第47卷 第8期 计 算 机 工 程 2021年8月
Vol.47 No.8 Computer Engineering August 2021

·热点与综述· 文章编号：1000-3428(2021)08-0037-08 文献标志码：A 中图分类号：TP391

MEC中卸载决策与资源分配的深度强化学习方法

杨 天，杨 军

（宁夏大学 信息工程学院，宁夏 银川 750021）

摘 要：在移动边缘计算（MEC）服务器计算资源有限且计算任务具有时延约束的情况下，为缩短任务完成时间并降低终端能耗，提出针对卸载决策与资源分配的联合优化方法。在多用户多服务器MEC环境下设计一种新的目标函数以构建数学模型，结合深度强化学习理论提出改进的Nature Deep Q-learning算法Based DQN。实验结果表明，在不同目标函数中，Based DQN算法的优化效果优于全部本地卸载算法、随机卸载与分配算法、最小完成时间算法和多平台卸载智能资源分配算法，且在新目标函数下优势更为突出，验证了所提优化方法的有效性。

关键词：移动边缘计算；计算资源；时延约束；卸载决策；资源分配；深度强化学习

开放科学（资源服务）标志码（OSID）：

中文引用格式：杨天，杨军.MEC中卸载决策与资源分配的深度强化学习方法[J].计算机工程，2021，47(8)：37-44.

英文引用格式：YANG T，YANG J.Deep reinforcement learning method of offloading decision and resource allocation in MEC[J].Computer Engineering，2021，47(8)：37-44.

Deep Reinforcement Learning Method of Offloading Decision and Resource Allocation in MEC

YANG Tian，YANG Jun

（School of Information Engineering，Ningxia University，Yinchuan，Ningxia 750021，China）

【Abstract】The computing resources of Mobile Edge Computing（MEC）servers are limited while the computing tasks have delay constraints.To reduce the completion time of computing tasks and the terminal energy consumption，a joint optimization method for offloading decision and resource allocation is proposed.In the multi-user and multi-server MEC environment，a new objective function is designed to build the mathematical model.Based on the model and the deep reinforcement learning theory，an improved Nature Deep Q-learning algorithm（Based DQN）is proposed.The experimental results show that among the various objective functions，the new objective function provides the Based DQN algorithm with more eminent advantages in optimization performance over all the local offloading algorithms，random offloading and allocation algorithms，Minimum Complete Time algorithms（MCT）and multi-platform offloading intelligent resource allocation algorithms.The effectiveness of the objective function and the algorithm is verified.

【Key words】Mobile Edge Computing（MEC）；computing resource；delay constraint；offloading decision；resource allocation；Deep Reinforcement Learning（DRL）

DOI：10. 19678/j. issn. 1000-3428. 0058730

0 概述

目前，智能化终端已经成为现代生活中不可缺少的一部分[1-2]，同时随着5G通信技术的发展，人们开始在智能终端设备上开展高清视频直播、增强现实等新型业务。然而，由于受到计算能力和电池容量的限制，终端设备无法高效地满足大量新型计算任务低时延、高计算的基本要求[3]，而若将计算密集型任务卸载至云端，则会增加传输的延迟和额外的网络负载[4-5]。为此，人们提出移动边缘计算（Mobile Edge Computing，MEC）[6-7]技术，将云端的计算与存储能力迁移至网络边缘，通过边缘进行任务计算，从而降低终端设备能耗与执行时延，提高服务质量[8]。

在MEC环境中，以卸载决策和资源分配为主的计算卸载技术是学者们重点研究的对象[9]。目前相关研究主要针对多用户单MEC服务器场景，且多数

基金项目：宁夏自然科学基金“基于边缘计算的大规模无线传感器网络关键技术研究及在特色农业中的应用”（2020AAC03036）。

作者简介：杨 天（1995—），男，硕士研究生，主研方向为移动边缘计算、普适计算；杨 军（通信作者），教授。

收稿日期：2020-06-23 **修回日期：**2020-08-20 **E-mail：**dragon@nxu.edu.cn

没有同时考虑计算资源约束与时延约束[10-17]，这将导致不能更准确地模拟真实的卸载情况，如在自动驾驶、紧急救援等场景下，需要在有限资源下完成时延敏感型任务的计算。本文将卸载场景转变为多用户多MEC服务器场景，同时考虑计算资源有限与时延约束的情况，结合深度强化学习理论和一种新型目标函数，提出卸载决策与资源分配的联合优化方法，从而在满足时延约束的情况下缩短计算任务完成时间并降低终端能耗。

1 相关研究

近年来，国内外学者已对MEC计算卸载技术进行了深入的研究。文献[10]将可再生绿色能源引入到MEC系统中，将执行时延与卸载失败率作为优化目标，基于Lyapunov优化提出一种卸载决策与资源分配算法，但该系统仅适用于单用户卸载情况。文献[11]根据任务剩余完成时间进行边缘服务器的计算切换来缩短任务完成时间，以提高任务的卸载效率。文献[12]结合K近邻(K Nearest Neighbor，KNN)算法与强化学习中的Q-learning算法，提出一种多平台卸载智能资源分配方法。该方法首先通过KNN算法选择卸载节点，然后通过Q-learning算法优化资源分配，以降低系统时延成本。文献[11-12]虽然研究多用户卸载问题，但更关注于时延的优化而忽略了设备能耗的优化。文献[13]为了在计算依赖任务时控制超出时延约束的任务比例，提出一种最优资源管理策略以最小化移动设备能耗，但该模型没有考虑边缘设备的计算资源约束。文献[14]在边缘节点计算资源受限的情况下提出基于非合作博弈论的传输功率分配算法，获得了较好的计算卸载性能。文献[15]针对多用户完全卸载决策提出一种基于博弈论的任务卸载算法。该算法将卸载博弈模型转换为势博弈模型，通过基于有限改进性质的分布式博弈方法寻找纳什均衡解，以同时优化计算时延和设备能耗。文献[16]提出一种基于深度神经网络(Deep Neural Network，DNN)的优化算法。该算法首先利用序列二次规划(Sequential Quadratic Programming，SQP)法得到优化结果，然后利用优化结果训练DNN，不断更新网络权值，直到训练完成。实验结果表明，训练完成的DNN可以很好地逼近SQP的优化结果且精度很高，运行时间也大幅缩短。文献[14-16]虽然考虑了计算资源约束，但提出的系统模型均建立在单个MEC服务器上，没有对多个MEC服务器的计算资源受限问题进行研究。文献[17]建立了一个同时考虑终端、边缘节点和云计算节点的半马尔科夫决策过程资源分配模型，并提出一种寻找最优资源分配方案的算法以降低能耗和时延，但该研究没有考虑任务计算的时延约束。

本文将多用户单MEC服务器卸载场景转变为多用户多MEC服务器卸载场景，同时考虑服务器计算资源约束与任务时延约束，研究卸载决策与资源分配的联合优化方法，以期使系统在满足时延约束时缩短完成时间并降低终端能耗。针对研究问题设计一种新的目标函数并数学建模，利用结合深度学习感知能力与强化学习决策能力的深度强化学习方法，基于Nature Deep Q-learning(Nature DQN)算法并根据问题模型进行部分改进，提出Based DQN算法，并将该算法与全部本地卸载算法ALO、随机卸载与分配算法ROA、最小完成时间(Minimum Complete Time，MCT)算法[11]和多平台卸载智能资源分配算法[12]进行实验对比，同时对比不同目标函数下的优化结果。

2 系统模型

本文系统模型场景为多用户多服务器应用场景，如图1所示，其中有N台终端设备与M台MEC服务器，并通过无线通信链路连接MEC服务器计算卸载终端设备的任务数据。本文假设每个终端设备都可以对自己的执行任务进行卸载计算或本地计算，卸载时任务只能卸载到一台MEC服务器上进行计算，并且每个终端设备处于无线连接的范围之内。而每台MEC服务器的计算能力有限，不能同时接受每一个终端的卸载请求。终端设备的集合为$U=\{1,2,\cdots,i,\cdots,N\}$，MEC服务器的集合为$S=\{1,2,\cdots,j,\cdots,M\}$，所有任务的集合为$R$。模型中每个终端设备$i$都有一个待处理的计算密集型任务$R_i$，具体包括计算任务$R_i$所需的数据$D_i$(代码和参数)、计算任务$R_i$所需的CPU工作量$W_i$以及任务$R_i$的完成时延约束$\eta_i$，即$R_i \triangleq (D_i, W_i, \eta_i)$。

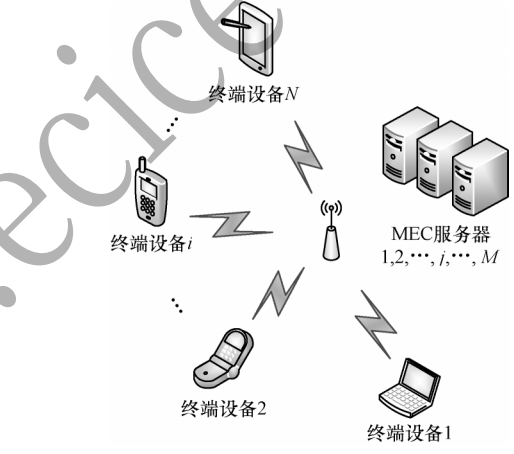

图1 系统模型场景

Fig.1 Scene of system model

以向量$\boldsymbol{X}=[x_1, x_2, \cdots, x_i, \cdots, x_N]$表示每个$R_i$的卸载决策。其中，$x_i \in \{0,1,\cdots,j,\cdots,M\}$，$x=0$表示当前为本地卸载，其余表示将$R_i$卸载至第$j$台MEC服务器。

2.1 计算模型

若R_i在本地处理，用T_i^{L}表示R_i本地执行的时间，具体定义如式(1)所示。

$$T_i^{\mathrm{L}}=\frac{W_i}{f_i^{\mathrm{L}}} \tag{1}$$

其中:工作量W_i具体为完成R_i所需的CPU周期总数;f_i^{L}表示终端设备i本地的计算能力,即每秒所执行的CPU周期数。

用E_i^{L}表示R_i本地执行的设备能耗,定义如式(2)所示。

$$E_i^{\mathrm{L}} = W_i \times J_i \tag{2}$$

其中:J_i为终端设备i计算每单位CPU周期的能耗,根据文献[18],$J_i = (f_i^{\mathrm{L}})^2 \times 10^{-27}$。

若R_i在边缘处理,R_i边缘执行下的时延T_i^{O}与设备能耗E_i^{O}应分别从数据上传、数据处理和数据回传3个部分进行计算,具体如下:

1)终端设备i将R_i的数据通过无线信道上传至相应的MEC服务器。

设T_i^{K}为设备i上传R_i数据的时间,定义如式(3)所示。

$$T_i^{\mathrm{K}} = \frac{D_i}{v^{\mathrm{K}}} \tag{3}$$

其中:D_i为R_i的数据大小;v^{K}为系统模型中的数据上传速率,即每秒上传的数据量。则终端设备i上传数据的能耗E_i^{K}如式(4)所示。

$$E_i^{\mathrm{K}} = T_i^{\mathrm{K}} \times p^{\mathrm{K}} \tag{4}$$

其中:p^{K}为终端设备i的上行传输功率。

2)MEC在接收到处理数据后分配计算资源进行计算。

用T_i^{C}表示卸载数据在MEC服务器中计算的时间,定义如式(5)所示。

$$T_i^{\mathrm{C}} = \frac{W_i}{f_{ij}^{\mathrm{O}}} \tag{5}$$

其中:f_{ij}^{O}为第j台MEC服务器为R_i卸载执行所分配的计算资源,即每秒所执行的CPU周期数。需要注意的是,当R_i卸载至本地或其他MEC服务器上时,f_{ij}^{O}为零且不能计算式(5)并作为模型中的约束条件,即:

$$f_{ij}^{\mathrm{O}} = 0, x_i \neq j \tag{6}$$

此时,终端设备i没有计算任务而处于等待状态并产生空闲能耗,设p_i^{I}为终端设备i的空闲功率,则卸载计算下终端设备i的空闲能耗E_i^{C}为:

$$E_i^{\mathrm{C}} = T_i^{\mathrm{C}} \times p_i^{\mathrm{I}} \tag{7}$$

3)MEC服务器将计算结果返回给终端设备i。

根据文献[19]可知,回传时计算结果较小且下行速率较高。因此,本文忽略终端设备接收时的时延与能耗。则R_i边缘执行下的时延T_i^{O}为传输时延T_i^{K}与MEC服务器计算时延T_i^{C}之和,即:

$$T_i^{\mathrm{O}} = T_i^{\mathrm{K}} + T_i^{\mathrm{C}} \tag{8}$$

R_i边缘执行下的设备能耗E_i^{O}为设备i的上传能耗E_i^{K}与设备i等待R_i在MEC服务器上计算完成的空闲能耗E_i^{C}之和,即:

$$E_i^{\mathrm{O}} = E_i^{\mathrm{K}} + E_i^{\mathrm{C}} \tag{9}$$

综上所述,终端设备i中任务R_i整个计算过程的时延T_i和能耗E_i分别为:

$$T_i = \begin{cases} T_i^{\mathrm{L}}, x_i = 0 \\ T_i^{\mathrm{O}}, x_i \neq 0 \end{cases} \tag{10}$$

$$E_i = \begin{cases} E_i^{\mathrm{L}}, x_i = 0 \\ E_i^{\mathrm{O}}, x_i \neq 0 \end{cases} \tag{11}$$

需要注意的是,T_i与f_{ij}^{O}应满足式(12)和式(13)所示的限制条件。

$$T_i \leqslant \eta_i \tag{12}$$

$$\sum_{i=1}^{N} f_{ij}^{\mathrm{O}} \leqslant F_j \tag{13}$$

其中:R_i的时延约束η_i参照文献[20],为计算能力是1.4 GHz并根据式(1)计算结果的2倍;F_j为第j台MEC服务器的整体计算资源,即每个卸载至第j台MEC服务器的R_i所分配的计算资源总和不应超过F_j。

2.2 问题模型

本文的研究目的是在多用户多MEC服务器场景下,考虑计算资源有限且计算任务具有时延约束的情况,设计联合优化系统的卸载决策和资源分配方案,使得所有计算任务在满足时延约束下缩短完成时间并最小化所有终端设备的能耗,同时延长终端设备的使用时间。因此,系统目标函数G定义如式(14)所示。

$$G = \sum_{i=1}^{N} E_i + 10 \times \sum_{i=1}^{N} \frac{T_i}{\eta_i} \tag{14}$$

其中:$\frac{T_i}{\eta_i}$为R_i完成时间与时延约束的比值。根据仿真实验的计算结果可知,$\sum_{i=1}^{N} E_i$与$\sum_{i=1}^{N} \frac{T_i}{\eta_i}$相差一个十进制数量级。因此,为保证两者数量级相同并共同优化,将$\sum_{i=1}^{N} \frac{T_i}{\eta_i}$乘以一个系数10。目标函数$G$通过求解最佳的卸载决策和资源分配方案,使得模型中终端设备的整体能耗与任务执行时间和时延约束比值最小化,以达到本文的研究目的。整体问题模型如下:

$$\min_{\boldsymbol{X},\boldsymbol{Y}} G \tag{15}$$

$$\boldsymbol{X} = [x_1, x_2, \cdots, x_i, \cdots, x_N] \tag{16}$$

$$\boldsymbol{Y} = [y_1, y_2, \cdots, y_i, \cdots, y_N] \tag{17}$$

$$y_i = \begin{cases} f_i^{\mathrm{L}}, x_i = 0 \\ f_{ij}^{\mathrm{O}}, x_i = j \end{cases} \tag{18}$$

s.t.

C1: $x_i \in \{0, 1, \cdots, j, \cdots, M\}, \forall i \in U$

C2: $y_i > 0, \forall i \in U$

C3: $f_{ij}^{\mathrm{O}} = 0, x_i \neq j$

C4: $T_i \leqslant \eta_i, \forall i \in U$

$$\text{C5:}\sum_{i=1}^{N} f_{ij}^{\text{O}} \leqslant F_j, \forall j \in S$$

其中：$\boldsymbol{X}$为任务卸载决策向量；$\boldsymbol{Y}$为计算资源分配向量；限制条件C1~C3表示每个任务R_i只能卸载到本地或其中一台MEC服务器上进行计算；C4表示任务完成时延的约束；C5表示分配的计算资源应满足的限制约束。

3 卸载决策与资源分配的联合优化方法

在上文建立的问题模型下，考虑采用结合强化学习与深度学习的深度强化学习方法进行问题求解，一方面是因为深度强化学习中的强化学习理论以“试错”的方式让智能体在与环境交互的过程中通过获得奖励来指导行为以改善决策，这适用于本文模型中任务卸载决策与计算资源分配的联合优化，另一方面是因为引入深度学习的深度强化学习方法可避免状态空间、动作空间过大而带来的存储困难问题。因此，下文将结合系统模型，首先设计系统状态（State）、系统动作（Action）、奖励函数（Reward）3个要素，然后对深度强化学习算法中的Nature DQN算法进行部分改进，提出一种基于深度强化学习的卸载决策与资源分配联合优化方法Based DQN，使得目标函数值G最小。

3.1 系统状态、动作与奖励函数设计

为联合优化卸载决策与资源分配方案以最小化目标函数值，令系统状态$\boldsymbol{s}$包括卸载决策向量$\boldsymbol{X}$、计算资源分配向量$\boldsymbol{Y}$、剩余计算资源向量$\boldsymbol{Z}$与G，如式(19)所示。

$$\boldsymbol{s}=[\boldsymbol{X}, G, \boldsymbol{Z}, \boldsymbol{Y}] \tag{19}$$

其中，$\boldsymbol{Z}=[z_1, z_2, \cdots, z_j, \cdots, z_M]$，$z_j$表示为第$j$台MEC服务器所剩的计算资源：

$$z_j = F_j - \sum_{i=1}^{N} f_{ij}^{\text{O}} \tag{20}$$

初始化时，系统状态为本地卸载状态，即$\boldsymbol{X}$为零向量，$\boldsymbol{Y}$中每个任务所分配的计算资源为f_i^{L}，G为全部本地卸载下的计算值，$\boldsymbol{Z}$中每个$z_j = F_j$。

系统动作$\boldsymbol{a}$应确定对哪一项任务进行怎样的卸载决策与计算资源分配，即对终端设备i下的任务R_i选择卸载与资源分配方案，调整系统状态，如式(21)所示。

$$\boldsymbol{a}=[i, \lambda, \psi] \tag{21}$$

其中：λ为R_i的卸载方案，$\lambda \in \{0, 1, \cdots, j, \cdots, M\}$；$\psi$为$R_i$的计算资源分配方案。需要注意的是，当$\lambda=0$时，$\psi = f_i^{\text{L}}$。

奖励函数r应关联目标函数，具体定义如式(22)所示。

$$r = \frac{G - G'}{G^{\text{L}}} \tag{22}$$

其中：G为当前t时刻状态s_t下的目标函数值；G'为s_t采取动作a_t到下一状态s_{t+1}下的目标函数值，两者分别通过各自状态中的卸载决策向量与资源分配向量计算出相应的时延与能耗后，再按照式(14)进行计算；G^{L}为全部本地卸载下的计算值，当G'结果更优时（$G > G'$）获得正奖励，即在状态s_t下采取动作a_t能够获得更优的目标函数值，反之奖励为非正值。

3.2 基于Nature DQN算法的联合优化

Nature DQN是在Q-Learning算法的基础上演变而来的。在Q-learning算法中，智能体在t时刻下观察环境中的状态s_t，根据概率以随机或者Q表的方式选择动作a_t执行，改变到状态s_{t+1}并获得奖励r_t，通过式(23)更新Q表与当前状态，并循环此学习过程，收敛于最大的Q函数Q^*，得到最优策略。

$$Q(s_t, a_t) = Q(s_t, a_t) + \delta \times \left[r_t + \gamma \times \max_{a_{t+1}} (s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \tag{23}$$

其中：δ是学习率；γ是折扣系数。

相较于Q-learning算法，Nature DQN算法不同点在于其Q值不是直接通过系统状态和系统动作计算，而是通过Q网络（神经网络）进行计算，即期望神经网络拟合Q表，如式(24)如示。以神经网络进行拟合，可以应对随着状态、动作维数的增大而带来的Q表存储困难问题，如在本文所提的状态与动作中，随着N与M的增加，自身的组合数量庞大，Q表将难以进行对应Q值的存储。

$$Q(s_t, a_t; \theta) \approx Q^*(s_t, a_t) \tag{24}$$

其中：θ为神经网络的参数。Nature DQN算法中使用了2个结构相同但θ不同的Q网络（当前网络Q与目标网络Q'），当前网络Q进行动作选择并更新θ，目标网络Q'计算目标Q值。目标网络Q'中的参数θ'不需要迭代更新，而是每隔一段时间复制θ进行延迟更新，以减少目标Q值和当前Q值相关性，使算法更好地收敛。

此外，Nature DQN采用经验回放训练强化学习的学习过程，即将s_t、a_t、r_t、s_{t+1}、done（判断学习是否结束的布尔值）五元组存储到一个经验池中，通过随机抽样进行学习，减少样本之间的相关性，更好地训练神经网络。

结合问题模型，本文根据约束条件C5，在原始Nature DQN算法的动作选择上增加了a_t中ψ是否满

足计算资源约束的判断，筛选有效的执行动作，以提高学习效率。具体算法如下：

算法1　动作筛选算法（AS）

输入　s_t, a_t

输出　动作合理判断布尔值bool

1.bool=False//初始化

2.if λ==0 then//若R_i选择在本地执行

3.bool=True//动作允许执行

4.elif λ!=0 and x_i==λ then

5.//若R_i选择在原卸载位置执行

6.if $y_i+z_\lambda \geqslant \psi$ then

7.//R_i在λ服务器上原分配的计算资源y_i与当前λ剩余的//计算资源z_λ之和大于等于ψ，即在$z_\lambda \geqslant \psi$或$y_i \geqslant \psi$或$z_\lambda \geqslant \psi - y_i$//的情况下，则可通过回收$R_i$原分配的$y_i$，重新为$R_i$分配ψ，并//满足计算资源的约束

8.bool=True

9.end if

10.elif λ! = 0 and X_i!=λ then

11.//若R_i选择在新卸载位置执行

12.if $\psi \leqslant z_\lambda$ then//若资源分配方案满足约束

13.bool=True

14.end if

15.end if

将动作筛选算法（AS）加入到Nature DQN算法中，若a_t满足计算资源约束则执行该动作，否则重新根据ε贪婪策略选取动作。具体算法如下：

算法2　Based DQN算法

输入　训练回合数Total，学习率δ，折扣系数γ，Q'的更新频率h，学习间隔步长σ，批量梯度下降的样本数b、ε

输出　s，θ

1.对Q网络所有参数θ与Q网络所有参数θ'进行初始化，其中θ'=θ

2.初始化状态s与经验回放集合Φ

3.for episode ←1 to Total do//迭代

4.在Q网络中使用s输入，得到所有动作Q值的输出，用ε贪婪法在所有Q值下选择动作a

5.if AS(s,a) then//满足计算资源约束

6.在状态s执行动作a，调整卸载方案，回收原计算资源并重新分配，得到新状态s'，奖励r与是否终止布尔值done

7.将[s,a,r,s',done]五元组存入Φ中

8.s←s'

9.if 满足学习间隔步长σ then

10.从Φ中采样b个样本计算当前目标Q值μ_k(k=1,2,…,b)

11.$$\mu_k = \begin{cases} r_k, \text{done} = \text{True} \\ r_k + \gamma \times \max_{a'} Q'(s'_k, a';\ \theta'), \text{done} = \text{False} \end{cases}$$

12.对μ_k与$Q(s_k, a_k, \theta)$利用均值方差损失函数，通过梯度反向传播更新Q网络的参数θ

13.end if

14.if 满足h then

15.θ'=θ

16.end if

17.若s'是终止状态且满足C_4，则当前回合迭代完毕，否则转到步骤4

18.else//未满足计算资源约束

19.返回步骤4

20.end if

21.end for

4　实验与结果分析

利用Python语言在Visual Studio Code平台上对本文算法与全部本地卸载算法（ALO）、随机卸载与分配算法（ROA）、最小完成时间算法（Minimum Complete Time，MCT）[11]、多平台卸载智能资源分配算法[12]进行实验对比，以验证本文算法的有效性，同时在不同目标函数下对比Based DQN算法的优化效果，以验证新提目标函数的有效性。具体仿真参数如下：

假设每一台设备i的计算能力为1 GHz，上行传输功率为700 mW，空闲功率为100 mW，上传速率为2 Mb/s，M=2，且每台MEC服务器的整体计算能力分别为5 GHz与4 GHz，$\psi \in \{f_i^L, 1.2, 1.4, 1.6\}$ GHz。任务R_i中的数据D_i服从（500，1 000）的均匀分布，单位为Kb。工作量W_i服从（1 000，1 500）的均匀分布，单位为Megacycles。

对于深度强化学习的参数，设ε为0.9，学习率δ为0.001，折扣系数γ为0.9，经验回放集合Φ大小为2 000，随机采样样本数b为32，更新频率h为50，学习间隔步长σ为5（学习步数需大于200）。

4.1　算法收敛情况

假设有7台终端设备，即所需执行的任务数量为7，执行回合数（episode）为150，比较目标函数值G的变化，如图2所示。可以看出：ROA算法在整个迭代过程震荡，无法收敛；ALO算法始终保持收敛，但由于全部任务卸载到本地，造成较大的时延与能耗，目标函数值较高；其余3种算法随着episode的增加逐步收敛，MCT算法在第96回合达到收敛；多平台卸载智能资源分配算法在第127回合后逐步收敛，且收敛目标函数值比MCT算法的计算结果降低3.12%；Based DQN算法自100回合后逐步收敛，其结果较于多平台卸载智能资源分配算法降低1.53%，在5种算法中结果最优。MCT算法与多平台卸载智能资源分配算法结果较差于Based DQN算法，这是因为两者对任务完成时延关注更多。此外，多平台卸载智能资源分配算法中使用Q-learning算法进行训练学习，由于本文中状态、动作维数较大，Q表存储问题导致探索不全面，使得多平台卸载智能资源分配算法不能得到最优结果。

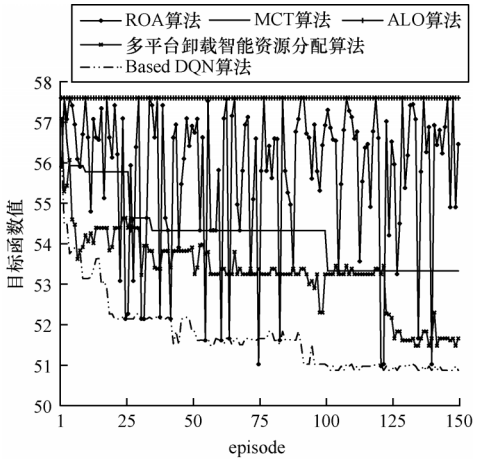

图2 5种算法的目标函数值变化

Fig.2 Change of objective function values of five algorithms

将ROA算法、MCT算法、多平台卸载智能资源分配算法和Based DQN算法的能耗分别与ALO算法的能耗总和做差，再分别除以ALO算法的能耗总和作为降低能耗比例（Energy Reduced Scale，ERS），并联合对比在满足时延约束下的缩短完成时间的比例（Time Reduced Scale，TRS），如表1所示。可以看出：MCT算法、多平台卸载智能资源分配算法与Based DQN算法可在缩短完成时间的同时降低终端能耗50%以上，且Based DQN算法中时延与能耗减少的比例更大。

表1 4种算法的TRS和ERS

Table 1 TRS and ERS of four algorithms %

算法	TRS	ERS
ROA算法	26.85	47.36
MCT算法	32.25	50.78
多平台卸载智能资源分配算法	32.50	51.12
Based DQN算法	33.51	51.51

4.2 不同学习率下的算法收敛情况

分别在0.01、0.001、0.0001这3种不同学习率δ下对比Based DQN算法的收敛情况，如图3所示。可以看出：当δ为0.01时，算法收敛速度较快，但较大的学习率导致收敛于局部最优解；当δ较小为0.000 1时，算法收敛速度较慢，较长的收敛时间影响了算法的优化效率。

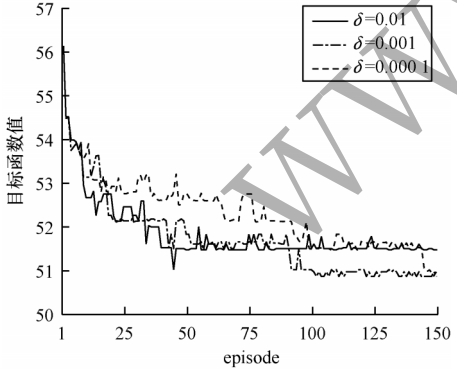

图3 不同学习率下Based DQN算法的收敛情况

Fig.3 Convergence of Based DQN algorithm under different learning rates

为进一步比较Based DQN算法在不同学习率δ下对时延与能耗的优化效果，分别对比不同学习率δ下的Based DQN算法在收敛过程中TRS与ERS的变化情况，如图4、图5所示。可以看出：当δ为0.01时，TRS与ERS收敛于局部最优解；当δ为0.000 1时，TRS与ERS收敛过慢；当δ为0.001时，Based DQN算法收敛后对时延与能耗的优化效果最佳。因此，本文算法采用0.001的学习率。

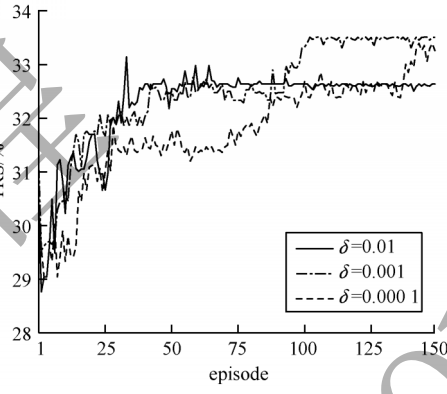

图4 不同学习率下Based DQN算法的TRS

Fig.4 TRS of Based DQN algorithm under different learning rates

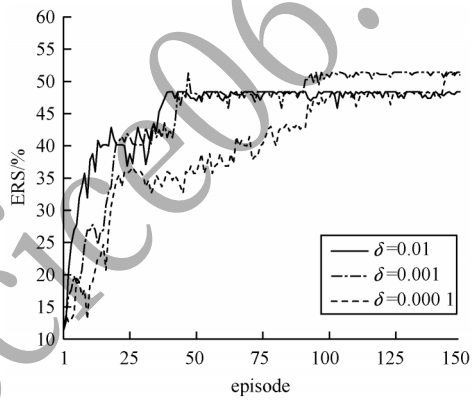

图5 不同学习率下Based DQN算法的ERS

Fig.5 ERS of Based DQN algorithm under different learning rates

4.3 不同累计任务数量下的算法目标函数值对比

分别模拟[20，100]的累计任务数量，对比5种算法的目标函数值，如图6所示。可以看出：随着累计任务数量的增加，5种算法的G值逐渐增大，而在不同累计任务数量下ALO算法、ROA算法的G值较大，这主要是由于两种算法没有对任务卸载方案与计算资源分配方案进行合理优化，导致任务执行时，时延与能耗较高。3种优化算法相比前述两种算法在不同累计任务数量下能够有效降低目标函数值。当累计任务数量为20时，3种算法差别较小，但随着累计任务数量的增加，Based DQN算法的优化效果得以体现。以累计任务数量等于100时为例，多平

台卸载智能资源分配算法、Based DQN算法相较于MCT算法G值分别降低3.62%、5.89%。

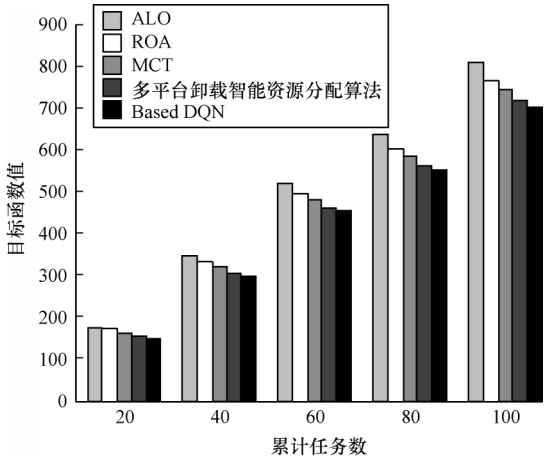

图6　不同累计任务数量下5种算法的目标函数值

Fig.6　Objective function values of five algorithm under different numbers of cumulative tasks

此外，本文将多平台卸载智能资源分配算法与Based DQN算法相较于MCT算法的时延与能耗分别降低的比例进行对比，如表2所示。可以看出：在大量累计任务数量下，Based DQN算法优化效果更佳。

表2　2种算法对MCT算法的优化效果

Table 2　Optimization effects of two algorithms for MCT algorithm　%

算法	时延减少比例	能耗减少比例
多平台卸载智能资源分配算法	1.86	24.50
Based DQN算法	2.85	29.76

4.4　不同目标函数下的优化情况

对于降低时延与能耗的多目标优化问题，通常以任务执行时延与终端执行能耗的加权和作为目标函数进行问题求解。将每一个任务执行时延与能耗加权和的平均值作为另一种目标函数（见式（25）），与本文所提目标函数（见式（14））进行时延与能耗的优化对比，终端设备数为7。

$$G^{\mathrm{R}}=\frac{\sum_{i=1}^{N}(\tau\times T_i+(1-\tau)\times E_i)}{N} \tag{25}$$

在式（25）所示的目标函数中：τ为执行时延的权重系数；$1-\tau$为执行能耗的权重系数。考虑到本文是在满足时延约束下缩短时延、降低能耗，将τ分别取值为0.7、0.6、0.5与式（14）在Based DQN算法下进行TRS、ERS联合实验对比，如表3所示。可以看出：当τ=0.7和τ=0.6时，算法更多关注时延的优化；当τ=0.5时，优化结果较为均衡，而在新目标函数下的Based DQN算法优化效果最好，能够在满足时延约束下最大程度地缩短时延并降低能耗。

表3　不同目标函数下Based DQN算法的TRS和ERS

Table 3　TRS and ERS of Based DQN algorithm under different objective functions　%

目标函数	TRS	ERS
τ=0.5	28.15	47.57
τ=0.6	32.21	40.34
τ=0.7	32.72	32.27
式（14）	33.51	51.51

为进一步比较不同目标函数对时延与能耗的优化程度，在累计任务为100时，对比4种目标函数下Based DQN算法相较于MCT算法时延与能耗分别降低的比例，如表4所示。可以看出：Based DQN算法在新目标函数下时延与能耗的优化效果更好，验证了本文所设计目标函数的有效性。

表4　不同目标函数下Based DQN算法对MCT算法的优化效果

Table 4　Optimization effect of Based DQN algorithm for MCT algorithm under different objective functions　%

目标函数	时延减少比例	能耗减少比例
τ=0.5	1.52	23.21
τ=0.6	1.79	22.61
τ=0.7	1.98	20.89
式（14）	2.85	29.76

5　结束语

本文在MEC服务器计算资源有限的情况下考虑时延约束，设计一种新的目标函数并构建数学模型，对深度强化学习中的Nature DQN算法进行改进，提出卸载决策与资源分配的联合优化算法：Based DQN，以缩短计算任务完成时间，降低终端能耗。实验结果表明，该算法的优化效果均优于ALO算法、ROA算法、MCT算法和多平台卸载智能资源分配算法，且其在本文设计的目标函数下结果更优。下一步将研究任务具有优先级与执行顺序以及无线干扰环境下的卸载决策和资源分配方案。

参考文献

[1] ZHANG K, LENG S P, HE Y J, et al. Mobile edge computing and networking for green and low-latency Internet of Things[J]. IEEE Communications Magazine, 2018, 56(5): 39-45.

[2] DINH T Q, TANG J H, LA Q D, et al. Offloading in mobile edge computing: task allocation and computational frequency scaling[J]. IEEE Transactions on Communications, 2017, 65(8): 3571-3584.

[3] BOUENT M, CONAN V. Mobile edge computing resources optimization: a geo-clustering approach[J]. IEEE Transactions on Network and Service Management, 2018, 15(2): 787-796.

[4] MAO Y Y, YOU C S, ZHANG J, et al. A survey on mobile edge computing: the communication perspective[J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322-2358.
[5] ZHANG K, MAO Y M, LENG S P, et al. Contract-theoretic approach for delay constrained offloading in vehicular edge computing networks[J]. Mobile Networks and Applications, 2019, 24(3): 1003-1014.
[6] ABBAS N, ZHANG Y, TAHERKORDI A, et al. Mobile edge computing: a survey[J]. IEEE Internet of Things Journal, 2018, 5(1): 450-465.
[7] SATYANARAYANAN M. The emergence of edge computing[J]. Computer, 2017, 50(1): 30-39.
[8] SHI W S, DUSTDAR S. The promise of edge computing[J]. Computer, 2016, 49(5): 78-81.
[9] 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): 142-159.
XIE R C, LIAN X F, JIA Q M, et al. Survey on computation offloading in mobile edge computing[J]. Journal on Communications, 2018, 39(11): 142-159. (in Chinese)
[10] MAO Y Y, ZHANG J, LETAIEF K B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices[J]. IEEE Journal on Selected Areas in Communications, 2016, 34(12): 3590-3605.
[11] 李波, 黄鑫, 牛力, 等. 车载边缘计算环境中的任务卸载决策和优化[J]. 微电子学与计算机, 2019, 36(2): 78-82.
LI B, HUANG X, NIU L, et al. Task offloading decision in vehicle edge computing environment[J]. Microelectronics & Computer, 2019, 36(2): 78-82. (in Chinese)
[12] 王汝言, 梁颖杰, 崔亚平. 车辆网络多平台卸载智能资源分配算法[J]. 电子与信息学报, 2020, 42(1): 263-270.
WANG R Y, LIANG Y J, CUI Y P. Intelligent resource allocation algorithm for multi-platform offloading in vehicular networks[J]. Journal of Electronics & Information Technology, 2020, 42(1): 263-270. (in Chinese)
[13] YOU C S, ZENG Y, ZHANG R, et al. Asynchronous mobile-edge computation offloading: energy-efficient resource management[J]. IEEE Transactions on Wireless Communications, 2018, 17(11): 7590-7605.
[14] 余翔, 石雪琴, 刘一勋. 移动边缘计算中卸载策略与功率的联合优化[J]. 计算机工程, 2020, 46(6): 20-25.
YU X, SHI X Q, LIU Y X. Joint optimization of offloading strategy and power in mobile-edge computing[J]. Computer Engineering, 2020, 46(6): 20-25. (in Chinese)
[15] 张艮山, 刘旭宁. 资源受限移动边缘计算任务拆分卸载调度决策[J]. 计算机应用与软件, 2019, 36(10): 268-273, 278.
ZHANG G S, LIU X N. Tasks split and offloading decision in mobile edge computing with limited resources[J]. Computer Applications and Software, 2019, 36(10): 268-273, 278. (in Chinese).
[16] LI J, LÜ T J. Deep neural network based computational resource allocation for mobile edge computing[C]// Proceedings of 2018 IEEE GLOBECOM Workshops. Washington D. C., USA: IEEE Press, 2018: 1-6.
[17] LIN C C, DENG D J, YAO C C. Resource allocation in vehicular cloud computing systems with heterogeneous vehicles and roadside units[J]. IEEE Internet of Things Journal, 2018, 5(5): 3692-3700.
[18] WEN Y G, ZHANG W W, LUO H Y. Energy-optimal mobile application execution: taming resource-poor mobile devices with cloud clones[C]//Proceedings of 2012 IEEE Conference on Computer Communications. Washington D. C., USA: IEEE Press, 2012: 2716-2720.
[19] WANG C M, RICHARD Y F, LIANG C C, et al. Joint computation offloading and interference management in wireless cellular networks with mobile edge computing[J]. IEEE Transactions on Vehicular Technology, 2017, 66(8): 7432-7445.
[20] 徐佳, 李学俊, 丁瑞苗, 等. 移动边缘计算中能耗优化的多重资源计算卸载策略[J]. 计算机集成制造系统, 2019, 25(4): 954-961.
XU J, LI X J, DING R M, et al. Energy efficient multi-resource computation offloading strategy in mobile edge computing[J]. Computer Integrated Manufacturing Systems, 2019, 25(4): 954-961. (in Chinese)

编辑 金胡考