battery charge consumed as a percentage of total device battery capacity of UE $i$ as of time step $t$. The selection of up-link transmission power influences the up-link transmission latency, which influences the fluidity. On the other hand, the selection of up-link transmission power also influences UE battery charge consumption.

### *C. Problem formulation*

To sum up, the DL sub-objective of the MSP is to find the optimal UE-MSPCS allocation arrangement $c^t$ which minimizes the total down-link latency $\ell_i^{d,t}$ while maximizing the total resolution-influenced UE earning potential $\omega(r_i^{d,t})$. We formulated our down-link utility function as:

$$\min_{\boldsymbol{c}^t} \sum_{t=1}^{T} \sum_{i\in\mathcal{N}} q \cdot \ell_i^{d,t} - (1-q) \cdot b \cdot \omega(r_i^{d,t}), \tag{8}$$

$$s.t. \quad \sum_{j=1}^{M} c_{i,j}^t = 1, \forall i \in \mathcal{N}, \forall t \in T, \tag{9}$$

$$D_i^t \geq 0, \forall i \in \mathcal{N}, \forall t \in T \tag{10}$$

where $T$ is the total number of down-link transmissions of in-game graphical data. The constraint (9) restricts each UE to be allocated to only one MSPCS in each iteration step. The constraint (10) ensures that the down-link data for all UEs in each iteration step is non-negative. $q$ is a scaling factor which seeks to balance the order difference in magnitude between $\ell_i^{d,t}$ and $\omega(r_i^{d,t})$. $b$ is a factor which places both terms $\ell_i^{d,t}$ and $b \cdot \omega(r_i^{d,t})$ on the same measurement unit.

On the other hand, the UL sub-objective of the MSP is to find the optimal UE up-link power allocation $p^t$ which minimizes the total up-link latency $\ell_i^{u,t}$ and UE device battery consumption $Q_i^t$. We formulate our up-link utility function as:

$$\min_{\boldsymbol{p^t}} \sum_{t=1}^{T} \sum_{i \in \mathcal{N}} h \cdot \ell_i^{u,t} - (1-h) \cdot f \cdot Q_i^t, \tag{11}$$

$$s.t. \quad \sum_{j=1}^{M} c_{i,j}^t = 1, \forall i \in \mathcal{N}, \forall t \in T \tag{12}$$

$$Q_i^t \leq 100, \forall i \in \mathcal{N}, \forall t \in T, \tag{13}$$

where $\ell_i^{u,t}$ is the up-link transmission delay of player (UE) $i$ at iteration step $t$, $Q_i^t$ is the UE battery consumption of UE $i$ at iteration step $t$, and $h$ is a factor which scales $Q_i^t$ to be on a balanced magnitude of order as $\ell_i^{u,t}$. Scaling factor $f$ in the objective function is essential in placing both terms $\ell_i^{u,t}$ and $f \cdot Q_i^t$ on the same measurement unit. Constraint (12) ensures that each UE is allocated to only 1 MSPCS. Finally, constraint (13) ensures that the UE battery consumption has to be less than 100% (or battery life (%) has to be greater than 0) for the up-link transmission.

## III. Reinforcement learning settings

For our work, we assign two reinforcement learning agents, one (DL agent) controlling the down-link transmission UE-MSPCS allocation, and the other (UL agent) controlling the up-link transmission power selection. The DL agent and UL agent serve their sub-objectives (8) and (11), respectively.

### A. State, Action, Reward Design

**State.** For the DL agent's observation state $s^d$, we have chosen to include 1) Channel gain between each UE and all MSPCSs: $g_{v,i}^t$, 2) in-game scene DL data size at each transmission iteration $t$: $D_i^t$, as $D_i^t$ influences latency $\ell_i^{d,t}$ and $g_{v,i}^t$ influences both data DL transmission rate, latency $\ell_i^{d,t}$ and resolution-influenced in-game earning potential $\omega(r_i^{d,t})$.

For the UL agent's observation state $s^u$, we have chosen to include 1) Channel gain between each UE and all MSPCSs: $g_{v,i}^t$, as it influences data UL transmission rate, latency $\ell_i^{u,t}$ and uplink transmission battery consumption $Q_i^t$. As we are enforcing that each UE device has positive battery-charge to continue the up-link transmission at each transmission iteration step $t$, we have to include 2) Battery charge consumed (%) of UE devices $Q_i^{t-1}$ from the previous iteration for the calculation of the battery charge consumed (%) $Q_i^t$ in the present iteration.

**Action.** In the DL communication model, the DL agent's action is to decide the UE to MSPCS allocation, in which the action space can be written as such: $a^{d,t} = \mathbf{c}^t = \{c_1^t, ...c_N^t\}$. The number of the discrete actions is $N^M$, where $N$ denotes the number of UEs and $M$ is the total number of MSPCSs. This signifies that each UE will be allocated one MSPCS.

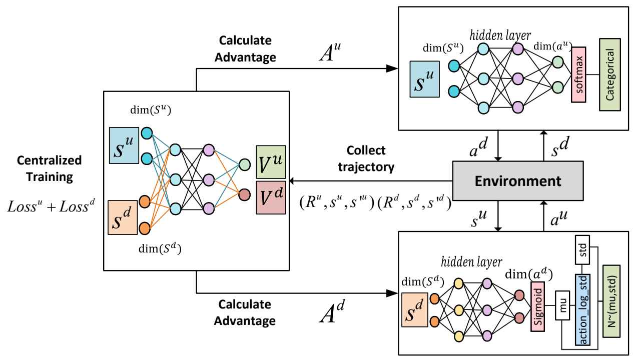


Fig. 2. Reinforcement Learning Multi-Agent-Loss-Sharing Proximal Policy Optimization (MALS-PPO) structure and model update.

In the UL communication model, the action space is continuous and action dimension is $N$, in which there is one power output value selected for each UE to transmit the in-game data to its assigned MSPCS. The up-link action space at each transmission iteration is written as such: $a^{u,t} = \mathbf{p}_t = \{p_1^t, ..., p_N^t\}$, where $p_i^t$ $(i \in \mathcal{N})$ is the up-link power selected at iteration $t$ for UE $N$.

**Reward.** For the **down-link** communication model, the reward given to the DL agent at transmission iteration $t$ is given as such:

$$R^{d,t} = -\frac{\sum_{i \in \mathcal{N}} q \cdot \ell_i^{d,t} - (1-q) \cdot b \cdot \omega\left(r_i^{d,t}\right)}{N} \tag{14}$$

while for the up-link communication model, the reward given to the UL agent at transmission iteration $t$ is given as such:

$$R^{u,t} = -\frac{\sum_{i \in \mathcal{N}} h \cdot \ell_i^{u,t} - (1-h) \cdot f \cdot Q_i^t}{N} \tag{15}$$

The intuition for the down-link and up-link reward assignment follows the utility function in equation (8) and (11), respectively. We divide the reward functions by $N$ players to find an average reward, as average reward received per UE is more intuitive than the reward sum.

### B. Multi-Agent-Loss-Sharing (MALS)

In our work, we equipped our MALS algorithm with a discrete-action space Actor, a continuous-action space Actor and a multi-head Critic, based on a Proximal Policy optimization (PPO) algorithm [16].

**Function process**: In each transmission iteration, the rewards $(R^{u,t}, R^{d,t})$ with states $(s^{u,t}, s^{d,t})$ and next states $(s^{u,t+1}, s^{d,t+1})$ will be sent to critic for generating the state-value $V$ and loss functions of Critic. Finally, $V$ will be used to calculate the advantage value for updating the Actors. This process repeats until the end of an episode. The above-mentioned process is illustrated in Fig 2.

**Asymmetric Actor**: In MALS, there are two Actors, one is responsible for arranging channel resources in the DL stage, and the other is responsible for allocating transmission power in the UL stage. Apart from their asymmetric task, the action space types for both DL and UL agents, and hence policy parameterizations, are dissimilar.

Each agent attempts to achieve its own sub-objectives. In MALS, $\theta_d$ and $\theta_u$ are the network weights of DL agent and UL agent and the gradients $\Delta\theta_d$, $\Delta\theta_u$ of the DL agent and UL agent are:

$$\Delta\theta_u = \mathbb{E}_{(s^{u,t},a^{u,t})\sim\pi_{\theta_u{'}}}[\nabla f^t(\theta_u)A^u(s^{u,t})] \tag{16}$$

$$\Delta\theta_d = \mathbb{E}_{(s^{d,t},a^{d,t})\sim\pi_{\theta_d{'}}}[\nabla f^t(\theta_d)A^d(s^{d,t})] \tag{17}$$

where $f^t(\theta) = min\{R^t(\theta), clip(R^t(\theta), 1-\epsilon, 1+\epsilon)\}$ in the case of PPO, in which $R^t$ is the reward obtained by an agent at iteration $t$, and $\epsilon$ is the clipping parameter. The advantage function $A(\cdot)$ of both UL and DL agents are estimated with a truncated version of GAE [17].

**Multi-head Critic**: In our problem, the objectives of each agent are different. Thus, we propose the MALS to aid the estimation of the value function in the reward problem. This Multi-head Critic have two heads and the value network is divided into two branches $V_\phi^u, V_\phi^d$. The value function can be estimated by the two-head neural network $F(\cdot)$ as shown below:

$$V_\phi^u = F(s^u) \tag{18}$$

$$V_\phi^d = F(s^d) \tag{19}$$

Then, we sum the weighted losses of each head as the loss function of the Multi-head Critic:

$$L^u(\phi) = (V_\phi(s^{u,t}) - A^{u,t} - \gamma V_{\phi'}(s^{u,t+1}))^2 \tag{20}$$

$$L^d(\phi) = (V_\phi(s^{d,t}) - A^{d,t} - \gamma V_{\phi'}(s^{d,t+1}))^2 \tag{21}$$

$$L(\phi) = \kappa_1 \times L^u(\phi) + \kappa_2 \times L^d(\phi) \tag{22}$$

where $\phi$, $\phi'$ are the weights of the critic network and the target network, respectively, and $\kappa_1$ and $\kappa_2$ are the weights of each head's loss. We update the Multi-head critic through (22) to improve the estimate of the multi-value function.

## IV. Experiment results

### A. Configuration

We use three congestion settings: 3 MSPCSs and 4 to 6 UEs (with 3 MPSPCSs and $z$ number of UEs denoted as "3$z$" from hereon) to test our proposed Multi-Agent-Loss-Sharing (MALS) reinforcement learning-based method. The bandwidth and noise are simulated to be $w = 10$ MHz and $\sigma^2 = -100$ dBm, respectively. We design the in-game resolution-influenced earning potential function to be: $\omega(r_i^{d,t}) = P \cdot \exp^{-r_i^{d,t}}$, where $P$ is a positive scaling factor which controls the maximum profit a user can make. Intuitively, as UE $i$ down-link data size at iteration step $t$ increases, the resolution of the in-game graphics increases, resulting in an increase in resolution-influenced in-game earning potential. This earning potential function is arbitrary and is chosen as such for simplicity. We set different MSPCS power output and initialized random locations within a 100m by 100m area for the MSPCS and for different UEs as: $(1.5, 2.0)$ Watt, $(0, 100)$ m, respectively. The metaverse in-game graphic sizes $D_i^t$ of each UE varies every episode from

**Algorithm 1** MALS-PPO (proposed RL structure)

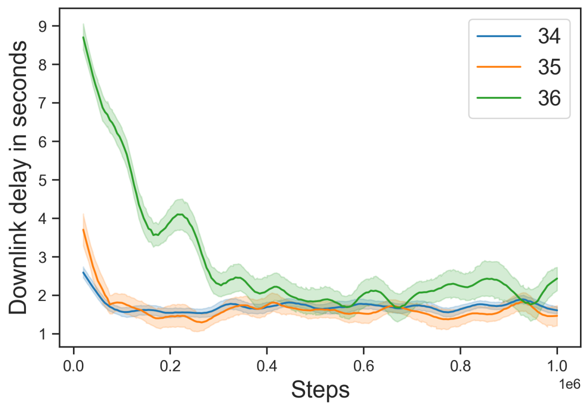
```
Initiate: critic parameter $\phi$ and target network $\phi^{'}$, down-link
    actor parameter $\theta_d$, up-link actor parameter $\theta_u$, initial state
    $s^{d,1}$, $s^{d,t} \leftarrow s^{d,1}$
 1: for iteration = $1, 2, ...$ do
 2:   DL agent execute action according to $\pi_{\theta_d'}(a^{d,t}|s^{d,t})$
 3:   Get reward $R^{d,t}$ and up-link state $s^{u,t}$
 4:   UL agent execute action according to $\pi_{\theta_u'}(a^{u,t}|s^{u,t})$
 5:   Get reward $R^{u,t}$ and next step down-link state $s^{d,t+1}$
 6:   if iteration $\geq 2$ then
 7:     Sample $(s^{d,t}, a^{d,t}, s^{u,t}, a^{u,t}, R^{d,t}, R^{u,t}, s^{d,t+1}, s^{u,t+1})$
        iteratively
 8:   end if
 9:   $s^{d,t} \leftarrow s^{d,t+1}$, $s^{u,t} \leftarrow s^{u,t+1}$
10:   Compute advantages $\{A^{d,t}, A^{u,t}\}$
11:   Compute target values $\{V_{targ}^{d,t}, V_{targ}^{u,t}\}$
12:   for $k$ = $1, 2, ..., K$ do
13:     Shuffle the data's order and set batch size $bs$
14:     for $j$=$0, 1, ..., \frac{T}{bs} - 1$ do
15:       Compute gradient for down- and up-link actors:
          $\nabla\theta_d, \nabla\theta_u$
16:       Apply gradient ascent on $\theta_d$ using $\nabla\theta_d$ by eq. (17)
17:       Apply gradient ascent on $\theta_u$ using $\nabla\theta_u$ by eq. (16)
18:       Update critic with loss from eq. (22)
19:     end for
20:   end for
21:   Assign target network parameters $\phi^{'} \leftarrow \phi$ every $C$ steps
22: end for
```

800 to 1000 Mb. We set $P$, $b$ and $f$ to be 100, 50 and 75, respectively, after empirical tuning, and $h$ and $q$ to be 0.5. In each transmission iteration step, UEs randomly move a maximum of 10m in $x$ and $y$ directions, in which $x$ and $y$ represents the relative longitudinal and latitudinal directions in our 100m by 100m map. We adopt the ADAM optimizer [18] for all our implemented algorithms. To better observe the final model performances, we trained the models for 1,000,000 iterations (steps) for all congestion settings. We conducted the training and simultaneous evaluation of the models for each of the configurations at different seed settings: seed 0 to seed 9.

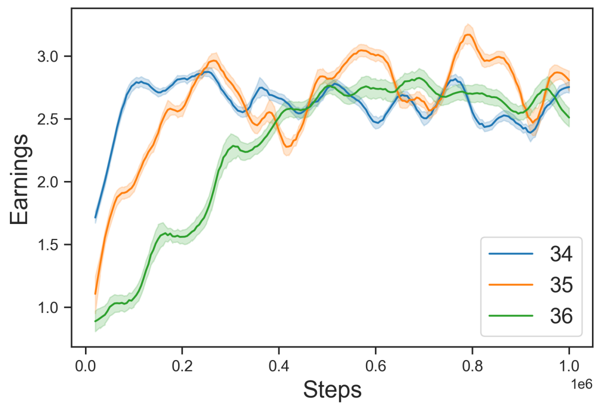
### B. Result analyses

*1) **MALS model convergence:*** In training our MALS model, all three configurations showed a general increment of achieved test reward for both the DL and the UL agents as training progresses (shown in Figure 4).

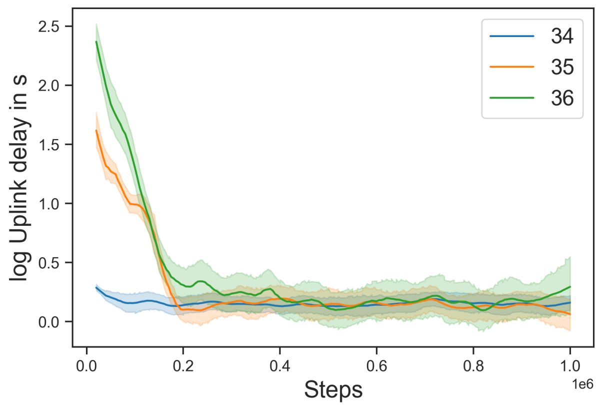
**Down-link.** The gradual increment in the MALS down-link agent reward, as training progresses, can be attributed to the decrease in in-game down-link transmission time and increase in resolution-influenced earning potential of the UE devices (shown in Figure 3) with each training step. These observations directly reflect that the down-link agent is allocating UE-MSPCS more efficiently, decreasing the down-link latency and



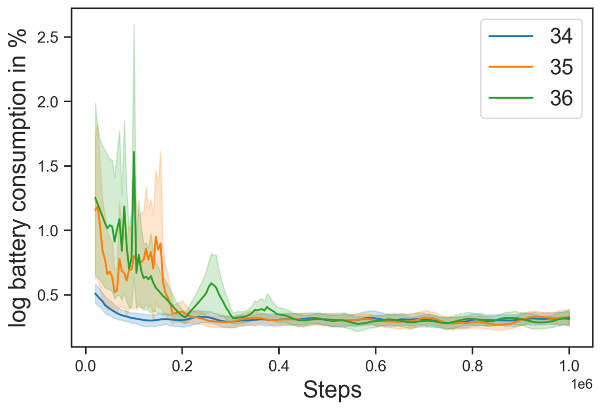
Average down-link delay.



Average UE earning potential.



Average up-link delay.



Average battery consumption percentage.

Fig. 3. Key metrics performance obtained through MALS across training steps.

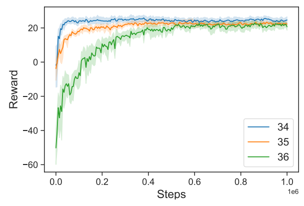
improving the resolution-influenced earning potential of the UEs.

**Up-link.** Similarly, we note that the improvement of the MALS up-link agent reward received, as training progresses, can be attributed to the more optimal selection of UE transmission output power for the up-link transmission, resulting in a lower average up-link delay and lower battery charge consumption (shown in Figure 3).

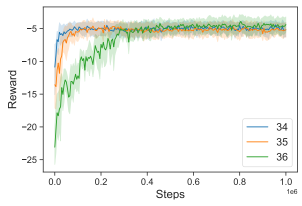
*2) **Comparison with other reinforcement learning model structures**:* We studied how our proposed MALS model compared to two other RL-based model structures; the most intuitive (ii) independent dual agent (IDA) (which employs two individual agents in which information transfer is only via the states), and (iii) CTDE (which is the framework used for MARL) [8]. Our proposed model (MALS model structure) achieves a smooth convergence for each of the configuration, exhibiting a narrow range of down-link and up-link rewards, across different seeds (shown in Figure 4). The narrow bands indicate that the proposed model's performance is stable across the different seed settings, for each of the congestion configurations.

On the other hand, the (ii) IDA model took significantly more steps to converge (shown in Figure 4). Furthermore, the variance of the down-link rewards obtained is much larger, with some seeds achieving excellent reward performance, yet failing in other seeds. The excellence of the down-link reward obtained by the IDA model in some seeds is reflected as a compromise in the performance of the independent dual agent's up-link reward. From Figure 4, we can observe for the IDA model in ”34” and ”35” configurations, there is a slight improvement in the up-link rewards in the early steps of training, followed by a subsequent decline in rewards obtained in later steps. This indicates instability and hyper-sensitivity of the model when handling different seed settings. It is apparent that the the down-link agent's rewards are prioritised over the up-link agent's rewards in this case, and this is not ideal in real-world.

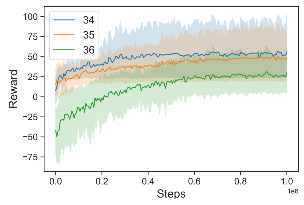
The (iii) CTDE model achieved lackluster performance when compared to our proposed methods (MALS) for both the down-link and up-link rewards (shown in Figure 4), and produced a large variance of down-link and up-link reward outputs across the different seeds, albeit not as severe as the IDA model. The lower rewards and higher variance across seeds reflect poorer performance, instability and unreliability of the CTDE model.



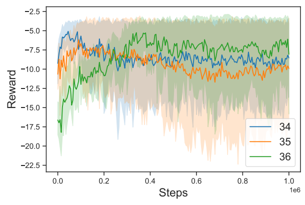
Down-link reward for MALS method.



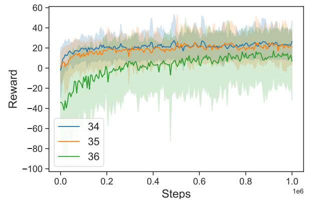
Up-link reward for MALS method.



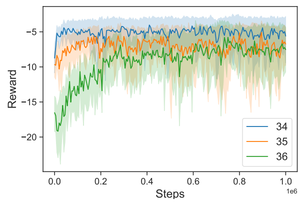
Down-link reward for IDA model.



Up-link reward for IDA model.



Down-link reward for CTDE model.



Up-link reward for CTDE model.

Fig. 4. Up-link and down-link agent rewards obtained for MALS model, Independent Dual Agent model, CTDE model, across congestion settings and seed 0 to 9. Bands within the graph indicate the range of reward values obtained.

## V. Conclusion

In this work, we considered a Metaverse Play-to-Earn mobile edge computing framework and formulated an asymmetric (discrete-continuous) and asynchronous (alternating DL and UL) dual-joint optimization where the MSPs' objective is to minimize in-game graphics DL, UL transmission latency, and UE device battery charge consumption, while maximizing UEs' in-game resolution-influenced earning potential. We then proposed a novel multi-agent loss-sharing (MALS) RL model to tackle the above-mentioned asynchronous and asymmetric problem, and demonstrated its superiority in performance over other methods. In future works, we will conduct thorough and in-depth joint optimization weighting-analyses of our proposed model.

## References

[1] E. Sheridan, M. Ng, L. Czura, A. Steiger, A. Vegliante, and K. Campagna, "Framing the future of web 3.0: Metaverse edition," https://www.goldmansachs.com/insights/pages/framing-the-future-of-web-3.0-metaverse-edition.html, 2021.

[2] R. Browne, "Cash grab or innovation? the video game world is divided over nfts," Dec 2021. [Online]. Available: https://www.cnbc.com/2021/12/20/cash-grab-or-innovation-the-video-game-world-is-divided-over-nfts.html

[3] A. N. Roos, "Best play-to-earn games with nfts or crypto," Sep 2022. [Online]. Available: https://www.playtoearn.online/games/

[4] [Online]. Available: https://www.polkacity.io/

[5] "Reality clash." [Online]. Available: https://realityclash.com/

[6] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1584–1607, 2019.

[7] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Compute-and data-intensive networks: The key to the metaverse," *arXiv preprint arXiv:2204.02001*, 2022.

[8] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[9] T. J. Chua, W. Yu, and J. Zhao, "Resource allocation for mobile metaverse with the internet of vehicles over 6g wireless communications: A deep reinforcement learning approach," *arXiv preprint arXiv:2209.13425*, 2022.

[10] H. Lu, C. Gu, F. Luo, W. Ding, and X. Liu, "Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning," *Future Generation Computer Systems*, vol. 102, pp. 847–861, 2020.

[11] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on sarsa," *IEEE Access*, vol. 8, pp. 54 074–54 084, 2020.

[12] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Transactions on Mobile Computing*, 2020.

[13] D. Guo, L. Tang, X. Zhang, and Y.-C. Liang, "Joint optimization of handover control and power allocation based on multi-agent deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13 124–13 138, 2020.

[14] C. He, Y. Hu, Y. Chen, and B. Zeng, "Joint power allocation and channel assignment for noma with deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2200–2210, 2019.

[15] J. R. Pierce, *An introduction to information theory: symbols, signals and noise*. Courier Corporation, 2012.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[17] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.