Communication-Assisted Multi-Agent Reinforcement Learning Improves Task-Offloading in UAV-Aided Edge-Computing Networks

Siyang Tan, Binqiang Chen, Dong Liu, Jianglong Zhang and Lajos Hanzo

***Abstract*—Equipping unmanned aerial vehicles (UAVs) with computing servers allows the ground-users to offload complex tasks to the UAVs, but the trajectory optimization of UAVs is critical for fully exploiting their maneuverability. Existing studies either employ a centralized controller having prohibitive communication overhead, or fail to glean the benefits of interaction and coordination among agents. To circumvent this impediment, we propose to intelligently exchange critical information among agents for assisting their decision-making. We first formulate a problem for maximizing the number of offloaded tasks and the offloading fairness by optimizing the trajectory of UAVs. We then conceive a multi-agent deep reinforcement learning (DRL) framework by harnessing communication among agents, and design a communication-assisted decentralized trajectory control algorithm based on value-decomposition networks (VDN) for fully exploiting the benefits of messages exchange among agents. Simulation results demonstrate the superiority of the proposed algorithm over the state-of-the-art DRL-based algorithms.**

***Index Terms*—Multi-agent reinforcement learning, UAV, trajectory planning**

I. Introduction

Mobile edge computing (MEC) is a key technique of improving the quality of experience (QoE) of mobile users by offloading the computation tasks from users [1, 2]. Typically, MEC servers are deployed at fixed locations near the wireless edge, limiting their capability of providing flexible services.

Given their maneuverability, high-end unmanned aerial vehicles (UAVs), having unexploited computing and storage hardware, might be harnessed as the flexible servers for mobile users [3–5]. Jeong *et al.* [6] optimized both the resource allocation and the UAV's trajectory for minimizing the overall energy consumption by utilizing a successive convex approximation-based algorithm. To tackle the resource allocation problem, Lyu *et al.* [7] propose a quantized dynamic programming algorithm for offloading delay-sensitive tasks in MEC. To reduce the complexity, Wu and Zhang [8] designed a UAV trajectory discretization method, and formulated a tractable problem for optimizing the consecutive UAV locations. However, due to the limited coverage range of UAVs and non-existence of a centralized control node, the UAV-aided MEC system environment is only partially observable for each UAV. Moreover, neither the model nor the dynamics of the environment are known *a priori*. Thus, it is difficult to formulate tractable problems for complex environments. Additionally, the computational complexity usually grows exponentially with the number of time slots considered in MEC settings, especially in multi-UAV scenarios.

To tackle the challenges of partial observations, unknown model and computational complexity, some authors harnessed reinforcement learning (RL) for UAV-aided MEC [9–13]. In [9], Zhang *et al.* proposed to minimize the energy consumption and computation latency by optimizing both the UAV trajectory control and task scheduling policy, by applying deep Q-network (DQN) methods for single-UAV scenarios. Wang *et al.* [10] optimized the user association, resource allocation and trajectory of UAVs for minimizing the energy consumption of all UEs. In order to maximize the energy efficiency while ensuring wide coverage by the UAVs, Liu *et al.* [11] proposed DRL-based methods for controlling the UAVs' trajectory. However, both [10] and [11] rely on a centralized controller for information processing and decisions for all UAVs, which would have an excessive overhead.

In order to obtain a decentralized policy, Hernandez-Leal *et al.* [12] and Yu *et al.* [2] employed a distributed learner for each MEC agent, which however may result in a non-stationary problem [12, 14]. To tackle this problem, Wang *et al.* in [13] resorted to a centralized training and decentralized execution (CTDE) framework to control a UAV's trajectory, aiming at maximizing the offloading fairness, while minimizing the overall energy consumption. Nonetheless, in both independent learner and CTDE methods, each agent chooses its action independently, unaware of the status and intentions of others, which hinders the cooperation among agents.

Against the above background and inspired by the observation that leveraging communications between the agents can provide critical information for decision making [14, 15], we conceive a novel communication assisted learning framework to deal with the aforementioned challenges in UAV trajectory control supporting MEC networks. Our major contributions can be summarized as follows:

- We formulate a new multi-agent RL (MARL) problem to optimize the trajectory of each UAV, aiming for maximizing both the number of offloaded tasks and the fairness.
- We propose a communication-assisted value decomposition network (CAVDN). By allowing agents to exchange the messages learned, each agent can take into account both its local observations and messages from others.
- Our simulation results show the superiority of CAVDN over state-of-the-art DRL-based UAV control algorithms.

II. System Model

In this section, we describe the UAV-aided MEC system. Let $\mathcal{K} = \{k|k = 1, 2, ..., K\}$ denote the IoT user equipment (UE) set. Due to the limited computing capability of IoT devices, their tasks have to be offloaded to UAVs for execution, where N rotary-wing UAVs with on-board MEC servers fly over the

S. Tan is with Chinese Aeronautical Establishment, Beijing 100029, China. B. Chen, D. Liu and J. Zhang are with Beihang University, Beijing 100191, China. L. Hanzo is with the University of Southampton, Southampton SO17 1BJ, U.K. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62001509 and 62301015, in part by the Youth Top Talent Support Program of Beihang University under Grant YWF-22-L-1269, and in part by the CAAC Key laboratory of General Aviation Operation under Grant CAMICKFJJ-2020-4. L. Hanzo would like to acknowledge the financial support of the Engineering and Physical Sciences Research Council projects EP/W016605/1, EP/X01228X/1 and EP/Y026721/1 as well as of the European Research Council's Advanced Fellow Grant QuantCom (Grant No. 789028) *(Corresponding author: Dong Liu)*

target area to provide computation services. Let $\mathcal{N} \triangleq \{n|n = 1, 2, ..., N\}$ represent the UAV set. We assume that all UAVs can be charged by the stations positioned on the roof tops before ruining out of battery as in [13] or laser-charged as in [16]. Thus, the energy consumption of UAVs is not considered.

The system is operated in discrete time steps (TSs), each with a duration of τ. In TS t, the coordinates of the nth UAV and the kth user are denoted by $\boldsymbol{d}_{n,t}^{uav} = [d_{n,t}^x, d_{n,t}^y, d_{n,t}^z]$ and $\boldsymbol{d}_{k,t}^{ue} = [d_{k,t}^x, d_{k,t}^y, 0]$, respectively.

A. UAV Motion and Offloading Decision

Let us denote the heading and normalized speed of the nth UAV at TS t by $\phi_{n,t} \in [0, 2\pi]$, and $\upsilon_{n,t} \in [0, 1]$, respectively. Consequently, the position of UAV n at TS $t+1$ is

$$\boldsymbol{d}_{n+1,t}^{uav} = \boldsymbol{d}_{n,t}^{uav} + (\boldsymbol{v}_{n,t} V_{\max} + \boldsymbol{w}_{n,t})\tau, \tag{1}$$

where $\boldsymbol{v}_{n,t} = [\upsilon_{n,t}\cos(\phi_{n,t}), \upsilon_{n,t}\sin(\phi_{n,t}), 0]$, $V_{\max}$ is the maximal speed of the UAVs, and $\boldsymbol{w}_{n,t}$ is a random variable reflecting the uncertainty of the environment, e.g., wind.

We consider the "full buffer" scenario, where each UE always has local tasks for offloading. If there is at least one UAV in the coverage of the UE, the UE will forward its local task to the nearest UAV. Otherwise, the UE will execute its task locally. Moreover, we assume that the computational capability of the UAV is powerful enough to complete each offloaded task in a single TS. The offloading decision of the kth UE for the nth UAV at TS t is denoted as $x_{k,n,t}$. Specifically, $x_{k,n,t} = 1$ represents that UE k offloads its task to UAV n at TS t, where $\sum_{n=1}^{N} x_{k,n,t} \leq 1$, and $x_{k,n,t} = 0$ otherwise.

B. Communication Channel and Task Offloading Process

Given the high probability of line-of-sight (LoS) connections between the UEs and UAVs, we only consider LOS channels. The channel gain between UE m and UAV n in TS t is $h_{k,n,t} = d_{k,n,t}^{-\alpha}\mu$, where α is the path loss exponent, and μ denotes the reference channel's power gain at one meter.

We assume that all UEs share the same bandwidth W to communicate with the UAVs, and the ground-to-air links are scheduled using TDMA. Then, the offloading data rate from the kth UE to the nth UAV at TS t can be written as

$$R_{k,n,t} = \frac{W}{\sum_{k=1}^{K}\sum_{n=1}^{N} x_{k,n,t}} \log_2\left(1 + \frac{P h_{k,n,t}}{\sigma^2}\right), \tag{2}$$

where P is the transmit power of the UE, and σ^2 is the additive white Gaussian noise power.

If UE k offloads its local task to UAV n at TS t, the duration of offloading the data is $\Delta_{k,n,t} = S(k)/R_{k,n,t}$, where $S(k)$ is the task size of UE k. If the offloading requires more than one time slot, i.e., $\Delta_{k,n,t} \geq \tau$, the UAV will refrain from maneuvering in the next TS for completing the offloading process. Let $\widetilde{x}_{k,n,t}$ represent the task offloading status. Specifically, if the task sent from UE k to UAV n is finished at TS t, we have $\widetilde{x}_{k,n,t} = 1$. Otherwise, $\widetilde{x}_{k,n,t} = 0$.

C. Performance Metric and Problem Formulation

Our primary objective is to maximize the average ratio of completed offloading tasks, where the ratio is expressed as

$$e_t = \sum_{k=1}^{K}\sum_{n=1}^{N} \frac{\widetilde{x}_{k,n,t}}{K}. \tag{3}$$

To balance the load among UAVs, we consider the fairness among each UAV's load as another objective to guide the optimization. We define $y_{n,t} = \sum_{k=1}^{K} x_{k,n,t}/K$ as the relative load of UAV n at TS t. Then, by applying Jain's fairness index, the UAV's fairness index f_t^{uav} can be expressed as

$$f_t^{uav} = \frac{(\sum_{n=1}^{N}\sum_{t'=1}^{t} y_{n,t'})^2}{K\sum_{n=1}^{N}(\sum_{t'=1}^{t} y_{n,t'})^2}, \tag{4}$$

which approaches 1 when the number of served tasks for all UAVs at each TS is similar.

Meanwhile, we also consider the fairness between UEs, which is defined similarly by

$$f_t^{ue} = \frac{(\sum_{k=1}^{K}\sum_{t'=1}^{t}\sum_{n=1}^{N} x_{k,n,t'})^2}{K\sum_{k=1}^{K}(\sum_{t'=1}^{t}\sum_{n=1}^{N} x_{k,n,t'})^2}, \tag{5}$$

which approaches 1 if the number of offloaded tasks for all UEs at each TS is similar.

In contrast to metrics like data rate, the offloading fairness indices have no explicit threshold regarding their impact on the QoE of users. Thus, we include them into the objective function as in [13]. By jointly considering the number of offloaded tasks and the fairness indices, we formulate the trajectory control problem as follows

$$\begin{aligned} \min_{\phi_{n,t},\upsilon_{n,t}} \quad & \mathbb{E}\left[-\sum_{t=1}^{T}(e_t + \lambda f_t^{uav} + \beta f_t^{ue})\right] \\ s.t.\ & \phi_{n,t} \in [0, 2\pi], \upsilon_{n,t} \in [0, 1], \end{aligned} \tag{6}$$

where λ and β are hyper-parameters used for adjusting the relative importance of the terms in the optimization objective function of (6). The expectation is taken over all random variables, including the UAV locations and fading channels in each TS.

At the initial TS, both the position of UEs and the randomness factor of UAV mobility are all unknown. Additionally, the objective function involving the derivation of $\widetilde{x}_{k,n,t}$ is not tractable. Therefore, traditional optimization methods are unsuitable, and we apply MARL to solve the problem in the following.

III. Communication-Assisted Value Decomposition Network Algorithm

In this section, we first introduce some basic notions of MARL. Then, we reformulate Problem (6) in the form of MARL by designing the observation, action and reward. Finally, we introduce the communication-assisted MARL framework and its solution.

A. MARL Problem Formulation

In MARL settings, usually the multi-agent decentralized partially observable Markov decision process (Dec-POMDP) is employed [17, 18]. Specifically, a Dec-POMDP can be represented by the state space $\mathcal{S}$, the action space $\mathcal{A}$, and the observation space $\mathcal{O}$ for each agent [14]. At the beginning of each TS t, the nth agent receives its local observation $\boldsymbol{o}_{n,t} \in \mathcal{O}$, which is a part of the state $\boldsymbol{s}_t$. Then, it chooses an action $\boldsymbol{a}_{n,t} \in \mathcal{A}$, according to its local observation $\boldsymbol{o}_{n,t}$ based on policy $\boldsymbol{\pi}_n$. After all agents' actions are executed, agent n

obtains reward $r_{n,t}$, which depends on the actions of all agents $\{\boldsymbol{a}_{n,t}|1 \leq n \leq N\}$. At the end of TS t, the current state $\boldsymbol{s}_t$ evolves into the next state $\boldsymbol{s}_{t+1}$. For cooperative tasks, all agents have the same objective of maximizing the team reward $\mathbb{E}\left[\sum_{t=1}^{T}\sum_{n=1}^{M}\gamma^{t-1}r_{n,t}\right]$, where γ is the discount factor.

To employ RL methods for solving Problem (6), we firstly define the local observation, as well as the action and reward function for each agent at TS t as follows.

1) **Observation** $\boldsymbol{o}_{n,t}$: Since both the offloading decision and offloading data rate depend on the distance between UAVs and users, we include their coordinates into the observation vector. Additionally, we include the accumulated number of tasks offloaded for UEs and the accumulated load of UAVs, which determine the fairness among UEs and among UAVs, respectively. Since the UAVs have limited reception range, we only take the information of the nearest M_{d}^{ue} users and the nearest M_{d}^{uav} UAVs into account. Then, the observation vector of UAV n can be formulated as

$$\boldsymbol{o}_{n,t} = \Bigg[\boldsymbol{d}_{1,t}^{ue}, ..., \boldsymbol{d}_{M_d^{ue},t}^{ue}, \sum_{t'=1}^{t}\sum_{n=1}^{N} x_{1,n,t'}, ..., \sum_{t'=1}^{t}\sum_{n=1}^{N} x_{M_d^{ue},n,t'}, \boldsymbol{d}_{1,t}^{uav}, ..., \boldsymbol{d}_{M_d^{uav},t}^{uav}, y_{1,t}, ..., y_{M_d^{uav},t}\Bigg]. \tag{7}$$

2) **Action** $\boldsymbol{a}_{n,t}$: The action of each UAV agent includes its heading and speed, i.e. $\boldsymbol{a}_{n,t} = [\phi_{n,t}, \upsilon_{n,t}]$.

3) **Reward** $r_{n,t}$: Since we consider cooperative tasks, the team reward

$$r_{n,t} = e_t + \lambda f_t^{uav} + \beta f_t^{ue} - \sum_{n=1}^{N}\frac{c_n^{\mathrm{col}}}{N}, \tag{8}$$

is shared among all agents, where c_n^{col} is a penalty term introduced for avoiding collisions between UAVs. Specifically, $c_n^{\mathrm{col}} = 1$ when UAV n collides with others, otherwise $c_n^{\mathrm{col}} = 0$.

Then, the goal of each agent is to cooperatively minimize the loss function expressed by

$$\min_{\boldsymbol{\pi}_n, 1\leq n\leq N} J = \mathbb{E}\left[-\frac{1}{N}\sum_{t=1}^{T}\sum_{n=1}^{N}\gamma^{t-1}r_{n,t}\right]. \tag{9}$$

B. Communication Assisted Decentralized Reinforcement Learning Framework

Existing RL-based papers on wireless MEC networks tend to implicitly assume conditional independence of actions from different agents [12, 13]. Consequently, each agent chooses its action purely based on its local observation during execution, while neglecting any interactions among agents. By contrast, we allow each agent to exchange its local information with the agents within its communication range during both training and execution, and more importantly, to learn what information should be exchanged for promoting cooperation among agents for better informed decision making.

Specifically, we introduce the communication-assisted value-based (CAVB) MARL framework of Fig. 1(a). Each agent stores and trains an agent network locally to estimate the local state-action value function of $Q_n(\boldsymbol{o}_n, \boldsymbol{a}_n) \triangleq \mathbb{E}[\sum_{i=t}^{T}\gamma^{i-t}r_{n,t}|\boldsymbol{o}_{n,t} = \boldsymbol{o}_n, \boldsymbol{a}_{n,t} = \boldsymbol{a}_n]$. Based on this,

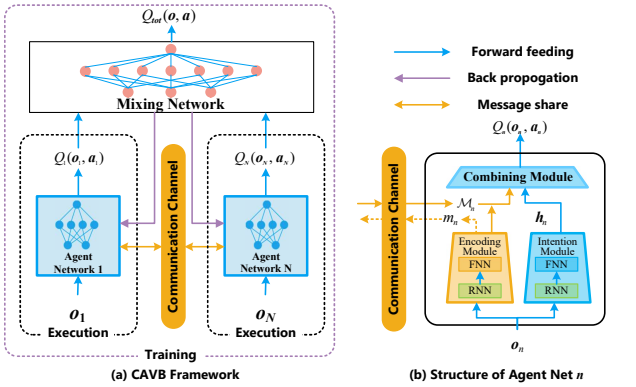

Fig. 1. CAVB framework and agent network structure.

the optimal action of agent n can be determined as $\boldsymbol{a}_n = \arg\max_{\boldsymbol{a}_n'} Q_n(\boldsymbol{o}_n, \boldsymbol{a}_n')$. In contrast to the existing MARL frameworks, where each agent only estimates the local state-action value based on its own local observation, we allow each agent to transfer messages based on its local observation between nearby agents. In this way, the observation gleaned from nearby agents can also be taken into account at a moderate overhead, when estimating the local state-action value. As a benefit, the local state-action can be estimated more accurately. Then, a mixing network combines the output of all agents' networks for approximating the global state-action value function of $Q_{tot}(\boldsymbol{o}, \boldsymbol{a}) \triangleq \mathbb{E}[\sum_{i=t}^{T}\gamma^{i-t}r_{n,t}|\boldsymbol{o}_t = \boldsymbol{o}, \boldsymbol{a}_t = \boldsymbol{a}]$, where we have $\boldsymbol{o} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, ..., \boldsymbol{o}_N\}$, and $\boldsymbol{a} = \{\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_N\}$. For instance, $Q_{tot}(\boldsymbol{o}, \boldsymbol{a})$ can be the summation of all the individual action value functions in the value decomposition network (VDN) [18], yielding $Q_{tot}(\boldsymbol{o}, \boldsymbol{a}) = \sum_{n=1}^{N} Q_n(\boldsymbol{o}_n, \boldsymbol{a}_n)$.

C. Agent Network Structure and Training Algorithm

In the following, we first design the agent network and then formulate the above-mentioned algorithm for training the agent network.

Each agent network contains three modules: *encoding* module, *intention* module, and *combining* module, as shown in Fig. 1(b). To mitigate the model's complexity and accelerate the training process, the same values of $\boldsymbol{\theta}$ are shared among all agents' network. Therefore, in the following, we consider agent n as an example for characterizing the agent network.

(1) The *encoding* module takes the local observation $\boldsymbol{o}_n$ as its input and outputs the encoded message $\boldsymbol{m}_n$. It consists of a recurrent neural network (RNN) followed by a fully-connected neural network (FNN). This recurrent structure can integrate the previous observations into the encoded messages that are transferred between agents.

(2) The *intention* module also takes the observation $\boldsymbol{o}_n$ as its input, and outputs the intention $\boldsymbol{h}_n$ that indicates the agent's behavior without considering others. It shares the same RNN with the encoding module. The motivation for introducing the intention module is to accelerate the learning process. At the beginning of the training phase, the messages from others may be noisy due to the randomly initialized neural networks, and hence they may negatively affect the decision making. The intention is used for compensating such an effect and helps learn local policies more quickly.

(3) The *combining* module is a FNN, which takes both the local intention $\boldsymbol{h}_n$ and the messages arriving from the nearest M_{d}^{uav} agents (denoted as $\mathcal{M}_n$, including local message m_n), and outputs the estimated local action-value function.

The training and execution process is shown in Fig. 2. The training process includes sample collection and agent network updates. During the sampling process, we harness the ϵ-greedy policy to determine each agent's action. In particular, the agent chooses $\boldsymbol{a}_n = \arg\max_{\boldsymbol{a}'_n} Q_n(\boldsymbol{o}_n, \boldsymbol{a}'_n)$ with a probability of $1-\epsilon$, and opts for a random action with probability ϵ. The interactions of all agents with the environment, including the messages received by each agent, are collected and stored in an experience *replay buffer* $\mathcal{D}$. Each sample is denoted by a five-tuple $(\boldsymbol{o}_t, \mathcal{M}_t, \boldsymbol{a}_t, r_t, \boldsymbol{o}_{t+1})$, where $\mathcal{M}_t = \{\mathcal{M}_{1,t}, ..., \mathcal{M}_{N,t}\}$ represents the received messages of all agents.

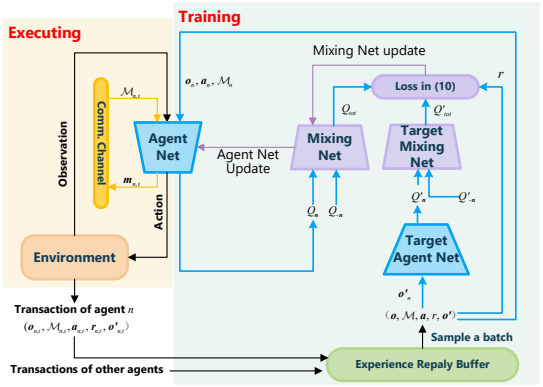

Fig. 2. The training and execution process of the nth UAV agent.

The agent networks are trained as shown in Fig. 2 via updating the parameters $\boldsymbol{\theta}$ by minimizing the following loss function via gradient descent,

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{o},\mathcal{M},\boldsymbol{a},r,\boldsymbol{o}')\sim\mathcal{D}}[(Q_{tot}(\boldsymbol{o},\boldsymbol{a};\boldsymbol{\theta}) - y)^2], \tag{10}$$

where $y = r + \gamma \max_{a'} Q'_{tot}(\boldsymbol{o}', \boldsymbol{a}'; \widehat{\boldsymbol{\theta}})$ and $\widehat{\boldsymbol{\theta}}$ is the parameter of the *target network*, which has the same architecture as the agent network, but have different parameter values. It is updated via $\widehat{\boldsymbol{\theta}} \leftarrow \tau\boldsymbol{\theta} + (1-\tau)\widehat{\boldsymbol{\theta}}$ using a small value of τ. This soft update helps reduce the correlations between $Q_{tot}(\boldsymbol{o}, \boldsymbol{a}; \boldsymbol{\theta})$ and the target value y, hence stabilizing learning. The detailed procedures are provided in Algorithm 1.

IV. Simulation Results

In this section, we compare the performance of our CAVDN algorithm to that of several baseline methods via simulation. Specifically, the following state-of-the-art DRL-based trajectory plan methods are compared: (1) VDN of [18] (2) multi-agent deep deterministic policy gradient (MADDPG) of [17]; (3) Double deep Q-network (DQN) of [19]; (4) CommNet of [15], which directly processes the received messages using the arithmetic mean; (5) Targeted Multi-Agent Communication (TarMAC) of [20], which processes the received messages using the weighted mean via multi-head attention. For methods that cannot be directly applied to continuous action spaces, we discretize the action space, where the heading set is $\{0, \pi/4, ..., 7\pi/4\}$ and the speed set is $\{0, 0.5, 1\}$, resulting

Algorithm 1 The CAVDN algorithm

Randomly initialize $\boldsymbol{\theta}$ and $\widehat{\boldsymbol{\theta}}$ following Gaussian distribution. Set $\mathcal{D} = \emptyset$.
for episode $= 1$ to max-episode-number **do**
  Reset the environment, and obtain the initial local observation $\boldsymbol{o}_n$ for $n = 1$ to N.
  for $t = 1$ to T **do**
    Get message $\boldsymbol{m}_{n,t} = e(\boldsymbol{o}_{n,t}; \theta_e)$ for agent $n = 1$ to N.
    Select action $\boldsymbol{a}_n$ from based on $\boldsymbol{o}_{n,t}$ and $\mathcal{M}_{n,t}$ using ϵ-greedy $Q_n(\boldsymbol{o}_{n,t}, \boldsymbol{a}_{n,t}; \boldsymbol{\theta})$ for agent $n = 1$ to N.
    Execute actions $\{\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_N\}$. Then, obtain reward r and next state observations $\boldsymbol{o}'_n$ for each agent n.
    Push $(\boldsymbol{o}, \boldsymbol{a}, \mathcal{M}, \boldsymbol{r}, \boldsymbol{o}')$ into replay buffer $\mathcal{D}$ and set $\boldsymbol{o}_n \leftarrow \boldsymbol{o}'_n$ for $n = 1$ to N.
    if length of $\mathcal{D}$ larger than given length **then**
      Randomly generate index set $\mathcal{B} = \{b_1, ..., b_B\}$.
      Obtain the sample batch of B samples $\{(\boldsymbol{o}^i, \boldsymbol{a}^i, \mathcal{M}^i, r^i, \boldsymbol{o}'^i)\}_{i\in\mathcal{B}}$ from $\mathcal{D}$.
      Compute the critic loss as $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{i}\sum_i (Q_{tot}(\boldsymbol{o}^i, \boldsymbol{a}^i) - y^i)^2$, where $y^i = r^i + \gamma \max_a Q_{tot}(\boldsymbol{o}'^i, a; \widehat{\boldsymbol{\theta}})$.
      Update θ by minimizing $\mathcal{L}(\boldsymbol{\theta})$.
      Update target network parameters by $\widehat{\boldsymbol{\theta}} \leftarrow \tau\boldsymbol{\theta} + (1-\tau)\widehat{\boldsymbol{\theta}}$.
    end if
  end for
end for

in an action space of size 24. Moreover, we also compare our CAVDN algorithm to the *random* and *greedy* UAV trajectory control methods, where each UAV moves randomly or towards the nearest UE sequentially.

We consider a 200×200 m^2 square area containing $M = 12$ randomly distributed UEs and $N = 4$ UAVs at the altitude of 100 m. Only the users with a "horizon distance" smaller than $D_{\mathrm{th}} = 20$ m can be discovered by and connected to the UAV, and we set $M_{\mathrm{d}}^{uav} = 3$ and $M_{\mathrm{d}}^{ue} = 6$. For the channel model, we set $\alpha = 2$, $\mu = -3$ dB as in [21]. The noise power is $\sigma^2 = -100$ dBm, and the transmit power is $P = 500$ mW. The maximal speed is $V_{\max} = 40$ m/s as in [22]. Detailed settings regarding the neural networks and the training process can be found at our github repository.[1]

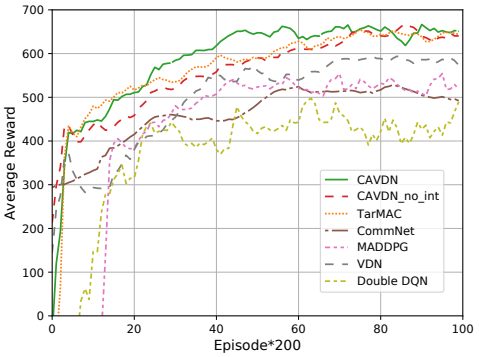

Fig. 3. Learning curves of CAVDN and other baselines. The reward is averaged over all UAVs and timesteps in each episode.

In Fig. 3, we compare the average reward versus the number of episodes in the training phase for different methods. It can be observed that VDN and MADDPG achieve better

[1]The code for reproducing the simulation results of this paper is available at: https://github.com/chenbq/CAVDN.

performance than Double DQN, because they maintaining a global value-function or global critic network during training. Among all methods, CAVDN achieves the highest average reward, due to the extra information received via communication during both training and testing phases, which provides each agent with extra information about the others for promoting cooperation. The performance of CommNet is inferior to that of CAVDN and that of TarMAC. This is because CommNet simply uses the arithmetic mean for processing the messages received, which results in information loss. Moreover, our proposed CAVDN performs slightly better than TarMAC because we also use RNN for encoding the messages. Besides, compared to CAVDN_no_int (CAVDN without intention module), the reward of CAVDN increases more rapidly and converges earlier, which validates the benefits of the intention module.

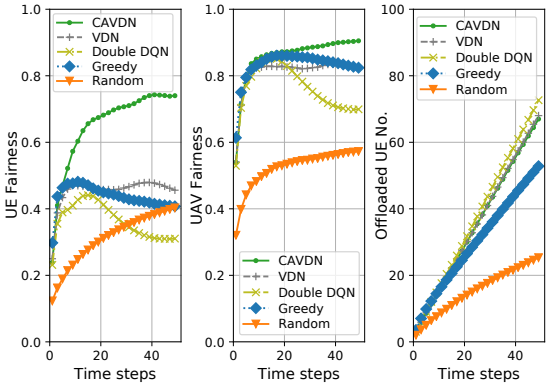

Fig. 4. Comparison of offloading fairness and the number of offloaded tasks.

The UE fairness, UAV fairness, and the number of offloaded tasks, are further investigated in Fig. 4 after the convergence of the RL-based methods. During testing, the positions of UEs are randomly generated and they are different from that used in the training phase. We can see that both the UE fairness and UAV fairness of CAVDN are the highest compared to others methods. Moreover, all learning-based methods achieve similar performance in terms of the number of offloaded tasks. It is worth noting that for the Double DQN and greedy methods, both the UE fairness and UAV fairness first increase and then decrease. This is because each UAV only has partial observations, hence some UAVs may have excessive load in the absence of efficient communication between agents, which prevents load-balancing.

V. Summary and Conclusions

A novel MARL based UAV trajectory planning scheme was proposed for UAV-aided MEC networks. The objective is to maximize the number of offloaded tasks and offloading fairness by optimizing the trajectory control policies of UAVs. To intelligently exchange critical information among agents to assist decentralized decision making, we conceived a communication-assisted MARL framework and proposed the CAVDN algorithm for training each UAV. Our simulation results showed that our proposed CAVDN algorithm improves the offloading fairness among users and balances the load among UAVs, compared to both the state-of-the-art DRL-based and classic greedy methods. This indicates that the messages learned and exchanged via communication provide valuable knowledge for each agent to "understand" the situation and the intention of others, which is critical for multi-agent cooperation in sophisticated trajectory control policies. In the future, we will investigate the impact of the cost introduced by message exchange.

References

[1] C. Park and J. Lee, "Mobile edge computing-enabled heterogeneous networks," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 2, pp. 1038–1051, 2021.

[2] J. Yu *et al.*, "IRS assisted NOMA aided mobile edge computing with queue stability: Heterogeneous multi-agent reinforcement learning," *IEEE Trans. on Wirel. Commun.*, 2022.

[3] N. H. Motlagh, M. Bagaa, and T. Taleb, "UAV-based IoT platform: A crowd surveillance use case," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 128–134, 2017.

[4] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy efficient resource allocation in UAV-enabled mobile edge computing networks," *IEEE Trans. Wirel. Commun.*, vol. 18, no. 9, pp. 4576–4589, 2019.

[5] M. Li *et al.*, "Energy-efficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3424–3438, 2020.

[6] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2049–2063, 2018.

[7] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, and R. P. Liu, "Energy-efficient admission of delay-sensitive tasks for mobile edge computing," *IEEE Trans. Commun.*, vol. 66, no. 6, pp. 2603–2616, 2018.

[8] Q. Wu and R. Zhang, "Common throughput maximization in UAV-enabled OFDMA systems with delay consideration," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6614–6627, 2018.

[9] L. Zhang *et al.*, "Task offloading and trajectory control for UAV-assisted mobile edge computing using deep reinforcement learning," *IEEE Access*, vol. 9, pp. 53 708–53 719, 2021.

[10] L. Wang *et al.*, "Deep reinforcement learning based dynamic trajectory control for UAV-assisted mobile edge computing," *IEEE Trans. on Mobile Comput.*, Early Access.

[11] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2059–2070, 2018.

[12] F. Khoramnejad and M. Erol-Kantarci, "On joint offloading and resource allocation: A double deep Q-network approach," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 4, pp. 1126–1141, 2021.

[13] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multi-agent deep reinforcement learning based trajectory planning for multi-UAV assisted mobile edge computing," *IEEE Trans. on Cognitive Commun. and Netw.*, vol. 7, no. 1, pp. 73–84, Mar. 2021.

[14] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 750–797, Oct. 2019.

[15] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *NeurIPS*, 2016, pp. 2244–2252.

[16] Q. Liu *et al.*, "Charging unplugged: Will distributed laser charging for mobile wireless power transfer work?" *IEEE Veh. Technol. Mag.*, vol. 11, no. 4, pp. 36–45, 2016.

[17] L. Ryan *et al.*, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *NeurIPS*, 2017, pp. 6379–6390.

[18] P. Sunehag *et al.*, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, 2018, pp. 2085–2087.

[19] J. Hu *et al.*, "Cooperative internet of UAVs: Distributed trajectory design by multi-agent deep reinforcement learning," *IEEE Trans. on Commun.*, vol. 68, no. 11, pp. 6807–6821, Nov. 2020.

[20] A. Das *et al.*, "TarMAC: Targeted multi-agent communication," in *ICML*, 2019.

[21] Y. Wang, Z. Hu, X. Wen, Z. Lu, J. Miao, and H. Qi, "Three-dimensional aerial cell partitioning based on optimal transport theory," *in IEEE ICC*, 2020.

[22] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Trans. on Wirel. Commun.*, vol. 18, no. 4, pp. 2329–2345, Apr. 2019.