Incentive Mechanism Design for Relay Communication in Semantic Communication-enabled UAV Networks

***Abstract*—**

***Index Terms*—Semantic communication, distributed spectrum sharing acesss, spectrum and power allocation, multi-agent reinforcement learning.**

I. Introduction

A. Related Work

B. Main Contributions

- We leverage a semantic information extraction framework for UAV-collected images through scene graph generation. We employ the RelTR method to transform images into scene graphs, which capture subject-predicate-object relationships. To optimize transmission efficiency, we introduce a triplet priority level that evaluates the importance of each triplet and filters them based on a dynamic priority level threshold, γ_i, determined by the UAV. This enables flexible control over the size and detail of the scene graphs sent to the BS. Furthermore, we define a full scene graph factor, ρ_k, to measure the completeness of transmitted data, ensuring the balance between data transmission efficiency and the preservation of essential image context. This framework optimizes the trade-off between reducing data size and maintaining accurate image reconstruction under constrained network resources.
- We investigate the optimization problem of maximizing long-term rewards of multi-UAV networks by jointly designing transmission power, channel and priority level threshold selection strategies. Specifically, we formulate reward as a function of transmission capacity and full scene graph factor and constraint it with quality of service (QoS). Because of dynamic environment, the formulated optimization problem is non-trivial.
- We model the problem above by employing the stochastic game theory [1], then develop a centralized training and decentralized implementation MARL based algorithm for solving the formulated stochastic game of multi-UAV networks. Each UAV is an independent learning agent and can only observe its local environment, which substantially reduce the computational burden of the system. During centralized training phase, the average reward of all UAVs in the system is used for training, which help the UAVs learn to cooperate more effectively and achieve better global performance.
- We investigate the performance of different adaptive optimization algorithms, namely Adam [2] and RMSProp [3] and the pure vallina SGD [4] on solving our specific problem. The hypermeters setting are carefully derived from simulations.
- Simulation results are provided to demonstrste the performance of the system over two different setups: system-centric mode and user-centric mode. In the system-centric mode, UAVs are allowed to proactively sacrifice their individual performance to minimize interference, thereby improving overall system flexibility and exploring the spectrum's true capacity in the given environment. In the user-centric mode, UAVs prioritize maintaining service quality above a certain threshold, even at the cost of increased interference and potential performance degradation. This mode enforces stricter power control selections and employs more aggressive penalties to drive the UAVs to manage interference effectively. Our work demonstrates how these two modes influence system performance, providing insights into trade-offs between global system optimization and individual user satisfaction.

II. System Model And Problem Formulation

We consider a system including a set of N UAVs, denoted as $\mathcal{N} = \{1, \dots, N\}$, a set of U ground users, denoted as $\mathcal{U} = \{1, \dots, U\}$ and a BS. Each UAV performs two functions simultaneously. Firstly, it acts as a relay, receiving data from users within its coverage and transmitting the data to the BS. Secondly, it collects images of the physical system and employs a SemCom technique, specifically Relation Transformer (RelTR) which proposed in [5], to extract scene graphs from the images. The scene graph consists of triplets, which will be further explained in the following sections. Then, the UAV transmits these triplets to the BS.

The whole transmission is divided into multiple time steps, and each time step has the duration of τ. Each time step is further divided into two short phases, i.e. phase 1 with duration of τ_1 and phase 2 with duration of $\tau_2 = \tau - \tau_1$. In phase 1, each UAV receives the data from its users while sensing an area of interest and leveraging RelTR to extract the triplets. In phase 2, the UAVs transmit the data collected from the users and the semantic data obtained in phase 1 to a remote BS. The BS has a limited number of channels that the UAVs select. Key elements of the proposed system are demonstrated in detail below.

A. User Association and Uplink Transmission

In phase 1, each user u selects a UAV as relay to forward its data to the BS. We consider a decentralized user association in which each user locally selects the UAV with the highest signal-to-interference-plus-noise ratio (SINR). We consider the network consisting of the UAV and its associated as a local network that uses a unlicensed channel. We denote $\texttt{SINR}_{u,i}$ as the SINR achieved by the user u as associating UAV i. Particularly, $\texttt{SINR}_{u,i}$ is defined by

$$\texttt{SINR}_{u,i} = \frac{p_u |h|^2_{u,i} d^{-2}_{u,i}}{\Gamma_i + W_i \sigma^2}, \tag{1}$$

where p_u is the tranmission power of user u, $h_{u,i}$ is user-UAV channel small-scale fading, $d_{u,i}$ is distance between the user u and the UAV i, Γ_i is the interference caused from other networks due to the unlicensed channels, W_i is the available bandwidth of UAV i and σ^2 is the variance of noise. The problem of each user u is expressed by

$$\arg\max_{i \in \mathcal{N}} \texttt{SINR}_{u,i}. \tag{2}$$

We assume that there are U_i users selecting UAV i. U_i can vary over time steps. To achieve the fast resource allocation, UAV i uses the round-robin technique for the user access. The transmission rate achieved by user u is expressed by

$$C_{u,i} = \frac{W_i}{U_i} \log_2(1 + \texttt{SINR}_{u,i}). \tag{3}$$

Denote the data size in bit that UAV i receives from a single user u as $Q^{\text{relay}}_{u,i}$ and from all U_i users as Q^{relay}_{i}. Mathematically,

$$Q^{\text{relay}}_{u,i} = \tau_1 C_{u,i}. \tag{4}$$

$$Q^{\text{relay}}_{i} = \sum_{u \in \mathcal{U}_i} Q^{\text{relay}}_{u,i}. \tag{5}$$

Under the dynamic environment of the UAV networks, channel small-scale fading $h_{u,i}$ and interference Γ_i vary over time steps. Also, locations of the users change over time steps. Thus the number of users associating with UAV i varies over time steps. These changes result in the variation of Q^{relay}_{i} over time steps.

B. Image Collection and Semantic Information Extraction

1) Scene graph extraction: We denote λ_i as the sensing rate of the UAV i. The number of images UAV i collecting in time step t is

$$I^t_i = \tau_1 \lambda_i. \tag{6}$$

Denote the set of collected images as $\mathcal{I}^t_i = \{1, \ldots, I^t_i\}$. These images are led into the semantic module of UAV i to be extracted to scene graphs. In this work, we utilize the RelTR method, proposed in [5], to generate graph scenes. Each scene graph represents relationships between subjects and objects in the image, which are expressed as triplets in the form of < `subject-predicative-object` >. Here, `predicative` describes how the object is related to the object. The process of generating a graph representation from the original image using RelTR was summarized in Fig 1.

The images sensed by UAV i in the same time step are highly similar in context, hence having highly similar size as well as the number of triplets. As such, in this work, we ignore the trivial difference between the images collected in the same time step and assume that image $k^t_i \in \mathcal{I}^t_i$ can represent all others images in $\mathcal{I}^t_i$. However, the context may vary between different time step. For example, assume the image in Fig. 1 is k^t_i, which captures a bus, a car and a motorcycle. In the next time step $t+1$, these vehicles may move out the sensing area of UAV i and no other changes happened. Then, scene graph k^{t+1}_i contains less subjects and objects than scene graph k^t_i, hence containing less triplets. To further investigate the changes of triplets, we denote $\mathcal{S}_k = \{1, \ldots, S_k\}$ as the set of S_k triplets of the scene graph k^t_i.

2) Triplet priority level: We assign each triplet $s \in \mathcal{S}_k$ an exclusive value called priority level ϕ_s, with $\phi_s \in [0, 1]$ to evaluate its importance. The higher ϕ_s, the more important the triplet is.

3) Priority level threshold: We denote priority level threshold γ_i, with $\gamma_i \in [0, 1]$, as a filter parameter of the semantic module of UAV i. Only the triplets $s \in \mathcal{S}_k$ with priority level $\phi_s \geqslant \gamma_i$ are transmitted to the BS. UAV i selects priority level threshold γ_i for all scene graphs $k^t_i \in \mathcal{I}^t_i$ in the beginning of phase 2 of time step t, based on the sensed context as well as their willingness for transmitting the triplets. Apparently, if γ_i is high, each output scene graph only includes a few most important triplets with high priority. In result, the scene graph is low in size, which can easily be transmitted to the BS. However, this also means the BS can not fully recover the original image. In other words, with a high value of γ_i, the scene graphs do not fully reflect the original context. In contrary, if γ_i is low, each output scene graph includes most of the triplets, hence heavy in size but fully reflects the original context.

We denote $Q^{\text{graph}}_{i,k}$ as the size in bit of scene graph k^t_i. Then, $Q^{\text{graph}}_{i,k}$ is a function of γ_i and expressed as follows:

$$Q^{\text{graph}}_{i,k}(\gamma_i) = \sum_{s \in \mathcal{S}_k} c_s ||s||, \tag{7}$$

where $||s||$ is the size of triplet s and c_s refers to the decision variable. Particularly, $c_s = 1$ if the priority of triplet l is greater or equal to γ_i and $c_s = 0$ otherwise. The total size of the scene graphs of UAV i that need transmitting in a time step is

$$Q^{\text{graph}}_{i}(\gamma_i) = \sum_{k^t_i \in \mathcal{I}^t_i} Q^{\text{graph}}_{i,k}(\gamma_i). \tag{8}$$

To this end, the total size of the data that the UAV i transmits to the BS during phase 2 is given by

$$Q_i = Q^{\text{graph}}_{i}(\gamma_i) + Q^{\text{relay}}_{i}. \tag{9}$$

4) Full scene graph factor: UAV i is expected to fully transmit Q_i. However, given the limited channel resources, Q_i may only be partially transmitted. In these cases, the value of γ_i is raised to filter out the less necessary triplets in order to lower size of the scene graphs. To evaluate the number of left triplets, we introduce full scene graph factor ρ_k as follows:

$$\rho_k = \frac{S_k(\gamma_i)}{S_k}, \tag{10}$$

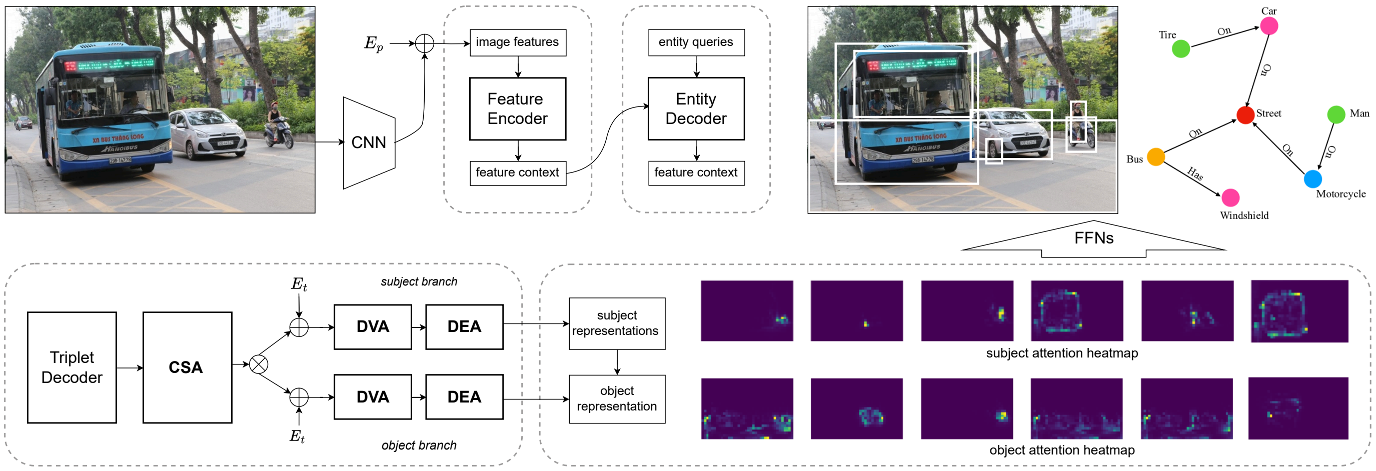


Fig. 1: Flowchart of RelTR for generating scene graph from original image. **CSA**, **DVA** and **DEA** are respectively Coupled Self-Attention, Decoupled Visual Attention and Decoupled Entity Attention. E_p is the positional encodings, E_t is the triplet encoding. $\oplus$ indicates element-wise addition, while $\otimes$ indicates concatenation or split.

in which $S_k(\gamma_i)$ is the number of triplets passing the filter with a selected value of γ_i. We observe that $\rho_k \in [0,1]$. We expect all S_k triplets to be transmitted to the BS such that the original image is fully recovered. In other words, we expect that $\rho_k = 1$. However, the transmissions of the scene graphs may not be finished within the second phase. Thus, ρ_k needs to be optimized.

C. UAV-BS Communications

We denote $\mathcal{C} = \{1, \ldots, C\}$ as the set of C channels available to the N UAVs. The bandwidth of each channel is W_{B}. The small-scale fading of the channel between the UAV i and BS over channel c is $h_{i,c}$. The interfering channel from UAV i' to UAV i ($i' \neq i$), when selecting the same channel c, expressed as $h_{i',c}$. The received SINR of the i-th UAV-to-BS link over the c-th channel are expressed as

$$\mathtt{SINR_{i,c}} = \frac{p_i |h|^2_{i,c} d^{-2}_{i,B}}{I_{i,c} + W_{\mathrm{B}} \sigma^2}, \tag{11}$$

where p_i is the tranmission power of UAV i, $d_{i,B}$ is the distance between the UAV i and BS, σ is the variance of noise. And

$$I_{i,c} = \sum_{i' \in \mathcal{N}, i' \neq i} \alpha_{i',c} p_{i'} |h|^2_{i',c} d^{-2}_{i',B}, \tag{12}$$

denotes the interference power, where $p_{i'}$ is the transmission power of UAV i', $d_{i',B}$ is the distance between the UAV i' and BS. $\alpha_{i,c}$ is the binary variable indicating channel association. Specifically, $\alpha_{i,c} = 1$ if UAV i is served over the c channel in time step t, otherwise $\alpha_{i,c} = 0$. Then, the transmission rate of UAV i in phase 2 is expressed as follows

$$C_{i,c} = a_{i,c} W_{\mathrm{B}} \log_2(1 + \mathtt{SINR_{i,c}}), \tag{13}$$

The number of available channels is limited, and more than one UAV can select the same channel. This results in the low transmission capacity due to the low SINR. In other words, each UAV i may not fully transmit Q_i. Note that the users require low latency and high quality of service. Meanwhile, the frequency of image updates can be slower based on the virtual system's requirements. Therefore, transmitting user data Q_i^{relay} is prioritized. Once Q_i^{relay} has been fully transmitted, UAV i proceed to transmit $Q_i^{\mathrm{graph}}(\gamma_i)$. Appropriate γ_i is selected to eliminate less important triplets, thereby decreasing the data size to fit the limited channel. We can formulate our problem in time step t as

$$\begin{aligned} \underset{a_{i,c}, p_i, \gamma_i}{\text{maximize}} \quad & C_{i,c} \sum_{k_i^t \in \mathcal{I}_i^t} \rho_k \\ \text{subject to} \quad & a_{i,c} \in [0,1], \quad \forall i \in \mathcal{N}, \forall c \in \mathcal{C}, \\ & 0 \leqslant p_i \leqslant p_{\max}, \quad \forall i \in \mathcal{N}, \\ & \gamma_i \in [0,1], \quad \forall i \in \mathcal{N}, \\ & Q_i \leqslant C_{i,c} \quad \forall i \in \mathcal{N}, \forall c \in \mathcal{C}. \end{aligned} \tag{14}$$

Traditional optimization algorithms can be used at the BS. However, the traditional approaches focus on designing spectrum access protocols for specific network scenarios, which require prior knowledge of the network dynamics to formulate the wireless networks model and cannot effectively be applied to handle complex real-world problems. Instead, we utilize deep reinforcement learning (DRL) which embraces the multi-dimensional perception ability of deep neural network (DNN) and the autonomous decision-making ability of reinforcement learning that can intelligently adapt to unknown network dynamics. In case of popular single-agent RL, the BS collects channel states, data sizes and locations of the UAVs to make the optimal decisions in centralized manner. Given the high density of the UAVs, the state space and action space may be very large. This leads to excessive delay caused by the information feedback and a heavy computational burden at the BS. Furthermore, the channel state information, the contexts captured by the UAVs and the user-UAV association vary over time steps. As such, the input parameters as well as the total data is highly dynamic. Updating these information to the BS at the beginning of phase 2 costs high overhead communication. In this work, we handle this problem by leveraging multi-agent RL (MARL) to optimize the network performance in distributed manner.

III. Multi-agent RL based resource management and priority level threshold selection scheme

In this section, we present decentralized MARL algorithms to solve the spectrum allocation, power control and priority level threshold selection problem of the UAVs. Conventionally, resource management problem is competitive. However, MARL allows coordination and collaboration among multiple UAVs by using the same reward shared across all UAVs. Multiple UAVs can explore different strategies to optimize the system's overall performance simultaneously, leading to a richer understanding of the environment and potentially faster convergence to optimal policies. Also, MARL simulates real-world scenario involves multiple interacting UAVs, allowing for better simulation and optimization.

To use MARL algorithm, we present the problem under the form of Markov decision process (MDP). At the beginning of phase 2 of each time step t, each UAV i would observe the local environment state s_i^t. Based on the that, each UAV would take an action a_i^t. Time is then incremented to $t+1$ and all the local environment is transitioned to the next state s_i^{t+1}. At this time, each UAV receives a shared numerical reward r^{t+1}, which is the average reward of all UAVs for their action in the previous time step, i.e. $r^{t+1} = \frac{1}{N}\sum_{i\in\mathcal{N}} r_i^{t+1}$.

The MARL based algorithm includes two separated phases, i.e., the centralized learning and the distributed implementation phases. In the training phase, the average reward of all UAVs is readily accessible to each UAV. The UAVs select their actions to improve the shared reward through updating its double deep Q-network (DDQN). In the implementation phase, each UAV observes the local environment and then takes an action according to its trained DDQN on the same time scale with the communication channel state. Key elements of the proposed MARL are demonstrated in detail below.

A. State space

At the beginning of phase 2, each UAV $i \in \mathcal{N}$ observes the local environment state s_i^t. We consider the dynamic and practical wireless environment in which the UAV-BS channels small-scale fading, i.e. $h_{i,c}$ for all $i \in \mathcal{N}, c \in \mathcal{C}$, vary over time step. These channel coefficients directly impact the data rate of the UAVs, so we include them in the state space. These can be estimated at the BS in each time step t and then broadcast to all UAVs in its coverage, trading-off by a small signaling overhead. We also introduce the interference power collected from the previous time step $I_{i,c}^{t-n}$ for all $c \in \mathcal{C}$ and $n \in [1,5]$, in the state space , with $I_{i,c}$ expressed in (12). This carries the environment changing pattern, which offers the UAV richer understanding of the environment. In addition, the state space consists size of user data Q_i^{relay} , size of all triplets Q_i^{graph} and a set of all triplets along with the priority levels associated with each triplets, denoted as $\mathcal{U}_i^{\text{triplet}}$, that collected by UAV i in phase 1. Specifically, $\mathcal{U}_i^{\text{triplet}}$ is defined as followed

$$\mathcal{U}_i^{\text{triplet}} = \{||s||, \phi_s\}_{s\in\mathcal{S}_k}, \tag{15}$$

This allows the UAV i to learn the size of the original scene graph and the importance of triplets included in the scene graph. Based on this, the UAV properly decides the priority level threshold γ_i. Also, including the value of interference caused from other networks due to the unlicensed channels Γ_i and the distance between UAV i and all users $u \in \mathcal{U}$ i.e. $d_{u,i}$ in the state space helps UAV i make better decision on actions. These can be collected by the UAV in phase 1 without causing further signaling overhead. As a result, the state space for UAV i is defined as follows:

$$\begin{aligned} s_i^t = \Big\{ & Q_i^{\text{relay}}, Q_i^{\text{graph}}, \mathcal{U}_i^{\text{triplet}}, \{h_{i,c}\}_{c\in\mathcal{C}, i\in\mathcal{N}}, \\ & \{I_{i,c}^{t-n}\}_{c\in\mathcal{C}, n\in[1,5]}, \Gamma_i, \{d_{u,i}\}_{u\in\mathcal{U}} \Big\}. \end{aligned} \tag{16}$$

Note that practically values of the inputs should be normalize to the range [0,1] before be passed into neural network. Normalization helps eliminate significant scale differences and handle the error values, e.g. false input values caused by unwanted reasons, among elements in the state space. This makes the data more interpretable and comparable, thus boosting the computation speed and improving convergence stability. Specifically, we normalize each component by dividing them by their maximum values, which can be obtained from a few trial episodes.

B. Action Space

At the beginning of phase 2, UAV i selects a channel c_i, tranmission power p_i and priority level threshold γ_i. Particularly, the spectrum naturally breaks into C disjoint channel, i.e. $c_i \in \{1, \ldots, C\}$. We set γ_i as a discrete variable, i.e. $\gamma_i \in \{0, \ldots, 1\}$. The transmission power p_i is continuous variables in reality. To limit the action space, we discretize p_i as $\mathcal{P} = \{p_1, p_2, \ldots, p_M\}$. Therein, p_M is the maximum value of transmission power. The action space of each UAV i is mathematically designed as

$$a_i^t = \left\{ c_i, p_i, \gamma_i \middle| c_i \in \mathcal{C}, p_i \in \mathcal{P}, \gamma_i \in \{0, \ldots, 1\} \right\} \tag{17}$$

Note that the selection of c_i is expressed by the binary variables $a_{i,c}$. Particularly, $a_{i,c_i} = 1$ if c_i is selected and $a_{i,c} = 0$ otherwise.

C. Reward Design

In this section, we design reward functions for each individual UAV. To evaluate the performance of UAV i, we define $Q_i^{\text{relay}-\text{remain}}$ as remaining amount of user data that cannot be delivered to the BS, and $Q_i^{\text{graph-remain}}$ as the size of the remaining triplets that cannot be delivered to the BS due to the low transmission capacity. Particularly,

$$Q_i^{\text{relay}-\text{remain}} = \begin{cases} Q_i^{\text{relay}} - \tau_2 C_{i,c}, & \text{if } \tau_2 C_{i,c} < Q_i^{\text{relay}}, \\ 0, & \text{if } \tau_2 C_{i,c} \geqslant Q_i^{\text{relay}}, \end{cases} \tag{18}$$

and

$$Q_i^{\text{graph}-\text{remain}} = \begin{cases} Q_i^{\text{graph}}(\gamma_i), \\ \qquad \text{if } \tau_2 C_{i,c} \leqslant Q_i^{\text{relay}}, \\ Q_i^{\text{graph}}(\gamma_i) + Q_i^{\text{relay}} - \tau_2 C_{i,c}, \\ \text{if } Q_i^{\text{graph}}(\gamma_i) + Q_i^{\text{relay}} > \tau_2 C_{i,c} > Q_i^{\text{relay}}, \\ 0, \\ \qquad \text{if } \tau_2 C_{i,c} \geqslant Q_i^{\text{graph}}(\gamma_i) + Q_i^{\text{relay}}. \end{cases} \tag{19}$$

The design above clearly show the priority of transmitting user data, then the scene graphs. We expect UAV i to transmit all the user data and triplets. For this, we design penalty function that is apllied when $Q_i^{\text{relay}-\text{remain}} \neq 0$ or $Q_i^{\text{graph}-\text{remain}} \neq 0$. Specifically

$$P_1 = -\exp\big(\omega_3 \frac{(\omega_1 Q_i^{\text{relay-remain}} + \omega_2 Q_i^{\text{graph-remain}})}{\max(\omega_1 Q_i^{\text{relay-remain}} + \omega_2 Q_i^{\text{graph-remain}})}\big). \tag{20}$$

In which, ω_1 and ω_2 are hyper-parameters that need to be tuned empirically based on practical network to ensure the priority of user data. Specifically, we want $\omega_1 Q_i^{\text{relay-remain}} > \omega_2 Q_i^{\text{graph-remain}}$ so that UAV i receives more penalties when cannot transmit all the user data.

We want each UAV receives exponentially increasing penalty as $\omega_1 Q_i^{\text{relay-remain}} > \omega_2 Q_i^{\text{graph-remain}}$ gets higher. To apply this idea, first, we normalize the remaining data to the range [0,1] by dividing it to its maximum value, which are achieved from running a few steps of random resource allocation. Using the normalized value as input of exponential function, we observe that when the input is in its low range, an increase of input only causes a slight increase of the exponential function. But when the input has a high value, the same increase of the input causes an explosive increase to the exponential function. To further control the increasing speed of the exponential function when the input increases, we introduce hyper-parameter ω_3. Specifically, the higher ω_3, the more significant the penalty upsurge is for the same increase of input. This design encourages all UAV to keep their remain data below a certain level. In other words, the system is encouraged to maintain the services quality in a majority of areas.

We expect UAV i maximize $\sum_{k_i^t \in \mathcal{I}_i^t} \rho_k$ by selecting higher value of γ_i at the same time as maximizing $C_{i,c}$, as long as all data are successfully transmitted to the BS. For this, if $Q_i^{\text{relay}-\text{remain}} = Q_i^{\text{graph}-\text{remain}} = 0$, we set the reward as

$$P_2 = \sum_{c=1}^{C} C_{i,c} + \omega_4 \sum_{k_i^t \in \mathcal{I}_i^t} (1 - \exp(-\rho_k)). \tag{21}$$

Note that $\rho_k \in [0,1]$ in natural, so no needs to normalize. When ρ_k is low, a small increase of ρ_k leads to significant increase of P_2. Because of this, all UAVs are likely to keep its ρ_k above a certain level to gain the easy reward. In other words, this design encourages the system to maintain certain level of services in all areas. However, as ρ_k grows higher, the increase of P_2 slows down and diminishes because $(1 - \exp(-\rho_k))$ approaches its maximum value of 1. This discourages each of the UAVs from raising ρ_k overly high. This is indeed unnecessary to accomplish overly detailed images of a few area by trading off poor-quality images of all other areas. Moreover, overly high ρ_k means a large number of triplets need transmitting, which causes large interference. ω_4 is empirically tuned hyper-parameters to demonstrate the importance of optimizing full scene graph factor over maximizing transmission rate in equation (21). We expect UAV i focuses on improving its spectrum sharing scheme because maximizing the full scene graph factor only makes sense when the transmission capacity is high enough. Otherwise, UAV i cannot transmit all triplets to the BS, and thereby trigger the the penalty P_1.

There is a downside of the use of exponential function in equation (21) that the UAV may lack of motivation to continue to maximize ρ_k when the exponential function approaches near its maximum value of 1, because then the reward nearly stays constant after ρ_k reachs to a certain value. To handle this problem, and also further boost learning speed and stability of UAV i, we offer it clear target. Specifically, we set reward to a constant number, β, when it can deliver all the data, and the value of P_2 achieves certain threshold. To this end, the reward design at each time step t is

$$r_i^{t+1} = \begin{cases} \beta, & \text{if } Q_i^{\text{relay}-\text{remain}} = Q_i^{\text{graph}-\text{remain}} = 0 \\ & \qquad \& \ P_2 \geqslant r_0, \\ P_2, & \text{if } Q_i^{\text{relay}-\text{remain}} = Q_i^{\text{graph}-\text{remain}} = 0, \\ \omega_5 P_1, & \text{otherwise.} \end{cases} \tag{22}$$

with β and r_0 that need empirical tuning. The value of β is set to be greater than the greatest value P_2 can achieve, which is collected from a few random training steps. Our tuning experience shows that it should not be too big and ideally twice of the greatest value. r_0 receives a value that is close to the maximum value of P_2. However, it should not be too hard to achieve. Otherwise, the UAV can rarely notices the superior reward, and thereby rarely learn anything from this design.

Finally, because later in this work we may modify ω_3, P_1 would be changed accordingly. To ensure the stability of the reward, we use hyper-parameter ω_5 to ensure balance between $|P_2|$ and $|\omega_5 P_1|$. This is important because if $\omega_5 P_1$ overwhelms P_2, the system would lack of motivation to learn to further enhance service quality after escaping from the penalty.

D. Learning Algorithm

Our MARL based approach is divided into 2 phases, i.e., the centralized training phase and the distributed implementation phase. The key elements are described below.

Algorithm 1 Channel, transmission power and priority level threshold selection with MARL

```
1: Stimulate environment, generating UAVs and users;
2: Initialize seperate DDQN and replay memory for each UAV $i$ for all $i \in \mathcal{N}$;
3: for each episode $k$ do
4:     if $k \equiv 0 \pmod 4$ then
5:         All UAVs update parameters of target network $\theta_i = \theta_i'$;
6:     end if
7:     for each time step $t$ do
8:         All UAVs $i \in \mathcal{N}$ estimate the user-UAV channels small-scale fading $h_{u,i}$ for all $u \in \mathcal{U}$, then broadcast to all users;
9:         for each user $u \in \mathcal{U}$ do
10:            Estimate $d_{u,i}$ and $\Gamma_i$ for all $i \in \mathcal{N}$;
11:            Calculate the $\texttt{SINR}_{u,i}$ for all $i \in \mathcal{N}$;
12:            Select the UAV with the largest SINR as relay;
13:        end for
14:        for each UAV $i \in \mathcal{N}$ do
15:            Receives and computes total size of all user data $Q_i^{\text{relay}}$;
16:            Captures and processes the images with RelTR, then giving out the total size of all triplets $Q_i^{\text{graph}}(0)$;
17:        end for
18:        The BS estimates the UAV-BS channels small-scale fading $h_{i,c}$ for all $i \in \mathcal{N}$ and $c \in \mathcal{C}$, then broadcasts to all UAVs;
19:        for each UAV $i \in \mathcal{N}$ do
20:            Estimate interference power $I_{i,c}^t$ for all $c \in \mathcal{C}$;
21:            Observes the state $s_i^t$ and select action $a_i^t$ following $\epsilon$-greedy policy;
22:            Executes actions and takes reward $r_i^{t+1}$;
23:            Sends reward $r_i^{t+1}$ to the BS
24:        end for
25:        The BS calculates the shared reward $r^{t+1}$ and sends to all UAVs;
26:        Update positions of users and user-UAV channels small-scale fading $h_{u,i}$ for all $u \in \mathcal{U}$ and $i \in \mathcal{N}$;
27:        Update UAV-BS channels small scale fading $h_{i,c}$ for all $i \in \mathcal{N}$ and $c \in \mathcal{C}$;
28:        for each UAV $i \in \mathcal{N}$ do
29:            Observes state $s_i^{t+1}$;
30:            Stores $(s_i^t, a_i^t, r_i^{t+1}, s_i^{t+1})$ in replay memory;
31:        end for
32:    end for
33:    for each UAV $i \in \mathcal{N}$ do
34:        Randomly samples mini-batche $D_i$ from the replay memory;
35:        Optimizes the parameters $\theta$ of the online network to minimize $L(\theta)$;
36:    end for
37: end for
38:
```

1) Centralized training: We leverage double deep Q-learning (DDQN) algorithm [6]. DDQN based on the concept of action-value function, denoted as $Q_\pi(s,a)$. It is defined as the expected return starting from the state s, taking the action a, and thereafter following the policy π. Particularly

$$Q_\pi(s,a) = \mathrm{E}_\pi[G^t | S^t = s, A^t = a], \tag{23}$$

where G^t is the cumulative discounted rewards with a discount rate γ and follow

$$G_t = \sum_{j=0}^{\infty} \gamma^k r^{t+j+1}, 0 \leqslant \gamma \leqslant 1. \tag{24}$$

DDQN uses two separate deep neural networks, namely online network and target network. Considering the UAV i, the online network, which is parameterized by θ_i, is used to evaluate the parameterized action-value function $Q(s,a;\theta_i)$. The target network, which is parameterized by θ_i', is used to estimate the target value Y that we want the value of action-value function $Q(s,a;\theta_i)$ to approach. Particularly, the target value, denoted as Y_i, follow

$$Y_i = r^{t+1} + \gamma Q(s^{t+1}, \mathrm{argmax}_{a^{t+1}} Q(s^{t+1}, a^{t+1}; \theta_i); \theta_i'). \tag{25}$$

To optimize $Q_\pi(s,a)$, DDQN leverages the experience replay mechanism, which stores the transitions each time step throughout the simulating process. Multiple time steps form a episode. At each episode k, UAV i randomly samples a mini-batch D_i from its replay memory and trains the two neural networks on these data. Specifically, θ_i is updated to minimize the loss function under the form of sum-square error:

$$L(\theta_i) = \sum_{\mathcal{D}} [Y - Q(s,a;\theta_i)]^2. \tag{26}$$

θ_i' is duplicated from θ_i every few episodes and remain unchanged between two consecutive updates. This helps avoid overestimation problem, as presented in [7] and [8]. In the scenario of MARL, each UAV i has a separate DDQN set up. The detailed training procedure is presented in Algorithm 1.

2) Distributed implementation: In this phase, at each time step t, each UAV i observes the local environment and compiles state space s_i^t following (15), then selects action a_i^t selected by the trained DDQN. Afterward, the UAVs transmit the data to the BS using the selected channel, transmission power and priority level threshold.

Note that the computationally intensive training procedure described in Algorithm 1 can be done offline for multiple episodes, considering various channel conditions and changes in network topology. On the other hand, the low-cost implementation process is carried out online during network deployment. It is only necessary to update the trained DDQN for all UAVs when there are significant changes in the environment, such as once a week or even a month. The frequency of updates depends on the dynamics of the environment.

IV. Methods for stochastic optimization

Mathematically, training the online network means solving the non-convex optimization problem, expressed as

$$\arg\min_{\theta \in \mathrm{R}} L(\theta) \tag{27}$$

In solve this problem, we apply and compare the performance of the two most popular adaptive optimization algorithms, namely Adam [2] and RMSProp [3]. They are two rare algorithms that are stand-out and proven to work well across a wide range of deep learning architectures. Adam has been widely used in many applications owning to its stunning performance despite minimal tuning. RMSProp, although introduced earlier, remains a go-to optimizer in online and non-stationary setting thanks to its competitive practical performance and significant training stability. [9] even shows that RMSProp outperforms Adam, thanks to its better generalizing ability, in some specific domains of tasks. Our goal is to investigate which optimization algorithm solves our problem better. Also, it is noted that recent works such as [10] and [9] suggests in certain circumstances, adaptive methods like Adam and RMSProp may perform worse than the pure vanilla SGD, e.g. in tasks that heavily require generalization. Considering the success of SGD in many reputable study, such as [11]–[14], we also include the simulation version with pure vanilla SGD-approach in the later section. Below, we present the key points of RMSProp and Adam algorithm.

A. RMSProp

Algorithm 2 Pseudo-code of RMSProp. All operations on vectors are element-wise.

```
1: Initialize: $k := 0$; $v_0 := 0$
2: while $\theta$ not converge do
3:     $k := k + 1$
4:     $g_k := \nabla_\theta L(\theta_{k-1})$
5:     $v_k := \beta_2.v_{k-1} + (1 - \beta_2).g_k \odot g_k$
6:     $\theta_k := \theta_{k-1} - \alpha.g_k/(\sqrt{v_k} + \eta)$
7: end while
8: return $\theta$
```

Algorithm 2 presents RMSProp algorithm order of computation in each episode k, with α is learning rate, $\beta_2 \in [0, 1)$ is exponential decay rate for moment estimation. The main ideas of RMSProp are as follow. First, gradient of the loss function, i.e. g_k, is calculated (step 4). The gradient represents the direction and magnitude of the steepest ascent or descent of the loss function. By calculating the gradient of the loss function with respect to the model's parameters, we can determine the direction in which the parameters should be adjusted to minimize the loss. Next, step 5 calculates v_k, which is the exponential weighted moving average of the squared gradient. This helps to smooth out the gradient updates and adaptively adjust the learning rate for each parameter. Finally, θ is updated following step 6. Dividing the learning rate α by square root of v_k serves as a normalization factor that scales the learning rate based on the magnitude of the gradient. It allows for larger updates when the gradients are small and smaller updates when the gradients are large. This adaptivity helps to prevent the learning rate from becoming too large and overshooting the optimal solution or too small and slowing down the convergence. η is a small constant, e.g. 10^{-8}, that added to denominator to avoid division by zero when v_k is very close to 0.

B. Adam

Algorithm 3 Pseudo-code of Adam. All operations on vectors are element-wise. β_1^k and β_2^k denotes β_1 and β_2 to the power k.

```
1: Initialize: $k := 0$; $v_0 := 0$; $s_0 := 0$
2: while $\theta$ not converge do
3:     $k := k + 1$
4:     $g_k := \nabla_\theta L(\theta_{t-k})$
5:     $s_k := \beta_1.v_{k-1} + (1 - \beta_1).g_t$
6:     $v_k := \beta_2.v_{k-1} + (1 - \beta_2).g_t \odot g_t$
7:     $\hat{s_k} := s_k/(1 - \beta_1^t)$
8:     $\hat{v_k} := v_k/(1 - \beta_2^t)$
9:     $\theta_k := \theta_{k-1} - \alpha.\hat{s_k}/(\sqrt{\hat{v_k}} + \eta)$
10: end while
11: return $\theta$
```

Algorithm 3 presents Adam algorithm order of computation in each episode k, with α is learning rate, β_1, $\beta_2 \in [0, 1)$ are exponential decay rate for moment estimation. v_k works the same as in RMSProp algorithm. The main difference lies in the introduction of s_k (step 5), which is the exponential weighted moving average of the gradient. Specifically, the use of s_k in step 9 incorporates the effect of prior gradients into the current update. When encounter noisy gradient, the value of s_k would be averaged out to near zero. This helps to smooth out the update trajectory. Meanwhile, when encounter flat gradient, the value of s_k build up over time and move faster towards the regions of lower loss. In short, this results in quicker convergence and faster training.

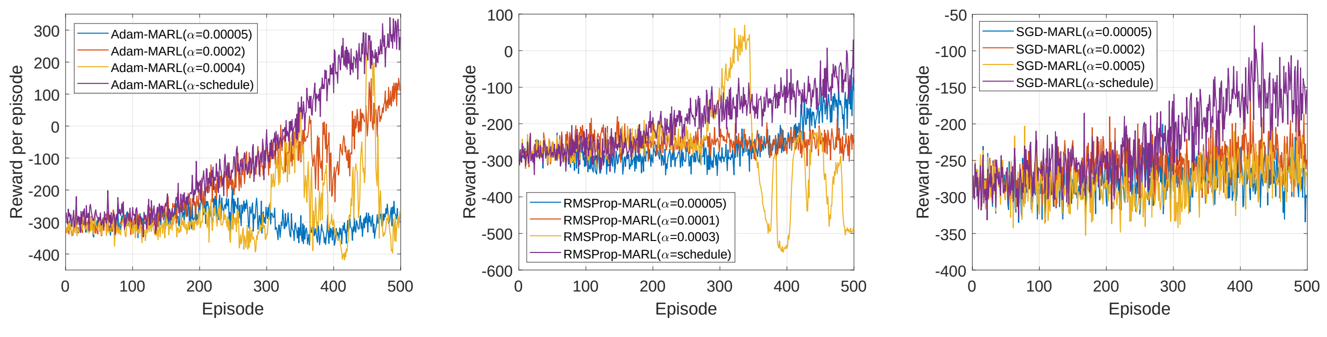
There is a problem is that because v_k and s_k are initially vector of 0's, they are biased towards zero at the beginning of the training phase, leading to large update magnitudes. By dividing the estimates by the bias correction factor (step 7 and 8), it ensures that the initial estimates are closer to their true values, thereby preventing the updates from being excessively large and destabilizing the optimization process.

V. Simulation evaluation

In this section, we present simulation results to evaluate the proposed MARL method. We set up the simulator range following Cartesian coordinate system with the origin point at the center of the map, the x-axis representing width, the y-axis representing length, and the z-axis representing height. Specifically, $x \in [-250, 250]$, $y \in [-250, 250]$ and $z \in [0, 100]$. The BS is placed at the origin point. 4 UAVs

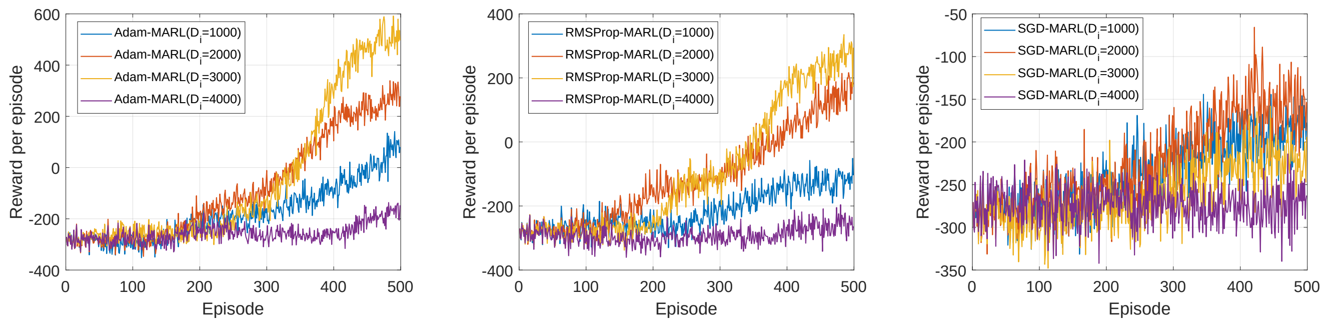
TABLE I: Simulation Parameters

Parameter	Value	Parameter	Value
Duration of time step (τ)	0.08 s	UAVs' sensing rate (λ_i)	4 images/s
Duration of the first phase (τ_1)	0.02 s	Priority level threshold selection (γ_i)	$\{0.1, 0.3, 0.5\}$
Duration of the second phase (τ_2)	0.06 s	Users' antenna height	0 m
Number of UAVs (N)	4		
Users' tranmission power levels (p_k)	15 dBm	Users' antenna gain	2 dBi
Number of ground users (U)	20 users	UAV antenna height	0 m
Bandwidth of UAV (W_i)	15 kHz	BS antenna height	25 m
$\{h_{i,c}, h_{u,i}\}$	$\lvert\mathcal{N}(0,1)\rvert$	UAV antenna gain	2 dBi
Interference power (Γ_i)	$\mathcal{U}\{-17, -18, -19\}$ dBm	BS antenna gain	8 dBi
Bandwidth of BS (W_{B})	10^6 Hz	$\{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5\}$	$\{0.005, 0.0004, 0.7, 82, 15\}$
Noise variance (σ^2)	-176 dBm	$\{\beta, r_0\}$	$\{96, 65\}$



(a) (b) (c)

Fig. 2: Impact of learning rate on reward achieved by schemes based on (a) Adam-MARL, (b) RMSProp-MARL and (c) SGD-MARL over training phase. Mini-batchsize $D_i = 2000$ is applied on all schemes.



(a) (b) (c)

Fig. 3: Impact of mini-batch size on reward achieved by schemes based on (a) Adam-MARL, (b) RMSProp-MARL and (c) SGD-MARL over training phase. α-schedule is applied on all schemes.

are evenly distributed in 4 directions of the BS, respectively coordinated at (125, 0, 100), (-125, 0, 100), (0, 125, 100) and (0, -125, 100). The users are generated randomly within the map for the purpose of providing diverse dataset. We assume that all transceivers use a single antenna and both user-to-UAV and UAV-to-BS links are dominated by LoS links. Further simulation parameters are listed in Table I. Note that all considering parameters follow Table I by default, but are modified to fit each figure whenever applicable.

We build each UAV a separate DDQN that includes 3 fully connected hidden layers, containing respectively 256, 128 and 64 neurons, respectively. Each uses the leaky rectified linear unit (Leaky ReLU), $f(x) = max(0.01x; x)$, as the activation function. The training phase of each UAV's DDQN-network lasts 500 episodes, each episode contains 200 step. The exploration rate ϵ linearly annealed from 1 to 0.02 in the first 400 episodes and remains 0.02 afterwards.

As mentioned earlier, we investigate the performance of MARL schemes using three optimization algorithms: Adam, RMSProp, and SGD, referred to as Adam-MARL, RMSProp-MARL, and SGD-MARL, respectively. We set fixed values for the exponential decay rates of moment estimation, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\eta = 10^{-8}$. Meanwhile, the learning rate α and minibatch size D_i are carefully tuned for all algorithms.

First, we try different learning rates with a fixed minibatch size of 2000. Fig. 2 illustrates the performance of Adam-MARL, RMSProp-MARL, and SGD-MARL with three specific α value levels: low, proper, and high. In Fig. 2a, we observe that with a low $\alpha = 0.00005$, the reward received in each episode hardly changes throughout the training phase. With proper $\alpha = 0.0002$, Adam-MARL achieves a good performance. However, with high $\alpha = 0.0004$, the learning process diverges rapidly after about 210 episodes from the beginning of training phase.

We observe that even with proper constant α, Adam-MARL and RMSProp-MARL schemes do not converge optimally, while SGD-MARL scheme hardly makes progress. To address this issue, we apply a learning rate schedule [15] in combination with Adam, RMSProp, and SGD. The learning rate schedule allows for faster progress in the initial stages of training with a higher learning rate, facilitating more efficient exploration of the solution space. As the training phase lasts, the learning rate decreases, allowing for finer adjustments to the model's parameters and preventing overshooting. This leads to stable training and improved convergence for all optimization algorithms. We tried some popular learning rate schedule setting such as exponential decay [16] and polynomial decay [17], but they do not help improve the performance. However, dividing the training time into intervals and applying different learning rates for each interval yield much better performance. Specifically, we set $\alpha = 0.01$ for the first 100 training episodes, $\alpha = 0.0003$ for the next 200 episodes, and $\alpha = 0.00005$ for the remaining episodes. As shown in Fig. 2, this improves the performance and convergence in all schemes.

Secondly, we explore the impact of different minibatch sizes D_i on the schemes with the learning rate schedule setting mentioned above. The results are presented in Fig. 3. It is observed that Adam-MARL and RMSProp-MARL perform best with $D_i = 3000$, while SGD-MARL achieves optimal performance with $D_i = 2000$.

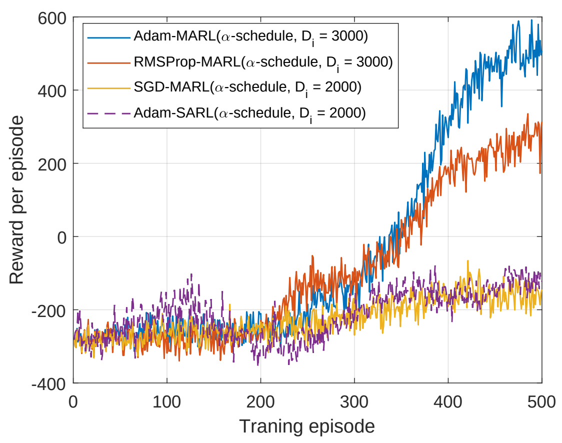


Fig. 4: Comparison of the reward achieved by the schemes with their best learning rate α and mini-batchsize D_i over training phase.

We investigate the reward gained by all considering schemes with their best learning rate α and mini-batchsize D_i in Fig. 4. Also, to benchmark the performance of MARL, we also consider the single-agent RL approach that is optimized with Adam. This approach is inspired by [18] and termed Adam-SARL, in which a single DDQN is shared across all UAVs. In each time step, only one UAV selects its action based on local information and a trained DDQN, while the other UAVs keep their action unchanged. Our tuning experience shows that Adam-SARL works efficiently with α-schedule and $D_i = 2000$. Following, we simulate the performance of the schemes with the α-schedule setting and the best minibatch size found above.

From Fig. 4, we observe that all schemes learn to improve their performance throughout the training phase. Adam-MARL achieves the highest reward and converges smoothly in the final episodes of the training phase. RMSProp-MARL also converges well, but end up at lower reward. Adam-SARL and SGD-MARL converge with minimal progress after the training phase. This indicates the superior performance of Adam and also highlights that MARL exploits dynamic-environment dramatically better than SARL.

Following, we consider the performance of the schemes in two different mode:

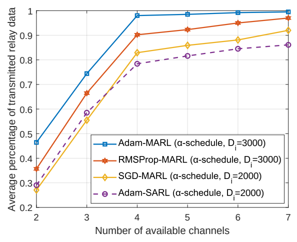
1) System-centric mode: In this approach, we allow each UAV to proactively sacrifice their performance to avoid causing interference, which offer the system flexibility to better exploit the environment. To do this, we include the option of shutting down in the transmission power selection set. Hyper-parameters ω_3 is adjusted to lower the rate of exponential growth of P_1, so that the UAVs do not receive a heavy penalty for choosing this option. ω_5 is adjusted to balance the elements in equation (22). This mode helps discover the true limitations of the spectrum in the specific environment by determining the group of UAVs in which areas can collaboratively operate to yield the best overall system performance. It is possibly worth considering a system implementation that only includes well-functioning UAVs and seeks alternative solutions for areas where UAVs are proactively restrained from operating.

In this mode, we set up the power control options for each UAV with four levels, i.e. $[23; 10; 5; -100]$ dBm. The choice of -100 dBm effectively represents zero transmission power. $\{\omega_3, \omega_5\}$ are set to $\{0.14, 15\}$. With these value of ω_3 and ω_5, the penalty would not increase much as the value of remaining data increase.

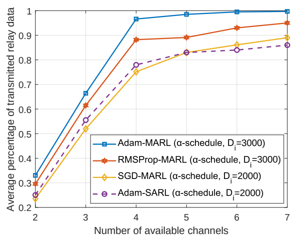
2) User-centric mode: In this approach, we want each UAV to maintain its service quality above a certain threshold. To do this, we fix the transmission power selections in which each element is greater than a certain threshold. Hyper-parameters ω_3 and ω_5 are modified so that the penalty increases explosively as the remaining data value increases. This forces the UAVs to learn how to deal with stronger interference caused by all the UAVs joint. This leads to performance degradation compared to the system-centric mode, which will be validated following.

In this mode, we set up the power control options for each

UAV with for levels, i.e. $[23; 18; 14; 8]$ dBm. $\{\omega_3, \omega_5\}$ are set to $\{0.2, 10\}$. ω_3 in this mode is higher than in system-centric mode, leads to more explosive increases of the penalty. ω_5 is lower to balance out $\omega_5 P_1$ and P_2.



(a)

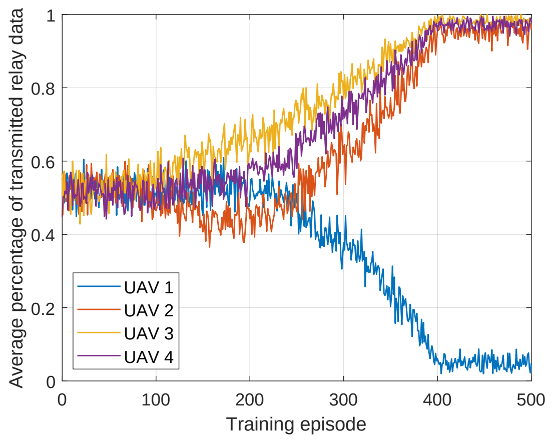


(b)

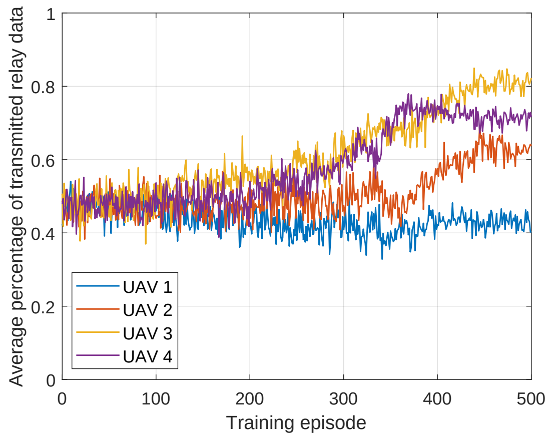
Fig. 5: Average percentage of transmitted user data of 4 UAVs sharing varying number of channels following (a)system-centric mode and (b)user-centric mode.

We investigate how well the considering schemes perform in relaying task in Fig. 5. From the figure, the performance of all schemes improves as the number of channels growing. This is because an increase in channels leads to more flexible channel selection and lower likelihood that multiple UAVs share the same channel. Consequently, each UAV learns to utilize higher transmission power while causing minimal interference to other UAVs that sharing the same channel and enhancing relay performance. In system-centric mode, we observe from Fig. 5a that Adam-MARL consistently outperforms the other schemes across all channel configurations. Specifically, the performance gap between Adam-MARL and the others is most apparent in the most challenging case of 2 available channels, which demonstrates its superior optimizing ability. As more channels are available, the optimization problem need solving is less difficult, and thereby the gap are narrower. Notably, for the current setting, Adam-MARL deliver nearly 100% of the relay data with no less than 4 available channels in the system-centric mode.

In user-centric mode, Fig. 5b shows that Adam-MARL also outperforms all other optimization algorithms. But we observe significantly lower performance for all schemes in cases of narrow spectrum with 2 and 3 channels available, compared to the system-centric mode. This is because in user-centric mode, each of the UAVs is forced to maintain their service above a certain threshold, which causes interference. The fewer number of available channels, the more significant the interference the system need to deal with. As such, more limited improvement can be made. However, when more than 4 channels are available, the system learns so that each UAV transmits over a separate channel, hence avoid causing interference while sharing channel. Without noise, the average performance of the system in different schemes gradually becomes similar to in system-centric mode, which is shown in Fig. 5b. For the current setting, Adam-MARL delivers above 96% of the relay data with no less than 4 available channels and no noticeable relay loss is spotted with no less than 5



(a)



(b)

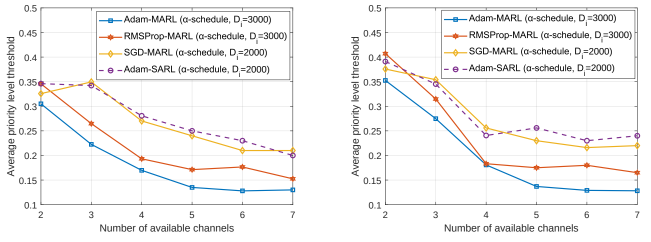
Fig. 6: Average percentage of transmitted relay data of each UAV achieve over the training process following (a)system-centric mode and (b)user-centric mode. 3 channels shared by 4 UAVs with Adam-MARL(α-schedule, $D_i = 3000$) scheme.

channels available in the user-centric mode.

There is another important observation from Fig. 5 that Adam-SARL performs a little bit better than SGD-MARL in case of 2 and 3 available channels. However, as more than 4 channels available, SGD-MARL achieves better and better results than Adam-SARL. This is because MARL allows for coordination and collaboration among multiple UAVs. Multiple UAV divide the large-scale areas and learn from the interactions with all the others, leading to a richer understanding of the environment. Meanwhile, SARL based scheme cannot be such sufficient to capture the dynamics of environment as it grows larger.

From perspective of relaying task, we show deeper insights about the learning process and the performance differences between the system-centric mode and user-centric mode in Fig. 6. From Fig. 6a, UAV 1 figures out a clever strategy to proactively diminish its performance to avoid interfering with other

UAVs. This is probably because it observes challenging local conditions such as unfavorable user-UAV channel states and heavy relay data and scene graphs transmission requirements. With better local conditions, other UAVs can flexibly share the available channels with less interference. At the end of the training phase, UAVs 2, 3, and 4 manage to transmit nearly all their relay data to the BS, but this comes at the cost of UAV 1 becoming nearly nonfunctional. Different from this, Fig. 6b demonstrates that in the user-centric mode, all UAVs learn while maintaining their performance above a certain threshold. Despite an clear overall performance improvement observed in the training process, the increased interference caused by all operating UAVs leads to a lower overall system performance compared to the system-centric mode.



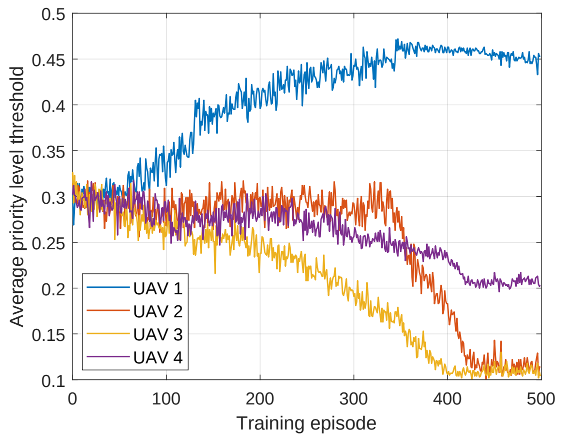
(a) (b)

Fig. 7: Average priority level threshold chosen by 4 UAVs sharing varying number of channels following (a)system-centric mode and (b)user-centric mode.

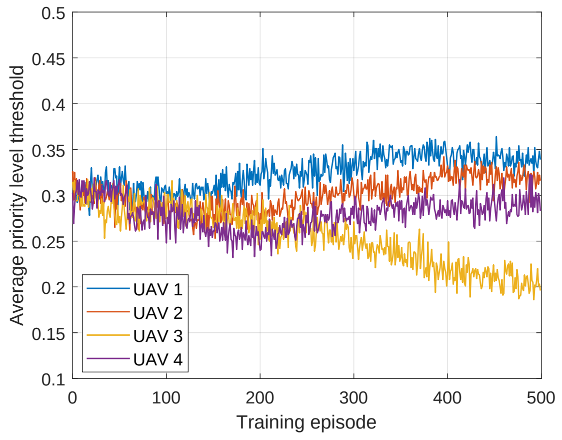
The second objective of the system is to select an appropriate priority level threshold. We investigate this in Fig. 7. From the figure, the average priority level threshold decrease as the number of channels growing. This is because an increase in available channels leads to greater transmission capacity upper boundary. As the system learn to improve its transmission capacity, it should select lower priority level threshold to keep more triplets compatibly.

Fig. 7a and Fig. 7b again indicates that in the system-centric mode, UAVs can better exploit the environment, resulting in lower average priority level threshold compared to the user-centric mode. We also observe that Adam-MARL based scheme still performs best. However, one unusual observation is that the average priority level threshold of the schemes based on SGD-MARL and Adam-SARL are higher than RMSProp-MARL in the difficult case of two available channels. Examining the results reveals that the they collapse into selecting low priority level threshold and cannot transmit all the triplets to the BS due to limited transmission capacity. This only happens in the most challenging-to-optimize case with only 2 channels available.

From images-sensing perspective, we again investigate the learning process and the performance differences between the system-centric mode and user-centric mode in Fig. 8. From Fig. 8a, we observe that UAV 1 learns to diminish its performance. It selects high priority level threshold to filter out almost all of the triplets that need transmitting to minimize



(a)



(b)

Fig. 8: Average priority level threshold chosen by each UAV over the training process following (a)system-centric mode and (b)user-centric mode. 3 channels shared by 4 UAVs with Adam-MARL(α-schedule, $D_i = 3000$) scheme.

the size of its scene graphs. While only transmit the most important triplets, it avoid causing interference to other UAVs. The other UAVs learn to select low priority threshold to keep a majority of their triplets and transmit them to the BS. In other words, the BS receives high quality scene graphs from the areas covered by these UAVs. On the other hand, Fig. 8b demonstrates that in the user-centric mode, all UAVs learn on the condition that their scene graphs quality are above a certain threshold. Due to interference caused by maintaining functionality of all UAVs, the overall system performance is improved throughout the training phase, but lower compared to the system-centric mode.

For better visualization, we demonstrate the output scene graph of UAV 3 and UAV 1 at the start and at the end of training phase in Fig. 9. It shows that after training phase, the scene graph collected by UAV 3 is densely filled with triplets, which representing high quality image. Meanwhile, Fig. 9b shows that only 3 triplets are kept by UAV 1 after the training

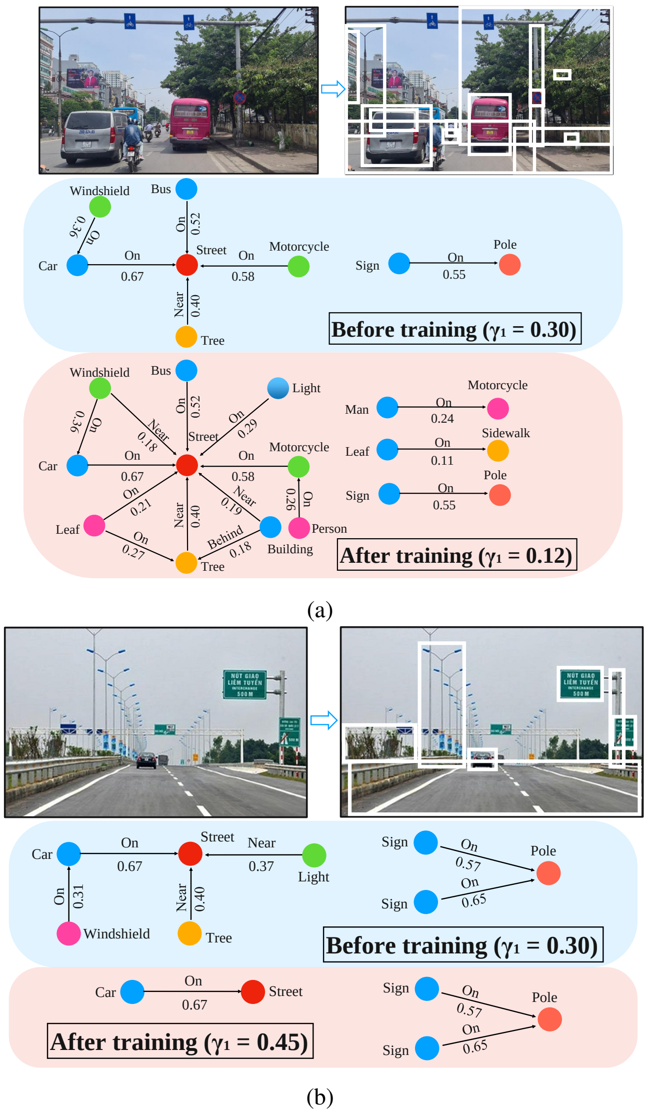


Fig. 9: Comparison of the output scene graphs before and after training phase of (a) UAV 3 and (b) UAV 1 in simulation run shown in Fig. 8a

phase, which only tells basic details of the image.

VI. Conclusions

References

[1] A. Nowé, P. Vrancx, and Y.-M. D. Hauwere, "Game theory and multi-agent reinforcement learning," in *Reinforcement Learning*, M. Wiering and M. van Otterlo, Eds. Berlin, Germany: Springer, 2012, pp. 441–470.

[2] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[3] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop, coursera: Neural networks for machine learning," *University of Toronto, Technical Report*, vol. 6, 2012.

[4] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[5] Y. Cong, M. Y. Yang, and B. Rosenhahn, "Reltr: Relation transformer for scene graph generation," *arXiv preprint arXiv:2201.11460*, 2022.

[6] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *AAAI*, vol. 30, no. 1, March 2016, pp. 2094–2100.

[7] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Proceedings of 4th Connectionist Models Summer School*, D. T. J. E. M. Mozer, P. Smolensky and A. Weigend, Eds. Erlbaum Associates, June 1993.

[8] H. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., vol. 23. Curran Associates, Inc., 2010.

[9] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4151–4161.

[10] P. Zhou, J. Feng, C. Ma, C. Xiong, S. C. H. Hoi *et al.*, "Towards theoretically understanding why sgd generalizes better than adam in deep learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 285–21 296, 2020.

[11] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 8599–8603.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[13] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[15] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," pp. 437–478, 2012.

[16] Z. Li and S. Arora, "An exponential learning rate schedule for deep learning," *arXiv preprint arXiv:1910.07454*, 2019.

[17] R. Ge, S. M. Kakade, R. Kidambi, and P. Netrapalli, "The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares," *Advances in neural information processing systems*, vol. 32, 2019.

[18] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for v2v communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 2019.