Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Supplementary materials for

Yang LI, Ziling WEI, Jinshu SU, Baokang ZHAO, 2024. A multi-agent collaboration scheme for energy-efficient task scheduling in 3D UAV-MEC space. *Front Inform Technol Electron Eng*, 25(6):824-838.
https://doi.org/10.1631/FITEE.2300393

1 Preliminary details of the MADDPG algorithm

As illustrated in the main article, the environment involves multiple unmanned aerial vehicles (UAVs), multiple communication channels, and many intricate random tasks. In addition, the optimization problem involves many optimization elements and requires mutual collaboration. The traditional deep reinforcement learning (DRL) approaches such as Deep Q Network (DQN), Actor–Critic (AC), and Deep Deterministic Policy Gradient (DDPG) (Wang S et al. (2018)) perform well in the single-agent environments (Wang ZQ et al. (2022)). However, they are poorly suited to multi-agent collaborative environments. During the training process, each agent's policy is constantly changing, making the environment unstable from the perspective of individual agents. This prevents the stability of the learning, resulting in an enormous challenge for convergence. The multi-agent deep deterministic policy gradient (MADDPG) algorithm (Lowe et al. (2017)) is introduced to solve the problem caused by environment dynamics. It is designed to learn its policies while considering other agents' policies.

Partially observable Markov games, known as multi-agent extension of Markov decision processes (MDPs), are utilized in MADDPG. Three sets of states, observations, and actions are adopted in these games to define N agents. Based on private observation o_i, each agent adopts a deterministic policy μ_i to select its action a_i. Let $o = \{o_1, ..., o_N\}$, $\mu = \{\mu_1, ..., \mu_N\}$, and $\vartheta = \{\vartheta_1, ..., \vartheta_N\}$, where ϑ contains all parameters of the set of policies μ. For agent i, the gradient of the expected return in the actor network can be represented as

$$\nabla_{\vartheta_i} J(\mu_i) = \mathbb{E}_{o,a\sim D}\left[\nabla_{\vartheta_i}\mu_i(a_i|o_i) \times \nabla_{a_i} Q_i^{\mu}(o, a_1, ..., a_N)|_{a_i=\mu_i(o_i)}\right], \tag{S1}$$

where D is the experience replay buffer, storing many transitions $(o, a_1, ..., a_N, r_1, ..., r_N, o')$. $Q_i^{\mu}(o, a_1, ..., a_N)$ represents a centralized action-value function of agent i, which inputs o and $\{a_1, ..., a_N\}$ and outputs the Q value. It is noted that agent i's action a_i is derived from its actor network, while the other agents' actions are still the original values selected from D. In the critic network, Q_i^{μ} is updated as

$$\mathcal{L}_i = \mathbb{E}_{o,a_1,...,a_N,r_1,...,r_N,o'}\left[\left(Q_i^{\mu}(o, a_1, ..., a_N) - y\right)^2\right], \tag{S2}$$

where y is the Temporal-Difference (TD) target. It is calculated as

$$y = r_i + \gamma Q_i^{\mu'}(o', a'_1, ..., a'_j, ..., a'_N)|_{a'_j=\mu'_j(o'_j),\forall j\in[1,N]}, \tag{S3}$$

where r_i represents the reward received by agent i, γ is the discount factor, and μ' denotes the target policy derived by the target network with delayed update parameters. Each action a'_j, $\forall j \in [1, N]$, including agent i's action a'_i, is obtained from its corresponding target policy μ'_j.

2 Flying actions in a 3D space

Fig. S1 presents an example of flying action with $p_n^t = hy$ and $\theta_n^t = 45°$. This denotes that the UAV selects axis plane h–y as its flying plane with a flying angle of 45° to fly in a 3D space. Since the co-channel interference is considered, k_n^t represents the number of sub-channels selected. $\{\{b_{u,e}^t\}_{u\in\mathcal{U},a_{u,n}=1}\}$ is the set of task schedule strategies for the associated users of UAV n. The action space is thus derived as $a^t = \{\{a_n^t\}_{n\in\mathcal{N}}\}$.

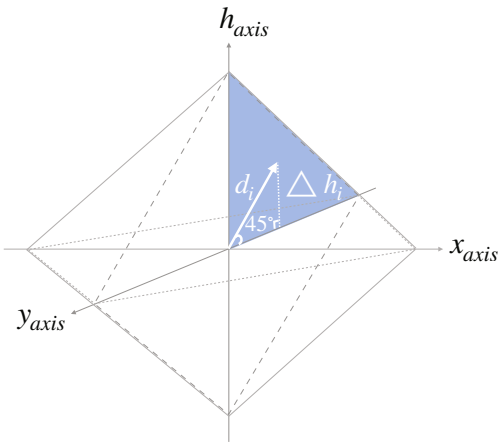


Fig. S1 Flying actions in a 3D space

3 Details of the function $K(\bar{o}_n^{t+1}, f_i)$

$d^2(\bar{o}_n^{t+1}, f_i)$ is a list with the size of $\bar{k}$. The operation $\frac{d^2(\bar{o}_n^{t+1},f_i)}{d_m^2}$ normalizes the distance $d^2(\bar{o}_n^{t+1}, f_i)$ with d_m^2. Subsequently, the fraction is processed with maximization, i.e., $\bar{d} \leftarrow \max\left(\frac{d^2(\bar{o}_n^{t+1},f_i)}{d_m^2} - \hat{d}_k, 0_{\bar{k}}\right)$, where $\hat{d}_k$ is the kernel cluster distance and $0_{\bar{k}}$ is a list containing $\bar{k}$ zeros. The kernel value $\bar{K}_i \leftarrow \frac{\epsilon}{\bar{d}+\epsilon}$ is derived. Then, the similarity between $\bar{o}_n^{t+1}$ and its neighbors is calculated as $\bar{s} \leftarrow \sqrt{\sum_{i=1}^{\bar{k}} \bar{K}_i} + \bar{c}$. If $\bar{s} > \bar{s}_{\mathrm{MAX}}$ indicates that the agent enters a similar observation, then $\bar{r}_n^t = 0$. Otherwise, $\bar{r}_n^t = \frac{1}{\bar{s}}$. $\bar{s}_{\mathrm{MAX}}$ represents the kernel maximum similarity. In brief, the clustering method is adopted in the inverse kernel of Dirac delta function. The intrinsic reward is a positive value when the agent enters a disparate state. Otherwise, the agent has no intrinsic reward. Therefore, the agent is motivated to explore different states to maximize reward.

4 Parameter setting

The key notations used are illustrated in Table S1. An area is utilized as the simulation target in UAV-MEC, in which N UAVs and U users are located. The uncertainty of the number of associated users would lead to dimension inconsistency of state and action. To tackle this issue, the maximum number of associated users is limited to $\overline{U} = 10$. The state and action dimensions are determined to be $4N + 6\overline{U} + 4$ and $4 + \overline{U}$, respectively. Taking $N = 3$ as an example, there are 76 states and 14 actions. The actor network of each UAV takes observation o_n^t and action a_n^t as the input and output, respectively. The network is structured with an input layer, three hidden layers, and an output layer. The critic network inputs state space s^t and action space a^t, and outputs action value Q. As illustrated in Table S2, we train these two kinds of networks with the learning rates of 0.0005 and 0.0001, separately. The CTMADDPG algorithm is trained 5000 episodes, each containing 100 time slots. As for the UAV-MEC environment setting, each UAV flies with a velocity that is in the range of [0, 100] m/s. The flying angle is selected in $[0, 2\pi]$. There are $N - 1$ sub-channels for N UAVs to choose from, which results in interference among the UAVs. For a more explicit

expression, the remaining parameters for the simulated environment are listed in Table S3, whose values are extracted from multiple studies in the literature (Dai et al., 2022; Lakew et al., 2022; Lu et al., 2022; Wang LY et al., 2022b; Wang ZQ et al., 2022; Zhao et al., 2022).

Table S1 Summary of the key notations

Notations	Description
N	The number of UAVs
K	The number of sub-channels
U	The number of users
$\overline{U}$	The limit number of associated users
T	The number of time slots
$X_{min}, Y_{min}, H_{min}$	The minimum values of X, Y, and H axis
$X_{max}, Y_{max}, H_{max}$	The maximum values of X, Y, and H axis
D_u	The task's data size
C_u	The required CPU cycles per bit
$max(D_{\mathcal{U}})$	The global maximum value of data size
$min(D_{\mathcal{U}})$	The global minimum value of data size
$max(\overline{D}_{\mathcal{U}})$	The global maximum value of deadline
v_n	The flying velocity of a UAV
θ_n	The flying angle in a flying plane
f_c	The carrier frequency
c	The light velocity
η_{LoS}, η_{NLoS}	The average additional losses for LoS and NLoS links
c_1, c_2	The constant values of environment
p_u	A user's transmission power
p_n	A UAV's transmission power
B_1	The bandwidth of a user
B_2	The bandwidth of a UAV
Θ	The noise power
f_n	The computation capacity of each UAV
f_e	The computation capacity of the remote BS
m	The mass of a UAV
g	The gravity coefficient
ρ	The density of the air
C_D	The drag coefficient of the air kinetics
S	The wing area of a UAV
$\overline{R}$	The gas constant
w_1, w_2, w_3	The parameter weights for flight energy

Table S2 Algorithm parameters

Notations	Description	Value
M	Training episodes	5000
T	Time slots in an episode	100
D	Experience replay buffer	10,000
X	Batch size	200
α_a	The learning rate of the actor network	0.0005
α_c	The learning rate of the critic network	0.0001
ι	The update interval for the actor network	2
ϵ	The ϵ-greedy value	0.1
γ	Discount factor	0.95
β	Discount coefficient of intrinsic reward	0.001
$\bar{c}$	The minimum amount of pseudo-counts	0.001
$\bar{\epsilon}$	The constant for kernel function	0.0001
$\bar{k}$	The number of the nearest neighbors	10
e	The proportion coefficient for soft update	0.01

The following six benchmarks are utilized for performance evaluation:

1. The MADDPG algorithm is a novel algorithm for the promotion of collaboration among multiple agents. As we have mentioned, the CTMADDPG algorithm is proposed based on MADDPG. Thus, MADDPG is adopted as an essential benchmark to evaluate the degree of performance improvement.

2. The Multiple-Independent-Agent DDPG (MIADDPG) algorithm encompasses a system wherein all agents run independently to maximize their own rewards without any coordination. Thus, it is meaningful

Table S3 Simulation values

Parameters	Value
N	3, 4, 5
K	2, 3, 4
U	40, 60, 80
$\overline{U}$	10
T	100
$X_{min}, Y_{min}, H_{min}$	0, 0, 10
$X_{max}, Y_{max}, H_{max}$	1300, 1300, 1000
D_u	[1, 10] Mb
C_u	[1000, 1500]
$min(D_{\mathcal{U}})$	1 Mb
$max(D_{\mathcal{U}})$	10 Mb
$max(\overline{D}_{\mathcal{U}})$	1000
v_n	[0, 100] m/s
θ_n	[0, 2π]
f_c	2×10^9
c	3×10^8
η_{LoS}, η_{NLoS}	1.44544, 199.526
c_1, c_2	12.081, 0.11395
p_u	1 W
p_n	10 W
B_1	0.2 MHz
B_2	20 MHz
Θ	-100 dbm
f_n	$\{10, 20\}$ GHz
f_e	80 GHz
m	0.895 kg
g	9.8 N/kg
ρ	1.29 kg/m^3
C_D	0.03
S	0.25 m^2
$\overline{R}$	8.314
w_1, w_2, w_3	0.01, 0.50, 0.49

to deploy this algorithm as a benchmark.

3. The MASAC algorithm adopts entropy to facilitate exploration, similar to the curiosity mechanism in the proposed CTMADDPG algorithm. Hence, we adopt it for comparison.

4. The MAPPO algorithm utilizes the actor–critic framework, which is the same as the one adopted in the CTMADDPG algorithm.

5. The Greedy algorithm is cost-effective and tends to make a fast decision with specific greedy goals. By contrast, DRL algorithm usually consumes a large amount of time for strategy making. Since the number of fulfilled tasks significantly influences the energy efficiency, the Greedy algorithm is designed for the UAVs to hover over the user-dense area with a proper flying height, further providing edge services for the ground users.

6. The Random algorithm represents a scheme under which all UAVs randomly choose their strategies, e.g., flying and scheduling.

5 Training rewards with $N = 3$

During the training procedure, the reward is an intuitive metric for performance evaluation. Thus, we record the training reward for three DDPG-based algorithms, i.e., MIADDPG, MADDPG, and CT-MADDPG. As shown in Fig. S2, the training reward is tested with two sub-channels and three UAVs $\{\text{UAV } 1, \text{UAV } 2, \text{UAV } 3\}$, which are equipped with $\{20 \text{ GHz}, 10 \text{ GHz}, 10 \text{ GHz}\}$ computation capacities. The point of departure is (600, 600, 500), where the middle height value allows the UAVs to fly upward or down. The left green figure represents the MIADDPG algorithm, whose reward fluctuates in the range of [100, 250]. The fluctuation range indicates that MIADDPG cannot converge to a relatively stationary state. The middle orange figure is the MADDPG algorithm, which converges in the compact range of [170, 250]. The right blue figure denotes the CTMADDPG algorithm, whose reward values are distributed in the range of [200, 300] at the convergence state. Compared with MADDPG, the reward shows a noticeable improvement

of 19% and rapidly converges to the optimal value.

Fig. S2 Training rewards of MIADDPG, MADDPG, and CTMADDPG with $N = 3$

6 Convergence rewards with $N = 3$

To better illustrate the rewards obtained under the convergence state, we divide the tasks into three classifications in terms of intensity:

1. Low-intensity task. The default parameters are used in this case.

2. Middle-intensity task. When $\max(\overline{D}_{\mathcal{U}}) = 100$, each task should be finished within the deadline of 100, which indicates a high emergency. We fix $\max(\overline{D}_{\mathcal{U}}) = 100$ for each task to emphasize its urgency degree.

3. High-intensity task. When $C_u = 5000$, each bit of task data requires 5000 CPU cycles. This means that each task needs a high computation capacity. Thus, we set $\max(\overline{D}_{\mathcal{U}}) = 100$ and $C_u = 5000$ to represent each task's high intensity.

For low-intensity tasks, the tested reward results are shown in Fig. S3(a) with four independent variables, i.e., task data size, the number of task CPU cycles, task deadline, and UAV computation capacity. The reward value shows a distinct increase with the growth of task data size. It is in positive correlation with the amount of data; i.e., the greater the quantum of data successfully computed, the greater the number of rewards derived.

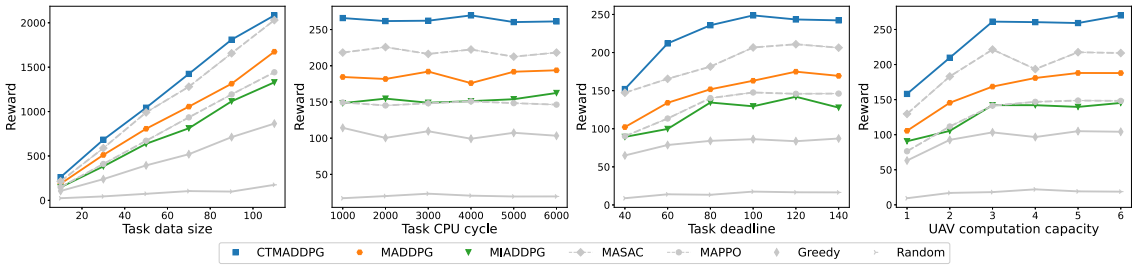
Since each UAV is equipped with enough computation capacity, increasing the number of task CPU cycles will not bring much burden on these UAVs in the low-intensity case. Therefore, no apparent change happens with the rise of the number of task CPU cycles. The task deadline represents the emergency level for a task. The fulfilled task data are calculated with the deadline. Only when a task is finished within its deadline range can it be added to the reward for energy efficiency calculation. Hence, the reward slightly grows with a change in task deadlines. When the UAV computation capacity is limited to a small value range, the reward grows in the $[1, 3]$ range. After that, the reward reaches the saturation point. Currently, the computation capacity is not the critical element determining the reward. Among the seven algorithms, the Random and Greedy algorithms show the worst results; this is because the Random algorithm randomly selects its strategies without any regularity, and as for the Greedy algorithm, although it demonstrates a slight increase in the performance in terms of reward, it is unable to adjust its strategy with the transformation of the environment due to the fixed greedy goal. The CTMADDPG algorithm performs the best, and it can fully utilize the task-related information for adjustment.

For the middle-intensity tasks, Fig. S3(b) shows the tested reward results with four independent variables. Since the deadline for the middle-intensity task is fixed, we adopt user bandwidth for measurement.

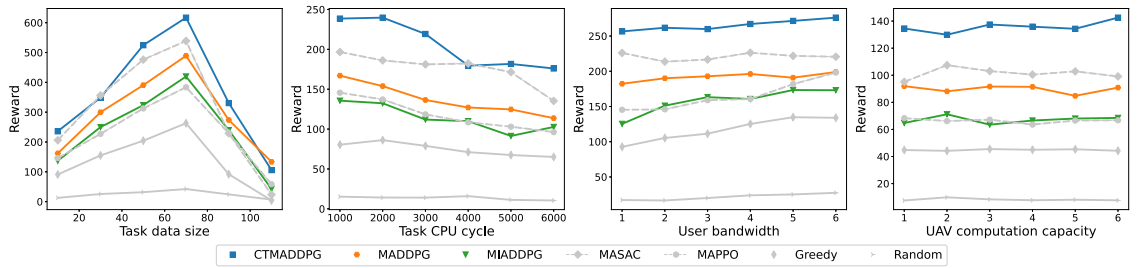
With the change in the task data size, the reward increases to the peak and begins to decline, which is different from that in the low-intensity case. As mentioned, the reward will increase when more task data are fulfilled. The amount of the fulfilled task data grows with the growth of task data size. However, when the task data size is beyond the processing capacity of the UAV in the middle-intensity case, the task

is failed. Hence, the reward is declined. For the middle-intensity task, its computation demand is higher compared with that of the low-intensity task. The greater the number of CPU cycles required, the more the computation burden that is loaded. Thus, in the middle-intensity case, the reward shows a downtrend with the growth of the number of task CPU cycles. The user bandwidth determines the transmission rate during the upload period. With the growth of the user bandwidth, the transmission time is reduced, which further saves time with the constraint of task deadline. Therefore, the reward related to the change in user bandwidth slightly increases. With the increase in the UAV computation capacity, the reward tends to be unchanged. In the middle-intensity case, the UAV computation capacity is highly demanded. The value scope $[1, 5]$ is too small to influence the reward. As we can see, the maximal reward is reduced from 250 to 140, in contrast with the low-intensity case. In three DDPG-based algorithms, CTMADDPG shows the best reward. Since MADDPG and MIADDPG cannot efficiently explore the environment to reach the optimal convergence state, they present a relatively low reward.

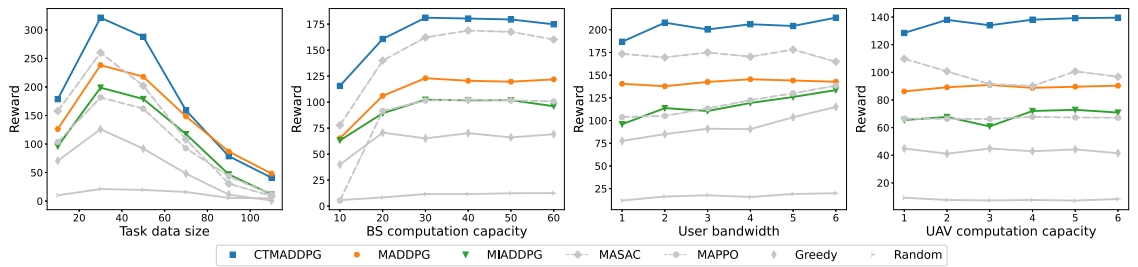
For high-intensity tasks, Fig. S3(c) shows the tested reward results. Since the number of the task CPU cycles is fixed in the high-intensity case, we take BS computation capacity as the succedaneum. The high-intensity task has more load than the middle-intensity task. Compared with the middle-intensity task, the reward arrives at the peak in advance with the increased task data size. The overall reward is smaller than that in the middle-intensity case. At the initial phase, with the growth of BS computation capacity, the available computation resource and the reward are increased. Nevertheless, the distance between the BS and the UAV is considerable, which prolongs the transmission time. The agent may not migrate the tasks to



(a) With the low-intensity task



(b) With the middle-intensity task



(c) With the high-intensity task

Fig. S3 Reward results in different intensities: (a) with the low-intensity task; (b) with the middle-intensity task; (c) with the high-intensity task

the BS. Hence, the reward shows a rising tendency and begins to be flat. For the two independent variables, i.e., user bandwidth and UAV computation capacity, the tendency of the overall reward is similar to the one in Fig. S3(b). The task burden in the high-intensity case is much more than that in the middle-intensity case. Therefore, with the same user bandwidth value, the reward in Fig. S3(b) is more significant than that in Fig. S3(c). In the figure, CTMADDPG gains the most significant reward, and MASAC takes the second place. The reward in MIADDPG is much the same as that in MAPPO. Both of them perform worse than MADDPG.

References

Dai C, Zhu K, Hossain E, 2022. Multi-agent deep reinforcement learning for joint decoupled user association and trajectory design in full-duplex multi-UAV networks. *IEEE Trans Mob Comput*, 22(10):6056-6070. https://doi.org/10.1109/TMC.2022.3188473

Lakew DS, Tran AT, Dao NN, et al., 2022. Intelligent offloading and resource allocation in heterogeneous aerial access IoT networks. Proc 31st Int Conf on Neural Information Processing Systems, p.6382-6393.

Lowe R, Wu Y, Tamar A, et al., 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. Proc 31st Int Conf on Neural Information Processing Systems, p.6382-6393.

Lu WD, Mo YD, Feng YQ, et al., 2023. Secure transmission for multi-UAV-assisted mobile edge computing based on reinforcement learning. *IEEE Trans Netw Sci Eng*, 10(3):1270-1282. https://doi.org/10.1109/TNSE.2022.3185130

Wang LY, Zhang HX, Guo SS, et al., 2022. Deployment and association of multiple UAVs in UAV-assisted cellular networks with the knowledge of statistical user position. *IEEE Trans Wirel Commun*, 21(8):6553-6567. https://doi.org/10.1109/TWC.2022.3150429

Wang S, Jia DY, Weng XS, 2018. Deep reinforcement learning for autonomous driving. https://arxiv.org/abs/1811.11329

Wang ZQ, Rong HG, Jiang HB, et al., 2022. A load-balanced and energy-efficient navigation scheme for UAV-mounted mobile edge computing. *IEEE Trans Netw Sci Eng*, 9(5):3659-3674. https://doi.org/10.1109/TNSE.2022.3188670

Zhao N, Ye ZY, Pei YY, et al., 2022. Multi-agent deep reinforcement learning for task offloading in UAV-assisted mobile edge computing. *IEEE Trans Wirel Commun*, 21(9):6949-6960. https://doi.org/10.1109/TWC.2022.3153316