ISSN 1004-9037,CODEN SCYCE4
Journal of Data Acquisition and Processing Vol. 40,No. 1,Jan. 2025,pp. 163－175
DOI:10. 16337/j. 1004-9037. 2025. 01. 012

http://sjcj. nuaa. edu. cn
E-mail:sjcj@nuaa. edu. cn
Tel/Fax:＋86-025-84892742

联合WPT和MEC的无线传感网时延优化算法

张　健[1]，刘鹏博[1]，汤　健[2]

(1. 南京信息工程大学计算机学院,南京 210044;2. 北京工业大学信息科学技术学院,北京 100124)

摘　要：无线传感网络(Wireless sensor network，WSN)受电池能量有限和计算能力不足的约束,使得电池续航能力成为其广泛部署的瓶颈。本文利用无线电能传输(Wireless power transmission，WPT)和多接入边缘计算(Multi-access edge computing，MEC)技术,在传感器节点能耗受限的情况下,通过联合优化节点卸载决策、无线供电时长和带宽资源分配,最大限度地降低了传感器节点的任务平均完成时延。本文将优化问题建模成混合整数规划问题,并且为了适应复杂动态的信道环境,提出了一种基于柔性动作评价(Soft actor critic，SAC)的时延最小化深度强化学习算法(Deep reinforcement learning delay minimization，DrlDM),将原始优化问题建模成马尔可夫决策过程(Markov decision process，MDP)。仿真结果表明,与3种基线实验相比,本文提出的DrlDM算法平均延迟降低62.11%,显著缩短了节点的任务平均完成时间。

关键词：多接入边缘计算;深度强化学习;无线传感网络;无线电能传输;计算卸载
中图分类号：TN92　　**文献标志码**:A

Delay Optimization Algorithm in Wireless Sensor Networks Combining WPT and MEC

ZHANG Jian[1]，LIU Pengbo[1]，TANG Jian[2]

(1. School of Computer Science，Nanjing University of Information Science and Technology，Nanjing 210044，China;2. School of Information Science and Technology，Beijing University of Technology，Beijing 100124，China)

Abstract：Wireless sensor network (WSN) is constrained by limited battery energy and insufficient computing power，and the limited battery life hinders its widespread deployment. In this paper，wireless power transmission (WPT) and multi-access edge computing (MEC) technologies are used to solve the problem of limited energy consumption of sensor nodes. By jointly optimizing the decision of the node offloading，wireless power supply duration and bandwidth resource allocation，the average task completion delay of sensor nodes is minimized to the greatest extent possible. The optimization problem is modeled as a mixed integer programming problem. In order to adapt to the complex and dynamic channel environment，a deep reinforcement learning delay minimization (DrlDM) algorithm based on soft actor critic (SAC) is proposed. The original optimization problem is modeled as a Markov decision process (MDP). Simulation results show that compared with three baseline experiments，the average delay of the DrlDM algorithm proposed in this paper is reduced by 62.11 %，significantly shortening the average task completion time of nodes.

基金项目:国家和北京市重点实验室基金(BGRIMM-KZSKL-2020-02);国家自然科学基金(62073006)。
收稿日期:2024-03-23;**修订日期**:2024-08-01

Key words: multi-access edge computing (MEC); deep reinforcement learning; wireless sensor network (WSN); wireless power transmission (WPT); computation offloading

引　言

无线传感网络(Wireless sensor network, WSN)具有能量供应受限、计算资源有限等缺点,一旦节点的能量耗尽,WSN将无法正常工作,且在特定场景下,例如工业物联网、森林火灾监控以及空气质量监测等[1],频繁地更换电池也不切实际。因此,解决传感器节点的电池容量有限和计算效率低下这两个问题成了WSN各项应用的关键。

多接入边缘计算(Multi-access edge computing, MEC)允许为靠近边缘服务器的移动节点提供计算资源,从而将任务卸载到边缘服务器,以降低计算负担,被认为是解决移动节点计算能力和能量受限问题的关键技术[2-6]。文献[7]针对单用户MEC系统提出了部分卸载调度和功率分配方法,在保证用户发射功率约束的同时,联合最小化MEC服务器中的任务执行延迟和能量消耗。文献[8]提出了一个低复杂度高效的启发式算法来解决MEC中的在线最优服务选择、资源分配和任务卸载等问题,以最大化用户服务质量。文献[9]提出分布式和集中式的解决方案,旨在通过合理的任务分配和调度策略,实现最大化MEC系统利润。文献[10]将MEC技术应用到WSN中,通过对WSN中网络的能量和时间约束建模,考虑网络的长期平均信息年龄的优化。文献[11]针对WSN中任务处理的能效问题引入MEC技术,提出了一种近似最优化的任务处理机制,实现动态卸载和任务处理。然而,目前有关MEC和WSN的研究很少考虑到受时延约束的随机任务到达问题。

无线电能传输(Wireless power transmission, WPT)技术可以使无线节点通过接收射频信号发射器广播的射频信号,获取能量以确保持续工作[12-13],是为MEC网络提供可持续能源的有效解决方案之一。文献[14]使用反向散射通信技术,对卸载决策、时间分配、后向散射系数和发射功率进行了优化,最大化计算速率。文献[15]中,用户基于非正交多址接入技术进行部分卸载,采用迭代算法对时间分配、传输功率和计算频率进行联合优化,最大化计算能效。文献[16]考虑最小化总计算延迟,提出了一种基于深度强化学习的离线任务卸载算法,设计了一种用于优化无线电能传输时长和传输时间分配的最差节点调整算法,以最小化计算延迟。以上文献均未考虑WPT与任务卸载时产生的信道竞争问题,而合理分配信道占用时间能够有效提高用户计算效率。

近年来大量的研究工作表明,深度强化学习算法可以有效解决MEC网络中的计算卸载以及资源分配问题。文献[17]提出了一种基于强化学习的资源管理算法,该算法学习卸载和边缘服务器供应的最优策略,以最小化长期系统成本。文献[18]研究了MEC场景下的计算卸载和资源分配问题,提出了基于值迭代的强化学习方法和基于双重深度Q网络的方法,最小化整个系统的能耗。文献[19]考虑了部分卸载的情况,提出了一个Q-learning方案和一个基于深度确定性策略梯度(Deep deterministic policy gradient, DDPG)的解决方案,最小化系统延迟。文献[20]将柔性动作评价(Soft actor critic, SAC)算法应用到无人机网络中,通过联合优化飞行接入点的三位部署和功率分配,实现最小用户速率最大化。文献[21]研究了多用户MEC的物联网系统中网络寿命优化问题,提出了一种基于SAC的生命周期最大化算法,通过实验验证了SAC算法的有效性。

综上所述,尽管引入WPT可以缓解节点的电池能量瓶颈问题,但由于半双工通信的限制,WPT占用了信道资源,不可避免地会增加节点的任务处理时间。例如,充电时间过短可能导致节点能量匮乏,无法完成任务处理;而充电时间过长,则会增加节点等待任务处理的时间。在过去的研究中,一些工作已经考虑了MEC中计算延迟最小化的问题。文献[19]使用部分卸载模式,但实际场景中,任务很难进

行细粒度划分。文献[16]通过采用时分多址的方式为用户分配信道资源,但这可能会导致不必要的等待时间,且忽略了用户的公平性。文献[14-16]中假设每次节点通过WPT获得的电能可以完全利用于本地计算或者计算卸载,但实际情况并非如此理想,因为节点在同一时间收集到的能量、需要计算的数据量以及信道状态可能不同,将导致能量剩余情况。本文围绕上述待解决问题展开研究。

1　系统模型

1.1　网络模型

考虑1个由多个传感器节点和1个接入点(Access point,AP)组成的无线供电MEC网络,将节点表示为集合$\boldsymbol{M}=\{1,2,\cdots,M\}$,AP和每个节点都配备1个单天线。AP拥有稳定的能量来源并且配备1个计算服务器,而每个节点则携带1个可充电电池。AP能够通过下行链路向每个节点广播射频信号,为节点提供能量。在这个网络中,假设每个节点的电池能量仅来自于AP[22]。节点可以通过收集到的能量本地计算所需的任务,或通过上行链路将任务上传到AP进行计算。

在实际场景中,节点需要计算的任务很难被任意划分进行计算,所以本文采取二进制的卸载策略[14,16],即节点会将任务本地计算或者全部上传到服务器进行计算,用x_m表示节点m的卸载决策,$x_m=1$表示节点m将任务卸载到服务器进行计算;$x_m=0$表示节点m将本地计算该任务。

将系统时间划分为每个长度为T的连续时间帧,在每个时间帧T中,按照节点能量的收集和消耗可以划分成两部分,如图1所示。在每个时隙开始时,每个节点都会有一定的概率随机产生1个具有时延约束的计算密集型任务,假设节点的任务都能在1个时隙结束前完成计算[22]。所有节点都会在时隙开始时进行能量收集,充电结束后,节点消耗电池的能量,进行任务卸载或者本地计算。假设服务器的计算能力远远高于本地计算能力,所以忽略了服务器处理任务的时间,且假设返回结果的数据量远远小于上传数据量,因此忽略了结果返回的时间[16,22]。

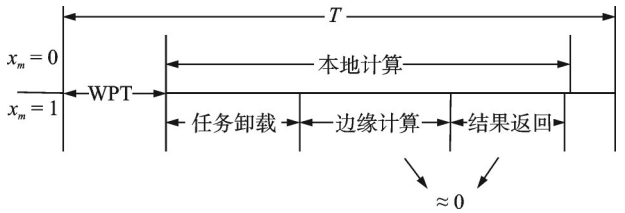

图1　系统时间模型图

Fig.1　System time model diagram

图2展示了1个由5个节点和1个AP组成的无线供电多接入边缘计算(Wireless power multi-access edge computing,WPMEC)网络,AP配有1个服务器且有着稳定的能量来源,每个节点配备了1个独立的可充电电池。在能量收集期间,AP将射频信号广播到每个节点,5个节点同时从射频信号中收集能量。收集结束后,节点1、2和5将同时将任务卸载到AP进行计算,节点3、4本地计算该任务。

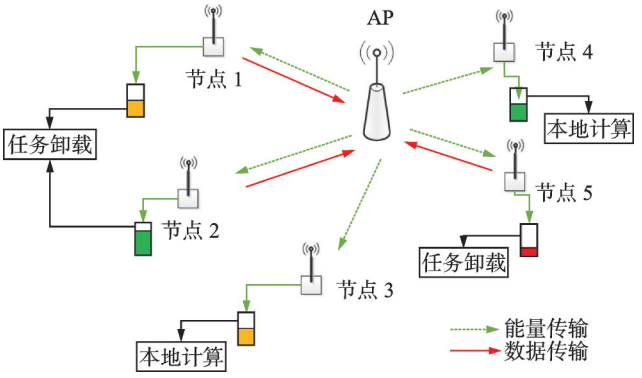

图2　WPMEC系统模型

Fig.2　WPMEC system model

1.2　任务模型

考虑到更加现实的场景,在每个时隙开始时,每个节点都会有一定的概率随机产生1个具有时延约束的计算密集型任务,为了更加直观地表示任务信息,用$\lambda_t^m=(L_m,D_m)$来表示节点m在时隙t产生的任

务信息，其中L_m代表任务的数据量，单位为bit，D_m表示这个任务的期望完成时间，单位为s。不失一般性，当节点m在时隙t没有产生任务时，$\lambda_t^m=(0,0)$。

1.3 通信模型

在通信模型中，采用频分复用作为通信方案，假设通信中的上行链路和下行链路的信道增益相同，且在1个时隙内保持不变，但可能在不同的时隙中发生变化。设h_m表示节点m和AP之间的信道增益，根据自由空间路径衰落模型[16]，上行链路和下行链路的大规模衰落$\overline{h_m}$可表示为

$$\overline{h_m}=A_d\left(\frac{3\times10^8}{4\pi f_c d_m}\right)^{d_e} \tag{1}$$

式中：A_d表示天线增益；f_c表示载波频率；d_m表示节点m和AP之间的距离；d_e表示路径损耗指数。根据瑞利衰落模型，信道增益$\boldsymbol{h}=[h_1,h_2,\cdots,h_m]$可以表示为

$$h_m=\overline{h_m}\gamma_m \tag{2}$$

式中γ_m表示独立随机信道衰落因子，服从单位均值的指数分布。给定节点计算任务的通道增益样本是独立的。

节点被分配的带宽资源取决于AP端的决策，用M_t^{off}表示在t时刻选择任务卸载的节点集合，即$M_t^{\text{off}}=\{m|\forall x_m=1,m\in\boldsymbol{M}\}$，节点$m$在$t$时刻被分配的带宽资源的权重$\xi_t^m\in\Omega_t$，$\Omega_t$是1组AP给出的权重系数，所以节点$m$在$t$时刻被分配的实际带宽资源$B_t^m$表示为

$$B_t^m=W\frac{\xi_t^m}{\sum\limits_{i\in M_t^{\text{off}}}\xi_t^i} \tag{3}$$

式中W为AP的总带宽资源。根据香农定理，可以得出t时刻节点m的上行传输速率r_t^m(bit/s)为

$$r_t^m=B_t^m\log_2\left(1+\frac{P_m h_m}{B_t^m\sigma^2}\right) \tag{4}$$

式中：P_m为节点的发射机功率；σ^2为噪声谱密度。

1.4 计算模型

在此网络模型中，用$E_{m,t}^{\text{mob}}$表示节点m在时隙t开始时的可用能量，不失一般性，$E_{m,0}^{\text{mob}}=0$。节点m在时隙t收集到的能量E_t^m可以表示为

$$E_t^m=\mu P h_m\theta \tag{5}$$

式中：θ表示AP发射射频信号的时间；$\mu\in(0,1)$表示能量收集效率；P表示AP的发射功率。那么在节点执行卸载决策前，节点可用电量可以表示为

$$E_{m,t}^{\text{mob}}=E_{m,t-1}^{\text{mob}}+E_t^m \tag{6}$$

1.4.1 本地计算

节点m选择本地计算时，设f_m为节点m本地计算能力，单位为周期/s，T_m^{loc}为节点m本地计算的时间，$0\leqslant T_m^{\text{loc}}\leqslant T$，用$\phi$表示节点本地处理1 bit任务数据所需的CPU周期数，其中$\phi>0$，因此T_m^{loc}可表示为

$$T_m^{\text{loc}}=\frac{L_m\phi}{f_m} \tag{7}$$

用e_m^{loc}表示节点m本地计算所产生的能耗，那么有

$$e_m^{\text{loc}}=kf_m{}^3T_m^{\text{loc}}\leqslant E_{m,t}^{\text{mob}} \tag{8}$$

式中k为计算能效系数。

为了使节点本地计算任务时的时间最小，节点要在满足能量约束的情况下最大化计算频率，当

$f_m=f_{\max}$时,本地计算时间为

$$T_m^{\mathrm{loc}}=\frac{L_m\phi}{f_{\max}} \tag{9}$$

本地最大计算能耗$e_m^{\mathrm{loc,max}}$为

$$e_m^{\mathrm{loc,max}}=kf_{\max}{}^2L_m\phi \tag{10}$$

所以当$e_m^{\mathrm{loc,max}}\leqslant E_{m,t}^{mob}$时,$f_m=f_{\max}$能在保证能量约束的情况下最小化计算时间,反之当$e_m^{\mathrm{loc,max}}>E_{m,t}^{\mathrm{mob}}$时,$f_m$应表示为

$$f_m=\left(\frac{E_{m,t}^{\mathrm{mob}}}{T_m^{\mathrm{loc}}k}\right)^{\frac{1}{3}} \tag{11}$$

所以本地计算时间T_m^{loc}表示为

$$T_m^{\mathrm{loc}}=\sqrt{\frac{\phi^3L_m^3k}{E_{m,t}^{\mathrm{mob}}}} \tag{12}$$

综上所述,本地计算时间T_m^{loc}表示为

$$T_m^{\mathrm{loc}}=\begin{cases}\dfrac{L_m\phi}{f_{\max}} & e_m^{\mathrm{loc,max}}\leqslant E_{m,t}^{\mathrm{mob}}\\ \sqrt{\dfrac{\phi^3L_m^3k}{E_{m,t}^{\mathrm{mob}}}} & e_m^{\mathrm{loc,max}}>E_{m,t}^{\mathrm{mob}}\end{cases} \tag{13}$$

本地计算结束后节点可用电量表示为

$$E_{m,t+1}^{\mathrm{mob}}=E_{m,t}^{\mathrm{mob}}-e_m^{\mathrm{loc}} \tag{14}$$

1.4.2　任务卸载

当节点选择任务卸载时,T_m^{tran}表示节点m任务卸载到AP所花费的时间,为了保证任务数据能完整的传输给AP,应有以下约束

$$T_m^{\mathrm{tran}}r_t^m=T_m^{\mathrm{tran}}B_t^m\log_2\left(1+\frac{P_mh_m}{B_t^m\sigma^2}\right)\geqslant L_m \tag{15}$$

用e_m^{off}表示任务传输所产生的能耗,那么有

$$e_m^{\mathrm{off}}=P_mT_m^{\mathrm{tran}} \tag{16}$$

同样,发射机功率P_m表示为

$$P_m=\frac{E_{m,t}^{\mathrm{mob}}}{T_m^{\mathrm{tran}}} \tag{17}$$

卸载时间T_m^{tran}满足以下等式

$$T_m^{\mathrm{tran}}=\frac{L_m}{B_t^m\log_2\left(1+\dfrac{E_{m,t}^{\mathrm{mob}}h_m}{T_m^{\mathrm{tran}}B_t^m\sigma^2}\right)} \tag{18}$$

任务卸载结束后节点可用电量表示为

$$E_{m,t+1}^{\mathrm{mob}}=E_{m,t}^{\mathrm{mob}}-e_m^{\mathrm{off}} \tag{19}$$

1.5　问题制定

节点m在t时刻的任务处理时延T_m^t表示为

$$T_m^t=(1-x_m)T_m^{\mathrm{loc}}+x_mT_m^{\mathrm{tran}}+\theta \tag{20}$$

在t时刻处理完所有节点的平均时延T_t^{total}可以表示为

$$T_t^{\text{total}}=\frac{1}{\boldsymbol{M}}\sum_{m\in\boldsymbol{M}}((1-x_m)T_m^{\text{loc}}+x_mT_m^{\text{tran}}+\theta) \tag{21}$$

本文的目标是最小化所有时间段的节点平均任务完成延迟,问题描述为P1,即

$$\begin{cases}\text{P1: }\min\lim\limits_{t\to\infty}\dfrac{1}{t}\sum\limits_{v=1}^{t}T_v^{\text{total}}\\ \text{s.t.}\\ \text{C}_1\text{: }0<\theta<T\\ \text{C}_2\text{: }x_m\in\{0,1\},\forall m\in\boldsymbol{M}\\ \text{C}_3\text{: }T_m^{\text{tran}}B_t^m\log_2\left(1+\dfrac{P_mh_m}{\sigma^2}\right)\geqslant L_m,\forall m\in\boldsymbol{M}\\ \text{C}_4\text{: }kf_m{}^3T_m^{\text{loc}}\leqslant E_m^{\text{mob}}(t),\forall m\in\boldsymbol{M}\\ \text{C}_5\text{: }0\leqslant e_m^{\text{loc}}\leqslant E_m^{\text{mob}}(t),\forall m\in\boldsymbol{M}\\ \text{C}_6\text{: }0\leqslant e_m^{\text{off}}\leqslant E_m^{\text{mob}}(t),\forall m\in\boldsymbol{M}\end{cases} \tag{22}$$

问题P1是一个混合整数规划问题,其在多项式时间内求解具有挑战性,深度强化学习可以适应复杂时变的系统信道状态,已经被广泛应用。为了应对这一挑战,本文提出了一种基于SAC的深度强化学习方法来求解这个问题。

2 基于SAC的时延最小化深度强化学习算法

在本节中,首先将原始问题建模成马尔可夫决策过程(Markov decision process,MDP),在每个时隙开始时,每个节点将本地信息发送给AP。AP通过观察系统状态,包括任务大小、可用电量和信道增益,然后进行充电时间、卸载决策以及相应的带宽资源分配,在1个任务结束后,会产生一定的成本。随后提出了一种基于SAC的时延最小化深度强化学习算法DrlDM,最后将原始问题转化为带有约束的MDP优化问题。

2.1 MDP制定

2.1.1 状态

在t时隙开始时,用$\boldsymbol{\lambda}_t=[\lambda_t^1,\lambda_t^2,\cdots,\lambda_t^m]$表示节点的任务信息,$\boldsymbol{H}_t=[h_1,h_2,\cdots,h_m]$表示节点与AP之间的信道增益,$\boldsymbol{E}_{m,t}^{\text{mob}}=[E_{1,t}^{\text{mob}},E_{2,t}^{\text{mob}},\cdots,E_{m,t}^{\text{mob}}]$表示节点可用电量。具体来说,AP在做出动作前将观察到状态$\boldsymbol{s}_t$,即

$$\boldsymbol{s}_t=(\boldsymbol{\lambda}_t,\boldsymbol{H}_t,\boldsymbol{E}_{m,t}^{\text{mob}}) \tag{23}$$

2.1.2 动作

当AP观察到系统状态$\boldsymbol{s}_t$后,AP将为节点选择动作:(a)节点在本地处理该任务或者进行任务卸载;(b)确定节点广播射频信号的时间;(c)为节点动态分配带宽资源。因此,AP做出的动作$\boldsymbol{a}_t$表示为

$$\boldsymbol{a}_t=(\boldsymbol{X},\theta,\boldsymbol{\Omega}_t) \tag{24}$$

式中$\boldsymbol{X}=[x_1,x_2,\cdots,x_m]$表示节点的卸载决策。

2.1.3 奖励

奖励函数的设计对于强化学习算法的收敛和性能有着重要的影响,为此,本文对该网络场景设计了一个奖励机制。具体来说,当1个时隙结束时,任务的执行情况有3种可能性:(1)任务在约束时间内被计算完成;(2)任务在计算时电量提前被耗尽;(3)任务的计算时间超出任务的时间约束。

对于情况(1),将节点的任务完成时间作为成本纳入到奖励计算中,具体表示为

$$C^{\mathrm{suc}}=\sum_{m\in I^{\mathrm{suc}}}T_m^t \tag{25}$$

式中I^{suc}表示情况(1)的节点集合。对于情况(2)和(3)将给定一个惩罚值,具体表示为

$$C^{\mathrm{drop}}=\sum_{m\in I^{\mathrm{drop}}}\bar{\omega} \tag{26}$$

式中I^{drop}表示情况(2)和(3)的节点集合,$\bar{\omega}$为1个常量惩罚值。故对于状态动作对$(\boldsymbol{s}_t,\boldsymbol{a}_t)$的奖励$r(\boldsymbol{s}_t,\boldsymbol{a}_t)$表示为

$$r(\boldsymbol{s}_t,\boldsymbol{a}_t)=-(C^{\mathrm{suc}}+C^{\mathrm{drop}}) \tag{27}$$

2.2　算法设计

SAC算法是一种适用于解决连续动作空间的深度强化学习算法,其独特之处在于引入了最大化熵的思想,以在探索和利用之间取得平衡。SAC的目标是通过优化策略网络和值函数网络来最大化期望累积奖励,同时最大化动作分布的熵,从而鼓励智能体产生更具有探索性的行为,SAC算法的优化目标可以表示为

$$J(\pi)=\sum_{t=0}^{T}\mathbb{E}_{(\boldsymbol{s}_t,\boldsymbol{a}_t)\sim\rho_\pi}[r(\boldsymbol{s}_t,\boldsymbol{a}_t)+\alpha\mathcal{H}(\pi(\cdot|\boldsymbol{s}_t))] \tag{28}$$

式中α为一个正则化的系数,用来控制熵的重要程度。α越大,探索性就越强,有助于加速后续的策略学习;反之α越小,就会增加策略陷入较差的局部最优的可能性。ρ_π表示策略π的状态分布,其中策略熵$\mathcal{H}$表示为

$$\mathcal{H}(\pi(\cdot|s))=\mathop{\mathbb{E}}_{a\sim\pi(\cdot|s)}[-\log\pi(a|s)] \tag{29}$$

具体来说,SAC网络模型由1个Actor网络和4个Critic网络组成,算法网络模型如图3所示。下面将分别从Critic网络、Actor网络以及自适应熵正则项来分别描述本文基于SAC设计的DrlDM算法。

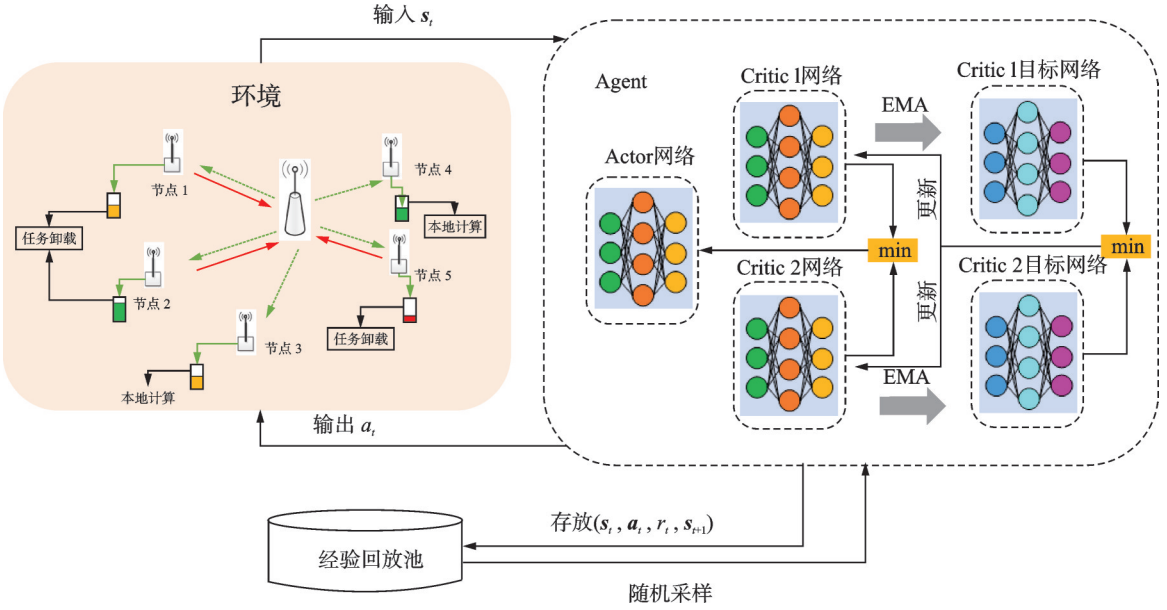

图3　DrlDM算法框架

Fig.3　Framework the DrlDM algorithm

2.2.1 Critic网络

对于Critic网络，由2个Q网络和2个目标Q网络组成。Critic网络接收来自Actor网络的动作状态对$(\boldsymbol{s}_t, \boldsymbol{a}_t)$作为输入，经过全连接的深度神经网络(Deep neural networks，DNN)输出一个期望值作为动作状态对的评估。Critic网络在每次使用Q网络时，会挑选一个Q值较小的网络，从而缓解Q值过高这一问题，目标Q网络这一设置用于提高训练当中的稳定性。任意一个Q网络的损失函数可以表示为

$$L_Q(w)=\mathbb{E}_{(\boldsymbol{s}_t,\boldsymbol{a}_t,r_t,\boldsymbol{s}_{t+1})\sim B}\left[\frac{1}{2}\left(Q_w(\boldsymbol{s}_t,\boldsymbol{a}_t)-(r(\boldsymbol{s}_t,\boldsymbol{a}_t)+\gamma(\min_{j=1,2}Q_{w_j^-}(\boldsymbol{s}_{t+1},\boldsymbol{a}_{t+1})-\alpha\log\pi(\boldsymbol{a}_{t+1}|\boldsymbol{s}_{t+1})))\right)^2\right] \tag{30}$$

式中：w代表Q网络的网络参数；w^-代表目标Q网络的网络参数；B代表经验回放池；γ为折扣因子。每个目标Q网络的参数都采用指数移动平均(Exponential moving average，EMA)的更新方式，即有

$$w^- \leftarrow \varepsilon w+(1-\varepsilon)w^- \tag{31}$$

式中$0<\varepsilon\ll 1$是一个更新因子，EMA方法允许目标Q网络以平滑的方式跟随Q网络，这有助于减少更新的差异，有效避免学习过程中的不稳定波动。主Q网络参数w可以通过以下公式更新

$$w\leftarrow w-\delta_c\nabla_w L_Q(w) \tag{32}$$

式中δ_c表示主Q网络的学习率。

2.2.2 Actor网络

对于Actor网络，由1个全连接的DNN网络组成。Actor网络的输入是智能体Agent观察到的系统状态$\boldsymbol{s}_t$，根据状态$\boldsymbol{s}_t$输出高斯分布的均值和标准差，然后进行采样。由KL散度得到的Actor网络损失函数可以表示为

$$L_\pi(v)=\mathbb{E}_{\boldsymbol{s}_t\sim B,\boldsymbol{a}_t\sim\pi_v}[\alpha\log(\pi_v(\boldsymbol{a}_t|\boldsymbol{s}_t))-Q_w(\boldsymbol{s}_t,\boldsymbol{a}_t)] \tag{33}$$

式中v代表Actor网络的网络参数。由于根据高斯分布来采样动作的过程不可导，因此使用重参数化的方式，将对动作的期望重写为对噪声的期望，于是Actor网络的损失函数被改写为

$$L_\pi(v)=\mathbb{E}_{\boldsymbol{s}_t\sim B,\varsigma_t\sim\mathcal{N}}\left[\alpha\log(\pi_v(f_v(\varsigma_t;\boldsymbol{s}_t)|\boldsymbol{s}_t))-\min_{j=1,2}Q_{w_j}(\boldsymbol{s}_t,f_v(\varsigma_t;\boldsymbol{s}_t))\right] \tag{34}$$

式中$f_v(\varsigma_t;\boldsymbol{s}_t)=b_{\text{mean}}+\varsigma_t\odot b_{\text{std}}$，$\varsigma_t$表示采样得到的期望值，$b_{\text{mean}}$和$b_{\text{std}}$分别是Actor网络输出的均值和方差，“$\odot$”表示Hadamard乘积。Actor网络参数$v$可以通过以下公式更新

$$v\leftarrow v-\delta_a\nabla_v L_\pi(v) \tag{35}$$

式中δ_a表示Actor网络的学习率。

2.2.3 自适应熵正则项

在SAC算法中，熵正则项系数α的调整影响着熵值的大小。在不同的状态下，熵的大小需求各异，对于存在最优动作不确定性的状态，熵的取值宜较大；而对于某个状态下最优动作相对确定的情况，熵的取值则可适度减小。为了实现熵正则项的自动调整，将强化学习的目标重新构建为一个受约束的优化问题

$$\begin{cases}\max\limits_{\pi}\ \mathbb{E}_\pi[\sum\limits_t r(\boldsymbol{s}_t,\boldsymbol{a}_t)]\\ \text{s.t.}\ \ \mathbb{E}_{(\boldsymbol{s}_t,\boldsymbol{a}_t)\sim\rho_\pi}[-\log(\pi_t(\boldsymbol{a}_t|\boldsymbol{s}_t))]\geqslant\mathcal{H}^*\end{cases} \tag{36}$$

式(36)通过约束熵的均值大于$\mathcal{H}^*$的同时，最大化期望奖励，通过化简可以得到

$$L(\alpha)=\mathbb{E}_{\boldsymbol{s}_t\sim B,\boldsymbol{a}_t\sim\pi(\cdot|\boldsymbol{s}_t)}[-\alpha\log\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)-\alpha\mathcal{H}^*] \tag{37}$$

通过最小化式(37)目标可以自适应调整α的值。

影响所提出的DrlDM算法训练时间的主要因素是训练过程中涉及的Actor网络和Critic网络中

CNN网络的计算时间。这些网络的复杂度和所需的计算资源决定了整体训练时间的长短。Actor网络和Critic网络的梯度下降复杂度可以表示为$O\left(N_{\mathrm{BS}}\left(\sum_{i=0}^{I-1}l_i l_{i+1}+\sum_{j=0}^{J-1}\hat{l}_j\hat{l}_{j+1}\right)\right)$,其中$N_{\mathrm{BS}}$表示批量大小,$l_i$和$\hat{l}_j$分别表示Actor网络和Critic网络的第$i$层和第$j$层的神经元数量,$I$和$J$分别代表Actor网络和Critic网络中的全连接层数。在执行阶段,只需将Actor网络训练到理想状态,因此每个时隙的计算复杂度等同于通过参与者网络进行前向传递的计算复杂度,可以表示为$O\left(\sum_{j=0}^{J-1}\hat{l}_j\hat{l}_{j+1}\right)$。DrlDM算法的伪代码如下。

算法 1　DrlDM算法

Input:初始化Critic网络参数w_1、w_2、Actor网络参数ν,经验回放池B,目标Critic网络参数$\overline{w_1}$、$\overline{w_2}$;

Output:w_1、w_2、ν

(1) for episode $=0,1,\cdots,$do

(2)　初始化环境,观察状态s_0

(3)　for $t=0,1,\cdots,T$ do

(4)　　选择动作$\boldsymbol{a}_t\sim\pi_\nu(\boldsymbol{a}_t\mid\boldsymbol{s}_t)$

(5)　　采取动作$\boldsymbol{a}_t$,观察奖励r_t和下个状态$\boldsymbol{s}_{t+1}$

(6)　　存储四元组$(\boldsymbol{s}_t,\boldsymbol{a}_t,r_t,\boldsymbol{s}_{t+1})$到经验回放池$B$

(7)　　从经验回访池中采样N条数据$\{(\boldsymbol{s}_i,\boldsymbol{a}_i,r_i,\boldsymbol{s}_{i+1})\}_{i=1,2,\cdots,N}$

(8)　　计算目标Q值基于:

(9)　　$r_i+\gamma(\min\limits_{j=1,2}Q_{\bar{w}_j}(\boldsymbol{s}_{i+1},\boldsymbol{a}_{i+1})-\alpha\log\pi_\nu(\boldsymbol{a}_{i+1}|\boldsymbol{s}_{i+1})),\forall i\in\{1,2,\cdots,N\}$

(10)　　更新Critic-nets参数w_1,w_2基于:

(11)　　$L(w_j)=\dfrac{1}{N}\sum\limits_{i=1}^{N}(y_i-Q_{w_j}(\boldsymbol{s}_i,\boldsymbol{a}_i)),\forall j\in\{1,2\},\forall i\in\{1,2,\cdots,N\}$

(12)　　通过重参数化采样$\tilde{\boldsymbol{a}}_i$,更新Actor-net参数ν基于:

(13)　　$L_\pi(\nu)=\dfrac{1}{N}\sum\limits_{i=1}^{N}(\alpha\log\pi_\nu(\tilde{\boldsymbol{a}}_i|\boldsymbol{s}_i)-\min\limits_{j=1,2}Q_{w_j}(\boldsymbol{s}_i,\tilde{\boldsymbol{a}}_i)),\forall j\in\{1,2\},\ \forall i\in\{1,2,\cdots,N\}$

(14)　　更新温度参数α基于式(37)

(15)　　更新Critic Target-nets参数基于:$\overline{w_j}\leftarrow\varepsilon w_j+(1-\varepsilon)\overline{w_j},\forall j\in\{1,2\}$

(16)　end for

(17) end for

2.3　问题转换

在本文制定的优化问题中,目标是找到最优策略π^*,使得预期的长期成本最小化,于是将优化问题P1转换成了MDP优化问题P2,旨在解决以下MDP问题来找到最优策略π^*,即

$$\begin{cases}\mathrm{P2}:\pi^*=\arg\max\limits_{\pi}\mathbb{E}_\pi\left[\sum\limits_t r(\boldsymbol{s}_t,\boldsymbol{a}_t)+\alpha\mathcal{H}(\pi(\cdot|\boldsymbol{s}_t))\right]\\ \text{s.t. } \mathrm{C}_1\text{—}\mathrm{C}_6\end{cases}\tag{38}$$

式中C_1—C_6为式(22)中的约束条件。

本节完成了原始优化问题的转换,将节点任务平均延迟最小化问题转化为长期奖励最大化问题。

3 仿真结果与分析

3.1 仿真参数设置

节点随机到达的任务数据大小随机分布在[10,200]bit,节点与AP之间的距离随机分布在[2.5,5.2] m,载波频率$f_c=715$ MHz,天线增益$A_d=4.11$,能量收集效率$\mu=0.7$,路径损耗指数$d_e=2.5$。能效系数$k=10^{-26}$,节点处理1位任务数据所需的CPU周期数$\phi=100$,节点的最大本地计算频率$f_{max}=1.5\times10^7$周期/s,背景噪声方差$\sigma^2=10^{-12}$ W,对比实验中未涉及到的变量设置:AP发射机功率$P=3$ W,带宽$W=10$ MHz,节点数量为7个,参数设置部分见于文献[22]。

本文通过Pytorch框架使用Python实现了DrlDM算法。在DrlDM算法中,Critic网络和Actor网络都由包含1个输入层、2个隐藏层以及1个输出层的全连接DNN网络组成,其中每一个隐藏层的神经元数量都是128个。折扣因子$\gamma=0.99$,经验回放池B的容量为10^5,采样大小N设置为64。

3.2 仿真结果分析

图4和图5分别描述了Actor网络和Critic网络在不同学习率(Learning rate, LR)下的收敛曲线。在1 000次的迭代下,可以观察到Actor网络和Critic网络分别在学习率等于0.000 1和0.01时取得了最好的收敛效果,之后的实验均采用此套参数。

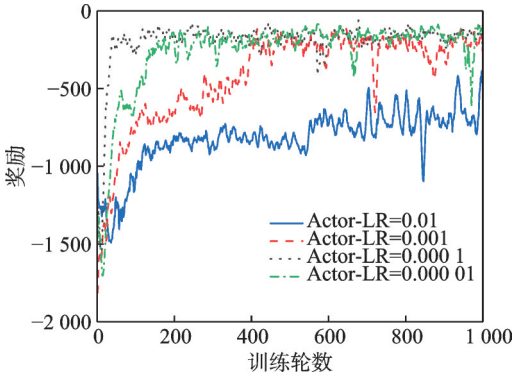

图4 Actor网络在不同学习率下的收敛结果

Fig.4 Convergence results of Actor networks under different learning rates

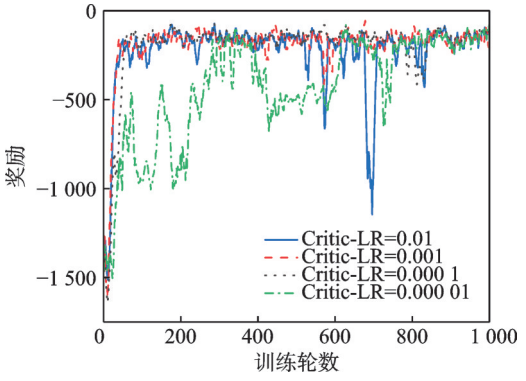

图5 Critic网络在不同学习率下的收敛结果

Fig.5 Convergence results of Critic network under different learning rates

为了展示DrlDM算法的性能,本文与基于DDPG的方案进行了1 000轮的奖励收敛对比。如图6所示,在7个节点数量的情况下,所提出的DrlDM算法大约在150轮可以收敛,与DDPG算法相比,DrlDM算法表现出了更快的收敛速度和更好的稳定性。

为了能够展示本文提出的DrlDM算法的有效性,与3种基线方案进行了性能比较,分别是本地自适应计算(Local computing,LC)、均分带宽边缘计算(Edge computing,EC)和基于DDPG的卸载方案。

(1) 本地自适应计算:节点将它们需要处理的任务全部在本地进行计算,在这个方案中,节点会根据当前可用电量自适应调整本地CPU计算频率,以保证任

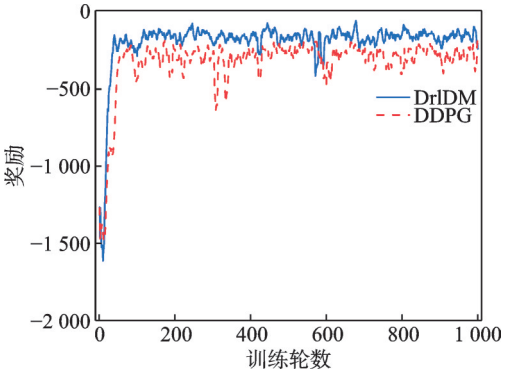

图6 DrlDM算法与DDPG算法收敛效果对比

Fig.6 Convergence effect comparison between DrlDM and DDPG algorithms

务能够完成计算。

(2) 均分带宽边缘计算:节点将待处理的任务全部卸载到AP通过远程服务器进行计算,AP会将传输带宽平均分配给所有需要处理任务的节点,节点会根据自身当前可用电量自适应调整发射机频率,以保证任务能够完成传输。

(3) 基于DDPG的卸载方案:采用基于DDPG的深度强化学习方案,对于所有节点,将节点的可用电量、节点与AP之间的信道增益以及任务负载使用DDPG算法输入到神经网络,DDPG输出节点的卸载决策、带宽资源分配决策和以及WPT的持续时间,然后同样使用本文设计的奖励机制去训练该网络。

DrlDM算法与3种基线方案的性能对比如图7~9所示。图7展示了本文所提出的DrlDM算法与3种基线方案在不同节点数量下的任务平均完成时延的性能对比。可以观察到,所提出的DrlDM算法实现了相比3种基线方案更低的任务处理时延。此外,随着节点数量的增加,两种基于Drl的方案延迟缓慢增加,而EC方案由于每个节点得到的带宽资源随着节点数量增加而变少,其延迟显著增加,在节点数量为11时,延迟大于LC方案。而LC方案由于是全部本地计算,其任务处理时延受节点数量影响很小。

图8展示了4种方案在不同带宽下的任务平均完成时延的性能对比。可以观察到,随着带宽总量的增加,LC方案由于是全部本地计算,与带宽大小无关因此延迟变化不大。其他3种方案的延迟随着带宽总量的增加缓慢下降,而基于Drl的两种方案都表现出了相较于其他两种方案更低的延迟性能,并且本文提出的DrlDM算法在不同带宽设置的实验下,都能够实现延迟低于1 ms的最优性能。

图9展示了4种方案在不同AP发射机功率下的任务平均完成时延的性能对比。从图中可以观察到,随着AP发射机功率的增加,4种方案实现的任务平均处理延迟都缓慢下降。这是由于AP的发射机功率增加使得在相同的WPT时间内,节点能够获得更多的能量,节点就能分配更多的能量用于本地计算或者任务卸载当中去,因此节点的任务处理时延会更短。

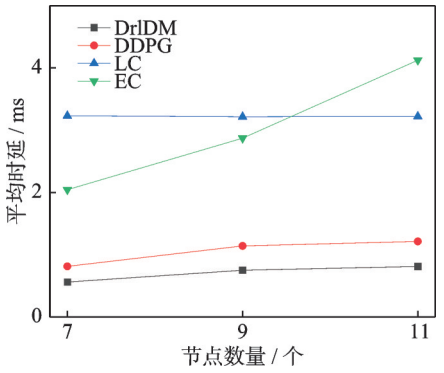

图7　4种方案在不同节点数量下的延迟对比

Fig.7　Delay comparison of four schemes under different numbers of nodes

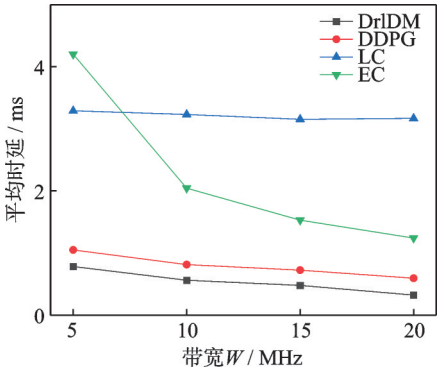

图8　4种方案在不同带宽设置下的延迟对比

Fig.8　Delay comparison of four schemes in different bandwidth settings

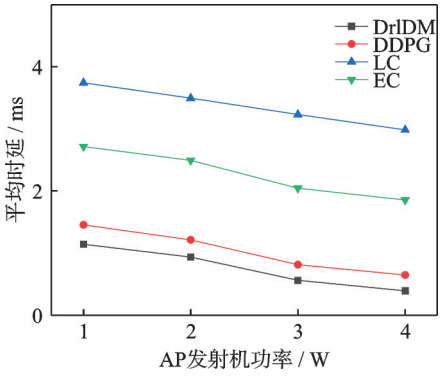

图9　4种方案在不同AP发射机功率下的延迟对比

Fig.9　Delay comparison of four schemes under different AP transmitter power

4　结束语

本文研究了在WPT和MEC辅助的WSN场景下,无线节点对计算任务的卸载问题。首先,本文将考虑的任务处理延迟最小化问题表述为一个混合整数规划问题,为了

解决这一挑战,设计了一种基于深度强化学习的解决方案。从缓解节点能量不足、降低任务计算时延的角度出发,将节点每个时刻的可用能量作为一个输入维度设计到算法中,将原问题转化为马尔可夫决策过程,并设计了奖励机制来训练网络。通过考虑节点随机到达且具有时延约束的计算任务、节点的电池剩余能量以及与AP连接的信道状态来联合优化节点的卸载决策、带宽分配以及WPT时间。仿真结果表明,与3种基线方案相比,所提出的DrlDM算法能够适应时变衰落的通信信道,平均延迟降低了62.11%,能显著缩短节点的任务平均完成时间。

参考文献:

[1] XIONG L, PENG T, LI F, et al. Privacy-preserving authentication scheme with revocability for multi-WSN in industrial IoT [J]. IEEE Systems Journal, 2023, 17(1): 38-49.

[2] TRAN T X, HAJISAMI A, PANDEY P, et al. Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges[J]. IEEE Communications Magazine, 2017, 55(4): 54-61.

[3] SIRIWARDHANA Y, PORAMBAGE P, LIYANAGE M, et al. A survey on mobile augmented reality with 5G mobile edge computing: Architectures, applications, and technical aspects[J]. IEEE Communications Surveys & Tutorials, 2021, 23(2): 1160-1192.

[4] MAO Y, YOU C, ZHANG J, et al. A survey on mobile edge computing: The communication perspective[J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322-2358.

[5] 郭永安, 王宇翱, 周沂, 等. 边缘网络下多无人机协同计算和资源分配联合优化策略[J]. 南京航空航天大学学报, 2023, 55(5): 757-767.
GUO Yongan, WANG Yuao, ZHOU Yi, et al. Multi-UAV collaborative computing and resource allocation joint optimization strategy in edge networks[J]. Journal of Nanjing University of Aeronautics & Astronautics, 2023, 55(5): 757-767.

[6] 杨宏青, 周子颉, 魏乐愚, 等. 面向边缘计算的数字化车间制造数据实时融合方法 [J]. 南京航空航天大学学报, 2024, 56(1): 80-87.
YANG Hongqing, ZHOU Zijie, WEI Leyu, et al. Edge computing oriented real time fusion method of digital workshop manufacturing data[J]. Journal of Nanjing University of Aeronautics & Astronautics, 2024, 56(1): 80-87.

[7] KUANG Z, LI L, GAO J, et al. Partial offloading scheduling and power allocation for mobile edge computing systems[J]. IEEE Internet of Things Journal, 2019, 6(4): 6774-6785.

[8] CHU W, YU P, YU Z, et al. Online optimal service selection, resource allocation and task offloading for multi-access edge computing: A utility-based approach[J]. IEEE Transactions on Mobile Computing, 2023, 22(7): 4150-4167.

[9] TENG H, LI Z, CAO K, et al. Game theoretical task offloading for profit maximization in mobile edge computing[J]. IEEE Transactions on Mobile Computing, 2023, 22(9): 5313-5329.

[10] ZHANG G, SHEN C, SHI Q, et al. AoI minimization for WSN data collection with periodic updating scheme[J]. IEEE Transactions on Wireless Communications, 2023, 22(1): 32-46.

[11] 张明杰, 朱江. WSN中基于强化学习的能效优化任务处理机制[J]. 信号处理, 2022, 38(3): 609-618.
ZHANG Mingjie, ZHU Jiang. Energy efficiency optimization task processing mechanism based on reinforcement learning in WSN[J]. Journal of Signal Processing, 2022, 38(3): 609-618.

[12] WANG F, XU J, CUI S. Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems[J]. IEEE Transactions on Wireless Communications, 2020, 19(4): 2443-2459.

[13] DENG X, LI J, SHI L, et al. Wireless powered mobile edge computing: Dynamic resource allocation and throughput maximization[J]. IEEE Transactions on Mobile Computing, 2022, 21(6): 2271-2288.

[14] NGUYEN P X, TRAN D-H, ONIRETI O, et al. Backscatter-assisted data offloading in OFDMA-based wireless-powered mobile edge computing for IoT networks[J]. IEEE Internet of Things Journal, 2021, 8(11): 9233-9243.

[15] SHI L, YE Y, CHU X, et al. Computation energy efficiency maximization for a NOMA-based WPT-MEC network[J]. IEEE Internet of Things Journal, 2021, 8(13): 10731-10744.

[16] ZHENG K, JIANG G, LIU X, et al. DRL-based offloading for computation delay minimization in wireless-powered

multi-access edge computing[J]. IEEE Transactions on Communications, 2023, 71(3): 1755-1770.

[17] XU J, CHEN L, REN S. Online learning for offloading and autoscaling in energy harvesting mobile edge computing[J]. IEEE Transactions on Cognitive Communications and Networking, 2017, 3(3): 361-373.

[18] ZHOU H, JIANG K, LIU X, et al. Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing[J]. IEEE Internet of Things Journal, 2022, 9(2): 1517-1530.

[19] DENG X, YIN J, GUAN P, et al. Intelligent delay-aware partial computing task offloading for multi-user industrial Internet of Things through edge computing[J]. IEEE Internet of Things Journal, 2023, 10(4): 2954-2966.

[20] XU F, RUAN Y, LI Y. Soft actor-critic based 3D deployment and power allocation in cell-free unmanned aerial vehicle networks[J]. IEEE Wireless Communications Letters, 2023, 12(10): 1692-1696.

[21] HEIDARPOUR A R, HEIDARPOUR M R, ARDAKANI M, et al. Soft actor-critic-based computation offloading in multi-user MEC-enabled IoT—A lifetime maximization perspective[J]. IEEE Internet of Things Journal, 2023, 10(20): 17571-17584.

[22] HUANG L, BI S, ZHANG Y-J A. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks[J]. IEEE Transactions on Mobile Computing, 2020, 19(11): 2581-2593.

作者简介:

张健(1980-),男,博士,讲师,研究方向:机器学习、边缘计算,E-mail: jianzhang@nuist.edu.cn。

刘鹏博(1999-),男,硕士研究生,研究方向:深度强化学习、边缘计算。

汤健(1974-),**通信作者**,男,博士,教授,博士生导师,研究方向:小样本数据建模和固废处理过程智能控制,E-mail: freeflytang@bjut.edu.cn。

(编辑:刘彦东)