基于DQN-DDPG的空地协作边缘计算任务卸载与资源分配研究

沈 乐

（南京邮电大学 通信与信息工程学院，江苏 南京 210003）

摘 要： 在基础设施有限的地区或紧急救援场景中，无人机辅助的移动边缘计算被认为是一种有效的解决方案，可处理资源受限的智能设备的计算密集型任务和时延敏感性计算任务。考虑到地面基站和多无人机辅助的多用户空地协作移动边缘计算场景，提出一种联合优化用户关联、子信道分配及边缘服务器计算资源的分配方法，以最小化长期平均时延的任务卸载和资源分配方案。首先，根据用户的随机任务生成无人机移动方案，基于不同的卸载决策建立卸载计算模型和本地计算模型。然后，以最小化长期平均时延为优化目标优化问题。最后，结合DQN与DDPG提出一种基于混合深度强化学习DQN-DDPG的任务卸载和资源分配算法（HDCR），解决离散和连续变量之间的问题和混合决策问题。仿真表明，所提算法相较于基于离散决策的DDCR等算法，在减少平均时延方面性能更优。

关键词： 移动边缘计算；空地协作；无人机；混合决策；深度强化学习；任务卸载；资源分配

DOI： 10. 11907/rjdk. 231389 **开放科学（资源服务）标识码（OSID）：**

中图分类号： TN929.5 **文献标识码：** A **文章编号：** 1672-7800（2024）004-0074-08

Task Offloading and Resource Allocation Based on DQN-DDPG for Aerial-Ground Cooperative Mobile Edge Computing

SHEN Le

（*School of Communications and Information Engineering*, *Nanjing University of Posts and Telecommunications*, *Nanjing* 210003, *China*）

Abstract: In areas with limited infrastructure or emergency rescue scenarios, UAV assisted mobile edge computing is considered an effective solution, which can handle computing intensive tasks and delay sensitive computing tasks of resource constrained intelligent devices. Considering the ground base station and multi UAV assisted multi-user air ground cooperative mobile edge computing scenario, a joint optimization method of user association, subchannel allocation and edge server computing resource allocation is proposed to minimize the long-term average delay of task unloading and resource allocation. Firstly, generate a drone movement plan based on the user's random tasks, and establish offloading calculation models and local calculation models based on different offloading decisions. Then, optimize the problem with the objective of minimizing long-term average latency. Finally, combining DQN and DDPG, a task offloading and resource allocation algorithm (HDCR) based on hybrid deep reinforcement learning DQN-DDPG is proposed to solve the problems between discrete and continuous variables and mixed decision problems. Simulation results show that the proposed algorithm performs better in reducing average latency compared to algorithms such as DDCR based on discrete decision-making.

Key Words: mobile edge computing; aerial-ground cooperation; unmanned aerial vehicle; hybrid decision; deep reinforcement learning; task offloading; resource allocation

0 引言

物联网和B5G/6G无线技术的快速发展推动了智能应用的空前发展。移动应用程序特别是在线3D游戏、人脸识别、增强现实、虚拟现实等应用日益广泛[1]，这些应用程序通常是计算密集型和延迟敏感型，用户对高数据速率和低时延的要求呈指数增长，给受到计算资源和电池容量限制的物联网设备带来了挑战。移动边缘计算（Mobile Edge Computing, MEC）已被设想为克服上述挑战的一种方案，

收稿日期： 2023-04-17

作者简介： 沈乐（1997-），男，CCF会员，南京邮电大学通信与信息工程学院硕士研究生，研究方向为边缘计算与深度强化学习。本文通讯作者：沈乐。

允许智能设备将任务卸载到边缘服务器,为用户设备提供计算资源,从而降低能耗和时延[2-4]。然而,传统MEC服务器通常固定在特定位置,在基础设施不够完善的地区(偏远的海洋、高山等)或一些紧急情况下(救援、灾难重建等)受到限制。

无人机(Unmanned Aerial Vehicle, UAV)可良好的配备MEC服务器,因其移动灵活、部署迅速的特点被人们广泛使用。作为飞行MEC服务器,UAV通过与地面用户建立视距(Line of Sight, LoS)链路,实时为用户提供计算卸载服务。为了充分利用UAV的优势,已有大量工作研究UAV辅助的MEC系统的设计[5-10],然而上述文献仅考虑空对地场景下的计算卸载,当用户计算任务急剧增多时UAV计算资源受限,此时将无法满足用户需求。

近年来,考虑到地面基站(Ground Base Station, GBS)强大的计算能力,针对空地协作边缘计算场景下的问题受到了广泛关注。目前,大部分研究学者正着手开发空地协作MEC系统中的任务卸载和资源分配方案[11-17],利用传统优化技术处理相关问题[11-12],然而配备边缘服务器的GBS和UAV的计算资源是预先设定的,但来自地面用户的计算任务和资源需求是时变的,这些信息在实际的MEC场景中很难得到。因此,针对未知环境下的动态决策问题,许多研究人员使用马尔科夫决策过程(Markov Decision Process, MDP)对MEC系统中的动态控制进行建模,应用强化学习(Reinforcement Learning, RL)方法处理空地协作MEC系统中的问题[13-14]。其中,RL的智能体通过与未知环境不断交互,快速学习应对策略,使智能体具备一定的决策能力。此外,深度学习(Deep Learning, DL)的神经网络也具有一定的感知能力,将DL的感知能力和RL的决策能力相结合进行深度强化学习(Deep Reinforcement Learning, DRL),可有效处理复杂系统中的感知决策问题。

在多用户空地协作卸载场景中,动作空间通常同时包括离散动作和连续动作。然而,大多数现有工作仅使用一种类型的动作空间处理不同类型的变量,即连续动作的精细离散化或离散动作的连续值输出映射[16-17]。离散化动作空间可能会承受高维动作空间的影响,导致复杂性更高;连续值对离散动作的输出映射会增加额外的复杂性,可能会使智能体偏向次优决策而无法获取最优策略。因此,单一的DRL算法在处理复杂场景中的混合决策问题时较难得到最优策略,探索新的强化学习方法对解决空地协作MEC系统中的混合决策、动态控制问题十分重要。

1　相关工作

目前,许多研究工作利用传统的优化方法处理空地协作MEC系统中的问题。Yao等[11]通过联合优化多用户卸载策略和无人机的轨迹,最大限度减少地面用户的总传输能耗,确保了无人机的机动性和任务延迟限制。Lu等[12]在考虑时变/随机地面信道和视距空地信道的情况下,提出一个鲁棒的优化问题以最大限度减少无人机和用户能耗。然而,传统优化方法在实际的多服务器空地协作场景中,难以获得准确的环境和完整的信息。考虑到非平稳环境和非凸性质,传统优化方法只能得到近似的最优解。此外,随着网络节点和覆盖面积增加,模型的优化复杂度将进一步增加。

RL方法用于处理空地协作MEC系统中的一些问题。Wang等[13]提出一种基于RL的联合在线轨迹规划和卸载调度方案,使无人机和多小区网络中的基站动态协作,提供边缘计算服务。Nie等[14]提出一个空地协作MEC系统实现功率最小化,开发了一种基于DQN的方法来优化用户关联、计算资源和功率控制。Li等[15]使用一种dueling DQN的方法获得最优控制策略,目的是最大化区块链系统的吞吐量和数据计算能力。Seid等[16]研究了海洋无线通信网络中的延迟最小化问题,基于DQN与DDPG方法优化UAVs的飞行轨迹和虚拟机的配置。Chen等[18]通过分配无线带宽和卸载比,基于DQN的方法平衡空地协作MEC系统的延迟和能耗。Peng等[19]提出一种基于DDPG的方法优化多UAVs空地协作车辆网络中的用户关联、GBS和UAVs的可用资源,然而利用DDPG梯度方法处理离散问题,会增加额外的复杂性并降低模型控制精度。

尽管,上述关于空地协作MEC系统已解决了连续—离散混合卸载决策这一挑战,但大多采用DQN、DDPG或选取两者其一结合启发式优化来寻找离散或连续动作空间的解。实验表明,简单地放缩离散动作或离散化连续动作可能不适合为混合决策场景提供更好的性能改进,并且未能全面考虑信道的分配问题。

为此,本文考虑在动态环境的单GBS与多UAVs的空地协作MEC系统中采取正交频分复用(Orthogonal Frequency Division Multiple Access, OFDM)[17]接入方案,以最小化长期平均时延的任务卸载和资源分配。同时,结合DQN与DDPG提出一种基于DRL的混合决策算法(HDCR)联合优化用户关联、子信道分配和边缘服务器资源分配。

2　系统模型

本文假设时隙长度足够小、用户位置大致不变,并且GBS和每个UAV与其相关地面用户之间的信道状态在每个时隙内保持不变[3],如图1所示。考虑到GBS和多UAVs辅助的多用户空地协作MEC场景,由一个GBSm、U架配备边缘服务器的UAVs和N位地面用户组成,通过用户生成的任务选择本地计算或卸载计算,UAVs按固定轨迹飞行在矩形区域内,GBSm和UAVs为选择卸载的地面用户分配带宽和计算资源。其中,用户和UAVs分别由集合$\boldsymbol{N}_s \triangleq \{n = 1, 2, ..., N\}$、$\boldsymbol{U}_s \triangleq \{u = 1, 2, \cdots, U\}$表示;系统的带宽被划分为$K$个正交的子信道,由集合$\boldsymbol{K}_s \triangleq \{k =$

1,2,⋯,K}表示,每个子信道带宽为B Hz。用户可被分配到一个或多个子信道,将计算任务卸载到关联的UAV上,并将系统任务期间划分为T个时隙,由集合$\boldsymbol{T}_s \triangleq \{t = 1,2,\cdots,T\}$表示。

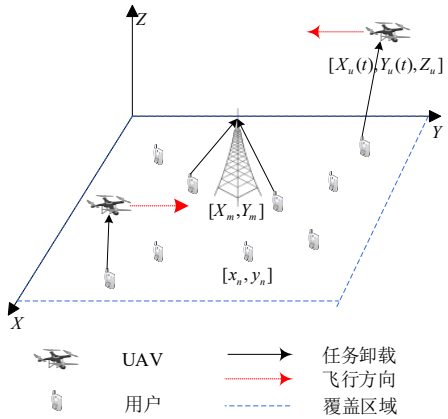

Fig. 1 System model

图1 系统模型

2.1 用户任务与关联模型

在每个时隙,各用户都会生成一个计算密集型的任务$\boldsymbol{A}_n(t) = \{D_n(t), F_n(t)\}, \forall n \in N_s, t \in T_s$,其中$D_n(t)$、$F_n(t)$分别表示用户$n$的任务数据大小和执行该任务所需的CPU周期。考虑到各物联网设备的计算能力有限,本文采用二进制卸载方案,即每个任务既可在本地执行,也可卸载到边缘服务器执行,以确保每个计算任务在最大时间允许$T^{\max}$内处理完成。同时,指定二进制变量$a_{n,m}(t)$、$a_{n,u}(t)$分别表示用户n与GBSm、n与UAVu的关联索引。其中,$a_{n,m}(t) = 1$(或$a_{n,u}(t) = 1, u \in U_s$)表示在时隙$t$,用户$n$与GBS$m$(或UAV$u$)的关联,反之$a_{n,m}(t) = 0$(或$a_{n,u}(t) = 0, u \in U_s$)。

在每个时隙中,各用户最多只能匹配一个边缘服务器。此外,将$a_{n,u}(t) = 1, u = 0$定义为本地处理,其约束表示为:

$$a_{n,m}(t) = \{0,1\}, \forall n \in N_s, t \in T_s \tag{1}$$

$$a_{n,u}(t) = \{0,1\}, \forall n \in N_s, u \in U_s, t \in T_s \tag{2}$$

$$a_{n,m}(t) + \sum_{u=0}^{U} a_{n,u}(t) = 1, \forall n \in N_s, t \in T_s \tag{3}$$

2.2 任务卸载模型

在卸载过程中,采用OFDM接入方案,每个子信道在每个时隙只分配给一个用户,但每个用户可同时占用多个子信道以提升传输效率。$\delta_{n,k}(t) \in \{0,1\}$表示子信道分配指示器,约束表示为:

$$\sum_{n=1}^{N} \delta_{n,k}(t) \leqslant 1, \forall k \in K_s, t \in T_s \tag{4}$$

$$\sum_{k=1}^{K} \delta_{n,k}(t) \leqslant K, \forall n \in N_s, t \in T_s \tag{5}$$

式中:$\delta_{n,k}(t) = 1$表示在时隙t子信道k被分配给用户n,否则$\delta_{n,k}(t) = 0$。

2.2.1 用户与GBSm关联模型

GBSm和用户n的2-D坐标分别为$[X_m, Y_m]$、$[x_n, y_n]$。在时隙t,GBSm与用户n间的水平距离为:

$$l_{n,m}(t) = \sqrt{\left(X_m - x_n\right)^2 + \left(Y_m - y_n\right)^2} \tag{6}$$

因此,结合用户n分配到信道资源,对应GBSm的上行链路的数据速率[20]为:

$$r_{n,m}(t) = \sum_{k=1}^{K} \delta_{n,k} B \log_2\left(1 + \frac{\alpha P_{\mathrm{Tr}}}{l_{n,m}^2(t)}\right), \forall n \in N_s, t \in T_s \tag{7}$$

2.2.2 用户与UAV关联模型

在时隙t,UAVu的3-D笛卡尔坐标表示为$[X_u(t), Y_u(t), Z_u]$,所有UAV都在固定高度飞行以避免遇到障碍物。具体的,每架UAV都从预定的初始坐标$[X_u(0), Y_u(0), Z_u]$出发,然后按固定轨迹飞行。用户n的2-D坐标为$[x_n, y_n]$。因此,用户n与UAVu间的水平距离可计算为:

$$l_{n,u}(t) = \sqrt{\left(X_u(t) - x_n\right)^2 + \left(Y_u(t) - y_n\right)^2} \tag{8}$$

结合用户n分配到的信道资源,对应UAVu的上行链路的数据速率为:

$$r_{n,u}(t) = \sum_{k=1}^{K} \delta_{n,k} B \log_2\left(1 + \frac{\alpha P_{\mathrm{Tr}}}{Z_u^2 + l_{n,u}^2(t)}\right), \forall n \in N_s, u \in U_s, t \in T_s \tag{9}$$

式中:B为子信道的带宽;P_{Tr}表示用户的传输功率;$\alpha = \frac{g_0 G_0}{\sigma^2}$;$G_0$为天线增益;$g_0$为参考距离为1 m时的信道功率增益;$\sigma^2$表示噪声功率。

由上可知,除了用户与GBSm或UAV关联模式及子信道分配方式外,每个边缘服务器还需要为相关卸载用户分配适当的计算资源。由于GBS和UAVs的发射功率远高于地面用户,且每个任务的处理结果的数据规模相对较小,因此本文忽略了从边缘服务器返回处理结果的时耗[21]。如果用户n选择卸载任务,则用户n与边缘服务器之间的传输时延计算为:

$$T_n^{\mathrm{Tr}}(t) = \begin{cases} \dfrac{D_n(t)}{r_{n,m}(t)}, & \text{if } a_{n,m}(t) = 1 \\ \dfrac{D_n(t)}{r_{n,u}(t)}, & \text{if } a_{n,u}(t) = 1 \end{cases} \tag{10}$$

指定$f_{n,m}^{\mathrm{C}}(t)$(或$f_{n,u}^{\mathrm{C}}(t)$)表示GBSm(或UAVu)分配给卸载用户n的计算资源,则用户n在对应边缘服务器上执行任务的计算时延表示为:

$$T_n^{\mathrm{C}}(t) = \begin{cases} \dfrac{F_n(t)}{f_{n,m}^{\mathrm{C}}(t)}, & \text{if } a_{n,m}(t) = 1 \\ \dfrac{F_n(t)}{f_{n,u}^{\mathrm{C}}(t)}, & \text{if } a_{n,u}(t) = 1 \end{cases} \tag{11}$$

因此,用户n卸载任务的总时间成本表示为:

$$T_n^{\mathrm{O}}(t) = T_n^{\mathrm{Tr}}(t) + T_n^{\mathrm{C}}(t), \forall t \in T_s \tag{12}$$

2.3　本地执行模型

如果用户 n 决定选择本地执行,则用户 n 的本地执行时延表示为:

$$T_n^{\mathrm{L}}(t)=\frac{F_n(t)}{f_n^{\mathrm{L}}(t)},\forall t\in T_s \tag{13}$$

式中:$f_n^{\mathrm{L}}(t)$ 表示用户 n 的计算资源。

综上所述,用户 n 在卸载模式或本地执行模式下完成计算任务时的总时间成本为:

$$T_n(t)=\begin{cases}T_n^{\mathrm{L}}(t), & \text{local execution}\\ T_n^{\mathrm{O}}(t), & \text{off loading}\end{cases} \tag{14}$$

3　问题公式化与HDCR算法

3.1　问题公式化

在时延和资源的约束下,通过联合优化用户关联、子信道分配和边缘服务器计算资源分配轨迹以最小化长期平均时延。具体的数学表达式为:

$$\begin{aligned}&\mathbf{P1}:\min_{\boldsymbol{A}_m,\boldsymbol{A}_u,\boldsymbol{K},\boldsymbol{F}_m,\boldsymbol{F}_u}\frac{1}{T}\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}T_n(t)\\&\text{s.t.}(1)-(5),\\&\text{C6}:\delta_{n,k}(t)\in\{0,1\},\forall n\in N_s,k\in K_s,t\in T_s\\&\text{C7}:\sum_{n=1}^{N}a_{n,m}(t)f_{n,m}^{\mathrm{C}}(t)\leqslant f_{\mathrm{GBS}}^{\max},\forall t\in T_s\\&\text{C8}:\sum_{n=1}^{N}a_{n,u}(t)f_{n,u}^{\mathrm{C}}(t)\leqslant f_{\mathrm{UAV}}^{\max},\forall u\in U_s,t\in T_s\\&\text{C9}:T_n(t)\leqslant T^{\max},\forall n\in N_s,u\in U_s,t\in T_s\end{aligned} \tag{15}$$

式中:$\boldsymbol{A}_m=\{a_{n,m}(t),\forall n\in N_s,t\in T_s\}$、$\boldsymbol{A}_u=\{a_{n,u}(t),\forall n\in N_s,u\in U_s,t\in T_s\}$ 表示用户与GBSm、用户与UAVu 的卸载变量;$\boldsymbol{K}=\{\delta_{n,k}(t),\forall n\in N_s,k\in K_s,t\in T_s\}$ 表示子信道分配指示变量;$\boldsymbol{F}_m=\{f_{n,m}^{\mathrm{C}}(t),\forall n\in N_s,t\in T_s\}$、$\boldsymbol{F}_u=\{f_{n,u}^{\mathrm{C}}(t),\forall n\in N_s,u\in U_s,t\in T_s\}$ 分别表示GBSm 分配给关联用户的计算资源变量、UAVu 分配给关联用户的计算资源变量;约束条件C6表示子信道分配指示器;约束条件C7、C8表示卸载用户可从边缘服务器分配到有限的计算资源;$f_{\mathrm{GBS}}^{\max}$、$f_{\mathrm{UAV}}^{\max}$ 分别表示GBSm、UAVu 可分配的最大计算资源;约束条件C9表示每个任务都需在规定时间内完成;$T^{\max}$ 为最大时延允许。

由此可知,优化变量包含离散变量 $\boldsymbol{A}_m,\boldsymbol{A}_u,\boldsymbol{K}$ 和连续变量 $\boldsymbol{F}_m,\boldsymbol{F}_u$,则 **P1** 是一个混合整数非线性规划,采用传统优化方法难以得到最优解。因此,本文引入马尔可夫决策过程,提出一种基于DQN与DDPG的混合决策HDCR算法处理混合动作并学习最优控制策略。

3.2　MDP建模

本文根据提出的优化问题进行MDP建模,模型包括状态、动作和奖励3个部分。每个时隙智能体通过与环境进行交互,根据状态 s_t 采取动作 a_t 并反馈给环境,从环境中获得一个新的状态 s_{t+1} 并获得一个奖励 r_t,将公式化后的问题转化为 U+2个智能体的MDP过程,包括1位用户智能体和 U+1个边缘服务器智能体,从而降低动作处理的维度[22]。

(1)状态。智能体的输入状态决定了智能体采取的行动,用户智能体需要观测UAVs的位置和所有用户任务大小作出相应的卸载决策和子信道分配策略。用户智能体的输入状态定义为:

$$\begin{aligned}s_t^N=\{&[X_1(t),Y_1(t)],[X_2(t),Y_2(t)],...,\\&[X_u(t),Y_u(t)],A_1(t),A_2(t),...A_n(t)\}\end{aligned} \tag{16}$$

同时,边缘服务器智能体需要观测用户的卸载决策、子信道分配策略及用户执行任务所需的CPU周期,以作出合理的资源分配策略。GBS智能体和UAV智能体的输入状态定义为:

$$\begin{aligned}s_t^m=\{&a_{1,m}(t),...,a_{n,m}(t),...,a_{N,m}(t),\\&\delta_{1,1}(t),...,\delta_{n,k}(t),...,\delta_{N,K}(t)\\&F_1(t),F_2(t),...F_n(t)\}\end{aligned} \tag{17}$$

$$\begin{aligned}s_t^u=\{&a_{1,u}(t),...,a_{n,u}(t),...,a_{N,u}(t),\\&\delta_{1,1}(t),...,\delta_{n,k}(t),...,\delta_{N,K}(t)\\&F_1(t),F_2(t),...F_n(t)\}\end{aligned} \tag{18}$$

式中:s_t^N 为DQN单元的输入状态,DQN单元进行离散决策,输出的卸载决策和子信道分配动作由UAVs的位置和用户的任务大小共同决定;s_t^m、s_t^u 分别为DDPGm 单元、DDPGu 的输入状态,即DDPG单元的资源分配策略取决于与之对应的卸载、子信道分配方案及用户执行任务所需的CPU周期。

(2)动作。DQN单元和DDPG单元的动作构成了动作空间。其中,DQN单元动作包含所有可能的用户关联和子信道分配策略。用户智能体的输出动作定义为:

$$\begin{aligned}a_t^N=\{&a_{1,m}(t),...,a_{n,m}(t),...,a_{N,M}(t),\\&a_{1,0}(t),...,a_{n,u}(t),...,a_{N,U}(t),\\&\delta_{1,1}(t),...,\delta_{n,k}(t),...,\delta_{N,K}(t)\}\end{aligned} \tag{19}$$

同时,边缘服务器智能体根据DQN的输出动作,作出相应的计算资源分配策略。GBS智能体和UAV智能体的输出动作定义为:

$$a_t^m=\{f_{1,m}^C(t),f_{2,m}^C(t),...,f_{N,m}^C(t)\} \tag{20}$$

$$a_t^u=\{f_{1,u}^C(t),f_{2,u}^C(t),...,f_{N,u}^C(t)\} \tag{21}$$

式中:a_t^m、a_t^u 分别为GBSm、UAVu 的资源分配策略。

(3)奖励。DQN单元的输出动作决定了用户的任务计算方式,如果用户 n 选择本地计算则需考虑本地计算时延 $T_n^{\mathrm{L}}(t)$,如果用户 n 选择卸载计算,卸载决策和子信道分配方式都决定了用户 n 与对应边缘服务器的上行链路数据速率,将间接影响传输时延 $T_n^{\mathrm{Tr}}(t)$。用户智能体从环境反馈的奖励设置为:

$$r_t^N=-\frac{1}{N}\sum_{n=1}^{N}\left(T_n^{\mathrm{Tr}}(t)+T_n^{\mathrm{L}}(t)\right) \tag{22}$$

边缘服务器智能体根据用户的卸载决策、子信道分配策略及用户执行任务所需的CPU周期，作出资源分配策略，从而影响卸载用户在对应边缘服务器上执行任务的计算时延 $T_n^{\mathrm{C}}(t)$。GBS智能体和UAV智能体从环境反馈的奖励定义为：

$$r_t^m = -\frac{\sum_{n=1}^{N} a_{n,m}(t) T_n^{\mathrm{C}}(t)}{\sum_{n=1}^{N} a_{n,m}(t)} - \rho \tag{23}$$

$$r_t^u = -\frac{\sum_{n=1}^{N} a_{n,u}(t) T_n^{\mathrm{C}}(t)}{\sum_{n=1}^{N} a_{n,u}(t)} - \rho \tag{24}$$

式中：ρ 表示惩罚项，如果用户 n 与边缘服务器 m（或边缘服务器 u）没有关联，边缘服务器 m（或边缘服务器 u）仍分配给用户 n 计算资源，则施加惩罚。

3.3 HDCR算法

HDCR算法的结构如图2所示。DQN单元包含两个DNNs（Deep Neural Networks，DNNs），即估计网络 $Q(s_t, a_t, \chi)$ 和目标网络 $Q(s_t, a_t, \chi^-)$，分别用于估计动作的 Q 值和生成用于训练的目标 Q 值。χ、χ^- 为两个DNNs的网络参数，χ 每个时隙都会更新，而 χ^- 则在一个固定的时间间隔复制 χ 进行更新。

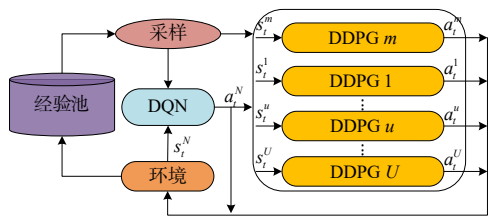

Fig. 2 HDCR algorithm framework

图2 HDCR算法框架

在时隙开始时，DQN单元从环境获取状态 s_t^N，然后根据 ε − *greedy* 贪婪策略得到一个动作 a_t^N，该动作根据概率 $1-\varepsilon$ 随机选取或根据概率 ε 选取状态 s_t^N 下 Q 值最大的动作。

$$a_t^N = \arg\max_{a_t^N} Q(s_t^N, a_t^N;\ \chi) \tag{25}$$

式中：ε 为贪婪因子；$0 < \varepsilon < 1$。

本文基于贪婪策略和奖励函数，DQN单元选择奖励更大的动作或尝试探索从未选择过的动作，以探索整个状态和动作空间，存储DQN单元将每个时隙的状态、动作、奖励和下一个状态表示为 $(s_t^N, a_t^N, r_t^N, s_{t+1}^N)$。当训练开始时，从经验池中抽取 S 组数据进行训练，然后得到目标 Q 值。

$$y_i^N = r_i^N + \gamma_{\mathrm{DQN}} \max_{a_{i+1}^N} Q(s_{i+1}^N, a_{i+1}^N, \chi^-) \tag{26}$$

式中：$\gamma_{\mathrm{DQN}} \in (0,1)$ 为DQN单元的折扣因子，在每次迭代中，通过最小化损失函数（见式(27)）更新估计网络。

$$L(\chi) = \left(y_i^N - Q(s_i^N, a_i^N;\ \chi)\right)^2 \tag{27}$$

在HDCR算法框图中共有 U+1个DDPG单元，能有效地处理连续动作空间。每个DDPG单元由4个DNNs组成，Actor网络 $\pi(s_t;\ \mu)$ 和Critic网络 $Q(s_t, a_t;\ \theta)$ 用于输出动作和估计动作的 Q 值；目标Actor网络 $\pi(s_t;\ \mu^-)$ 和目标Critic网络 $Q(s_t, a_t;\ \theta^-)$ 用于获取目标 Q 值，其参数分别为 μ, θ, μ^- 和 θ^-。每个DDPG单元结构及参数更新方式相同，本文将以DDPGm为例进行阐述。

在DQN单元作出动作 a_t^N 后，DDPGm 获取输入状态 s_t^m，由DDPGm的Actor网络 $\pi(s_t^m;\ \mu_m)$ 输出动作 a_t^m，得到GBSm的资源分配策略。为了平衡对新动作的探索和对已知动作的利用，添加了一个服从正态分布随机噪声 N_c：

$$a_t^m = \pi(s_t^m;\ \mu_m) + N_c \tag{28}$$

与DQN单元类似，存储DDPGm将每个时隙的状态、动作、奖励和下一个状态表示为 $(s_t^m, a_t^m, r_t^m, s_{t+1}^m)$。需要注意的是，DDPG单元的输入状态是根据观测DQN单元的输出动作所得到。在时隙 t，DQN单元作出决策后，输出动作中的部分信息结合用户执行任务所需的CPU周期将作为DDPGm的输入状态。具体的，在时隙 t，s_t^m 中的部分信息由 a_t^N 中的部分信息表示，在时隙 $t+1$，s_{t+1}^m 中的部分信息由 a_{t+1}^N 中的部分信息表示。因此，该单元需要存储 T 组数据，而状态 s_{T+1}^m 是在时隙 $T+1$ 获取的，则要新增一个虚拟时隙存储时隙 T 所需的状态 s_{T+1}^m。为此，在时隙 t 需要存储 s_t^m, a_t^m, r_t^m，在时隙 $t+1$ 需要DQN单元输出动作 a_{t+1}^N，即可获取 s_{t+1}^m，最后将 $(s_t^m, a_t^m, r_t^m, s_{t+1}^m)$ 存入经验池中。

在训练开始时，从经验池中随机抽取 S 组数据，DDPGm的Actor网络 $\pi(s_t^m;\ \mu_m)$ 通过策略梯度进行训练。

$$\nabla_{\mu_m} J \approx \frac{1}{S}\sum_i \left[\nabla_{\mu_m}\pi(s;\ \mu_m)|_{s=s_i^m} \nabla_a Q(s, a;\ \theta)|_{s=s_i^m, a=\pi(s_i^m;\ \mu_m)}\right] \tag{29}$$

为解决过拟合问题，使用 $\pi(s_t^m;\ \mu_m^-)$ 和 $Q(s_t^m, a_t^m;\ \theta_m^-)$ 根据经验抽取池数据，即Critic的目标 Q 值为：

$$y_i^m = r_i^m + \gamma_{\mathrm{DDPG}} Q(s_{i+1}^m, \pi(s_{i+1}^m;\ \mu_m^-);\ \theta_m^-) \tag{30}$$

式中：$\gamma_{\mathrm{DDPG}} \in (0,1)$ 为DDPG单元的折扣因子。

然后，通过最小化损失函数更新Critic网络：

$$L(\theta) = \frac{1}{S}\sum_i \left(y_i^m - Q(s_i^m, a_i^m;\ \theta_m)\right)^2 \tag{31}$$

最后，对两个目标网络参数采用软更新策略：

$$\begin{aligned} \mu_m^- &\leftarrow \tau\mu_m + (1-\tau)\mu_m^-, \\ \theta_m^- &\leftarrow \tau\theta_m + (1-\tau)\theta_m^- \end{aligned} \tag{32}$$

式中：τ 为软更新系数。

算法1 基于HDCR的任务卸载和资源分配算法

输入：用户智能体的状态 s_t^N，边缘服务器智能体的状态 $s_t^m, s_t^1, ..., s_t^U$。

输出：用户智能体的动作 a_t^N，边缘服务器智能体的动作 $a_t^m, a_t^1, ..., a_t^U$。

1. 初始化经验池
2. 初始化DQN单元的估计网络和目标网络
3. 初始化每个DDPG单元的Actor和Critic网络的估计网络和目标网络

4. FOR Episode = 1, 2, ...$k^{\max}$, DO
5. 初始化状态s_t^N
6. FOR DDPGm, DO
7. 初始化s_m, a_m, r_m
8. END
9. FOR DDPGu = 1, 2, ...U, DO
10. 初始化s_u, a_u, r_u
11. END
12. FOR t = 1, 2, ..., T + 1, DO
13. DQN单元根据贪婪策略选取动作a_t^N
14. DDPG单元根据a_t^N得到输入状态s_t^m, s_t^1, ...s_t^U
15. IF t > 1, DO
16. 将$\{s_m, a_m, r_m, s_t^m\}$存储到经验池$\mathcal{D}$中
17. FOR DDPGu = 1, 2, ...U, DO
18. 将$\{s_u, a_u, r_u, s_t^u\}$存储到经验池$\mathcal{D}$中
19. END
20. END
21. DDPG单元输出动作a_t^m, a_t^1, ...a_t^U
22. 与环境进行交互,获取奖励r_t^N, r_t^m, r_t^1, ...r_t^U
23. 与环境进行交互,获取DQN单元的下一状态s_{t+1}^N
24. IF t < T + 1, DO
25. 使$s_m = s_t^m$, $a\_m = a_t^m$, $r\_m = r_t^m$
26. FOR DDPGu = 1, 2, ...U, DO
27. 使$s_u = s_t^u$, $a\_u = a_t^u$, $r\_u = r_t^u$
28. END
29. 将$\{s_t^N, a_t^N, r_t^N, s_{t+1}^N\}$存储到经验池$\mathcal{D}$中
30. END
31. IF 训练过程开始, DO
32. 从经验池中抽取S组数据
33. DQN单元的估计网络的参数χ由公式(27)训练,目标网络的参数χ^-每W步复制χ得到
34. DDPG单元的Actor网络和Critic网络的参数μ和θ分别由公式(29)和(31)训练,目标Actor网络和目标Critic网络的参数μ^-和θ^-由公式(32)更新
35. END
36. END
37. END

4　实验结果与分析

4.1　参数设置

为了使仿真结果具有一般性,参数设置参考文献[9]与文献[19],地面用户数量设为N = 4,为用户提供服务的GBS为1个,UAV数量为U = 2,UAV的飞行高度固定在Z_m = 75 m,用户和UAV被限制在$R^{\max}$ = 400 m的矩形目标区域内。每个回合时隙T = 60,最大时延允许为$T^{\max}$ = 1 s。系统可用子信道个数K = 4且每个子信道的带宽为B = 2 MHz。每个时隙$D_n(t)$均匀分布在10~12.5 KB,执行任务所需的CPU周期数$F_n(t)$均匀分布在2 × 10^9~2.5 × 10^9之间。对于UAV移动,UAV1、UAV2的2-D初始坐标为[50, 350], [350, 50],UAV1纵坐标每时隙减少5 m,UAV2纵坐标每时隙增加5 m。

对于卸载计算,发射功率为P_{Tr} = 0.1 W,天线增益为G_0 = 2.284 6,噪声功率为σ^2 = −90 dbm,GBSm上的计算资源为$f_{\mathrm{GBS}}^{\max}$ = 100 GHz,UAVu上的计算资源为$f_{\mathrm{UAV}}^{\max}$ = 30 GHz,信道功率增益g_0 = 1.42 × 10^{-4}。对于本地计算,用户的计算资源$f_n^{\mathrm{L}}(t)$ = 2.5 GHz。

对于DDPG单元,Actor网络、Critic网络都包含两个全连接的隐藏层,分别有256和128个神经元,学习率分别为0.000 1和0.001,采用Adam Optimizer对Actor和Critic网络进行更新,惩罚项为ρ = 1,随机噪声为$N_c(0, 2)$,探测噪声每时隙的衰减率为0.999 95,折扣因子γ_{DDPG} = 0.99,软更新系数τ = 0.001。对于DQN单元,估计网络和目标网络包含两个完全连接的隐藏层,分别有100和20个神经元,以学习率为0.001采用Adam Optimizer训练估计网络,折扣因子γ_{DQN} = 0.9,初始贪婪因子为ε = 0.3。

在开始训练后,ε每时隙增加0.000 1,最终增加至0.95。DQN单元中目标网络的更新间隔为W = 100,最大训练回合数为$k^{\max}$ = 1 500。在训练过程中,采样数据数量S = 64,经验池大小$\mathcal{D}$ = 10 000。为了验证本文所提HDCR算法的性能优势,将其与RANDOM与DDCR算法进行比较。其中,RANDOM通过用户随机选择卸载决策,将子信道资源随机分配,然后通过边缘服务器随机分配计算资源;DDCR为基于离散决策DRL的任务卸载和资源分配算法,使用DQN单元取代HDCR方案中的DDPG单元,使得边缘服务器的资源分配动作被离散化,即使用U+2个DQN单元处理优化问题。

4.2　性能分析

图3为HDCR和DDCR算法下每回合的平均时延。由此可见,HDCR、DDCR算法控制下的平均时延随着回合数增加而降低,最终达到收敛。DDCR算法在170个回合时开始收敛,并在300个回合后稳定,而HDCR算法在170个回合开始收敛,在700个回合后稳定。虽然,DDCR算法收敛速度较快,但离散化资源分配动作使得DQN难以处理大维度动作空间,因此收敛性能较差,而HDCR算法能处理混合决策问题,平均时延也低于DDCR算法,所表现的性能更优。

图4、图5为不同边缘服务器计算资源下不同方案的平均时延。由此可见,随着边缘服务器计算资源增加,3种方案的平均时延均在下降,原因为用户能被分得的计算资源增加,从而进一步减小了卸载时处理任务的计算时延。具体的,RANDOM方案表现最差,原因为RANDOM方案是随机选取动作,无法保证选取合适的用户关联和资源分配策略,导致平均时延较高,而基于DRL的DDCR和HDCR方案则在降低平均时延方面表现较好。特别是,HDCR方案得益于边缘服务器资源的精确分配,所表现的性能

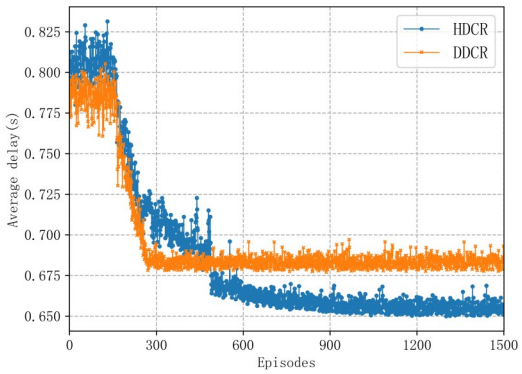

Fig. 3 Average delay of each episode under the HDCR and DDCR algorithm

图3 HDCR和DDCR算法下每回合的平均时延

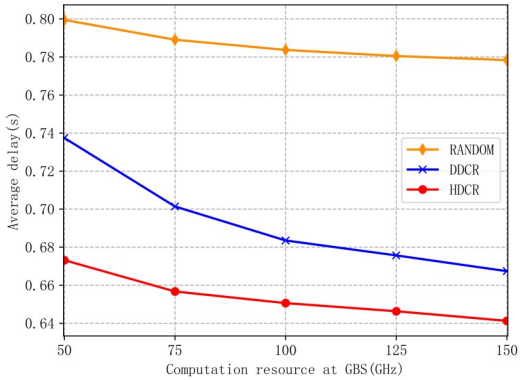

Fig. 4 Average delay achieved by different schemes under different GBS computation resources

图4 不同GBS计算资源下不同方案的平均时延

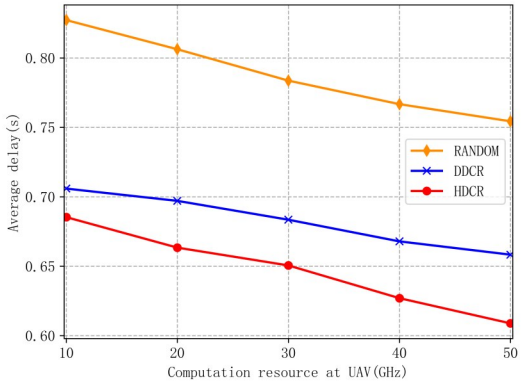

Fig. 5 Average delay achieved by different schemes under different UAV computation resources

图5 不同UAV计算资源下不同方案的平均时延

最优。

图6为不同子信道个数下不同方案的平均时延。由此可见,随着子信道个数增加,所有方案的平均时延迅速下降,原因为选择卸载的用户可被分得的信道资源增加,减少了用户与边缘服务器间的传输时延。由于动作选取不适当,RANDOM方案性能表现最差,HDCR方案的性能优于DDCR方案,原因为离散化动作空间使动作空间维度较大,造成DDCR方案性能变差。

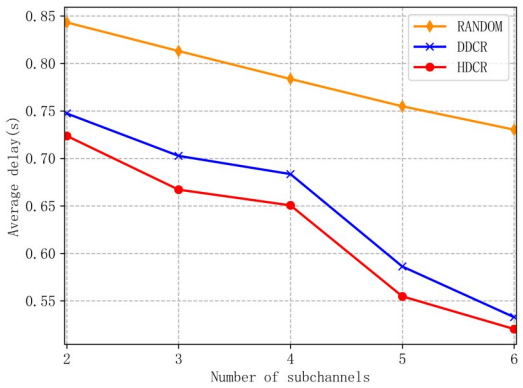

Fig. 6 Average delay achieved by different schemes under different subchannel numbers

图6 不同子信道个数下不同方案的平均时延

图7为HDCR控制下不同子信道个数下不同方案的平均时延,3种方案分别为只有UAVs辅助的方案、只有GBS辅助的方案和空地协作方案。随着子信道个数增加,所有方案的平均时延迅速下降,由于卸载用户可被分配的信道资源增加,对应卸载用户的传输时延减少。实验表明,只有UAVs辅助的方案性能表现最差,只有GBS辅助的方案的性能表现次之,而空地协作方案表现最好,原因为UAVs可分配的计算资源相较于GBS较少,导致处理任务计算时延较高,而固定基站的GBS方案会影响远距离卸载用户的上传速率。空地协作方案中边缘服务器覆盖面更广,UAVs可在飞行途中缩短与地面用户的距离,为地面用户提供服务,再结合GBS较高的计算资源,进一步提升了系统性能。

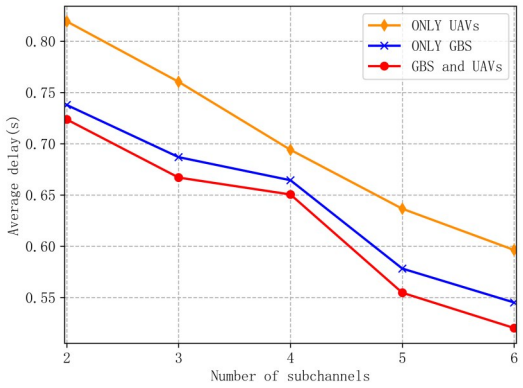

Fig. 7 Average delay achieved by different schemes of different subchannel numbers under HDCR control

图7 HDCR控制下不同子信道个数下不同方案的平均时延

4 结语

本文针对GBS和多UAVs辅助的多用户空地协作MEC场景,提出一种联合优化用户关联、子信道分配及边缘服务器计算资源分配,以最小化长期平均时延的任务卸载和资源分配方案。同时,为了处理包含离散和连续变量的优化问题,提出一种基于混合深度强化学习的任务卸载

和资源分配算法(HDCR),结合DQN与DDPG解决了GBS与多UAVs辅助的空地协作MEC场景中多用户混合决策问题。仿真表明,所提算法相较于离散决策DDCR等算法,在减少平均时延方面具有更好的性能。

在HDCR算法控制下的空地协作方案相较于非协作方案,能获取更低的平均时延。然而,本文所提HDCR算法包含多个智能体,在实际场景中多个DRL算法的混合会占据更多资源,因此仍需进一步研究基于DRL的混合决策框架,以减少资源消耗并加快网络收敛速度。

参考文献:

[1] STANKOVIC J A. Research directions for the Internet of Things[J]. IEEE Internet of Things Journal, 2014, 1(1): 3-9.

[2] HU H, SONG W W, WANG Q, et al. Energy efficiency and delay tradeoff in an MEC-enabled mobile IoT network[J]. IEEE Internet of Things Journal, 2022, 9(17): 15942-15956.

[3] HU H, ZHOU X, WANG Q, et al. Online computation offloading and trajectory scheduling for UAV-enabled wireless powered mobile edge computing[J]. China Communications, 2022, 19(4): 257-273.

[4] HU H, WANG Q, HU R Q, et al. Mobility aware offloading and resource allocation in a MEC-enabled IoT network with energy harvesting[J]. IEEE Internet of Things Journal, 2021, 8(24): 17541-17556.

[5] WANG Y, RU Z Y, WANG K Z, et al. Joint deployment and task scheduling optimization for large scale mobile users in multi-UAV-enabled mobile edge computing[J]. IEEE Transactions on Cybernetics, 2020, 50(9): 3984-3997.

[6] WU Z W, YANG Z L, YANG C, et al. Joint deployment and trajectory optimization in UAV-assisted vehicular edge computing networks[J]. Journal of Communications and Networks, 2022, 24(1): 47-58.

[7] JIANG C Z, LI Y R, SU R S, et al. A load balancing based resource allocation algorithm in UAV-aided MEC systems[C]// 2020 IEEE 6th International Conference on Computer and Communications, 2020: 519-523.

[8] DU Y, YANG K, WANG K Z, et al. Joint resources and workflow scheduling in UAV-enabled wirelessly-powered MEC for IoT systems[J]. IEEE Transactions on Vehicular Technology, 2019, 68(10): 10187-10200.

[9] WANG L, WANG K Z, PAN C H, et al. Deep reinforcement learning based dynamic trajectory control for UAV-assisted mobile edge computing[J]. IEEE Transactions on Mobile Computing, 2022, 21(10): 3536-3550.

[10] WANG L, WANG K Z, PAN C H, et al. Multi-agent deep reinforcement learning-based trajectory planning for multi-UAV assisted mobile edge computing[J]. IEEE Transactions on Cognitive Communications and Networking, 2021, 7(1): 73-84.

[11] YAO Y, DIAO X B, WANG X D, et al. Joint offloading strategy and trajectory optimization for aerial-ground cooperative mobile edge computing systems[C]// IET 8th International Conference on Wireless, Mobile & Multimedia Networks, 2019: 1-37.

[12] LU Y W, HUANG Y, HU T Y. Robust resource scheduling for air-ground cooperative mobile edge computing[C]// 2021 IEEE/CIC International Conference on Communications in China, 2021: 764-769.

[13] WANG S Y, HUANG Y, CLERCKX B. Dynamic air-ground collaboration for multi-access edge computing[C]// ICC 2022 - IEEE International Conference on Communications, 2022: 5365-5371.

[14] NIE Y W, ZHAO J H, GAO F F, et al. Semi-distributed resource management in UAV-aided MEC systems: a multi-agent federated reinforcement learning approach[J]. IEEE Transactions on Vehicular Technology, 2021, 70(12): 13162-13173.

[15] LI M, YU F R, SI P B, et al. UAV-assisted data transmission in blockchain-enabled M2M communications with mobile edge computing[J]. IEEE Network, 2020, 34(6): 242-249.

[16] SEID A M, BOATENG G O, ANOKYE S, et al. Collaborative computation offloading and resource allocation in multi-UAV-assisted IoT networks: a deep reinforcement learning approach[J]. IEEE Internet of Things Journal, 2021, 8(15): 12203-12218.

[17] WU Y H, WANG Y H, ZHOU F H, et al. Computation efficiency maximization in OFDM-based mobile edge computing networks[J]. IEEE Communications Letters, 2020, 24(1): 159-163.

[18] CHEN L M, KUANG X Y, ZHU F S, et al. Intelligent mobile edge computing networks for Internet of Things[J]. IEEE Access, 2021, 9: 95665-95674.

[19] PENG H X, SHEN X S. DDPG-based resource management for MEC/UAV-assisted vehicular networks[C]// 2020 IEEE 92nd Vehicular Technology Conference, 2020: 1-6.

[20] LI X H, XIAO R Y, PAN M M, et al. Risk-averse investment strategy for MEC service provisioning: a data-driven distributionally robust solution[J]. IEEE Internet of Things Journal, 2022, 9(23): 24148-24160.

[21] ZHOU F H, WU Y P, HU R Q, et al. Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems[J]. IEEE Journal on Selected Areas in Communications, 2018, 36(9): 1927-1941.

[22] WANG X M, ZHANG Y H, SHEN R J, et al. DRL-based energy-efficient resource allocation frameworks for uplink NOMA systems[J]. IEEE Internet of Things Journal, 2020, 7(8): 7279-7294.

(责任编辑:刘嘉文)