光电技术及应用

DOI：10.16818/j.issn1001－5868.2022030601

MEC 系统中基于多智能体深度增强学习的协作式任务卸载和资源分配

夏炳森*，唐元春，李　翠

（国网福建省电力有限公司经济技术研究院，福州 350000）

摘　要： 移动边缘计算(MEC)通过将计算任务卸载到 MEC 服务器上，在缓解智能移动设备计算负载的同时，可以降低服务时延。然而目前在 MEC 系统中，关于任务卸载和资源分配仍然存在以下问题：1)边缘节点间缺乏协作；2)计算任务到达与实际环境中动态变化的特征不匹配；3)协作式任务卸载和资源分配动态联合优化问题。为解决上述问题，文章在协作式 MEC 架构的基础上，提出了一种基于多智能体的深度确定性策略梯度算法(MADDPG)的任务卸载和资源分配算法，最小化系统中所有用户的长期平均成本。仿真结果表明，该算法可以有效降低系统的时延及能耗。

关键词： MEC；协作任务卸载；资源分配；多智能体

中图分类号： TN925.5　**文章编号：** 1001－5868(2022)03－0602－07

Cooperative Offloading and Resource Allocation with Multi-Agent Deep Reinforcement Learning in MEC

XIA Bingsen, TANG Yuanchun, LI Cui

(State Grid Fujian Economic Research Institue, Fuzhou 350000, CHN)

Abstract: Mobile edge computing (MEC) has emerged as a key technology to alleviate the computation workloads and decrease service latency for computation-intensive applications by offloading the tasks to MEC servers. However, the existing computation offloading and resource allocation studies present some several problems: poor collaboration between edge servers; mismatch between the computational task arrival and the dynamic characteristics in the real environment; and the dynamic joint optimization problem of the collaborative task unload and resource allocation. To solve such issues, based on the collaborative MEC framework, a multi-agent based deep deterministic policy gradient (MADDPG) is proposed for task unloading and resource allocation to minimize the overall long-term average cost. Simulation results reveal that the proposed scheme can reduce the delay and energy consumption.

Key words: mobile edge computing; cooperative computation offloading; resource allocation; multi-agent deep reinforcement learning

0　引言

智能移动设备的快速增长加速了诸如虚拟/增强现实、人脸识别和自动驾驶等计算密集型和时延敏感性应用的发展[1]。移动云计算，通过将计算密集型应用从智能 IoT 设备和移动终端处卸载至远端资源丰富的云服务器，能够很好地解决计算能力受限制的难题[1]。然而，云服务器与智能 IoT 设备之间远距离的特性，会带来传输时延的增加，还会对回传链路容量带来相当大的挑战。这些复杂的应用

收稿日期：2022－03－06.

* **通信作者：**夏炳森　**E-mail：** bingsxia@163.com

需要更高的计算能力、存储容量和电池寿命[2]。为了进一步解决该问题，移动边缘计算(Mobile Edge Computing，MEC)应运而生。MEC 将计算资源部署在更靠近无线接入网侧的边缘服务器中，使得智能 IoT 设备可以将其计算密集型任务卸载至更靠近自己的边缘服务器中。欧洲电信标准化协会(European Telecommunications Standardization Institute，ETSI)于 2014 年首次提出 MEC 的概念，以解决云计算模型中存在的问题[3]。与移动云计算相比，MEC 具有以下优势：实现更低的延迟、为移动设备节约能源、支持上下文感知计算、增强移动应用程序的隐私和安全性[4]。

MEC 系统在任务卸载和资源分配方面也面临着诸多挑战。任务卸载的一个关键问题是选择合适的边缘服务器，这不仅可以帮助卸载任务，而且可以保证服务需求，特别是在多用户和多边缘服务器 MEC 系统中。计算卸载和资源分配的联合优化可以显著缩短任务响应时间并节约能耗，研究人员已经进行了大量的研究工作[5-7]。文献[5]提出了一种用于物联网边缘计算系统的联合计算卸载和多用户调度算法，以最小化随机流量到达下的延迟和功耗的长期平均加权和。在文献[6]中，作者研究了支持 MEC 的密集 C-RAN 中的任务卸载和资源分配问题，旨在优化网络能量效率。文献[7]针对多接入边缘计算中的低延迟物联网服务，提出了一种动态卸载和调度算法，以最大化允许的任务数量。

然而，以上研究没有考虑到服务器之间的合作。由于服务在时间和空间上的动态不均匀分布，导致不同边缘节点之间的工作负载不同。在文献[8]中，作者利用设备、边缘节点和云服务器之间的垂直协作以及边缘节点之间的水平协作来考虑合作的三层计算网络，并联合优化卸载决策和计算资源分配，在设备电池容量有限的情况下最小化平均任务持续时间。但是文献[8]中没有考虑边缘节点的能量消耗。文献[9]提出了一种 MEC 服务器计算任务卸载的在线优化策略，该策略采用睡眠控制方案，以最小化 MEC 服务器网络的长期能耗。然而，文献[9]只关注边缘节点的能量消耗，没有考虑资源分配。为了解决安全和隐私问题，文献[10]为支持区块链的 MEC 系统提出了一个协作计算卸载和资源分配框架，通过联合优化卸载决策、功率分配，以及块大小和块间隔。但中继节点用于实现协同计算卸载，并假设每个基站仅服务于一个移动设备。与上述工作不同的是，本文提出的方法实现了边缘服务器之间的协作，可以显著降低任务执行延迟。

针对上述研究中存在的问题，本文在多 IoT 设备多边缘节点的架构下，研究了基于随机任务模型的协作任务卸载和资源分配策略。本文考虑了边缘节点之间的协同计算卸载，这不仅可以提高边缘节点之间的计算资源，而且可以减少执行任务时的计算延迟。本文使用部分卸载任务模型，以充分利用物联网设备和边缘节点中的计算资源。为了更好地反映环境动态变化的特征，本文假设每个 IoT 设备上任务的到达服从泊松分布，并将任务卸载决策、计算资源分配决策及通信资源分配决策的联合优化问题建模为 MDP 问题。为了求解上述动态优化问题，本文拟采用基于多智能体增强学习协作任务卸载和资源分配算法。为了解决多智能体环境中的优化和不稳定问题，本文提出了一种基于多智能体深度确定性策略梯度(Multi-Agent Deep Deterministic Policy Gradient，MADDPG)的协同计算卸载和资源分配算法。通过集中培训，每个代理可以相互获取信息。

1 系统模型

1.1 系统模型

本文所提的协作式 MEC 系统模型如图 1 所示，考虑有 M 个基站，集合为 $\boldsymbol{M}=\{1,2,\cdots,M\}$，每个基站都有一个相对应的、具有计算和存储能力的边缘服务器，多个相邻的基站能够通过有线链路相互通信。假定网络中有 N 个 IoT 设备，定义 IoT 用户设备集合为 $\boldsymbol{N}=\{1,2,\cdots,N\}$，每个 IoT 设备在接入网络时连接到距离最近的基站，定义由基站 m 服务的用户集合为 $\boldsymbol{U}_m$，$\forall m \in \boldsymbol{M}$，因此 $\sum_{m=1}^{M}|\boldsymbol{U}_m|=N, \forall m \in \boldsymbol{M}$。不失一般性，本文定义设备 n 由基站 m 为其提供服务，即 $\forall n \in \boldsymbol{U}_m$。

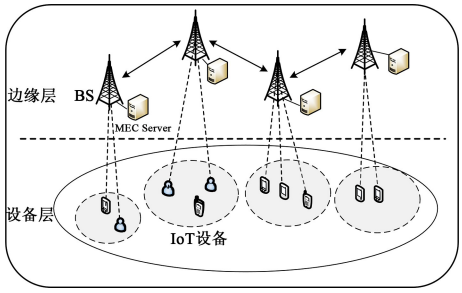

图 1 系统模型

1.2 任务模型

随机任务模型更符合环境动态变化的特性，因此本文假设任务的到达服从泊松分布，定义 $\lambda_n(t)$ 为 t 时隙设备 n 的任务到达率，并且不同任务之间是相互独立的。定义 $\Omega_n(t)=D_n(t),X_n(t),\tau_n(t)$ 为设备 n 在时隙 t 产生的任务，其中 $D_n(t)$ 表示 t 时刻任务 $\Omega_n(t)$ 的数据大小，单位是 bit，$X_n(t)$ 表示处理任务需要的计算资源强度(CPU 周期数/bit)，$\tau_n(t)$ 表示该任务执行的时延约束。

为了更好地利用 IoT 设备和边缘服务器上的计算资源，本文采用部分卸载策略。假定在每个时隙内，IoT 用户设备将计算任务部分卸载到边缘服务器端执行，剩余部分则在本地进行处理。由于不同边缘服务器服务的用户数、用户的任务达到率等不同，从而导致业务时空分布不均，使得不同节点间具有差异化的通信负载和计算负载。计算能力和计算负载间的不匹配将使得某些边缘节点特别繁忙，产生较大的计算时延，因此，本文考虑通过边缘节点间的协作来实现计算负载均衡，从整体上提高系统的资源利用率，降低任务执行时延。定义协作调度因子 $\alpha_n^{m'}(t)=\{0,1\},\forall m'\in \boldsymbol{M}$，其中 m' 表示用于协作的边缘服务器，$\alpha_n^{m'}(t)=0$ 表示用户需要卸载的任务不需要进行协作处理。反之，$\alpha_n^{m'}(t)=1$ 表示任务需要由基站 m 转移至基站 m' 进行协作处理，并且满足如下约束：$\sum_{m'=1}^{M}\alpha_n^{m'}(t)\leqslant 1$。

定义 $\rho_n(t)\in[0,1]$ 表示 IoT 设备在 n 时隙 t 任务的卸载率，当 $\rho_n(t)=0$ 时，表示用户设备 n 产生的所有任务在本地执行计算，当 $\rho_n(t)=1$ 时，表示用户设备 n 将所有任务通过接入点 m 卸载至边缘服务器执行计算。

1.3 通信模型

采用基于 OFDMA 的无线传输机制，因此不存在用户间的干扰。假设不同的基站间采用频分复用，因此也不考虑同频干扰。定义子信道集合为 $\boldsymbol{W}=1,2,\cdots,W$，且每个子载波带宽为 B(Hz)。定义子信道分配因子为 $\beta_n^w(t)=\{0,1\},\forall w\in \boldsymbol{W}$，其中 $\beta_n^w(t)=1$ 表示在时隙 t 子信道 w 分配给用户 n，且 $\sum_{w=1}^{W}\beta_n^w(t)=1,\forall w\in \boldsymbol{W}$。定义时隙 t 用户 n 的发射功率为 $p_n(t)\in[0,P^{\max}]$，其中 $P^{\max}$ 表示其最大发射功率，σ^2 表示信道噪声功率，$g_n(t)$ 为信道增益，则传输速率为

$$R_n(t)=B\sum_{w=1}^{W}\beta_n^w(t)\log_2\left(1+\frac{p_n(t)g_n(t)}{\sigma^2}\right) \quad (1)$$

1.4 计算模型

1.4.1 本地计算模型

本文假定每个 IoT 设备的计算能力是固定的，定义为 $f_n^l(t)$，可以在用户进行本地计算时建模为一个平均到达率为 $(1-\theta_{U_n,m}(t))\lambda_{U_n,m}(t)$ 的 $M/M/1$ 队列。因此本地计算的平均时延包含队列时延和服务时延，为

$$T_n^l(t)=\frac{1}{\mu_n^l(t)-(1-\rho_n(t))\lambda_n(t)} \quad (2)$$

其中，$\mu_n^l(t)$ 为用户 n 的本地 CPU 的服务速率，其大小为 $\mu_n^l(t)=f_n^l(t)/X_n(t)$。

本地计算能耗表达式为[11]

$$E_n^l(t)=\kappa(1-\rho_n(t))D_n(t)X_n(t)(f_n^l(t))^2 \quad (3)$$

其中，κ 是一个与硬件有关的有效电容常数[12]。

1.4.2 卸载计算模型

由上述可知，IoT 用户设备可以将时隙 t 产生的计算任务卸载到 ECS 进行计算，本文定义时隙 t 时对于 IoT 用户 U_n 产生的任务的卸载比例为 $\rho_n(t)$，因此进行卸载的数据服从一个到达率为 $\rho_n(t)\lambda_n(t)$ 的泊松分布。

由于本文只考虑系统内用户设备的 QoE，因此本文对用户 U_n 卸载到 ECS 的时延和能耗只考虑用户设备 U_n 的传输时延、计算时延、在每个边缘服务器中的排队时延、服务器之间的迁移时延及用户设备的传输能耗。

IoT 设备 U_n 卸载到 ECS m 的传输时延为

$$T_n^{\mathrm{tr}}(t)=D_n^{\mathrm{off}}(t)/R_n(t) \quad (4)$$

其中，$D_{U_n,m}^{\mathrm{off}}(t)=\rho_n(t)D_n(t)$。

IoT 设备 U_n 卸载到边缘服务器 m 的传输能耗为

$$E_n^{\mathrm{tr}}(t)=p_n(t)T_n^{\mathrm{tr}}(t) \quad (5)$$

进一步地，定义边缘服务器 m 在时隙 t 分配给 IoT 用户设备 U_n 的计算能力为 $f_m^c(t)$，到达边缘服务器 m 的到达率定义为 $\lambda_m(t)$，可推导出到达边缘服务器 m 的任务调度器的平均到达率为

$$\lambda_m(t)=\sum_{U_n\in \boldsymbol{U}_m}\rho_n(t)\lambda_n(t)+\sum_{n\in U_{m'}}\sum_{m'\in M}\alpha_n^{m'}(t)\cdot\rho_n(t)\lambda_n(t) \quad (6)$$

其中等式右边前半部分为服务器 m 本地处理的任务到达率，后半部分为其余基站迁移至边缘服务器 m 的到达率。

IoT 用户设备 U_n 在时隙 t 卸载到边缘服务器 c 的任务量的计算时延为

$$T_n^{\mathrm{c}}(t)=\frac{1}{\mu_m^{\mathrm{edg}}(t)-\lambda_m(t)} \tag{7}$$

其中，$\mu_m^{\mathrm{edg}}(t)=f_m^{\mathrm{c}}(t)/X_n(t)$ 为用户 U_n 的本地 CPU 的服务速率。

令 T^{tr} 表示边缘节点之间任务转移的平均时间。

因此，卸载到边缘节点上的时延为

$$T_n^{\mathrm{off}}(t)=T_n^{\mathrm{tr}}(t)+\sum_{m'\in M}\{(1-\alpha_n^{m'}(t))T_n^{\mathrm{c}}(t)+\alpha_n^{m'}(t)T_{\mathrm{coop}}^{\mathrm{c}}(t)+\alpha_n^{m'}(t)T^{\mathrm{tr}}\} \tag{8}$$

其中，$T_{\mathrm{coop}}^{\mathrm{c}}(t)$表示在协作节点上的计算时延，表示为

$$T_{\mathrm{coop}}^{\mathrm{c}}(t)=\sum_{m'\in M}\frac{1}{\mu_{m'}^{\mathrm{edg}}(t)-\lambda_{m'}(t)} \tag{9}$$

边缘节点上的计算能耗表达式为

$$E_n^{\mathrm{c}}(t)=\sum_{m'\in M}\alpha_n^{m'}(t)D_n^{\mathrm{off}}(t)X_n(t)\kappa(f_{m'}^{\mathrm{c}})^2 \tag{10}$$

综上所述，卸载到边缘节点的时延及能耗为

$$T_n(t)=\max\{T_n^l(t),T_n^{\mathrm{off}}(t)\} \tag{11}$$

$$E_n(t)=E_n^l(t)+E_n^{\mathrm{tr}}(t)+E_n^{\mathrm{c}}(t) \tag{12}$$

2 问题建立

IoT 用户的 QoE 反映了一个用户通过计算卸载得到的效益。因此，本文将 QoE 定义为通过协作计算卸载降低的长期平均通信成本，其中通信成本表示执行一个任务需要的能耗和时延的加权和。由前文可得，对于用户设备 U_n，在时隙 t 到达的任务通过协作计算卸载的开销为

$$C_n^{\mathrm{off}}(t)=c_n^{\mathrm{E}}(t)E_n(t)+c_n^{T}(t)T_n(t) \tag{13}$$

其中，$c_n^E(t)\in(0,1)$ 和 $c_n^T(t)\in(0,1)$ 分别表示能耗和时延的权重。

因此，本文可以计算出用户 U_n 的长期平均开销为

$$C_n=\lim_{T\to\infty}\frac{1}{T}\boldsymbol{E}\left[\sum_{t=0}^{T}C_n^{\mathrm{off}}(t)\right] \tag{14}$$

本文在多用户设备多边缘节点的架构下，通过联合优化卸载决策、任务分配决策、计算资源分配决策及通信资源分配决策，考虑在通信资源约束、任务调度约束及边缘节点的计算资源约束下，最小化系统内所有用户设备的长期平均开销。因此，本文将建立如下模型：

$$\begin{aligned}&\min_{\substack{\alpha(t),\rho(t)\\ f(t),\beta(t)\\ p(t)}}\sum_m\sum_{n\in U_m}C_n\\ &s.t.\\ &C1:\rho_n(t)\in[0,1],\forall n\in U_m\\ &C2:T_n(t)\leqslant\tau_n(t),\forall n\in U_m\\ &C3:\beta_n^w(t)\leqslant 1,\forall w\in\boldsymbol{W},\forall n\in U_m\\ &C4:p_n(t)\leqslant P^{\max},\forall n\in U_m\\ &C5:f_m^{\mathrm{c}}(t)\leqslant f_m^{\max}\\ &C6:\sum_{m'=1}^{M}\alpha_n^{m'}(t)\leqslant 1,\forall m'\in\boldsymbol{M}\end{aligned} \tag{15}$$

其中，C_1 表示计算任务的卸载比例不能超过 1，C_2 表示每个计算任务的时延不能超过最大允许时延，C_3 表示只能为每个设备分配一个子载波，C_4 表示发射功率不能超过最大功率，C_5 表示分配的计算资源不能超过最大的计算资源，C_6 表示每个计算任务只能在一个边缘节点上处理。

观察式(15)可得，由于约束条件中的二进制变量，使得本文所提目标函数是一个非凸随机优化问题，难以应用到实时动态网络中解决动态优化问题。当前用于求解动态网络优化问题的方法主要是 Lyapunov 优化算法[14-16]，然而 Lyapunov 优化算法不能解决决策空间的维度灾难问题。因此本文考虑通过将深度神经网络与强化学习结合，在解决维度灾难问题的同时，求解动态网络优化问题。

2.1 多智能体深度增强学习 MADRL 模型建立

在本文的 MEC 边缘架构中，基于多智能体架构采用分布式 DRL 算法进行任务调度及资源分配，最终最大化系统用户平均 QoE。本文将每个边缘节点作为一个智能体，在每个时隙，根据当前环境的状态及训练好的策略选择一个行动。在每个时隙 t，边缘节点 m 作为单个智能体，观察状态 $s(t)$，并相应地选择一个行动(任务调度及资源分配)$a(t)$。在选择了行动 $a(t)$后，环境的状态 $s(t)$转换为一个新的状态 $s(t+1)$，智能体 m 得到一个瞬时回报$r(t)$。

在本文中，各智能体的 MDP 模型定义如下：

(1)状态空间 S：将智能体 m 在时隙t 的环境状态定义为：在时隙 t 时有任务产生的用户设备U_n 的任务特征 $\Omega_n(t)$，任务到达率 $\lambda_n(t)$，边缘节点 m 的资源剩余状态 $f_m^{\mathrm{idle}}(t)$。智能体 m 在时隙 t 时的状态定义为 $S_m(t)=[\lambda_n(t),\Omega_n(t),f_m^{\mathrm{idle}}(t)]$。

(2)行动空间 A：在每个时隙 t，智能体 m 的行动定义为：其服务的 IoT 用户设备 U_n 的卸载决策变量$\alpha_n^{m'}(t)$，其服务的用户设备 U_n 的卸载率为 ρ_n

(t)，用户设备的功率分配为 $p_n(t)$，分配给用户设备 U_n 的计算频率为 $f_{\mathrm{m}}^{\mathrm{c}}(t)$，子载波分配为 $\beta_n^w(t)$，因此智能体 m 在时隙 t 时的行动定义为 $a_m(t)=\{\rho_n(t),\alpha_n^{m'}(t),\beta_n^w(t),p_n(t),f_{\mathrm{m}}^{\mathrm{c}}(t)\}$。

(3)回报函数 R：由上文可得，本文将智能体 m 在时隙 t 获得的瞬时回报定义为

$$R_m(t)=\sum_{n\in U_m}C_n^{\mathrm{off}}(t) \tag{16}$$

因此所有智能体获得的总回报在时隙 t 可以表示为

$$R(t)=\sum_{m\in \boldsymbol{M}}R_m(t) \tag{17}$$

状态转换概率为 $P_m(s'_m \mid s_m,a_m)s$。

2.2 基于 MADDPG 的任务卸载和资源分配算法

在多智能体环境下，每个智能体在学习过程独立更新它们自己的策略可能会造成训练过程的不稳定性。另外，智能体之间需要协作来获取其他智能体的决策信息以计算本地 Q 函数，因此为了克服多智能体环境的不稳定性，且使得智能体实现相互协作，本文采用多智能体深度确定性策略梯度算法(Multi-Agent Deep Deterministic Policy Gradient，MADDPG)[17]。MADDPG 是基于 AC 算法，并能更好地适用于多智能体环境中，能够很好地解决环境的变化特性以及多个智能体之间的相互作用。

每个智能体分为两部分：评判家部分和行动者部分。每个智能体在每个时隙会获取他们自己的状态信息 S_m，然后将该状态信息输入行动者网络，便会得到在当前策略下的最优行动 a_m。每个智能体通过执行该行动 a_m，会得到一个当前回报 r_m 和新的状态 S'_m。为了打破经验数据之间的相关性，本文采用经验池 D 回访。该经验池 D 位于管理员智能体中，并会存储所有智能体上传的的四元组$\langle s,a,r,s'\rangle$。每个智能体都有一个本地经验回访池 D_m^{loc}，并且会将$\langle s_m,a_m,r_m,s'_m\rangle$储存在 D_m^{loc} 中。在管理员智能体中有一个全局经验池 D，并且每隔一定的时间，每个智能体会将自己的本地经验池的内容上传到全局经验池 D 中。因此，在全局经验池中储存有所有智能体的状态，包括行动、回报和新的状态。当全局经验池 D 的可用容量为 0 时，智能体从经验池 D 中随机采样 H 个样本，其中每个样本由所有智能体的四元组组成，即对于样本 D_{h}，其表达式为

$$D_h=(\langle s_1^h,a_1^h,r_1^h,s'^h_1\rangle,\cdots,\langle s_m^h,a_m^h,r_m^h,s'^h_m\rangle) \tag{18}$$

评判家网络则通过计算状态-行动值函数来评估策略的好坏。当集中式的训练过程完成后，每个智能体从智能体管理员处下载已经训练好的参数，在执行过程中，每个智能体的行动者只需要根据当前时隙的状态及策略选择自己的行动即可。

在 MADDPG 算法中，将会在每个片段后更新行动和批判家网络。本文的目标是获取最大化长期期望回报下的最优策略的选取，可以表述为

$$\pi^*=\underset{\pi}{\operatorname{argmax}}\,E_\pi\Big[\sum_{t=1}^{\infty}\gamma^{t-1}R(t)\Big] \tag{19}$$

其中，$\pi=\{\pi_1,\pi_2,\cdots,\pi_M\}$，表示所有 M 个智能体的策略，γ 表示折扣因子。

综上，基于 MADDPG 的任务卸载和资源分配算法的训练和执行过程如下：

在每个时隙的开头，每个智能体将会从全局经验池 D 中随机抽取 H 个样本，然后将这些样本各自输入评判家网络中，将会得到状态值函数。行动者网络则采用如式进行梯度更新：

$$\nabla_{\theta_m}J\approx\frac{1}{H}\sum_h\nabla_{\theta_m}\log\pi_m(a_m\mid s_m)\cdot Q_m^{\pi}(s^h,a_1^h,\cdots,a_M^h)\big|_{a_m=\pi_m(s_m^h)} \tag{20}$$

其中，θ_m 表示智能体 m 的策略参数。

然后，评判家网络将以时间差分(Temporal Difference，TD)误差更新其权重，这里采用$TD(0)$，具体公式如下：

$$\delta_m=y^h-Q_m^{\pi}(s^h,a_1^h,\cdots,a_M^h) \tag{21}$$

其中，y^h 表示目标网络中的值函数，具体如下式所示：

$$y^h=r_m^h+\gamma Q'^{\pi}_m(s'^h,a'_1,\cdots,a'_M)\big|_{a'_i=\pi'_i(s'^h_i)} \tag{22}$$

为了使均方差最小，定义损失函数，评判家网络在进行参数更新时需要满足最小化损失函数。

$$Loss(\theta_m)=\frac{1}{H}\sum_h(y^h-Q_m^{\pi}(s^h,a_1^h,\cdots,a_M^h))^2 \tag{23}$$

本文采用软更新的方式来更新每个智能体 m 目标网络的参数，

$$\theta'_m\leftarrow\zeta\theta_m+(1-\zeta)\theta'_m \tag{24}$$

其中，ζ 表示软更新因子，一般将其定义为 $\zeta=0.01$。

当训练完成后，每个智能体则只需要进行分步式执行即可。在每个执行阶段，每个智能体只需要将自己的当前状态输入行动者网络即可得到当前状态下的最优行动策略。

3 性能仿真与结果分析

在本节中，将通过仿真结果对所提出的算法进行评估。本文考虑的设备和基站总数分别为 40 和 4。基站半径设置为 $R=500$ m，路径损耗模型可为 $127+30\log d$（d 单位为 km），基站的带宽为 20 MHz，子通道的数量为 10。BSs 和 IoT 设备的计算能力分别为 12 和 0.8 GHz。每个任务的数据大小为 0.5 Mbit。设备的最大传输功率为 0.5 W。每个任务所需 CPU 周期的平均数为 0.3 G cycles。假设任务到达率为 25 个任务/分钟。学习率固定为 10^{-2}，折扣因子 γ 设置为 0.95。$\kappa=10^{-27}$ Watt·s^3/$cycles^3$，平均传输时间 T_{tr} 设置为 0.002 s[9]，$\sigma^2=10^{-10}$ W/Hz。为了展示该算法的性能，本文与 Actor-Critic 算法进行了比较。

图 2 显示了不同算法下 MECS 的平均延迟和计算能力之间的关系。可以观察到，随着 MECS 计算能力的增加，平均延迟减小。这是因为计算容量越大，MECS 的服务率越高，从而降低了平均延迟。此外，所提出的 MA-DDPG 算法具有较低的平均延迟。

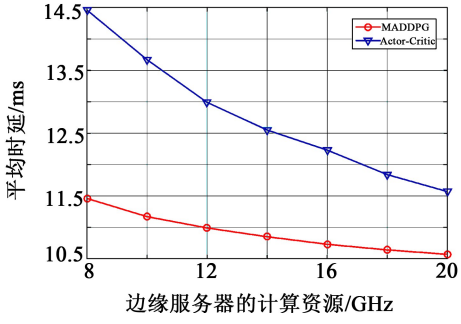

图 2 不同算法下 MECS 的平均延迟和计算能力的关系

图 3 显示了不同算法下 MECS 计算能力对平均能耗的影响。可以观察到，对于这两种算法，能量消耗随着 MECS 计算容量的增加而增加。还可以看到，所提出的 MA-DDPG 算法的能耗高于 Actor-Critic 算法。这是因为协同卸载有助于实现 MECS 之间的计算负载平衡，这使得能耗高于 Actor-Critic 算法。

图 4 显示了不同方案下所需 CPU 周期数与平均延迟之间的关系。可以看到，所需的 CPU 周期越大，延迟越高，两种算法之间的差距也在加大。这是因为该算法能够有效地利用系统计算资源。

图 5 显示了不同方案下的系统奖励。可以看到，MADDPG 算法的性能高于 Actor-Critic 算法。这是因为该算法可以通过学习协作策略获得互计算资源信息。

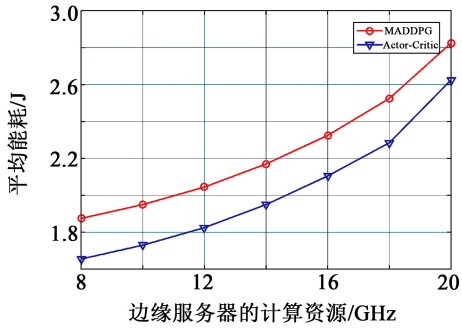

图 3 不同算法下 MECS 计算能力对平均能耗的影响

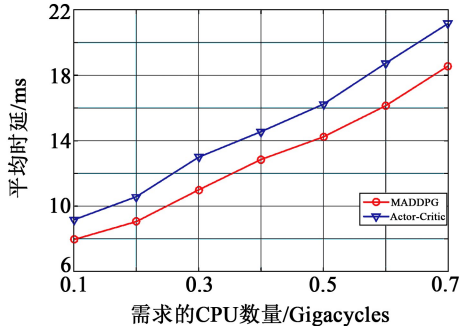

图 4 不同方案下所需 CPU 周期数与平均延迟的关系

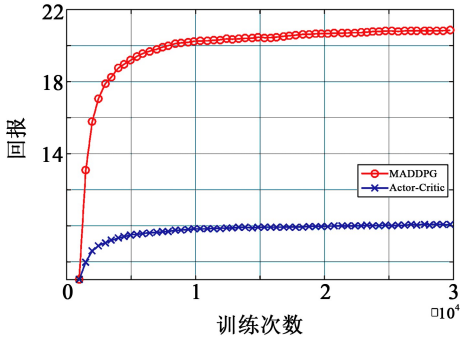

图 5 不同方案下的系统奖励

4 结束语

本文提出了一种在物联网 MEC 中基于多智能体深度强化学习的协同计算卸载和资源分配方法，将联合协同计算卸载和资源分配问题描述为一个优化问题，并提出了一种 MADDPG 算法来解决该问题。MADDPG 算法可以实现集中训练和分散执行，有助于学习相互协作的策略。仿真结果表明了所提出的 MA-DDPG 算法在不同系统参数下的有效性。

参考文献：

[1] Wu Y, Shi J, Ni K, et al. Secrecybased delay-aware computation offloading via mobile edge computing for internet

of things[J]. IEEE Internet of Things J., 2019, 6(3): 4201-4213.

[2] Guo S, Liu J, Yang Y, et al. Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing [J]. IEEE Trans. on Mobile Computing, 2019, 2(18): 319-333.

[3] Kuang Z, Li L, Gao J, et al. Partial offloading scheduling and power allocation for mobile edge computing systems[J]. IEEE Internet of Things J., 2019, 6(4): 6774-6785.

[4] Mao Y, You C, Zhang J, et al. A survey on mobile edge computing: The communication perspective [J]. IEEE Communications Surveys and Tutorials, 2017, 19(4): 2322-2358.

[5] Lei L, Xu H, Xiong X, et al. Multiuser resource control with deep reinforcement learning in IoT edge computing[J]. IEEE Internet of Things J., 2019, 6(6): 10119-10133.

[6] Zhang Q, Gui L, Hou F, et al. Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN[J]. IEEE Internet of Things J., 2020, 4(7): 3282-3299.

[7] Alameddine H A, Sharafeddine S, Sebbah S, et al. Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing[J]. IEEE J. on Sel. Areas in Communications, 2019, 3(37): 668-682.

[8] Wang Y, Tao X, Zhang X, et al. Cooperative task offloading in three-tier mobile computing networks: An ADMM framework[J]. IEEE Trans. on Vehicular Technol., 2019, 3(68): 2763-2776.

[9] Wang S, Zhang X, Yan Z, et al. Cooperative edge computing with sleep control under nonuniform traffic in mobile edge networks[J]. IEEE Internet of Things J., 2019, 3(6): 4295-4306.

[10] Feng J, Yu F R, Pei Q, et al. Cooperative computation offloading and resource allocation for blockchain-enabled mobile edge computing: A deep reinforcement learning approach[J]. IEEE Internet of Things J., 2020, 7(7): 6214-6228.

[11] Xiao Y, Krunz M. Distributed optimization for energy-efficient fog computing in the tactile internet[J]. IEEE J. on Sel. Areas in Communications, 2018, 36(11): 2390-2400.

[12] Wang Y, Sheng M, Wang X, et al. Mobile-edge computing: Partial computation offloading using dynamic voltage scaling [J]. IEEE Trans. on Communications, 2016, 64(10): 4268-4282.

[13] Mao Y, Zhang J, Letaief K B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices[J]. IEEE J. on Sel. Areas in Communications, 2016, 34(12): 3590-3605.

[14] Lyu X, Ni W, Tian H, et al. Optimal schedule of mobile edge computing for internet of things using partial information[J]. IEEE J. on Sel. Areas in Communications, 2017, 11(35): 2606-2615.

[15] Mao Y, Zhang J, Song S H, et al. Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems [J]. IEEE Trans. on Wireless Communications, 2017, 9(16): 5994-6009.

[16] Deng Y, Chen Z, Yao X, et al. Parallel offloading in green and sustainable mobile edge computing for delay constrained IoT system[J]. IEEE Trans. on Vehicular Technol., 2019, 12(68): 12202-12214.

[17] Cao Z, Zhou P, Li R, et al. Multi-agent deep reinforcement learning for joint multi-channel access and task offloading of mobile edge computing in industry 4.0[J]. IEEE Internet of Things J., 2020, 7(7): 6201-6213.

作者简介：

夏炳森(1981—)，男，福建省人，硕士，高级工程师，主要研究方向包括电力与系统、移动边缘计算和人工智能。