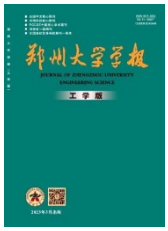


郑州大学学报(工学版)
Journal of Zhengzhou University(Engineering Science)
ISSN 1671-6833,CN 41-1339/T

《郑州大学学报(工学版)》网络首发论文

题目：基于深度强化学习的无人机边缘计算任务卸载策略
作者：王峰，马星宇，孟鹏帅，赵薇，翟伟光
DOI：10.13705/j.issn.1671-6833.2025.01.018
收稿日期：2024-07-10
网络首发日期：2024-08-22
引用格式：王峰，马星宇，孟鹏帅，赵薇，翟伟光．基于深度强化学习的无人机边缘计算任务卸载策略[J/OL]．郑州大学学报(工学版).
https://doi.org/10.13705/j.issn.1671-6833.2025.01.018

网络首发时间：2024-08-22 16:42:33
网络首发地址：https://link.cnki.net/urlid/41.1339.t.20240821.1136.001

基于深度强化学习的无人机边缘计算任务卸载策略

王　峰[1]，马星宇[2]，孟鹏帅[2]，赵　薇[2]，翟伟光[2]

（1.太原理工大学 电气与动力工程学院，山西 太原 030024；2.太原理工大学 电子信息与光学工程学院，山西 太原 030600）

摘　要：针对地理条件较为复杂的环境中存在的缺乏基础设施、任务延时高和带宽需求量大等问题，提出一种联合任务卸载和功率分配的多级移动边缘计算（MEC）系统模型。所提模型考虑将配备 MEC 的服务器部署在无人机附近提供计算服务，综合分析无人机的任务卸载、功耗和计算资源分配等问题并给出度量方法，同时考虑无人机可执行的任务类型以及任务对无人机的 CPU 和 GPU 要求，将该问题表述为混合整数非线性问题。针对该问题提出一种基于深度强化学习的计算任务卸载算法，该算法基于改进双深度 Q 学习算法，在深度强化学习中利用深度神经网络找到无人机之间的映射，从状态空间中找到潜在的模式并估计最优动作，并使用无模型的 DRL 方法，使每个无人机根据局部观察快速做出卸载决策。仿真结果表明：所提算法相比 LCGP 算法，平均卸载成本降低了 42.8%；相比 DDPG 算法，能耗减少了 16%；相比 DDQN 算法，任务执行延迟减少了 12.9%。

关键词：无人机；边缘计算；任务卸载；深度强化学习；资源分配

中图分类号：TN929.5；TP391.9　　**文献标志码**：A　　**doi**：10.13705/j.issn.1671-6833.2025.01.018

伴随人工智能（AI）与大数据领域的飞速发展，物联网（IoT）应用对低延迟以及高速率传输的需求也日益增长，给现行的云计算运行模式带来了巨大的挑战[1]。然而，物联网设备需要在地理条件较为复杂的环境或应急响应行动中运行，如林区、深海、沙尘暴频发的荒漠地带和高速公路等地方。近年来，无人机因其高度机动性和灵活性而被广泛应用于各个领域[2]。考虑到无人机的这些优势，将其用于移动方式部署边缘服务器，提供可靠高效的计算服务已成为一项具有巨大发展潜力的创新技术[3]。

传统优化方法在处理在线任务卸载难题时效果普遍不佳。包括启发式算法在内的优化算法，往往依赖于大量数值迭代获取大规模网络中的近似最优解。Ma 等[4]指出，这些固定策略难以应对随机 MEC 环境动态特性，实现最佳收益较为困难。尤其当数值方法[5]在大型应用场景中运行时，等效搜索空间的爆炸性增长和计算费用的显著增长显得格外突出。Li 等[6]的研究成果表明，仅用内点法来解决发射功率与实时需求功率的联合优化问题，存在计算复杂度较高的缺点。为应对这一难题，学者们提出基于强化学习（RL）的方法。Wang 等[7]提出一种基于强化学习的用户关联和资源分配算法，最小化智能设备的能耗、决定卸载决策，但在处理大规模问题时难以实现，因为需要对动作状态的值进行计算。为进一步提高此类方法的可拓展性并提高性能，众多研究学者引入基于深度强化学习的各类技术。例如，王丙琛等[8]通过在演员网络中加入 LSTM 预测机制，增强模型的泛化能力。Ye 等[9]采用分布式 DQN 方法优化协同工作的无人机与边缘计算服务器之间的卸载决策。Wang 等[10]基于深度确定性策略梯度的方法通过优先经验回放来最小化能源消耗。Lillicrap 等[11]则通过优先运用最优经验数据来进一步优化收益。值得注意的是，以上这些方法只关注了能源消耗而忽略了较长任务期限的影响。Mao 等[12]引入了一个天线辅助的云边混合计算框架，共同调度计算任务分配、无人机计算资源、关联控制、传输功率、带宽分配和部署位置优化，同时最小化物联网设备之间的最大计算延迟。在 Liu

收稿日期：2024-07-10；**修订日期**：2024-07-27
基金项目：山西省重点研发计划（202102150101008）；山西省留学人员科技活动项目择优资助项目（20230063）
作者简介：王峰（1975—），男，山西太原人，太原理工大学教授，博士，主要从事图像处理深度学习研究，E-mail：tyut18834573950@163.com。

等[13]的研究中,提出一种基于 DDQN 结构的新颖解决方案,旨在最大化能源供应方的质量服务表现的同时确保延长无人机电池使用寿命。Zhang 等[14]通过使用端到端 DQN 技术,以精准方式确定最优卸载比例和无人机运行路径,进一步降低无人机和能源供应方案的整体能耗水平。

针对以上问题,本文提出一种联合任务卸载和功率分配的多级 MEC 系统无人机网络系统模型,将问题表述为混合整数非线性问题。针对该问题提出一种基于深度强化学习的计算任务卸载算法,在深度强化学习中利用深度神经网络找到无人机侦察机之间的映射,在没有任何中央控制器的情况下使用本地观测做出卸载决策。通过使用多个性能指标在不同实验场景下进行仿真来评估所提方法的性能,并将结果与其他传统方案(LCGP、DDPG、DDQN)在不同参数配置下的结果进行了比较。

1 协同卸载任务的系统模型

1.1 系统模型

本文考虑一种多服务器、多用户 MEC 系统支持的无人机网络,如图 1 所示。该网络配备 MEC 服务器的无人机母舰(MUEC)和配备 MEC 服务器的地面基站(GSC)为无人机侦察机(US)提供任务卸载服务。假设 G 为 GSC 的集合,M 为 MUEC 的集合,S 为 US 的集合。每个 $S_i \in S$ 在每个时隙的开始处都有一组独立任务 K_i。另外,总操作时间以固定的时间段划分,并且在每个时隙开始时领导 MUEC 执行决策算法,并提供决策所需的数据信息。

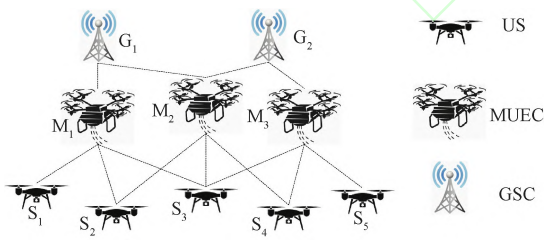


图 1 支持 MEC 的无人机网络系统模型

Figure 1 UAV network system model supporting MEC

每个任务 $j \in J_i$ 是多个参数的元组$<d_i^j, c_i^j, g_i^j, t_i^j, k_i^j>$的集合,其中,$d_i^j$ 为输入数据的大小,以位为单位,要传输到 MUEC;c_i^j 为完成任务所需 CPU 周期总数;g_i^j 为完成任务所需的 GPU FLOP 总数;t_i^j 为期望任务执行延迟时间(期望延迟时间),优先执行该任务;k_i^j 为任务类型,一个 US 可以有不同类型的任务,如人体检测、人脸或姿势识别,$k_i^j \in T'$ 且 $k_i^j \in k_i$,其中 T' 为包含无人机网络所能执行所有类型任务的通用任务集,k_i 为 S_i 所能执行的 S_i 任务集。

当在 S_i 中生成任务 $j(j \in J_i)$ 时,US 自行判断是否需要对任务进行卸载。如果 S_i 的任务 $j(j \in J_i)$ 被决定卸载到 M_i,则 US 直接将任务卸载到 MUEC。另一种选择是,如果选择任务 $j(j \in J_i, i \in S)$ 卸载到 $G_i \in G$,则任务通过与 G_i 关联的 M_i 传输到 G_i。注意,每个 M_i 最多只能与一个 G_i 关联。

由于 MUEC 也是无人机,有资源限制。因此,假设一个 MUEC 不能执行所有的任务,即每个 M_i 都有一组特定的可执行的任务 T_m($T_m \in \mathbf{Z}^+$),其中 $T_m \subseteq T'$。

此外,每个 S_i 和 M_i 分别具有发送和接收当前的能量容量 e_i 和 e_m、CPU 本地计算能力(CPU 周期/秒)f_i 和 f_m、GPU 本地计算能力(FLOP/秒)u_i 和 u_m,以及传输功率预算 p_i 和 p_m。

当 S_i 的计算资源耗尽且计算任务延长时,可以根据任务特点将任务卸载到 MUEC 或 GSC 来辅助 US。这些任务分为两种类型:计算密集型或延迟敏感型任务[15]。

如果任务对时延敏感,必须在某个时间段之前进行计算,将任务卸载到 MUEC 上,避免传输延迟。如果任务具有延迟容限性,并且需要高计算能力和更多能量,则通过具有足够计算能力的无线链路将任务从 US 卸载到 GSC。

1.2 通信模型

如果 S_i 将任务卸载到 MUEC 或 GSC 中的任何一个,这一过程产生的总延迟包括:发送和接收任务请求、决策控制信号的时间、决策时间、任务发送时间、传播延迟、任务执行时间以及处理数据后发送回 US 所需的时间。由于任务执行后的输出数据量通常远小于输入数据,且下行链路的数据速率远高于上行链路[16],本文省略了处理后数据传输所需的时间。

本文在上行链路中使用正交频分多址(OFDMA)[17]作为多址的解决方案,将总带宽划分为大小为 B 的 $|S|+|M|$ 个子带,每个 MUEC 或 US 可以单独占用一个子带。由于无人机是高空飞行,无人机与 GSC 和 MUEC 之间的信道遵循自由空间路径损失模型,视距信道在无人机通信中占主导地位。因此,S_i 与 M_i 之间的上行信道增益 w_{im} 可以表示为

$$w_{im} = \frac{\mu}{D_{im}^2}。\tag{1}$$

式中:D_{im} 为 M_i 与 M_m 之间的距离,m;μ 为参考距离 $D_0 = 1$ 米处的接收功率,W。

那么,从 S_i 到 M_i 可实现的上行比特率 r_{im} 可以

表示为

$$r_{im} = B\log_2\left(1 + \frac{w_{im}p_i^k}{\sigma}\right)。\tag{2}$$

式中:p_i^k 为 S_i 分配给任务 j 的传输功率,W;σ 为噪声功率,W。

同理,可求出 M_i 和 G_i 之间的上行比特率 r_{mg}。

1.3 计算模型

任务可以在 US 中本地计算,也可以卸载到 MUEC 或 GSC 进行计算。本节讨论两层的本地计算和边缘计算。

1.3.1 本地计算

当 S_i 在本地执行任务 j 时,任务执行时间 I_i^j 计算如下:

$$I_i^j = \frac{c_i^j}{f_i^j} + \frac{g_i^j}{u_i^j}。\tag{3}$$

式中:f_i^j 和 u_i^j 分别为 S_i 分配给任务 j 的本地计算资源(CPU 和 GPU)的总量。

执行任务 j 的能耗 O_i^j 包括 CPU 和 GPU 的处理能耗:

$$O_i^j = q_i \times c_i^j + v_i \times g_i^j。\tag{4}$$

式中:q_i 和 v_i 分别为 S_i 每个 CPU 周期和每个 GPU FLOP 所消耗的能量,J。

本文假设在执行任务时 CPU 和 GPU 之间没有并发处理。

1.3.2 任务卸载

随着任务数量的增加,由于计算资源的短缺,S_i 的计算能力被耗尽。这样,一段时间后生成的任务就会不断被丢弃。本文认为 US 可以根据任务特性将任务卸载给 MUEC 或 GSC。如前所述,任务可以分为两类:延迟敏感型任务和计算密集型任务。这种考虑的直觉是,基于深度学习的图像识别技术涉及许多阶段,这些阶段有各种类型,具有不同的大小和计算需求。因此,对于任意任务 $j \in J_i$ 表示第 i 架 US 可以执行的动作,当任务对延迟敏感时,将其卸载给 MUEC。因此,总延迟由 MUEC 的传输和计算延迟组成。

当 S_i 的任务 j 被卸载到 M_i 时,传输任务 j 的传输时间 A_{im}^j、传播延迟 U_{im} 和传输任务 j 的能耗 E_{im}^j 可由计算得到。

任务传输后在 MUEC 上执行,与本地计算中的执行时间和能耗计算类似,在 MUEC 上执行任务的执行时间和能耗分别为

$$I_{im}^j = \frac{c_i^j}{b_{im}^j} + \frac{g_i^j}{h_{im}^j};\tag{5}$$

$$R_{im}^k = q'_m \times c_i^j + v'_m \times g_i^j。\tag{6}$$

式中:b_{im}^j 和 h_{im}^j 分别为 M_i 分配给 S_i 的任务 j 的计算资源(CPU 和 GPU);q'_m 为 M_i 每个 CPU 周期的能耗,J;v'_m 为执行任务的 GPU FLOP。

1.3.3 GSC 计算

如果任务 j 被卸载到 G_i,则传输延迟包括从 S_i 到 G_i 的关联 M_i 和关联 M_i 到 G_i 的传输时间。因此,任务 j 在卸载到 GSC 时的传输时间和传播延迟分别为

$$A_{img}^j = A_{im}^j + \frac{s_i^j}{r_{mg}};\tag{7}$$

$$U_{img} = U_{im} + \frac{D_{mg}}{c}。\tag{8}$$

式中:D_{mg} 为 M_i 与 G_i 之间的距离,m。

任务在 G_i 处的执行延迟为

$$I_{img}^j = \frac{c_i^j}{z_{ig}^j} + \frac{g_i^j}{u_{ig}^j}。\tag{9}$$

式中:z_{ig}^j 和 u_{ig}^j 分别为 G_i 分配给任务 j 的计算资源(CPU 和 GPU)的总量。

通常假设 GSC 具有丰富的供电能力,因此,忽略在 GSC 执行任务所消耗的能量。将任务卸载给 GSC 时,只考虑任务传输所消耗的能量,计算公式为

$$E_{img}^j = O_{img}^j \times \frac{s_i^j}{r_{mg}}。\tag{10}$$

需要注意的是,本文假设大部分任务的计算密集型部分都在 GPU 中处理,CPU 只处理日常任务,占用的时间并不多。因此,本文只关注每个 US、MUEC 和 GS 的 GPU 资源分配,同时假设它们的 CPU 资源在所有分配的任务中平均分配。

1.4 问题公式化

为了将 MUEC 与 GSC 的关联关系纳入问题表述中,定义关联矩阵 $\boldsymbol{Q} \in B^{|M|\times|G|}$ 表示 M_i 与 G_i 的关联关系。若 M_i 与 G_i 相关联,则元素 $Q(m,g)=1$;否则,$Q(m,g)=0$。同样,本文将 $\boldsymbol{Z}$ 定义为 US 与 MUEC 之间的矩阵,其中 $Z(i,m)=1$,表示 S_i 与 M_i 之间存在通信链路;$Z(i,m)=0$,表示两者之间不存在通信链路。

问题表述中使用的优化变量定义如下。当任务 j 由 S_i 卸载到 M_i 时,$x_{im}^j=1$;其他情况时 $x_{im}^j=0$。当任务 j 通过 MUEC 由 S_i 卸载到 G_i 时,$x_{img}^j=1$;其他情况时 $x_{img}^j=0$。若 x_{img}^j 和 x_{im}^j 均等于 0,则表示任务在本地处理。每个 S_i 和 M_i 的总能耗分别为

$$O_i = \sum_k \Big(\sum_{m\in M} E_{im}^j(x_{im}^j + x_{img}^j) + (1-(x_{im}^j + x_{img}^j))O_i^j\Big)。\tag{11}$$

$$O_m = \sum_{i\in S}\sum_{k\in K_i}((\tau_m \times s_i^j + R_{im}^j)x_{im}^j + \sum_{g\in G}(\tau_m \times s_i^j + E_{img}^j)x_{img}^j)。 \tag{12}$$

式中:τ_m 为 M_i 接收每个任务所消耗的能量,J。

S_i 的每个任务 j 所需的总执行时间为

$$T_i^j = ((A_{im}^j + U_{im} + I_{im}^j)x_{im}^j + (A_{img}^j + U_{img} + I_{img}^j)x_{img}^j) + (1 - (x_{im}^j + x_{img}^j))I_i^j。 \tag{13}$$

主要问题表述如下:

$$\min\ w_1(\chi) + w_2\left(\sum_{i\in S}\sum_{j\in J_i}\frac{T_i^j}{d_i^j}\right)。 \tag{14}$$

式中:χ 为 $i\in S$ 和 $m\in M\left\{\frac{O_i}{\sum_{i\in S}e_i},\frac{O_m}{\sum_{m\in M}e_m}\right\}$ 的最大值。

为了解决式(14)中的问题,需要在每个时隙的时延约束下找到最优的卸载动作 x_{im}^j 和 x_{img}^j。应该注意的是,x_{im}^j 和 x_{img}^j 是整数决策变量,代表了公式化问题中的决策,以降低总体成本。该系统需要广泛的网络信息(如无人机的任务信息和剩余计算能力)来根据当前状态做出最优卸载决策。在这种情况下,无人机需要根据在时变环境下事先未知的时隙内收集到的到达任务信息来确定卸载决策。

1.5 RL 框架

如上所述,由于以下问题,使用传统优化技术很难解决优化问题:动态环境下[18]卸载决策是根据任务的具体需求确定的;对特定的任务,需要重新设计并运行解决方案;随着无人机数量的增加,状态和操作空间呈指数级增长,非凸问题将变得非常难以解决,收敛时间变长,卸载决策变得具有挑战性[19]。

为了解决传统优化技术中的上述缺陷,本文建议使用一种无模型方法 DRL。使每个 US 旨在根据局部观察快速做出卸载决策,减少 US 之间的信息交换。在传统强化学习中,问题通常被建模为马尔可夫决策过程(MDP)。在 RL 框架下,提供了状态空间、动作空间和所述问题的奖励函数的定义。

(1)状态空间 $m_{j,t}$。为了做出卸载决策,网络中的每个代理都有一组输入指标,在做出卸载决策时要考虑这些指标。令 $m_{j,t} = \{d,c,f,d^t\}$ 表示状态空间,其中 d 为任务大小;c 为完成任务所需的周期;f 为 MUEC 的计算能力;d^t 为任务类型。

(2)行动空间 $a_{j,t}$。每个 US 在生成任务后选择一个特定的动作。US 可以从本地信息中选择一个动作。本文考虑二进制卸载,可以在 MUEC 上执行,也可以卸载到地面边缘服务器。

(3)奖励功能。每个 US 的目标是使总奖励最大化。假设每个 US 都有相同的奖励数。基于以上讨论,奖励函数定义如下:

$$Z_{j,k} = -x_{im}^j x_{img}^j\frac{O_i + O_m}{\max(O_i + O_m)} - x_{im}^j(1 - x_{img}) \times \frac{E_{img}^j}{\max E_{img}^j} - (1 - x_{im}^j)\frac{O_i^j}{\max O_i^j}。 \tag{15}$$

成本越高,回报越少,反之亦然。因此,对于每个代理:

$$Z_j = \sum_{k=1}^{\beta} Z_{j,k}。 \tag{16}$$

每个代理的目标都是最大化这一目标,强调产生更好回报价值的行为。由于行为选择取决于智能体交互的网络动力学的独特特征,因此直接根据获得的效用来定义奖励函数会影响学习过程。这种考虑的直觉是,奖励的增加意味着改进,因此应该强调特定的行动[20]。因此,本文将即时时间步长中的奖励值之差视为奖励。基于以上讨论,奖励值定义如下:

$$u_t^j = \begin{cases} p, & 若\quad Z_{j,t} - Z_{j,t-1} < 0; \\ q, & 若\quad Z_{j,t} - Z_{j,t-1} > 0; \\ 0, & 其他。 \end{cases} \tag{17}$$

式中:$Z_{j,t}$ 为在第 t 个时段获得的累计奖励,当两个即时时间步长中的奖励差为负值时,是一种改进,此处 p 是一个正值;而即时时间步长获得奖励之间的正差异产生的 q,是一个负奖励。

为简单起见,本文认为每个代理共享相同的奖励。

2 基于深度强化学习的计算任务卸载

2.1 DRLCO 算法

基于前面的讨论,本节提出一种改进的分布式多智能体计算任务卸载(DRLCO)算法,如图 2 所示。此算法融合了神经网络和 Q-learning,实现最佳卸载决策。每当需要在本地计算或在 MEC 服务器上处理新的任务时,每个 US 在本地运行 DRLCO 算法,并以分布式的方式学习卸载策略,根据其本地知识选择动作,并立即获得特定动作的奖励。利用神经网络的感知特性与强化学习的决策能力,确定动态环境下最佳卸载策略。考虑到被计算的任务类型及数量的不确定性,将 DRLCO 算法设计为无模型导向的强化学习算法。

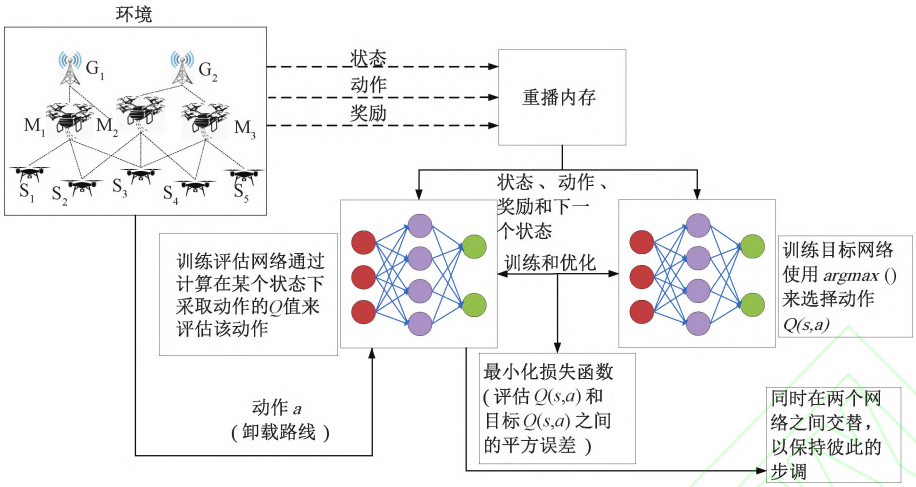


图 2　DRLCO 方案框图

Figure 2　Block diagram of DRLCO scheme

本文使用两个相同的神经网络用于构建 DRLCO 的整体结构,称之为目标网络 φ_j^{t} 和评估网络 φ_j^{e},其 Q 函数表示为 $Q_j^{\mathrm{t}}(m_{j,t},a_{j,t}|\varphi_j^{\mathrm{t}})$ 和 $Q_j^{\mathrm{e}}(m_{j,t},a_{j,t}|\varphi_j^{\mathrm{e}})$。为了提升训练效率和早期收敛,使用经验回放 $O_k(t)$ 存储过往样本。定义经验样本为 $\{m_{j,t},a_{j,t},u_{j,t},m'_{j,t}\}$。

在 DRLCO 中,两个神经网络是为了确定最佳动作。典型的 DQN 算法[21]中,max 算子用来选择和评估导致结果值被高估的动作。为此,本方案通过两个相同的网络来实现分离选择和评估行动的任务,从而量化评估每个贪婪决策的结果。目标网络会优先选择能产生最大 Q 值的动作,评估网络将通过计算在当前状态下执行该操作的 Q 值,对先前选定的行为进行评估。因此,使用两个神经网络均匀地评估策略的值,可以推导出目标网络的期望累计奖励为

$$Rew_j = u_{j,t} + u_k Q_j^{\mathrm{t}}(m'_{j,t},\max Q(j))。\quad (18)$$

式中:u_k 为控制进一步奖励的折扣系数;$Q(j) = argmax\ Q_j^{\mathrm{e}}(m'_{j,t},a_{j,t}|\varphi_j^{\mathrm{e}})|\varphi_j^{\mathrm{t}}$。

因此,$Q_j^{\mathrm{t}}(m_{j,t},a_{j,t}|\varphi_j^{\mathrm{t}})$ 和 $Q_j^{\mathrm{e}}(m_{j,t},a_{j,t}|\varphi_j^{\mathrm{e}})$ 之间的损失公式为

$$Loss_j(Q_j^{\mathrm{t}},Q_j^{\mathrm{e}}) = E(m_{j,t},a_{j,t},u_{j,t},m'_{j,t}) \sim O_k \times (Rew_j - Q_j^{\mathrm{e}}(m_{j,t},a_{j,t}|\varphi_j^{\mathrm{e}}))^2。\quad (19)$$

为了保证训练性能的稳定性,本文采取两个网络交替使用的方法。使用评估网络,用逐步更新的方式调整目标网络参数直至达到收敛结果。因此,每个 US 都能通过自己的信息学习最佳卸载策略。由于每个 US 只能观察到局部信息,因此采用相同的奖励来整合全局,实现更好的收敛效果。所有 US 的目标都是最大化整体奖励。

算法 1 具体步骤如下所示。

算法 1　基于改进的 DRLCO 算法

输入:状态空间 $< d,c,f,d^t >$;

输出:给定输入的最佳卸载位置。

① 初始化 $U_i \in S$ 的目标网络和评估网络的参数和重放内存 χ_j;

② while $episode = 1$ to $episode_{\max}$ do

③ 对 U_i 的整个环境进行重置;

④ while $j = 1$ to M do

⑤ USs 作用于动态环境;

⑥ while $t = 1$ to T do

⑦ U_i 观察状态 $m_{j,t}$ 的参数,其中有输入数据大小 d、完成任务所需的周期 c、MUEC 的计算能力 f、任务类型 d^t;

⑧ U_i 根据 ε-greedy 策略选择动作 $a_{j,t}$,将任务卸载给 MUEC 或者 GSC;

⑨ U_i 获得奖励 $u_{j,t}$ 和下一个状态 $m'_{j,t}$;

⑩ 当重放内存没满时,将样本 $\{m_{j,t},a_{j,t},u_{j,t},m'_{j,t}\}$ 添加到其中;

⑪ if 经验回放 $O_k(t)$ 中样本足够:

⑫ 从经验回放 $O_k(t)$ 中选择一组小批量;

⑬ 用式(18)计算累计奖励,用式(19)计算损失函数;

⑭ 更新评估网络 φ_j^{e},将参数从评估网络交替到目标网络中 $(\varphi_j^{\mathrm{e}} \rightarrow \varphi_j^{\mathrm{t}})$;

⑮ end if

⑯ end while

⑰ end while

⑱ end while

⑲ return 给定输入的卸载位置

本文所述 DRLCO 算法包括两大环节:一是收集网络环境中的数据样本;二是依据这些样本进行

智能体训练。在初始化部分,目标网络参数与初始权重一同被确定(第 1 行);之后设定集数,USs 开始与网络环境互动,采集所需数据(第 2~10 行)。在收集过程中,US 需要监控状态、决策行动、获取奖励,以及更新状态(第 7~9 行)。使用重播记忆存储已获取的经验作为后续训练之用。训练期则会从重播记忆中随机选取小批量样本用于智能体训练及损失计算(第 13 行)。接着,根据损失值调整评估网络,并让两个网络交替使用参数维持训练平衡(第 14 行)。本文提出的算法流程如图 3 所示。

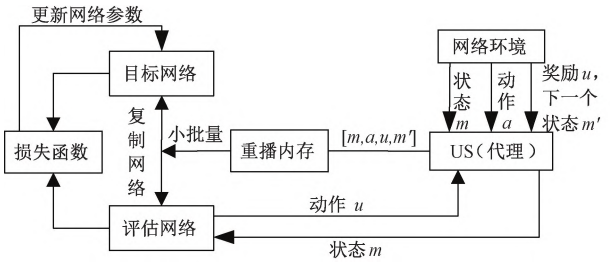


图 3　卸载策略流程图

Figure 3　Offload policy flow chart

3　仿真与数据分析

3.1　仿真设置

在仿真设置中,本文考虑了具有多个 GSC、MUEC 和 US 的 MEC 无人机网络。假设从 US 到 MUEC 和 MUEC 到 GSC 的 1 m 参考距离处的噪声方差和信道增益相同,每个 US 的任务以 λ 速率遵循泊松分布过程到达。本文设置了每个 US 的神经网络来模拟 DRLCO 算法。DRLCO 算法的神经网络由一个输入层、一个输出层和两个全连接层组成。两个隐藏层的大小分别为 350 和 400,并用 ReLU 作为激活函数,使用 Adam optimizer 来优化损失函数。在训练阶段,将最大迭代次数设置为 2 000,回访内存的大小设为 50 000。

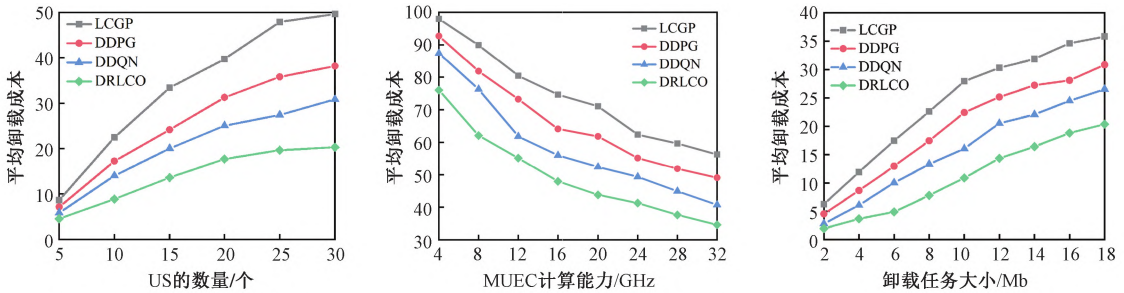
为了凸显未来奖赏的重要性,本文将折扣因子设置为 0.9。由于 $\alpha=0.001$ 时产生了更高的奖励和稳定的训练效果,在实际模拟环境中,采取了同样的学习率设定值。对于目标函数中的权重,采取等值设定方式,以此来保障对总成本衡量的公平性与全面性。这些权重可以依据任务特性(即能耗或延迟时间)进行灵活调整,从而更好地应对不同的任务需求。在实验中,计算任务大小在 2~18 Mb 随机变化。US、MUEC 和 GSC 的配置分别为 Intel Core i5-9300 H 和 1.9 TFLOPS (Nvidia GTX 1050)、Intel Core i5-12400 和 20 TFLOPS (Nvidia GeForce RTX 3070) 以及 Intel Core i7-12700 和 40 TFLOPS (Nvidia GeForce RTX 3080 Ti)。

3.2　模拟结果

将所提 DRLCO 算法与其他 3 种基准策略在不同参数下的仿真结果进行了比较。其中,3 种基准策略分别为本地计算贪心策略(local computing greedy policy, LCGP)、深度确定性策略梯度策略(deep deterministic policy gradient, DDPG)和双深度 Q 网络策略(double deep Q-network policy, DDQN)。

图 4 展示了 DRLCO 算法与其它 3 种基准算法在 US 数量、MUEC 计算能力和卸载任务大小不同时,分别对平均成本的影响。相比 LCGP 算法、DDPG 算法和 DDQN 算法,所提算法的平均卸载成本分别降低了 42.8%、33.2%和 20.7%。从图 4(a)可以看出,随着 US 数量的增加,需要计算的任务会更多,数据传输和任务执行延迟也会增加,进而增加整体卸载成本。尽管平均卸载成本随着 US 的数量增多而变多,但 DRLCO 算法相较于 DDQN 算法具有更好的成本管控能力,能够有效应对可伸缩性问题,在无人机数量配置方面与其他基准方案竞争优势更为明显。从图 4(b)可以看出,随着 MUEC 计算能力的增加,平均成本明显降低。因为 MUEC 有足够的计算资源时,为最大限度减少总成本,会将计算密集型任务卸载到 MUEC 上进行处理,这大大减少了任务传输和处理延迟。从图 4(c)可以看出,当从 US 上卸载的任务规模增加时,总成本也会增加。这是因为 MUEC 执行大型复杂任务时,需要消耗更多的 CPU 周期,进而导致处理时间显著增加。DRLCO 方案能够根据其他任务的特性做出最佳选择,将大型任务分配给具备适当计算能力的节点。因此,DRLCO 方案可以获得更高的回报,最大程度地优化整体成本。

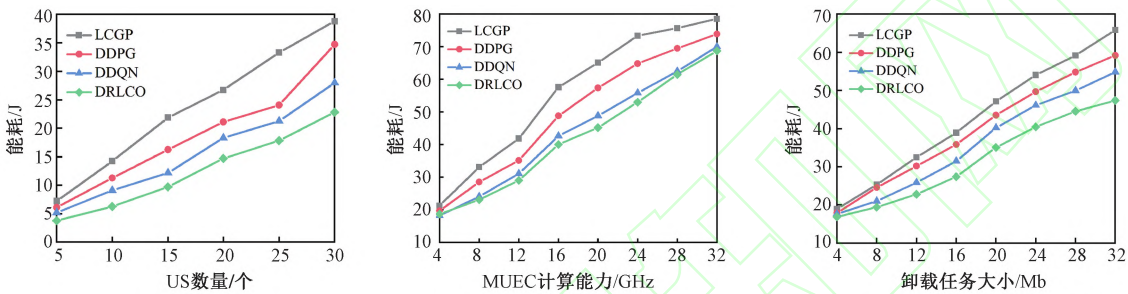
图 5 展示了 DRLCO 算法与其它 3 种基准算法在 US 数量、MUEC 计算能力和卸载任务大小不同时,分别对能耗的影响。相比 LCGP 算法、DDPG 算法和 DDQN 算法,所提算法的能量消耗分别降低了 28.5%、16%和 8.4%。从图 5(a)中可以看出,当 US 数量增加时,总能耗增加,这是因为 US 越多时,需要处理的任务也会越多,网络中的能耗增加。从图 5(b)可以看出,MUEC 的计算能力增加时,执行任务的时间减少,能耗增加。这是因为由于 MUEC 计算能力增加时,US 更倾向于将更多的计算密集型任务卸载到边缘计算节点上,与将任务卸载到 GSC 端相比,可以有效减少任务传输延迟时间。从图 5(c)可以看出,增加任务大小也会增加能耗。当任务规模较大时,US 需要更多的发射功率来传输任务,处



(a) 无人机数量对平均成本的影响　(b) MUEC计算能力对平均成本的影响　(c) 卸载任务大小对平均成本的影响

图 4　平均卸载成本

Figure 4　Average offloading cost



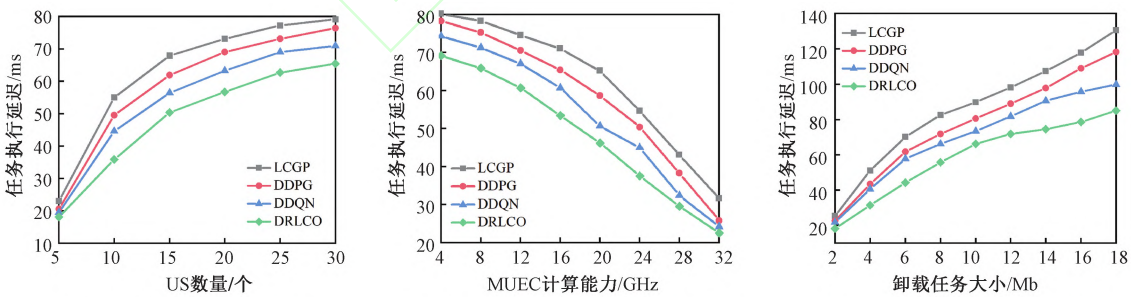
(a) US数量对能耗的影响　(b) MUEC计算能力对能耗的影响　(c) 卸载任务大小对能耗的影响

图 5　能耗

Figure 5　Energy consumption

理设备也需要更多 CPU 周期来完成任务。

图 6 展示了 DRLCO 算法与其它 3 种基准算法在 US 数量、MUEC 计算能力和卸载任务大小不同时,分别对任务执行延迟的影响。相比 LCGP 算法、DDPG 算法和 DDQN 算法,所提算法的执行任务延迟分别降低了 27.1%、20.4%和 12.9%。如图 6(b)所示,可以观察到,增加 MUEC 的计算能力时,执行任务延迟就会相应减少。这是因为当计算资源足够时,US 更有可能将复杂的任务分配给 MUEC 执行,以降低任务传输过程中的延迟。从图 6(c)可以看出,任务执行延迟会随着任务大小的增加而增加。由于大型任务需要更多的 CPU 周期来处理,计算时间就更长;当大型任务从 US 卸载到 MUEC 或者 GSC 时,也会产生更多的传输延迟。



(a) US数量对任务执行延迟的影响　(b) MUEC计算能力对任务执行延迟的影响　(c) 卸载任务大小对任务执行延迟的影响

图 6　任务执行延迟

Figure 6　Task execution delay

4　结论

本文针对地理条件较为复杂的环境或应急响应行动中存在的缺乏基础设施、任务延时高和带宽需求量大等问题,设计了一种联合计算任务卸载和功率分配的多级移动边缘计算系统模型,并提出一个 DRLCO 方案的多智能体强化学习策略,利用深度神经网络找到无人机侦察机之间的映射,并使用无模型的 DRL 方法。通过仿真实验将本文算法与基准算法从多方面进行对比,结果证明:相比于 LCGP 算法,所提算法的平均卸载成本降低了 42.8%;相比于 DDPG 算法,所提算法能耗减少了 16%;相比于 DDQN 算法,所提算法任务执行延迟减少了 12.9%。

参考文献:

[1] PANWAR P, SHABAZ M, NAZIR S, et al. Generic edge computing system for optimization and computation offloading of unmanned aerial vehicle[J]. Computers and Electrical Engineering, 2023, 109: 108779.

[2] 陈新颖, 盛敏, 李博, 等. 面向 6G 的无人机通信综述[J]. 电子与信息学报, 2022, 44(3): 781-789.
CHEN X Y, SHENG M, LI B, et al. Survey on unmanned aerial vehicle communications for 6G[J]. Journal of Electronics & Information Technology, 2022, 44(3): 781-789.

[3] NGUYEN V, KHANH T T, VAN NAM P, et al. Towards flying mobile edge computing[C]//2020 International Conference on Information Networking (ICOIN). Piscataway: IEEE, 2020: 723-725.

[4] MA M L, GONG C Y, WU L T, et al. FLIRRAS: fast learning with integrated reward and reduced action space for online multitask offloading[J]. IEEE Internet of Things Journal, 2023, 10(6): 5406-5417.

[5] AHANI G, YUAN D. BS-assisted task offloading for D2D networks with presence of user mobility[C]//2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring). Piscataway: IEEE, 2019: 1-5.

[6] LI N N, HAO W M, ZHOU F H, et al. Smart grid enabled computation offloading and resource allocation for SWIPT-based MEC system[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2022, 69(8): 3610-3614.

[7] WANG L, HUANG P Q, WANG K Z, et al. RL-based user association and resource allocation for multi-UAV enabled MEC[C]//2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC). Piscatawy: IEEE, 2019: 741-746.

[8] 王丙琛, 司怀伟, 谭国真. 基于深度强化学习的自动驾驶车控制算法研究[J]. 郑州大学学报(工学版), 2020, 41(4): 41-45, 80.
WANG B C, SI H W, TAN G Z. Research on autopilot control algorithm based on deep reinforcement learning [J]. Journal of Zhengzhou University (Engineering Science), 2020, 41(4): 41-45, 80.

[9] YE Y T, WEI W S, GENG D Q, et al. Dynamic coordination in UAV swarm assisted MEC via decentralized deep reinforcement learning[C]//2020 International Conference on Wireless Communications and Signal Processing (WCSP). Piscataway: IEEE, 2020: 1064-1069.

[10] 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.

[11] LILLICRAP T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning[EB/OL]. (2019-07-05)[2024-06-01]. https://arxiv.org/abs/1509.02971.

[12] MAO S, HE S F, WU J S. Joint UAV position optimization and resource scheduling in space-air-ground integrated networks with mixed cloud-edge computing[J]. IEEE Systems Journal, 2021, 15(3): 3992-4002.

[13] LIU Q, SHI L, SUN L L, et al. Path planning for UAV-mounted mobile edge computing with deep reinforcement learning[J]. IEEE Transactions on Vehicular Technology, 2020, 69(5): 5723-5728.

[14] ZHANG L, ZHANG Z Y, MIN L, et al. Task offloading and trajectory control for UAV-assisted mobile edge computing using deep reinforcement learning[J]. IEEE Access, 2021, 9: 53708-53719.

[15] WANG Y T, CHEN W W, LUAN T H, et al. Task offloading for post-disaster rescue in unmanned aerial vehicles networks[J]. IEEE/ACM Transactions on Networking, 2022, 30(4): 1525-1539.

[16] 李斌, 刘文帅, 费泽松. 面向空天地异构网络的边缘计算部分任务卸载策略[J]. 电子与信息学报, 2022, 44(9): 3091-3098.
LI B, LIU W S, FEI Z S. Partial computation offloading for mobile edge computing in space-air-ground integrated network[J]. Journal of Electronics & Information Technology, 2022, 44(9): 3091-3098.

[17] HUO Y, LIU Q Y, GAO Q H, et al. Joint task offloading and resource allocation for secure OFDMA-based mobile edge computing systems[J]. Ad Hoc Networks, 2024, 153: 103342.

[18] XU Y C, CHEN L L, LU Z H, et al. An adaptive mechanism for dynamically collaborative computing power and task scheduling in edge environment[J]. IEEE Internet of Things Journal, 2023, 10(4): 3118-3129.

[19] 李斌. 基于多智能体强化学习的多无人机边缘计算任务卸载[J]. 无线电工程, 2023, 53(12): 2731-2740.
LI B. Multi-agent reinforcement learning-based task offloading for multi-UAV edge computing[J]. Radio Engineering, 2023, 53(12): 2731-2740.

[20] LI W, ZHOU F, CHOWDHURY K R, et al. QTCP: adaptive congestion control with reinforcement learning [J]. IEEE Transactions on Network Science and Engineering, 2019, 6(3): 445-458.

[21] WANG P, NI W L. An enhanced dueling double deep Q-network with convolutional block attention module for traffic signal optimization in deep reinforcement learning[J]. IEEE Access, 2024, 12: 44224-44232.

Task Offloading Strategy of UAV Edge Computing Based on Deep Reinforcement Learning

WANG Feng[1], MA Xingyu[2], MENG Pengshuai[2], ZHAO Wei[2], ZHAI Weiguang[2]

(1. College of Electrical and Power Engineering, Taiyuan University of Technology, Taiyuan 030024, China; 2. College of Electronic Information and Optical Engineering, Taiyuan University of Technology, Taiyuan 030600, China)

Abstract: Aiming at the problems such as lack of infrastructure, high task delay and high bandwidth demand in complex geographical conditions, a multi-stage mobile edge computing system model which combined computing offloading and power distribution was proposed. In the proposed model, a server equipped with MEC was deployed near the UAV to provide computing services, and the problems such as task offloading, power consumption and computing resource allocation of the UAV were comprehensively analyzed and the measurement methods were given. At the same time, the types of tasks that the UAV could perform and the requirements of the CPU and GPU on the UAV were considered. The problem was expressed as a mixed integer nonlinear problem. A task computing offloading algorithm based on deep reinforcement learning was proposed to solve this problem. Based on the improved double deep Q learning algorithm, the algorithm used deep neural network to find the mapping between UAVs in deep reinforcement learning, finding potential patterns from the state space and estimating the optimal action, and used model-free DRL method to enable each UAV to make quick offloading decisions based on local observations. Simulation results showed that the proposed algorithm reduced the average offloading cost by 42.8% compared with LCGP algorithm. Compared with DDPG algorithm, the energy consumption was reduced by 16%. Compared with DDQN algorithm, the task execution delay was reduced by 12.9%.

Keywords: UAV; edge computing; task offloading; deep reinforcement learning; resource allocation