DOI:10.11918/202310003

边缘计算中时延敏感的启发式任务卸载方法

黄腾飞,杜永文,刘 帅,丁 元,王 欢

(兰州交通大学 电子与信息工程学院,兰州 730070)

摘 要:针对移动边缘计算在多用户环境中卸载决策设计困难导致负载失衡、总时延过高和响应延迟问题,提出一种时延敏感的启发式任务卸载方法。首先,为解决边缘设备计算资源匮乏和电量不足的问题,建立以边缘服务器为主体的卸载架构,构建系统模型与时延优化模型;其次,提出一种改进的近端策略优化算法 I-PPO,通过增加离线训练阶段、设计考虑多智能体决策影响的奖励机制、在特征中融入基于特定智能体的全局信息,使算法能够适用于多用户场景;进一步,在 I-PPO 的基础上,在任务卸载执行过程中引入任务优先级调度决策,构造时延敏感的轻量级启发式任务卸载算法,以降低系统时延,并提升用户体验。仿真实验表明,相比同类算法,所提 I-PPO 算法在收敛速度、寻优能力和鲁棒性方面表现更优,且适用于多智能体环境;所提算法在系统总时延和边缘服务器负载均衡方面优于对比算法,并具有良好的稳定性。

关键词:边缘计算;深度强化学习;任务卸载;多智能体;启发式算法

中图分类号:TP391 **文献标志码**:A **文章编号**:0367-6234(2026)03-0205-09

Latency-sensitive heuristic task offloading method in edge computing

HUANG Tengfei, DU Yongwen, LIU Shuai, DING Yuan, WANG Huan

(College of Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China)

Abstract: In order to address the challenge of designing reasonable offloading decisions for mobile edge computing (MEC) in multi-user environments, which leads to load imbalance, excessive total latency, and response delays, this paper proposed a latency-sensitive heuristic task offloading method. Firstly, to address the issues of limited computational resources and insufficient battery power of edge devices during computation task processing, the paper introduced an edge server-centric offloading paradigm and established a system model and a latency optimization model. Subsequently, it introduced an improved proximal policy optimization algorithm (I-PPO), which extended the offline training process, designed a reward mechanism that considers the impact of multi-agent decisions, and incorporated global information based on specific agents into the features, enabling the algorithm to be suitable for multi-user environments. Furthermore, building upon I-PPO, the paper introduced task priority scheduling decisions into the task offloading execution process, resulting in the development of a latency-sensitive lightweight heuristic task offloading algorithm, denoted as HTAI. This further optimized system latency and enhanced user satisfaction. Simulation experiments demonstrate that the I-PPO algorithm proposed in this paper, compared to similar algorithms, effectively improves convergence speed, optimization capability, and robustness, and it can be applied in multi-agent environments. Moreover, the algorithm proposed herein outperforms other algorithms in terms of total system latency and edge server load balance, exhibiting strong stability.

Keywords: edge computing; deep reinforcement learning; task offloading; multi-agent; heuristic algorithm

近年来,移动边缘计算(MEC)在众多领域得到快速发展和广泛应用。其通过将数据处理下沉到网络边缘,缓解了终端设备计算能力有限与高计算资源需求、高时延敏感任务之间的矛盾[1-2]。在边缘计算中,如何制定合理的任务卸载策略,以最小化系统总时延与能耗,是一个具有挑战性的问题[3]。

Teng 等[4]将任务卸载决策问题转化为非合作博弈问题,并证明了纳什均衡的存在;Zhao 等[5]、Wu 等[6]基于 Lyapunov 优化理论设计了在线动态卸载算法,可以有效降低系统能耗和延迟;Bozorgchenani 等[7]提出一种进化算法,能够找到能耗和时延之间帕累托最优前沿;Hou 等[8]考虑了具有依赖关系任务的卸载策略,通过建立无环图,将决策问题转化为能够在多项式时间内求解的离散问题。由于强化学

收稿日期:2023-10-07;**录用日期**:2023-11-16;**网络首发日期**:2023-12-14
网络首发地址:https://link.cnki.net/urlid/23.1235.T.20231214.1310.013
基金项目:甘肃省自然科学基金(1610RJZA056)
作者简介:黄腾飞(1996—),男,硕士研究生;杜永文(1974—),男,副教授,硕士生导师
通信作者:黄腾飞,849138454@qq.com

习(reinforcement learning, RL)在协同决策、动态分配和约束优化问题中表现优异,研究人员开始将其应用于任务卸载决策问题。

Li 等[9]提出一种基于近端策略优化(proximal policy optimization,PPO)算法的强化学习框架,解决了利润最大化目标下的卸载决策问题;Li 等[10]设计了一种基于 RL 的自适应服务部署算法,实现了用户端自主决策;Li 等[11]以长期效益最大化为目标,设计了一种基于深度强化学习的资源预留和服务器协作卸载策略。然而,将 RL 算法直接应用于多智能体环境,可能会导致非平稳性问题[12],其解决方案具有挑战性。Liu 等[13]将任务卸载问题描述为随机博弈,并开发了基于独立学习的 MAQ 算法以寻求纳什均衡;Zhang 等[14]提出了一种多智能体分布式方案,各智能体通过推断其他智能体的决策共同确定任务卸载和资源分配策略;Huang 等[15]设计了一种基于深度强化学习(deep reinforcement learning, DRL)的方案,在保证时延要求的同时,最小化了小蜂窝网络中的总能耗,能够处理非平稳性的 DRL 算法在多用户任务卸载决策中表现出良好效果。但现有研究存在两方面问题:1)部分研究中,边缘设备供能困难且计算资源有限[16],而边缘服务器供电充足,若边缘设备承担卸载决策和数据预处理任务,会增加其能量消耗并缩短使用寿命;2)部分研究仅考虑边缘设备到边缘服务器、边缘服务器到云端及边缘设备到云端 3 种卸载模式,没有考虑边缘服务器到边缘服务器的协作卸载模式,使得系统负载失衡、整体效率低下。为此,提出一种轻量级启发式任务卸载方法,以边缘服务器为卸载主体,使用 DRL 算法优化任务卸载策略,通过多边缘服务器和云端服务器的协作,提升系统运行效率和用户体验。

1 系统模型

1.1 边缘环境模型

采用经典 3 层边缘计算结构,边缘设备层、边缘节点层和云端之间通过有线网络和无线链路互联[17],如图 1 所示。边缘设备层由传感器、智能摄像头、智能手机等具有有限计算能力和电量的网络终端设备组成,通过无线链路与边缘节点层相连,随机产生时延敏感且高计算资源需求的任务,能够进行简单的信息预处理。边缘节点层位于边缘设备之上,包括边缘服务器、无线接入点、路由器等,具有较高的计算和存储能力,能为一定范围内的边缘设备提供计算资源。本文中该层主要指边缘服务器。边缘服务器之间及其与云端服务器通过有线网络连接,可将计算任务通过有线网络进行卸载。云端为云计算服务中心,拥有充足的计算能力和存储能力,可为其他设备提供计算资源支持,由于其地理位置过远,处理数据时时延较高。

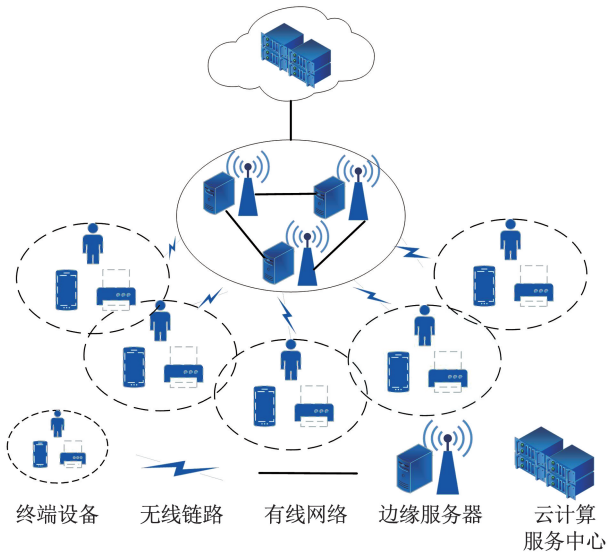


图 1 以边缘服务器为卸载主体的边缘计算结构

Fig. 1 Edge computing structure with edge server-centric offloading

在边缘网络中,边缘设备集 U 由 $n(u)$ 个移动设备组成,可表示为 $U=\{u_1,u_2,\cdots,u_{n(u)}\}$;任务集 I 由 $n(i)$ 个任务组成,与边缘设备一一对应,可表示为 $I=\{i_1,i_2,\cdots,i_{n(i)}\}$,单个任务可表示为 $i=(s_i,k_i,d_i,D_i)$,其中,s_i 为任务卸载目标边缘服务器序号,k_i 为卸载的次数,d_i 为任务数据量,D_i 为任务计算量(量化为执行单位任务数据所需的 CPU 时钟周期),任务大小服从正态分布;边缘服务器集 S 由 $n(s)$ 个边缘服务器组成,可表示为 $S=\{s_1,s_2,\cdots,s_{n(s)}\}$,单个边缘服务器包含 4 个维度信息,表示为 $s=[f_s,\varphi_s,B_s,b_s]$,其中 f_s 为边缘服务器的最大计算能力(量化为设备 CPU 最大时钟频率),φ_s 为有线传输功率,B_s 和 b_s 分别为最大无线链路带宽和有线链路带宽;云计算服务中心 C 实质为云服务器中心分配给该环境的虚拟容器,拥有丰富的计算资源,表示为 $C=\{f_C\}$;任务卸载位置集 $Z=\{z_1,z_2,\cdots,z_{n(i)}\}$,表示每个任务的卸载位置;$t$ 表示时刻,s。

设定边缘设备与边缘服务器存在固定从属关系,从属的边缘服务器为本地服务器。在每个时刻 t 开始时,云计算服务中心与边缘服务器向有线网络广播其负载情况,将系统中最空闲的若干个边缘服务器设定为备选服务器。边缘服务器接收到边缘设备卸载的任务后,有 3 种处理方式:1)本地计算:在本地边缘服务器上进行处理,总时延用 T_L 表示;2)卸载至备选服务器:当本地边缘服务器负载较重时,将任务二次卸载至备选服务器处理,总时延用

T_M表示;3)卸载至云端:当任务计算量较大且本地和备选边缘服务器也负载较重时,将任务卸载至云计算服务中心进行处理,总时延用 T_C表示。

在该边缘环境中,边缘服务器作为卸载决策的主体,供电充足;而边缘设备仅负责将任务卸载至本地边缘服务器,卸载功耗不受卸载模式影响,因此优化目标仅考虑时延因素。

1.2　时延计算模型

边缘服务器对接收到的任务共有 3 种卸载选择,以下分别构建 3 种方式的时延计算模型。考虑到任务结果数据量远小于任务上传的数据量,本文不考虑结果返回时延。

1)本地计算

当任务大小合适且边缘服务器计算资源充足时,选择本地计算,时延由任务上传至本地服务器的传输时延和任务处理时延两部分组成。本地计算时延 T_L 可表示为

$$T_L = \frac{d_i}{r_{u,s}} + \frac{L_s + D_i}{f_s} \tag{1}$$

式中:$d_i/r_{u,s}$为边缘设备 u 到边缘服务器 s 的传输时延;$r_{u,s}$为数据传输速率;L_s 为边缘服务器 s 上等待处理的计算量。$r_{u,s}$的计算公式为

$$r_{u,s} = B_{u,s}\log_2(1 + \mathrm{SINR}) \tag{2}$$

式中:$B_{u,s}$为边缘设备 u 到边缘服务器 s 的链路信道带宽;SINR 为信号与干扰加噪声比[18],表示接收到的有用信号强度与干扰信号强度的比值,其求解公式为

$$\mathrm{SINR} = \frac{P_{u,s}O_{u,s}}{\sum_{u'=1}^{n} P_{u',s}O_{u',s} + \sigma_{u,s}} \tag{3}$$

式中:$P_{u,s}$和 $P_{u',s}$为移动设备和边缘服务器之间的无线传输功率;$O_{u,s}$和 $O_{u',s}$为信道增益系数; $\sum_{u'=1}^{n} P_{u',s}O_{u',s}$ 为该边缘服务器所属其他边缘设备的无线干扰信号功率;$\sigma_{u,s}$为服从高斯分布的噪音。增益系数计算公式为

$$O_{s,s'} = k_{s,s'}d_{s,s'}^{-\alpha} \tag{4}$$

式中:$k_{s,s'}$为信号衰落因子;$d_{s,s'}$为 s 到 s'的距离;α 为路径损耗因子。

2)卸载至备选服务器计算

本地边缘服务器负载较重而备选服务器相对空闲时,选择将任务卸载到备选服务器。总时延包括 3 个部分:任务卸载至本地服务器的时延、本地服务器卸载至备选服务器的时延、任务处理时延。卸载至备选服务器的时延 T_M可表示为

$$T_M = \frac{d_i}{r_{u,s}} + \frac{d_i}{r_{s,s'}} + \frac{L_{s'} + D_i}{f_{s'}} \tag{5}$$

式中 $r_{s,s'}$表示本地服务器 s 到备选服务器 s'之间链路的数据上传速率。

3)卸载至云端计算

当边缘服务器和备选服务器任务负载都较重时,选择将任务卸载到云端进行计算,总时延同样包括 3 个部分:任务卸载至本地服务器的时延、本地服务器卸载到云端的时延、云端处理时延。卸载至云端的时延 T_C表示为

$$T_C = \frac{d_i}{r_{u,s}} + \frac{D_i}{r_{s,C}} + \frac{L_C + D_i}{f_C} \tag{6}$$

4)总时延

任务由边缘设备卸载至边缘服务器后,3 种卸载模式互斥,因此,总时延 T_i 可表示为

$$\begin{cases} T_i = K_1T_L + K_2T_M + K_3T_C \\ K_1 + K_2 + K_3 = 1, K_1, K_2, K_3 \in \{0,1\} \end{cases} \tag{7}$$

式中 K_1, K_2, K_3为决策因子,取值为 1 时表示选择对应卸载模式,取值为 0 时表示不选择。

1.3　问题描述

边缘设备随机生成大小不固定的任务,边缘服务器接收任务后,需结合任务大小、本地负载、备选服务器负载和云端负载选择合理的卸载模式。由式(7)可知,单个任务时延受卸载决策影响,系统总时延 T_{all}为 n 个任务时延之和,可表示为

$$T_{all} = \sum_{i=1}^{n} T_i \tag{8}$$

其中 $T_i = \begin{cases} T_L, & K_1 = 1 \\ T_M, & K_2 = 1 \\ T_C, & K_3 = 1 \end{cases}$。

优化目标为最小化系统总时延,综合约束条件建立优化公式如下:

$$\begin{cases} \min\ T_{all}, & \forall i \in I \\ \text{s.t. C1}: B_{u,s} \leqslant B_s, & \forall u \in U, s \in S \\ \text{C2}: r^i_{s,s'} \leqslant r_s, & s \in S \\ \text{C3}: r^i_{s,C} \leqslant r_s, & s \in S \\ \text{C4}: K_1 + K_2 + K_3 = 1, & K_1, K_2, K_3 \in \{0,1\} \end{cases} \tag{9}$$

式中:C1 表示边缘设备与边缘服务器之间的无线链路带宽不超过边缘服务器的最大无线链路带宽;C2 和 C3 分别表示任务由本地边缘服务器卸载到备选边缘服务器、云端的数据传输速率不超过边缘服务器的有线链路带宽;C4 表示任务仅可在本地边缘服务器、备选边缘服务器、云端中选择一个进行卸载。

2　基于 I-PPO 的任务卸载算法

边缘计算中的任务卸载问题为 NP 难问题,针对多用户环境对 DRL 算法进行改进,引入任务优先

级调度构建启发式任务卸载算法,进一步优化任务卸载性能。

2.1 改进的近端策略优化算法 I-PPO

近端策略优化(PPO)算法[19]是一种基于策略梯度的强化学习方法,采用 Actor-Critic 架构,其特点在于单次训练中可进行多次小批量梯度更新,从而提高算法收敛速度和寻优能力。研究表明,PPO 经过适当调整后可在多种合作多智能体环境中取得良好性能[20]。本文针对边缘计算环境对 PPO 进行改进:一是增加离线训练阶段,将算法分为离线训练和分散执行两个阶段;二是在离线训练过程中设计考虑多智能体决策影响的奖励机制;三是在特征中融入基于特定智能体的全局信息,以提升算法在多智能体环境中的寻优能力。

1)离线训练

在多智能体环境中,单个智能体难以获取全局信息,不满足马尔可夫博弈条件[21],导致算法难以收敛。为此,增加离线训练阶段,在此阶段设置一个中央控制器,其能够观测到全局信息并将处理后的信息通过网络广播给所有智能体。

在任务卸载问题中,中央控制器在时刻 t 可观测的全局信息 θ_{all}^t 可表示为

$$\theta_{\text{all}}^t = \{D_i, L_s, L_{\text{C}}, [K_1, K_2, K_3]_i\}_t, \forall i \in I \quad (10)$$

式中 $[K_1, K_2, K_3]_i$ 表示任务 i 在边缘服务器上的卸载决策。

中央控制器观测全局信息后,对数据进行处理,得到训练所需数据,如每个任务在 3 种决策下的理论运行时间和实际运行时间,计算公式如下:

$$T_{\text{all}}^{i,t} = \{T_{\text{L}}, T_{\text{M}}, T_{\text{C}}\}_t^i \quad (11)$$

$$T_{\text{real}}^{i,t} = \{T_i, [K_1, K_2, K_3]_i\}_t \quad (12)$$

2)适应多智能体环境的奖励机制

在 DRL 算法训练过程中,考虑其他智能体的决策对算法收敛至关重要。通过设计奖励机制体现环境在多智能体决策下的变化特点时,帮助智能体适应环境变化。设计奖励机制如下:

$$r_t^i = \begin{cases} 1, & T_{\text{real}}^{i,t} = \min T_{\text{all}}^{i,t} \\ -1, & T_{\text{real}}^{i,t} \neq \min T_{\text{all}}^{i,t} \end{cases} \quad (13)$$

该奖励机制规定,当选择的卸载模式为 3 种模式中时延最小的决策是给予奖励,否则给予惩罚。3 种卸载模式下的时延由所有智能体共同决定,其中隐含了环境在多智能体决策下的变化趋势,智能体可通过学习找到使时延最小的决策,最终所有智能体决策达成纳什均衡。

3)包含特定全局信息的特征

Yu 等的研究表明,在特征中增加特定于智能体的全局信息可有效提升算法在多智能体环境中的收敛效果。传统单智能体 RL 算法的特征多为智能体本地信息,可表示为

$$S_t^i = \{D_i, L_s\}_t \quad (14)$$

即任务大小及本地服务器负载。

多智能体环境中,在特征中增加基于智能体的全局信息,设计特征如下:

$$S_t^i = \{D_i, L_s, L_{s'}, L_{\text{C}}\}_t \quad (15)$$

即任务大小、本地服务器负载、备选服务器负载、云端负载。式中 $L_{s'}$ 为备选边缘服务器 s' 上等待处理的计算量。以上特征包含了边缘服务器决策所需信息,且信息量远小于所有全局信息,可在每个时隙开始时由云端服务器和各边缘服务器在网络内广播获取。

4)I-PPO 算法描述

改进近端策略优化算法(I-PPO)通过适配多智能体环境的奖励机制和特征调整,增强算法在多智能体环境中的寻优能力。算法的离线训练流程如算法 1 所示。

算法 1 I-PPO 离线训练

输入:边缘设备集 U,任务集 I,边缘服务器集 S,云服务器 C,训练的最大迭代次数 iteration,每轮迭代中总时长 T

输出:训练完成的策略网络 π

初始化:初始化每个智能体的策略网络 π 和价值网络 V;初始化记忆库 D

```
while k ∈ iteration do
  for t = 1 to T
    for s ∈ S
      for i ∈ s
        由式(10)、(15)得到初始状态 θ_t^i
        由策略网络 π 及 θ_t^i 获得动作 a_t^i
      end for
    end for
    for s ∈ S
      for i ∈ s
        时隙 t 中所有任务决策完成后,由式(11)、(12)、(13)得智能体奖励 r_t^i
        将 θ_t^i, a_t^i, r_t^i 存入记忆库 D 中
      end for
    end for
  end for
  for agent s ∈ S
    使用记忆库 D 中的决策数据,更新策略网络 π 及价值网络 V
  end for
end while
返回训练完成的策略网络 π
```

步骤 2 ~ 15 获取决策数据,其中步骤 3 ~ 8 对每个任务由状态 s_t^i 和策略网络 π 获得动作 a_t^i,步骤 9 ~ 14根据全局信息获得每个任务的奖励 r_t^i,并将相关信息存入记忆库 D;步骤 16 ~ 18 对每个智能体,利用记忆库 D 中的数据更新策略网络 π 及价值网络 V;步骤 20 返回训练完成的策略网络 π。

2.2　基于 I-PPO 的启发式任务卸载算法

获得训练完成的策略网络 π 后,为进一步优化任务卸载性能,考虑任务在服务器队列中的等待时间,引入任务优先级调度,提出基于 I-PPO 的启发式任务卸载算法(HTAI)。用户对计算量越小的任务,期望处理时间越短,设计任务预期处理时间为

$$T_e^i = \frac{D_i}{f_s} \tag{16}$$

设计用户满意度函数如下:

$$\xi = \frac{T_e^i}{T_i} \tag{17}$$

ξ 越大,表示用户满意度越高。任务被卸载到执行位置后,对队列中的所有任务根据满意度进行优先级排序,满意度越低,任务优先级越高,满意度低的任务先执行。该方法可在不影响卸载决策的前提下,进一步降低系统总时延。结合基于满意度的任务优先级调度方案和 I-PPO 离线训练得到的策略网络 π,通过边缘服务器和云端的协作,设计 HTAI。算法流程如算法 2 所示。

步骤 1 ~ 9 获取时间 T 内每个任务的卸载决策,其中,步骤 4 ~ 5 步由策略网络 π 及状态 s_t^i 获得动作 a_t^i,步骤 6 在卸载完成后,对任务在执行终端中的等待队列进行任务优先级调度;步骤 10 步输出卸载决策集 Z。

算法 2　HTAI

输入:边缘设备集 U,任务集 I,边缘服务器集 S,云服务器 C,训练完成的策略网络 π,运行时长 T

输出:卸载决策集 Z

1) for $t \in T$
2)　for agent $s \in S$
3)　for $i \in s$
4)　由式(10)、(15)得到初始状态 θ_t^i
5)　由策略网络 π 及 s_t^i 获得动作 a_t^i
6)　卸载至执行终端后,由式(16)、(17)进行任务优先级调度,决定任务执行顺序
7)　end for
8)　end for
9) end for
10) 输出卸载决策集 Z

3　仿真结果分析

所有算法及仿真基于 Python 实现,在搭载 AMD Ryzen 5 2600CPU 和 16 GB RAM 的台式计算机上执行。仿真环境中部署云服务器、若干边缘服务器和边缘设备进行任务卸载。主要参数设置如表 1 所示。

表 1　实验环境参数

Tab. 1　Parameters of experimental environment

参数	任务数据量	任务计算量	边缘服务器最大计算量	云端服务器最大计算量	最大无线链路带宽	边缘服务器之间有线链路带宽	边缘服务器与云之间有线链路带宽
取值	100 ~ 1100	100	600	2	60	60	200
单位	kb	cycle/b	MHz	GHz	Mbps	Mbps	Mbps

3.1　改进近端策略优化算法性能分析

将仅改进特征的 PPO 命名为 PPO-1,仅改进奖励机制的 PPO 命名为 PPO-2。为了评估分析I-PPO 算法的性能,选用 PPO、PPO-1、PPO-2 作为基准算法进行比较。

设置边缘服务器数 $n(s) = 10$,任务数 $n(i) = 100$,每个边缘服务器为固定 10 个边缘设备提供服务。边缘设备每秒产生 1 次任务,算法最大迭代次数 iteration = 50,每次迭代时长 100 s,优化目标为减少每轮训练中的系统总时延,训练结果如图 2 所示。

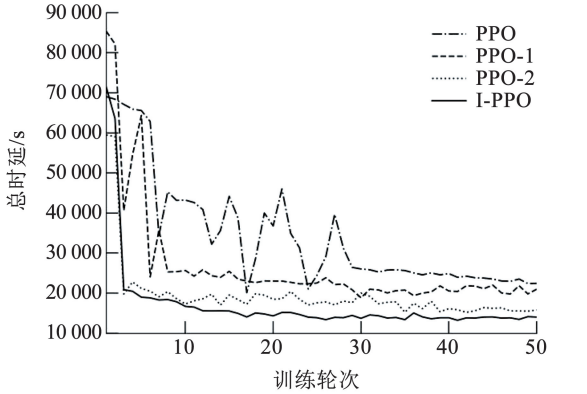
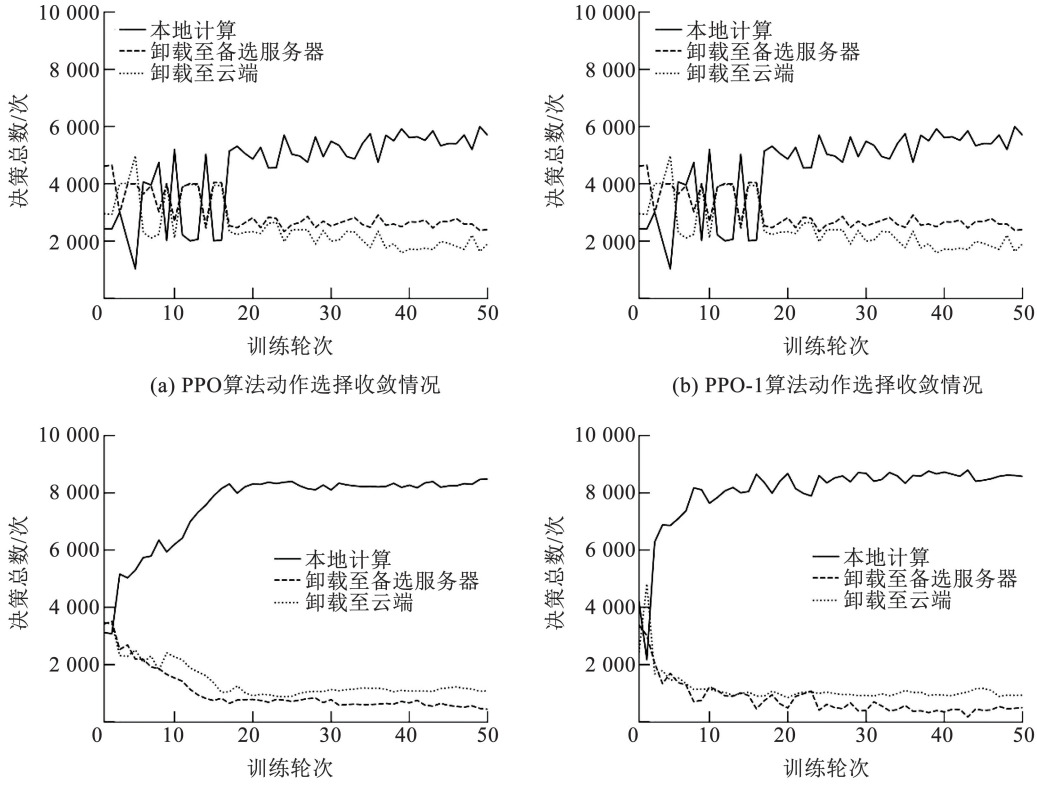


图 2　各算法系统总时延收敛情况

Fig. 2　Convergence status of various algorithmic systems in terms of total latency

由图 2 可以看出,各算法系统总时延总体上随着训练轮次的增加而下降。PPO 算法在约 30 轮训练后收敛,收敛极值在 22 000 ~ 27 000 之间波动;PPO-1 算法在约 8 轮训练后收敛,收敛极值在 22 000 ~ 26 000 之间波动;PPO-2 算法在 3 ~ 4 轮训练后收敛,收敛极值在 16 000 ~ 23 000 之间波动;I-PPO算法同样在 3 ~ 4 轮训练后收敛,收敛极值在 14 000 ~ 21 000 之间波动。收敛极值排序为:PPO > PPO-1 > PPO-2 > I-PPO;在收敛平稳性上,I-PPO > PPO-1 > PPO-2 > PPO。结果表明,特征改进可有效提升收敛的平稳性,奖励机制的改进可加速收敛,两项改进均能够提升算法的寻优能力。因此,I-PPO 算法在多智能体环境中较原 PPO 算法具有更快的收敛速度、更强的寻优能力与更高的鲁棒性。

进一步统计各算法动作选择情况,如图 3 所示。由图 3 可以看出,PPO 算法在 17 轮训练后达到平衡,3 种动作比值约为 11∶5∶4;PPO-1 算法在 10 轮训练后达到平衡,3 种动作比值约为 12∶3∶5;PPO-2 算法在 15 轮后达到平衡,3 种动作比值约为 17∶1∶2;I-PPO算法在 8 轮左右达到平衡,3 种动作比值约为 17∶1∶2,且动作差略大于 PPO-1。收敛速度和最终结果排序为:I-PPO > PPO-2 > PPO-1 > PPO,表明 I-PPO算法动作寻优精度更高。



(a) PPO算法动作选择收敛情况　(b) PPO-1算法动作选择收敛情况

(c) PPO-2算法动作选择收敛情况　(d) I-PPO算法动作选择收敛情况

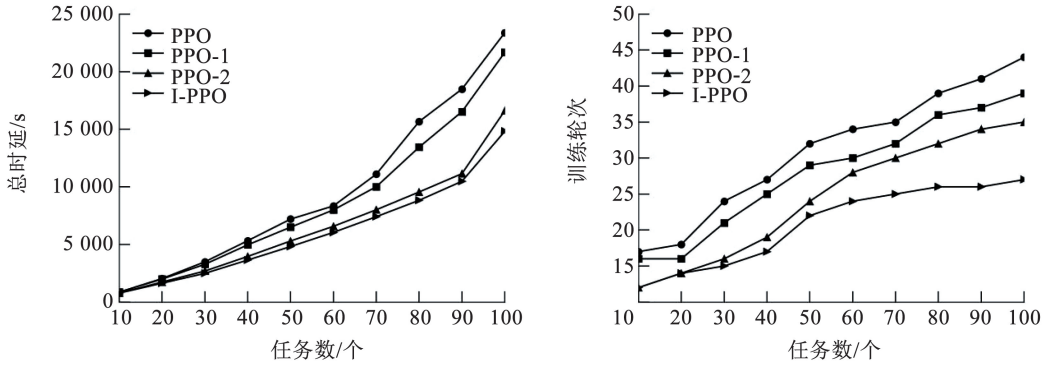
图 3　各算法动作选择收敛情况

Fig. 3　The convergence status of action selection for each algorithm

通过调整任务数 $n(i)$(范围 10 ~ 100),进一步观察算法效果,结果如图 4 所示。从图 4(a)可以看出,各算法收敛极值均随任务数增加而上升,当任务数趋近系统负载上限时上升加速。PPO 算法收敛极值波动范围为 860 ~ 24 000;PPO-1 算法收敛极值波动范围为 860 ~ 220 00;PPO-2 算法收敛极值波动范围为 780 ~ 17 000;I-PPO 算法收敛极值波动范围为 750 ~ 15 000。总体上,不同任务数下收敛极值排序为:PPO > PPO-1 > PPO-2 > I-PPO。

将趋近于收敛极值时的训练轮次定义为收敛所需训练轮次,从图 4(b)可以看出,其总体上随任务数增加而上升,且上升趋势趋于平缓。PPO 算法收敛所需训练轮次在 17 ~ 44 之间;PPO-1 算法在 16 ~ 39 之间;PPO-2 算法在 12 ~ 35 之间;I-PPO 算法在 12 ~ 27 之间。不同任务数下收敛所需训练轮次排序为:PPO > PPO-1 > PPO-2 > I-PPO。综合以上结果

可以看出,I-PPO 在不同任务数下收敛极值和收敛所需训练轮次均优于其他算法,在多智能体环境中有较高的鲁棒性。



(a) 不同任务数下收敛极值变化　　(b) 不同任务数下收敛所需训练轮次变化

图 4　不同任务数下收敛极值和收敛所需训练轮次变化

Fig. 4　Variation in convergence extrema and required training epochs under different task quantities

3.2　任务卸载算法性能分析

为评估所提启发式任务卸载算法性能,从系统总时延、边缘服务器负载均衡度两方面对 HTAI 算法进行测试,选择以下基准算法进行对比:

1)TI:基于 I-PPO 的卸载算法,不含任务优先级调度;

2)TP:基于 PPO 的启发式卸载算法;

3)TD:基于 DQN 的启发式卸载算法;

4)TM:传统卸载方法,边缘设备将任务卸载至所属的边缘服务器执行[22];

5)TR:经典随机卸载算法,任务随机卸载至本地、备选服务器或云端[23]。

各算法的参数设置如表 2 所示。

表 2　算法参数设置

Tab. 2　Setting of algorithm parameters

算法	参数
HTAI	gamma = 0.98, lam = 1, epsilon = 0.8, lr = 0.01, $n(s)=10$, $n(C)=1$
TI	gamma = 0.98, lam = 1, epsilon = 0.8, lr = 0.01, $n(s)=10$, $n(C)=1$
TP	gamma = 0.98, lam = 1, epsilon = 0.8, lr = 0.01, $n(s)=10$, $n(C)=1$
TD	gamma = 0.9, epsilon = 0.9, lr = 0.01, $n(s)=10$, $n(C)=1$
TM	$n(s)=10$
TR	$n(s)=10$, $n(C)=1$

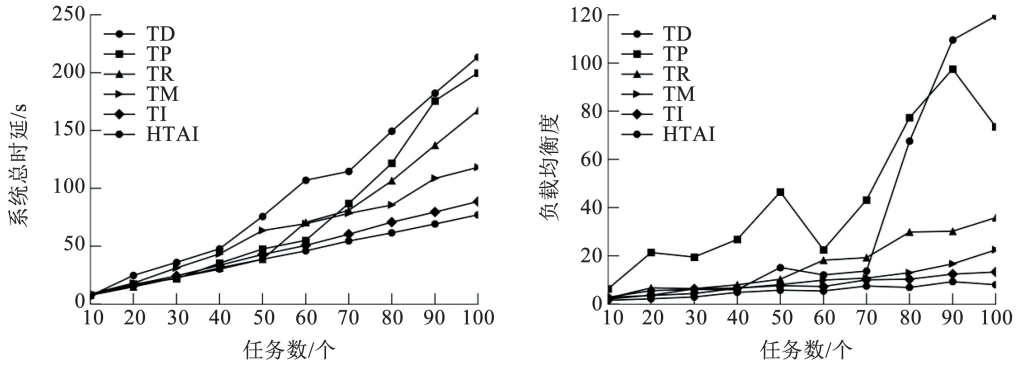
为展示边缘服务器负载均衡度变化情况,设计负载均衡度计算公式:

$$\sigma = K\sqrt{\frac{1}{n(s)}\sum_{s=1}^{n(s)}(t_s-\bar{t})^2} \tag{18}$$

式中:K 为常数,取 $K=10$;t_s 为边缘服务器 s 执行所有计算任务所需的时间;$\bar{t}$ 为所有服务器的平均执行时间,负载均衡度越低,说明服务器负载越均衡。

观察各算法在单位时间内不同任务数 $n(i)$ 下总时延与负载均衡度变化情况,结果如图 5 所示。系统总时延方面,从图 5(a)可以看出,TR 算法总时延在 7 ~ 220 之间波动;TM 在 7 ~ 200 之间波动,其受任务数影响较大,当任务较少时,算法具有良好的性能,当任务数增加时,算法表现不佳;TD 在 8 ~ 170 之间波动;TP 在 8 ~ 120 之间波动;TI 在 7 ~ 90 之间波动;HTAI 在 7 ~ 80 之间波动。可以看出,在系统总时延上,TI 与 HTAI 算法效果显著优于其他算法,且 HTAI 算法相比 TI 算法,总时延相对降低了 2% ~ 13%。

在边缘服务器负载均衡度方面,从图 5(b)可以看出,TD 算法负载均衡度在 2 ~ 120 之间波动;TP 在 6 ~ 97 之间波动;TR 在 2 ~ 35 之间波动;TM 在 2 ~ 22 之间波动;TI 在 2 ~ 14 之间波动;HTAI 在 1 ~ 8 之间波动。可以看出,TM、TI、HTAI 算法负载均衡度稳定性显著大于 TD、TP、TR 算法,且负载均衡度越低,算法稳定性越强。负载均衡度稳定性排序为 HTAI > TI > TM > TR > TP > TD,且不同任务数下 HTAI 算法负载均衡度均优于其他算法。综上,HTAI 任务卸载算法在不同任务数下的系统总时延和负载均衡度均优于其他卸载算法,且具有更强的稳定性。



(a) 不同任务数下系统总时延变化　　(b) 不同任务数下负载均衡度变化

图 5　不同任务数下系统总时延和负载均衡度变化

Fig. 5　Variations in total system latency and load balance under different task quantities

4　结　语

为解决边缘计算中边缘设备高计算资源需求与高延迟敏感的矛盾,利用深度强化学习解决多智能体边缘环境中的任务卸载决策问题。结合边缘设备供电不足且计算能力有限和边缘服务器负载失衡的特点,建立以边缘服务器为核心的边云协作任务卸载模型,通过边缘服务器之间的协作优化任务卸载,改进 PPO 算法使其适配多智能体环境,通过离线训练、设计兼顾其他智能体影响的奖励机制、在特征中融入基于特定智能体的全局信息,提出 I-PPO 算法,该算法对多智能体环境具有良好的适应性;基于 I-PPO算法提出任务卸载算法 HTAI,通过在任务卸载执行过程中引入任务优先级调度,进一步优化系统总时延。仿真实验验证了所提算法的有效性,I-PPO算法相比 PPO、PPO-1、PPO-2 算法,收敛速度更快、寻优能力更强,且具有更高的鲁棒性。HTAI 算法在不同任务数下,相比 TI、TP、TD、TM、TR 算法,能有效降低系统总时延,均衡边缘服务器负载,并具有更强的稳定性。

参考文献

[1] SHI Weisong, CAO Jie, ZHANG Quan, et al. Edge computing: vision and challenges[J]. IEEE Internet of Things Journal, 2016, 3(5): 637. DOI: 10.1109/JIOT.2016.2579198

[2] WU Huaming, SUN Yi, WOLTER K. Energy-efficient decision making for mobile cloud offloading[J]. IEEE Transactions on Cloud Computing, 2018, 8(2): 570. DOI: 10.1109/TCC.2018.2789446

[3] 李小平,周志星,陈龙,等. 异构边缘资源的任务卸载和协同调度[J]. 计算机研究与发展,2023,60(6):1296
LI Xiaoping, ZHOU Zhixing, CHEN Long, et al. Task offloading and collaborative scheduling of heterogeneous edge resources[J]. Journal of Computer Research and Development, 2023, 60(6): 1296. DOI: 10.7544/issn1000-1239.202110936

[4] TENG Haojun, LI Zhetao, CAO Kun, et al. Game theoretical task offloading for profit maximization in mobile edge computing[J]. IEEE Transactions on Mobile Computing, 2022, 21(10): 3519. DOI: 10.1109/TMC.2022.3175218

[5] ZHAO Fengjun, CHEN Ying, ZHANG Yongchao, et al. Dynamic offloading and resource scheduling for mobile-edge computing with energy harvesting devices[J]. IEEE Transactions on Network and Service Management, 2021, 18(2): 2154. DOI: 10.1109/TNSM.2021.3069993

[6] WU H M, CHEN J Q, NGUYEN T N, et al. Lyapunov-guided delay-aware energy efficient offloading in IIoT-MEC systems[J]. IEEE Transactions on Industrial Informatics, 2023, 19(2): 2117. DOI: 10.1109/TII.2022.3206787

[7] BOZORGCHENANI A, MASHHADI F, TARCHI D, et al. Multi-objective computation sharing in energy and delay constrained mobile edge computing environments[J]. IEEE Transactions on Mobile Computing, 2021, 20(10): 2992. DOI: 10.1109/TMC.2020.2994232

[8] HOU Chen, ZHAO Qianchuan. Optimal task-offloading control for edge computing system with tasks offloaded and computed in sequence[J]. IEEE Transactions on Automation Science and Engineering, 2023, 20(2): 1378. DOI: 10.1109/TASE.2022.3176745

[9] LI Shuyang, HU Xiaohui, DU Yongwen. Deep reinforcement learning and game theory for computation offloading in dynamic edge computing markets[J]. IEEE Access, 2021, 9: 121456. DOI: 10.1109/ACCESS.2021.3109132

[10] LI Gang, MIAO Jingbo, WANG Zihou, et al. An adaptive user service deployment strategy for mobile edge computing[J]. China Communications, 2022, 19(10): 238. DOI: 10.23919/JCC.2022.00.032

[11] LI Haiyuan, ASSIS K D R, YAN Shuangyi, et al. DRL-Based long-term resource planning for task offloading policies in multi-server edge computing networks[J]. IEEE Transactions on Network and Service Management, 2022, 19(4): 4151. DOI: 10.1109/TNSM.2022.3191748

[12] HERNANDEZ-LEAL P, KAISERS M, HERNANDEZ-LEAL T B,

et al. A survey of learning in multi-agent environments: Dealing with non-stationarity[J/OL]. arXiv, (2019 - 3 - 11)[2023 - 6 - 24]. https://arxiv.org/abs/1707.09183

[13] LIU Xiaolan, YU Jiadong, FENG Zhiyong, et al. Multi-agent reinforcement learning for resource allocation in IoT networks with edge computing[J]. China Communications, 2020, 17(9): 220. DOI: 10.23919/JCC.2020.09.017

[14] ZHANG Yutong, DI Boya, ZHENG Zijie, et al. Distributed multi-cloud multi-access edge computing by multi-agent reinforcement learning[J]. IEEE Transactions on Wireless Communications, 2021, 20(4): 2565. DOI: 10.1109/TWC.2020.3043038

[15] HUANG Xiaoyan, LENG Supeng, MAHARJAN S, et al. Multi-agent deep reinforcement learning for computation offloading and interference coordination in small cell networks[J]. IEEE Transactions on Vehicular Technology, 2021, 70(9): 9282. DOI: 10.1109/TVT.2021.3096928

[16] 谢人超,廉晓飞,贾庆民,等. 移动边缘计算卸载技术综述[J]. 通信学报,2018,39(11):138
XIE Renchao, LIAN Xiaofei, JIA Qingmin, et al. Overview of mobile edge computing offload technology[J]. Journal on Communications, 2018, 39(11): 138. DOI: 10.11959/j.issn.1000 - 436x.2018212

[17] VARGHESE B, WANG Nan, BARBHUIYA S, et al. Challenges and opportunities in edge computing[C]//2016 IEEE International Conference on Smart Cloud (SmartCloud). New York, USA: IEEE, 2016: 20. DOI: 10.1109/SmartCloud.2016.18

[18] XIAO M B, SHROFF N B, CHONG E K P. A utility-based power-control scheme in wireless cellular systems[J]. IEEE/ACM Transactions on Networking, 2003, 11(2): 210. DOI: 10.1109/TNET.2003.810740

[19] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal policy optimization algorithms[J/OL]. arXiv, (2017 - 8 - 28)[2023 - 6 - 24]. https://arxiv.org/abs/1707.06347

[20] YU C, VELU A, VINITSKY E, et al. The surprising effectiveness of PPO in cooperative, multi-agent games[J/OL]. arXiv, (2022 - 11 - 4)[2023 - 6 - 24]. https://arxiv.org/abs/2103.01955

[21] ZHANG Kaiqing, YANG Zhuoran, BASAR T. Multi-Agent reinforcement learning: A selective overview of theories and algorithms[J/OL]. arXiv, (2021 - 4 - 28)[2023 - 6 - 24]. https://arxiv.org/abs/1911.10635

[22] HOAT D, KIM D S. Dynamic collaborative task offloading for delay minimization in the heterogeneous fog computing systems[J]. Journal of Communications and Networks, 2023, 25(3): 465. DOI: 10.23919/JCN.2023.000008

[23] 苏命峰, 王国军, 李仁发. 基于利益相关视角的多维 QoS 云资源调度方法[J]. 通信学报, 2019, 40(6): 102.
SU Mingfeng, WANG Guojun, LI Renfa. Multidimensional QoS cloud computing resource scheduling method based on stakeholder perspective[J]. Journal on Communications, 2019, 40(6): 102. DOI: 10.11959/j.issn.1000 - 436x.2019115

(编辑　丁晓清)

(上接第 109 页)

[22] LIU Xuya, HAO Caiyan, SU Zezhao, et al. Image inpainting algorithm based on tensor decomposition and weighted nuclear norm[J]. Multimedia Tools and Applications, 2022, 82(3): 1. DOI: 10.1007/s11042 - 022 - 12635 - 3

[23] WANG Li, HOU Wensheng, WU Xiaoying, et al. 3D MR image denoising using a modified adaptive high order singular value decomposition method[C]//2020 42nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). Montreal, QC, Canada, 2020: 1580. DOI: 10.1109/EMBC44109.2020.9175418

[24] 蒋明峰, 陆亮, 吴龙, 等. 基于加权 Schatten p 范数最小化的磁共振图像重构方法研究[J]. 电子学报, 2019, 47(4): 784
JIANG Mingfeng, LU Liang, WU Long, et al. The Research of MRI reconstruction method by using weighted Schatten P-norm minimization[J]. Acta Electronica Sinica, 2019, 47(4): 784. DOI: 10.3969/j.issn.0372 - 2112.2019.04.003

[25] WANG Z, BOVIK A, SHEIKH H, et al. Image quality assessment: From error visibility to structural similarity[J]. IEEE Transactions on Image Processing, 2004, 13(4): 600. DOI: 10.1109/TIP.2003.819861

[26] RAVISHANKAR S, BRESLER Y. MR image reconstruction from highly undersampled k-space data by dictionary learning[J]. IEEE Transactions on Medical Imaging, 2011, 30(5): 1028. DOI: 10.1109/TMI.2010.2090538

[27] JUNG H, SUNG K, NAYAK K S, et al. k-t FOCUSS: A general compressed sensing framework for high resolution dynamic MRI[J]. Magnetic Resonance in Medicine, 2009, 61(1): 103. DOI: 10.1002/mrm.21757

[28] CHEN Chong, LIU Yingmin, SCHNITER P. OCMR (v1.0)-open-access multi-coil k-space dataset for cardiovascular magnetic resonance imaging[J]. Image and Video Processing, 2020: 1. DOI: 10.48550/arXiv.2008.03410

[29] TYGERT M, ZBONTAR J. Simulating single-coil MRI from the responses of multiple coils[J]. Image and Video Processing, 2018, 15(2): 115. DOI: 10.2140/camcos.2020.15.1

[30] SEGARS W P, TSUI B M. Study of the efficacy of respiratory gating in myocardial SPECT using the new 4-D NCAT phantom[J]. IEEE Transactions on Nuclear Science, 2002, 49(3): 675. DOI: 10.1109/TNS.2002.1039548

[31] SHARIF B, BRESLER Y. Adaptive real-time cardiac MRI using PARADISE: Validation by the physiologically improved NCAT phantom[C]//2007 4th IEEE International Symposium on Biomedical Imaging: From Nano to Macro. Arlington, VA, USA, 2007: 1020. DOI: 10.1109/ISBI.2007.357028

(编辑　丁晓清)