文章编号:1672-3961(2024)01-0083-08 DOI:10.6040/j.issn.1672-3961.0.2022.344

基于深度强化学习的物联网服务协同卸载方法

曹宇慧,黄昱泽*,冯北鹏,张淼,郭珍珍
(重庆交通大学信息科学与工程学院, 重庆 400074)

摘要:针对边缘计算中终端算力不足、资源有限和时延较大的问题,提出一种基于深度强化学习的物联网服务协同卸载方法。通过 3 种不同的卸载方式建立时延模型,挖掘服务之间的关联关系,对关联服务进行协同卸载,加入关联服务的通信时延以建立完善的卸载时延模型,结合整体模型考虑卸载率的取值以及关联服务如何协同卸载使时延最小,从而实现服务调用时延和服务间通信时延的最小化。试验结果表明,与其他算法相比,该算法在获取最优服务卸载策略的同时,系统总服务时延能降低 20%左右。

关键词:边缘计算;服务卸载;关联服务;协同卸载;深度强化学习
中图分类号:TP491.8 **文献标志码**:A
引用格式:曹宇慧,黄昱泽,冯北鹏,等. 基于深度强化学习的物联网服务协同卸载方法[J].山东大学学报(工学版),2024,54(1):83-90.
CAO Yuhui, HUANG Yuze, FENG Beipeng, et al. A collaborative service offloading approach for Internet of Things based on deep reinforcement learning[J]. Journal of Shandong University (Engineering Science), 2024, 54(1):83-90.

A collaborative service offloading approach for Internet of Things based on deep reinforcement learning

CAO Yuhui, HUANG Yuze*, FENG Beipeng, ZHANG Miao, GUO Zhenzhen
(School of Information Science & Engineering, Chongqing Jiaotong University, Chongqing 400074, China)

Abstract: Aiming at the problems of insufficient terminal computing power, limited resources and large delay in edge computing, a collaborative service offloading approach for Internet of Things based on deep reinforcement learning was proposed. The delay model was established through three different offloading methods, and the association relationship between services was mined. The associated services were cooperatively offloaded, and the communication delay of the associated services was added to establish a perfect offloading delay model, and the value of the offload rate and how the associated services were cooperatively offloaded to minimize the delay was combined with the overall model. Therefore, the service request delay and the communication delay between services were minimized. The experiment results showed that our approach could reduce about 20% service delay in the system than other baseline algorithms on searching the optimal service offloading strategy.

Keywords: edge computing; service offloading; interacting service; collaborative offloading; deep reinforcement learning

0 引言

随着信息技术的不断发展,物联网作为一种新型的网络基础设施得到快速发展,广泛应用于工业[1]、农业[2]、交通[3]、医疗[4]和家居[5]等各个行业。连接于物联网中的传感器、智能电表、摄像头等无线接入设备不断增加,全球将进入万物互联时代[6]。近年来,云计算技术的广泛应用虽然能够有效地解决终端有限的计算、存储和功耗问题[7],但

收稿日期:2022-10-13
基金项目:重庆市自然科学基金资助项目(CSTB2022NSCQ-MSX0368);重庆市教育委员会科学技术研究计划青年项目(KJQN202200702, KJQN201900708, KJQN202100738);国家自然科学基金资助项目(62101080)
第一作者简介:曹宇慧(1998—),女,安徽蚌埠人,硕士研究生,主要研究方向为边缘计算。E-mail:caoyuhui10@163.com
*通信作者简介:黄昱泽(1983—),男,四川甘洛人,讲师,硕士生导师,博士,主要研究方向为边缘计算、服务计算。
E-mail:huangyz@cqjtu.edu.cn

将所有数据采用集中式的方式集中于云计算机中心处理,将会增加数据传输时延,给时延敏感型服务及用户体验质量带来不容忽视的影响[8-9]。为了解决上述问题,边缘计算应运而生,得到学术界和产业界的广泛关注。

边缘计算作为一种新型的计算模型,主要包括边缘设备(如智能移动终端、物联网设备、智能车等)、边缘云和远端云(也称大规模云计算中心)3个部分[10]。边缘计算可大大降低数据传输时延,为用户提供低时延的高质量服务[11],将服务卸载于算力更强的边缘服务器上执行[12]。

物联网服务主要指物联网环境中为完成特定业务目标而调用的一系列服务,例如汽车导航时会调用位置服务、查询路况信息和红路灯信息等,这些服务在执行过程中需要处理大量数据,随着数据量的剧增,传统的云计算无法提供低时延的物联网服务,因此边缘计算应运而生。边缘计算中,由于边缘设备计算资源不足,边缘端计算资源有限,而云端服务时延较大,故需要将终端设备无法处理的任务卸载到边缘服务器上执行。目前已有不少研究工作提出了有效的服务卸载策略:文献[13]用深度强化算法(deep Q-network, DQN)进行迭代优化来获得最优的时延;文献[14]采用深度强化学习深度 Q 网络算法优化用户累积时延;文献[15]采用拉格朗日乘子法和迭代法获取纳什均衡解;文献[16]采用拉格朗日乘子法来最小化移动设备能耗和任务执行时间;文献[17]用 Q-Learning 算法和卷积神经网络减少时延和能耗。

以上大多都只考虑用户-任务-边缘服务器三者之间的关系,并没有考虑服务与服务之间的关联性,并且大多使用线性规划或强化学习来求解该问题,传统的强化学习局限于动作空间和样本空间都很小,且一般是在离散的情境下。比较复杂的任务往往有着很大的状态空间和连续的动作空间。深度确定性策略梯度(deep deterministic policy gradient, DDPG)算法可以解决有着高维或者连续动作空间的情境,是一种结合了深度网络的 Actor-Critic 方法,更加适用于边缘计算卸载场景。

由于物联网服务需要处理大规模的数据,且服务之间存在着数据传输,协同完成某些计算任务,因此,物联网服务卸载不仅需要降低服务时延,更应综合考虑降低服务之间的数据传输时延。而现有的方法对于服务间数据传输时延的优化考虑较少。为此,本研究提出一种基于深度强化学习的物联网服务协同卸载方法。基于用户调用服务历史记录,使用频繁模式挖掘算法挖掘出具有关联关系的关联服务对。考虑服务卸载时延和服务之间的数据传输时延建立服务协同卸载模型,提出一种基于深度确定性策略梯度算法的服务协同卸载算法。利用仿真试验和其他基准算法进行对比,验证了该算法能够有效地减少服务卸载时延。

1 系统模型

如图 1 基站中部署边缘服务器,用来处理用户卸载的计算任务。卸载方法主要有 3 种:本地执行、部分卸载、全部卸载。对于算力要求不高的服务,移动终端不需要将任务卸载到边缘服务器进行计算。对于大部分任务,需要将任务分为可卸载部分和不可卸载部分,可卸载部分将被卸载到边缘服务器进行处理,不可卸载部分进行本地计算。对于计算密集型任务,需要全部卸载到边缘服务器中进行计算。

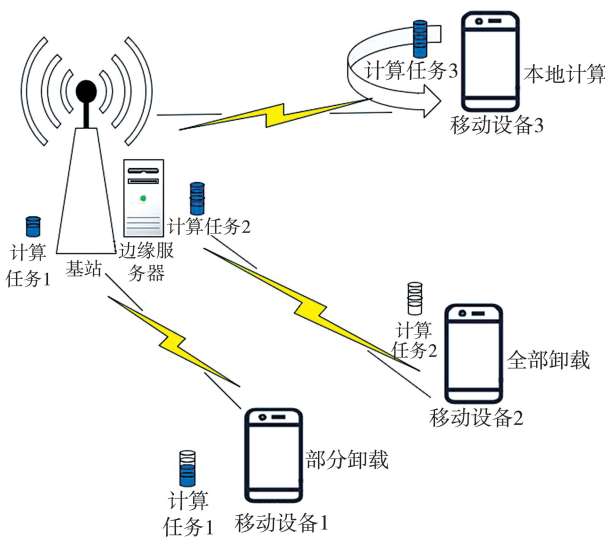

图 1 边缘计算卸载情况分类

Fig.1 Classification of edge computing offloading

物联网中,各类传感器等终端设备分布式部署于各地,不断获取原始数据并对数据进行加工处理,从而实现业务目标。由于单个原子服务所能完成的业务目标有限,通常情况下,不同服务之间存在着大量数据传输,协同完成某一个复杂的业务目标。因而对于物联网服务的卸载需要将存在数据传输的关联服务进行协同卸载,实现服务卸载时延和数据传输时延的联合优化。本研究提出一种基于深度强化学习的服务协同卸载方法,该方法框架如图 2 所示,系统利用用户调用服务的历史记录,挖掘出服务关联模式,建立关联服务协同卸载时延模型,该时延模型包括本地计算时延、边缘端计算时

延、部分卸载时延以及关联服务交互时延四部分,在此基础之上,本研究建立了基于深度强化学习的服务卸载算法对关联服务进行协同卸载,以实现服务调用时延和服务间数据传输时延的最小化。

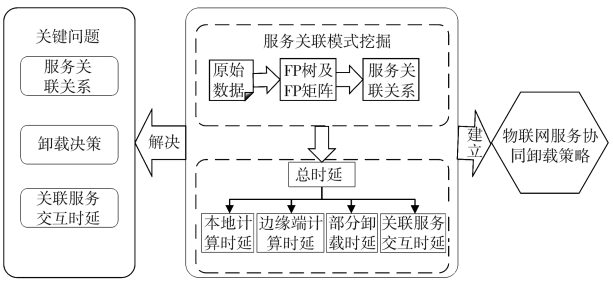

图 2　服务协同卸载框架

Fig.2　Service collaborative offloading framework

1.1　网络模型

信号在传输过程中会受到高斯白噪声等一系列因数的影响,建立网络信道模型时考虑了高斯白噪声对时延的影响。假设传输功率 P_i 是固定的,θ 为标准的路径损失传播指数,用户与边缘服务器之间的距离用 d_i 表示,则通信信道的信噪比

$$R_{\mathrm{SN}}=\frac{P_iH_id_i^{-\theta}}{\sigma^2},\tag{1}$$

式中,σ^2 为高斯白噪声的功率,H_i 为信道增益。因此,信道传输中的最大传输速率

$$R=W\,\mathrm{lb}(1+R_{\mathrm{SN}}),\tag{2}$$

式中 W 为信道带宽。

1.2　计算模型

1.2.1　服务关联模式挖掘

为了对具有关联关系的服务进行协同卸载,本研究提出基于频繁模式挖掘的服务关联模式挖掘算法。该算法将用户调用服务日志作为原始数据集,用户调用服务日志 E_{L} 可定义为形如 $E_{\mathrm{L}}=(C_{\mathrm{id}},T_S,\Pi)$ 的三元组,其中 C_{id} 为服务调用实例 ID 号,每次服务调用涉及一系列关联服务的执行;T_S 为时间戳的有限序列,用于表示原子服务的调用时间;Π 表示服务有限序列,由一系列原子服务 $M_i=(m_1,m_2,\cdots,m_n)$ 组成。本研究提出一种基于频繁模式的服务关联模式挖掘算法,该算法首先对原始数据集进行分析,基于数据集分布特征,自动设定了最小支持度阈值,然后构建相应的频繁模式(frequent-pattern, FP)树和 FP 矩阵,其中 FP 矩阵记录了大于支持度阈值的服务数据频度计数和兴趣度量值,然后通过并行化查找算法查找 FP 树挖掘出的具有区分力的关联服务对。具体算法可参见文献[18-19]。

1.2.2　本地执行

在挖掘出服务关联模式对后,本研究对关联服务进行协同卸载。服务卸载可分为本地计算、部分卸载、全部卸载,下面讨论各卸载模式下时延建模问题。

对于算力要求不高的服务,本研究直接在移动终端执行,假设 C_i^{local} 是移动设备本地计算的能力,F_i^{local} 表示任务执行要求的计算能力。因此本地计算时延

$$T_i^{\mathrm{local}}=\frac{F_i^{\mathrm{local}}}{C_i^{\mathrm{local}}}。\tag{3}$$

1.2.3　部分卸载

部分卸载将卸载任务分为两个部分,且需要将可卸载部分上传到边缘服务器进行计算,因此传输时延成为本研究要考虑的一个重要因素。整个卸载过程的总时延

$$T_i^{\mathrm{part}}=\max(T_i^{\mathrm{local}},T_i^{\mathrm{mec}}+2T_i^{\mathrm{tra}}),\tag{4}$$

式中:T_i^{tra} 为传输时延,$T_i^{\mathrm{tra}}=U_i/R$,其中 U_i 为卸载部分计算任务的大小;T_i^{mec} 为边缘服务器计算任务的时间,$T_i^{\mathrm{mec}}=F_i/C_i$,其中 F_i 为任务执行需要的计算能力,C_i 为边缘服务器的计算能力。因此总时延

$$T_i^{\mathrm{part}}=\max\left(\frac{F_i^{\mathrm{local}}}{C_i^{\mathrm{local}}},\ F_i/C_i+2U_i/R\right)。\tag{5}$$

1.2.4　全部卸载

对于计算密集型服务,本研究将其全部卸载到边缘服务器中执行,其计算时延

$$T_i^{\mathrm{all}}=2T_i^{\mathrm{tra}}+T_i^{\mathrm{mec}}=2M_i/R_i+F_i/C_i,\tag{6}$$

式中 M_i 为卸载计算任务的大小。

综上所述,假设边缘计算卸载率为 λ,$\lambda\in[0,1]$。$\lambda=0$ 时表示本地执行,$\lambda=1$ 时表示全部卸载,那么用户请求任务时所需时延可以表示为:

$$T_i=\begin{cases}\dfrac{F_i^{\mathrm{local}}}{C_i^{\mathrm{local}}}, & \lambda=0\\ \max\left(\dfrac{F_i^{\mathrm{local}}}{C_i^{\mathrm{local}}},\dfrac{F_i}{C_i}+2\dfrac{U_i}{R}\right), & 0<\lambda<1\\ 2\dfrac{M_i}{R}+\dfrac{F_i}{C_i}, & \lambda=1\end{cases}。\tag{7}$$

服务之间存在着大量数据传输,故仅考虑服务卸载时延无法满足现实需求,本研究加入了服务之间的数据传输时延。假设卸载服务中具有关联性的服务对有 n 对,则需要在时延中加入这 n 对子任务对数据交互的时延,每个子任务的交互时间用 t_{re} 表示,当服务全部在边缘端执行时,服务之间的数据传输时延远小于计算时延,故可忽略不计,因而本研究进一步优化时延表达式为:

$$T_i=\begin{cases}\dfrac{F_i^{\text{local}}}{C_i^{\text{local}}}, & \lambda=0\\ \max\left(\dfrac{F_i^{\text{local}}}{C_i^{\text{local}}},\dfrac{F_i}{C_i}+2\dfrac{U_i}{R}\right)+(n-\alpha)t_{\text{re}}, & 0<\lambda<1,\\ 2\dfrac{M_i}{R}+\dfrac{F_i}{C_i}+nt_{\text{re}}, & \lambda=1\end{cases} \tag{8}$$

式中 α 为本地执行的有数据交互的服务对。

1.3 数学模型

本研究根据用户任务量和边缘服务器负载情况动态调整 λ 的取值,将用户计算卸载时延表示为:

$$T_i=\max\left((1-\lambda)\frac{F_i^{\text{local}}}{C_i^{\text{local}}},\lambda\left(2\frac{U_i}{R}+\frac{F_i}{C_i}+(n-\alpha)t_{\text{re}}\right)\right)。\tag{9}$$

因此,优化函数为:

$$\min(T_i), \tag{10}$$

其约束条件为:

$$F_i^{\text{local}}\leqslant C^{\text{local}}, \tag{11}$$

$$F_i\leqslant C, \tag{12}$$

式中,C 为边缘服务器的最大计算能力,C^{local} 为用户移动设备的最大计算能力。

2 基于 DDPG 的服务卸载

强化学习模型如图 3 所示。在学习过程中的每个时间步长 t 中,代理观察状态为 s_t,并根据当前策略 φ 采取操作。当环境状态变为 s_{t+1} 时,在下一个时间步长中收到奖励值 r_t。环境的状态转换和所获得的奖励具有马尔可夫特征,即状态转换的概率和奖励只取决于环境的状态 s_t 和动作 a_t。代理根据决策策略接收这些状态,与环境进行交互和反向传播,以最大限度地获得期望奖励 r_t。

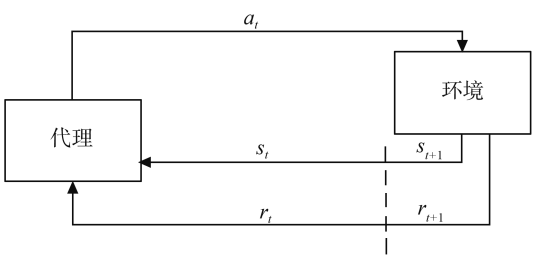

图 3 强化学习模型

Fig.3 The reinforcement learning model

2.1 状态空间

将网络情况或信道和资源情况作为系统的状态,在负载场景下会带来状态空间过于庞大等问题。为简化系统,本研究只考虑单用户的卸载需求,为此只需要每次观测一个服务需求的状态即可,从而可减少状态空间的维数,使强化算法易于收敛。

状态空间的形式可以描述为:

$$S_i=(q(i),p(i),D_{\text{remain}}(i),D(i),D_{\text{relation}}(i)), \tag{13}$$

式中,$q(i)$ 为边缘服务器位置信息,$p(i)$ 为用户的位置信息,$D_{\text{remain}}(i)$ 为剩余子任务量,$D(i)$ 为用户产生的随机任务量,$D_{\text{relation}}(i)$ 为关联任务量。

2.2 动作空间

本研究将动作空间定义为进行卸载时的卸载任务量,用元组表示为:

$$a_i=(k(i),\lambda(k)), \tag{14}$$

式中,$k(i)$ 为所服务用户的第 i 个卸载任务,$\lambda(k)$ 为任务卸载率。

2.3 奖励函数

本研究的目的是使用户的服务时延最小,定义系统状态 s_t 时执行动作 a_t 所获得的即时奖励

$$r_t=-T_i。 \tag{15}$$

强化学习算法中使用的动作值函数在时间步长 t 中描述了以下策略 φ 的预期回报

$$Q^{\varphi}(s_t,a_t)=E_{r_t,s_{t+1}\sim E}[r_t(s_t,a_t)+\gamma E_{a_{t+1}\sim\gamma}[Q^{\nu}(s_{t+1},a_{t+1})]]。 \tag{16}$$

因此,如果目标策略的更新是连续的,则将目标策略描述为函数 $\mu:A\leftarrow S$:

$$Q^{\varphi}(s_t,a_t)=E_{r_t,s_{t+1}\sim E}[r_t(s_t,a_t)+\gamma E_{a_{t+1}\sim\gamma}[Q^{\nu}(s_{t+1},\mu(s_{t+1}))]]。 \tag{17}$$

2.4 基于 DDPG 的服务卸载算法

DDPG 算法可以学习连续动作空间中的策略。如图 4 所示,DDPG 的结构形式类似 Actor-Critic。DDPG 可以分为策略网络和价值网络两个大网络。将每个网络再细分为目标网络和现实网络。Actor 输出的是一个确定性的动作,产生这个确定性动作的网络定义为 $\partial=\mu_\theta(s)$,Actor 的估计网络就是 $\mu_\theta(s)$,下标 θ 是神经网络的参数,这个估计网络就是用来输出实时的动作。此外,Actor 还有一个相同结构但不同参数的目标网络,用来更新价值网络 Critic。两个网络都是输出动作。

Critic 的作用是拟合价值函数 $Q_\omega(s,a)$,因此 Critic 也有一个估计网络和一个目标网络。这两个网络在输出端都输出当前状态的价值 q_{value},在输入端则有所不同。Critic 的目标网络输入有两个参数,分别是当前状态的观测值和 Actor 的目标网络输出的动作。Critic 的估计网络的输入则是当前 Actor 的估计网络输出的动作。目标网络是用来计算 Q_{target}。算法 1 展示了基于 DDPG 的计算卸载

算法。

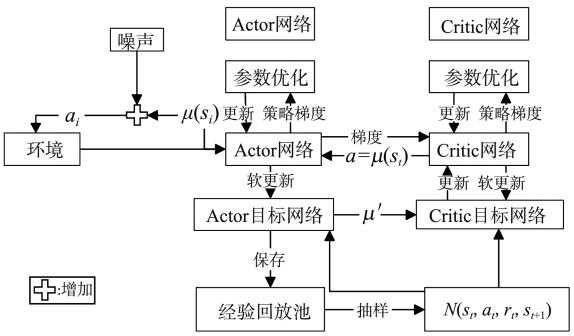

图 4　DDPG 模型

Fig.4　DDPG model

算法 1　DDPG 计算卸载算法

输入　步数长度 E;训练样本总数 I;Critic 网络学习率 λ_c;Actor 网络学习率 λ_a;折扣因子 γ;软更新系数 τ;经验回放池 B_m;最小抽样量 N;高斯噪声 n_i;最新状态 $q(i)$,$p(i)$,$D_{\text{remain}}(i)$,$D(i)$,$f(i)$;状态归一化参数 $\gamma_{D_{\text{reamin}}}$,$\gamma_{D_{\text{UE}}}$,$\gamma_x$,$\gamma_y$。

输出　回报 r_i。

(1) 随机初始化 Actor 网络和 Critic 网络权重 θ^{μ} 和 θ^{Q},清空 B_m;

(2) 使用初始化后的 Actor 网络和 Critic 网络权重对 Actor 目标网络和 Critic 目标网络进行初始化;

(3) for each episode E do;

(4)　重置仿真参数并且得到最初的观测状态;

(5)　for $i=1,2,\cdots,I$ do;

(6)　初始化 s_i 为当前状态;

(7)　状态归一化 $s_i \rightarrow s_i'$得到特征向量$\boldsymbol{\phi}(s')$;

(8)　根据状态 s_i 从 Actor 网络中获得带着 θ^{μ} 和 n_i 的动作 $a_i=\mu(s_i|\theta^{\mu})+n_i$;

(9)　执行 a_i,得到 r_i,并且观测下一个状态 s_{i+1};

(10)　将 s_{i+1}归一化下一个状态 s_{i+1}';

(11)　if B_m 没有满 then;

(12)　将 $(s_i', a_i, r_i, s_{i+1}')$ 状态存储到 B_m 中;

(13)　else;

(14)　用 $(s_i', a_i, r_i, s_{i+1}')$ 随机代替一组已经存在的状态值;

(15)　从 B_m 中随机抽取 $(s_i', a_i, r_i, s_{i+1}')$,$j=1,2,3,\cdots,I$;

(16)　计算当前目标的 Q $y_j=r_j+\gamma Q'(s_{j+1}', \mu'(s_{j+1}'|\theta^{\mu'}), \theta^{Q'})$;

(17)　使用均方差损失函数在 Critic 网络中更新当前网络的所有参数 $J(\theta^{Q})=\frac{1}{N}\sum_{j=1}^{N}((y_j-Q(s_j', a_j|\theta^{Q}))^2)$;

(18)　通过神经网络的梯度反向传播来更新 θ^{μ};

(19)　使用 τ 更新 Actor 目标网络和 Critic 目标网络参数;

(20)　end if;

(21)　end for;

(22)　end for;

(23)　得到 Actor 网络的 $\mu(s'|\theta^{\mu})$;

(24)　for each episode, $e=1,2,\cdots,E$　do;

(25)　重置边缘服务器的仿真参数并得到初始观测状态 s_i;

(26)　for $i=1,2,\cdots,N$ do;

(27)　状态归一化 $s_i \rightarrow s_i'$;

(28)　得到训练过的 Actor 网络的行为:$a_i=\mu(s_i'|\theta^{\mu})$;

(29)　执行动作 a_i 获得回报 r_i;

(30)　end for;

(31)　end for;

(32)　return r_i。

3　试验评估

本节通过仿真试验验证所提出的服务协同卸载算法的性能。采用 DQN[20] 和演员评论家[21](actor-critic,AC)算法两种深度强化学习的算法作为基准算法对比这 3 种算法的时延。所有算法测试均在 $T=60$ s 这一时间范围内对相同数量的服务进行卸载,并测试相应的时延大小。

3.1　仿真试验设置

试验中,假设用户位置在 $L\times W=100$ m×100 m 的正方形区域内随机取值,信道功率增益 H_i 在距离为 1 m 设置为−50 dB,传输带宽 B 设置为100 MHz,假设没有信号阻塞的高斯白噪声功率 $\sigma^2=-100$ dBm,用户端(user equipment,UE)和边缘服务器的计算能力分别设置为 $f_{\text{ue}}=0.8$ GHz 和 $f_{\text{mec}}=3$ GHz。用户和边缘服务器之间的距离为 d_i,由随机生成的用户位置决定。

3.2　试验结果分析

本研究进行了一系列的试验来确定在算法比较中使用的重要超参数的最优值[22]。该算法在不同学习速率下的收敛性能如图 5 所示。由图 5 可以发现,当 $\lambda_a=0.01$、$\lambda_c=0.02$ 时,算法收敛但收敛

时延较大;当 $\lambda_a=0.000\ 1$、$\lambda_c=0.000\ 2$ 时,训练步长较大时才能收敛到最优时延。因此,演员网络和评论家网络的最佳学习率分别为 $\lambda_a=0.001$、$\lambda_c=0.002$。

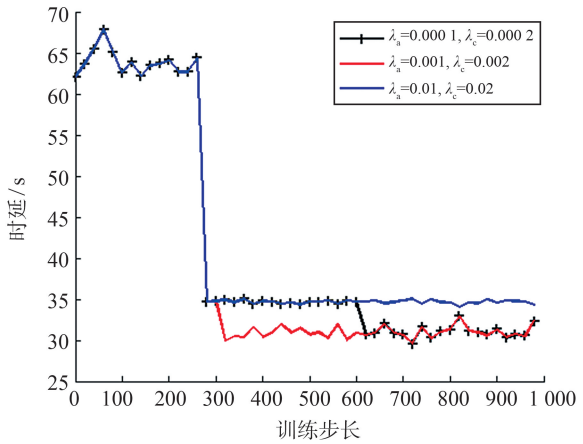

图 5 不同学习率下时延对比

Fig.5 Latency comparison under different learning rates

本研究也比较了不同折扣因子 γ 对该算法收敛性能的影响,如图 6 所示。γ 表示对动作进行估值时,下一步动作的价值所占权重。γ 越大,越能预见更遥远的未来。过小的 γ 将使评估网络无法及时地预见未来的事件,γ 过大,距离现在越远,评估网络对未来的预测精度越低。γ 越大,表示 Q 会将一个时间段内收集到的数据作为长期数据,导致算法的泛化能力较差。当折扣因子 $\gamma=0.50$ 时,经过训练的计算卸载策略的性能最好。因此,本研究在试验时将 γ 设置为 0.50。

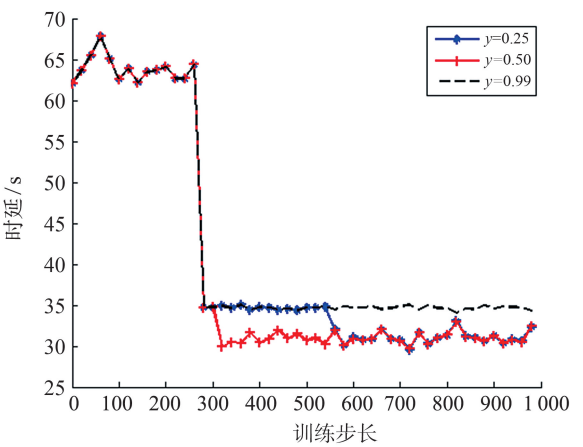

图 6 不同折扣因子下时延对比

Fig.6 Latency comparison under different discount factors

对该算法来说,探索参数 σ 的取值对算法的收敛性能有很大的影响,因此本研究比较不同探索参数下的时延变化,如图 7 所示。较大的 σ 会让智能体选择稳妥策略,使评估网络对 q 的估计变得更加准确,训练变得更加稳定。当算法在 $\sigma=0.01$ 时,最优时延在 28 s 左右进行波动。当 $\sigma=0.1$ 或 $\sigma=0.001$ 时,在训练步长接近 280 时下降较快,时延收敛在 35 s 和 30 s。因此,本研究选择 $\sigma=0.01$ 进行试验。

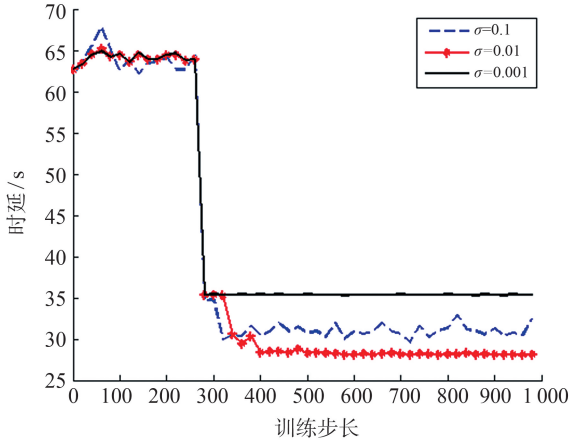

图 7 不同探索参数下时延对比

Fig.7 Latency comparison under different exploration parameters

参数确定后进行训练,不同卸载算法的训练时延对比如图 8 所示。

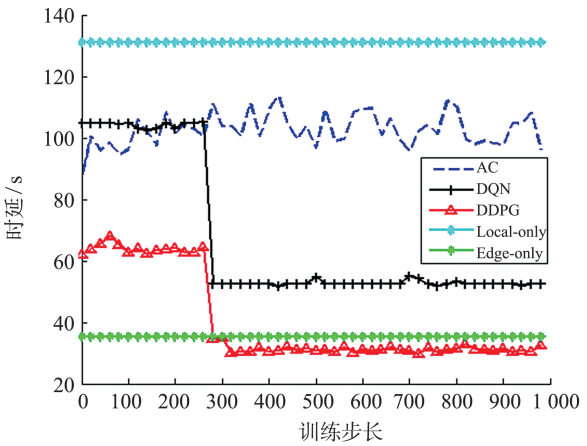

图 8 不同卸载算法时延对比

Fig.8 Latency comparison of different offloading algorithms

由图 8 可知,AC 算法无法收敛,DDPG 算法和 DQN 算法能够收敛并且训练步长在大约 300 时开始收敛。但 DQN 算法只能收敛到 50 左右,DDPG 的双网络模式使该系统能够收敛到 30 左右,收敛结果是 DQN 算法的 1/2。由于 AC 算法的 Actor 网络和 Critic 网络同时更新,Actor 网络的动作选择依赖于 Critic 网络的值函数,但 Critic 网络本身难以收敛,造成时延无法收敛。相比之下,DQN 和 DDPG 都具有 Actor 网络和 Critic 网络的双重网络结构。由于 DQN 只适用于离散的动作空间,而计算卸载的状态空间是连续的,因此 DQN 无法准确地找到最优的卸载策略。DDPG 算法探索了一个连续的动

作空间,并采取了精确的动作,最终得到了最优策略,显著减少了时延。而 DQN 算法的收敛性和处理时延远高于 DDPG。因此,在相同的任务量下,DDPG 算法的处理时延明显低于其他两种强化学习算法,并且 DDPG 算法优化的处理时延在训练步长较多时有更好的收敛性。

实际应用中任务量是不确定的[23-24],因此本研究通过改变任务量来观察在任务量增加的情况下时延的变化[24]。由于 AC 算法无法收敛,故仅对比了 DDPG 算法与 DQN 算法,如图 9 所示。随着数据量的增长,DDPG 算法的时延增长率减少且时延差距较小,说明该算法在一个大数据量的情况下时延增长较慢且比较稳定。而 DQN 算法在任务量为 160 Mbit 时有一个较大的增长,在200 Mbit时又出现了较大的波动。DQN 算法增长率较大且稳定性较差。因此在数据量增加时,DDPG 的性能明显优于 DQN。

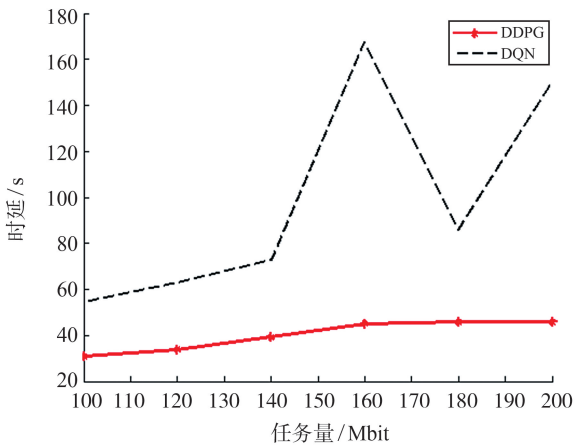

图 9　不同任务量下时延对比

Fig.9　Latency comparison under different task sizes

4　结论

本研究提出一种基于深度强化学习的物联网服务协同卸载方法,通过频繁模式挖掘算法挖掘出具有关联关系的服务对,对关联服务进行协同卸载,最小化服务时延,基于 DDPG 服务卸载算法来获得最优的卸载策略。通过仿真试验验证算法性能,试验通过比较学习率、折扣因子、探索参数等不同参数下对服务时延的影响,选取了最优参数,并比较了 3 种基准算法的服务时延。试验结果表明,该算法具有较好的服务时延。在未来的工作中,将研究多用户环境下服务协同卸载的问题,并改进相关算法以获取更好的算法性能。

参考文献:

[1] CANO J C, BERRIOS V, GARCIA B, et al. Evolution of IoT: an industry perspective[J]. IEEE Internet of Things Journal, 2018, 1(2): 12-17.

[2] RUAN J, WANG Y, CHAN F T S, et al. A life cycle framework of green IoT-based agriculture and its finance, operation, and management issues[J]. IEEE Communications Magazine, 2019, 57(3): 90-96.

[3] RIAHI A, CHALLAL Y, MOYAL P, et al. A game theoretic approach for privacy preserving model in IoT-based transportation[J]. IEEE Transactions on Intelligent Transportation Systems, 2019, 20(12): 4405-4414.

[4] YONGJOH S, SO-IN C, KOMPUNT P, et al. Development of an internet-of-healthcare system using blockchain [J]. IEEE Access, 2021, 9: 113017-113031.

[5] POPA D, POP F, SERBANESCU C, et al. Deep learning model for home automation and energy reduction in a smart home environment platform[J]. Neural Computing and Applications, 2019, 31: 1317-1337.

[6] DIN I U, GUIZANI M, HASSAN S, et al. The internet of things: a review of enabled technologies and future challenges[J]. IEEE Access, 2019, 7: 7606-7640.

[7] ARMBRUST M, FOX A, GRIFFITH R, et al. A view of cloud computing [J]. Communications of the ACM, 2010, 53(4): 50-58.

[8] 施巍松, 孙辉, 曹杰, 等. 边缘计算:万物互联时代新型计算模型[J]. 计算机研究与发展, 2017, 54(5): 907-924.
SHI Weisong, SUN Hui, CAO Jie, et al. Edge computing: a new computing model in the internet of everything era [J]. Computer Research and Development, 2017, 54 (5): 907-924.

[9] SHI W S, CAO J, ZHANG Q, et al. Edge computing: vision and challenges[J]. IEEE Internet of Things Journal, 2016, 3(5): 637-646.

[10] AHMED A, AHMED E. A survey on mobile edge computing[C]//Proceedings of the 10th International Conference on Intelligent Systems and Control. Coimbatore, India: IEEE, 2016: 1-8.

[11] 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): 344-352.

[12] WU Z, LU Z, HUNG P C K, et al. QaMeC: a QoS-driven iovs application optimizing deployment scheme in multimedia edge clouds[J]. Future Generation Computer Systems, 2019, 92: 17-28.

[13] ZHANG H H, LI J L, YUAN Q. Edge service migration

for vehicular networks based on multi-agent deep reinforcement learning[C]//Technologies and Services Toward Smart Cities: 6th International Conference. Kaohsiung, Taiwan, China: Lecture Notes in Computer Science, 2019: 344-352.

[14] 葛海波, 李文浩, 冯安琪,等. 改进遗传算法的边缘计算卸载策略[J]. 西安邮电大学学报, 2020, 25(3): 7-13.
GE Haibo, LI Wenhao, FENG Anqi, et al. Improved genetic algorithm for edge computing offloading strategy [J]. Journal of Xi′an University of Posts and Telecommunications, 2020, 25(3): 7-13.

[15] 张文柱, 曹琲琲, 余静华. 移动边缘计算中一种多用户计算卸载方法[J]. 西安电子科技大学学报, 2020, 47(6): 131-138.
ZHANG Wenzhu, CAO Beibei, YU Jinghua. A multi-user computing offloading method in mobile edge computing [J]. Journal of Xidian University, 2020, 47(6): 131-138.

[16] 罗斌. MEC 计算卸载策略的研究与应用[D]. 沈阳:中国科学院大学,2020.
LUO Bin. Research and application of MEC computing offloading strategy[D]. Shenyang: University of Chinese Academy of Sciences, 2020.

[17] 詹文翰. 移动边缘网络计算卸载调度与资源管理策略优化研究[D]. 成都:电子科技大学, 2020.
ZHAN Wenhan. Research on computing offload scheduling and resource management strategy optimization in mobile edge network [D]. Chengdu: University of Electronic Science and Technology of China, 2020.

[18] HUANG Y Z, HUANG J W, CHEN J L, et al. Poster: interacting data-intensive services mining and placement in mobile edge clouds[C]//Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking. New York, USA: Association for Computing Machinery, 2017: 558-560.

[19] HUANG Y Z, HUANG J W, LIU C, et al. PFPMine: a parallel approach for discovering interacting data entities in data-intensive cloud workflows[J]. Future Generation Computer Systems, 2020, 113: 474-487.

[20] RICHARD S, DAVID M, SATINDER S, et al. Policy gradient methods for reinforcement learning with function approximation[C]//Neural Information Processing Systems (NIPS). Cambridge, MA, USA: MIT Press, 1999: 1057-1063.

[21] MUIH V, KAVUKCUOGLUC K, SILVER D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518: 529-533.

[22] 葛志诚, 徐恪, 陈靓. 一种移动内容分发网络的分层协同缓存机制[J]. 计算机学报, 2018, 41(12): 2769-2786.
GE Zhicheng, XU Ke, CHEN Liang, et al. A hierarchical cooperative caching strategy for mobile content delivery network[J]. Chinese Journal of Computers, 2018, 41(12): 2769-2786.

[23] LUO Q Y, LI C L, SHI W S, et al. Self-learning based computation offloading for internet of vehicles: model and algorithm[J]. IEEE Transactions on Wireless Communications, 2021, 20(9): 5913-5925.

[24] HUANG J W, LÜ B F, WU Y A, et al. Dynamic admission control and resource allocation for mobile edge computing enabled small cell network[J]. IEEE Transactions on Vehicular Technology, 2022, 71 (2): 1964-1973.

(编辑:孙亚彤)