2020 年 10 月 第 10 期
Vol. 41 No. 10 2020

面向多用户移动边缘计算轻量任务卸载优化

张文献,杜永文,张希权

(兰州交通大学 电子与信息工程学院,兰州 730070)
E-mail: 0218654@ stu. lzjtu. edu. cn

摘 要: 在移动设备资源受限的情况下,移动边缘计算(MEC)通过合理分配边缘服务器和多个移动设备的计算资源来提高移动设备用户的计算体验. 然而这种密集计算问题是一种高维的 NP 难问题,传统机器学习方法在解决该问题的时候并没有良好的效果. 本文将最佳计算卸载问题建模为马尔可夫决策过程,目标是最大化长期效用性能,根据队列状态,能量队列状态以及移动用户与基站之间的信道质量做出卸载决策. 为了降低状态空间中高维性的问题,提出了应用 DDPG 的基于候选网络优化 ECOO(Edge Computing Optimize Offloading)算法,从而产生一种用于解决随机任务卸载的新型学习算法. 通过实验证明,提出的 ECOO 算法在时延和能耗方面均优于其它传统机器学习方法.

关 键 词: 移动边缘计算; 任务卸载; 深度强化学习; 卸载决策; 多用户
中图分类号: TP391 **文献标识码:** A **文 章 编 号:** 1000-1220(2020)10-2056-07

Optimization of Lightweight Task Offloading for Multi-user Mobile Edge Computing

ZHANG Wen-xian, DU Yong-wen, ZHANG Xi-quan
(School of Electronics and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China)

Abstract: In the case of limited resources of mobile devices, mobile edge computing (MEC) improves the computing experience of mobile device users by reasonably allocating computing resources of edge servers and multiple mobile devices. However, this kind of intensive computing problem is a kind of high dimensional NP hard problem, and some machine learning methods do not have a good effect in solving this problem. In this paper, the Markov decision process model is established to find the best task off-loading scheme, which maximizes the long-term utility performance, so as to make the best offloading decision according to the queue state, energy queue state and channel quality between mobile users and BS. In order to explore the curse of high dimension in state space, we propose a candidate network-based Edge Computing Optimize Offloading (ECOO) algorithm with the application of DDPG algorithm. Experimental results show that the proposed ECOO algorithm is superior to other traditional machine learning methods in terms of time delay and energy consumption.

Key words: mobile edge computing; task offloading; deep reinforcement learning; offloading decision; multi-user

1 引 言

随着移动通信技术的不断发展,移动终端服务给人们的生活带来了极大的便利. 同时,新的需求相继涌现,给移动计算技术带来了许多新的挑战. 其中最显著就是计算需求和计算能力不匹配的问题. 同时移动设备通常存在高能耗问题,这使得它无法支撑规模较大的计算任务. 根据思科的报告表明[1],移动数据流量将在未来 5 年增长 7 倍,到 2021 年将达到每月 49 艾字节,同时全球 IoT 设备数量将从目前的 80 亿增加到 120 亿.

由于数据流量的激增,终端已经无法承载这样大规模的计算需求. 利用中央云技术处理大量数据,一时被认为是一个有效的解决方案. 但是,云计算在解决终端计算需求的过程中存在一些劣势: 延迟性高、能量消耗大以及网络带宽占用高等. 为了提高移动应用的服务质量(QoS),移动边缘计算技术(MEC)作为一种新的解决方案应运而生,同时,移动边缘计算技术可以用来解决移动设备上不断增加的移动应用程序计算需求问题.

作为 MEC 的一项关键技术——任务卸载,它首先通过卸载决策去确定哪些计算需要在本地处理,哪些计算需要上传到边缘服务器去处理. 最后通过计算资源分配来确定最后卸载的位置. 任务卸载技术可以有效解决 MEC 框架在资源存储、计算性能以及能耗等方面存在的不足. 这不仅减轻了核心网的压力,而且降低了因传输带来的时延.

此外,在无线衰落环境中,时变无线信道条件在很大程度上影响着无线 MEC 系统的最佳卸载决策[0]. 在多用户场景中,主要问题是个体计算模式的联合优化(即,卸载或者本地计算)和无线资源分配(例如,在进行卸载时该选择哪些基站). 由于场景中存在二进制卸载变量,这些问题通常被表述为混合整数规划(MIP)问题. 为了解决 MIP 问题,大多研究

[1] http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-whitepaper-c11-520862.html.

收稿日期: 2019-12-05 收修改稿日期: 2020-04-10 基金项目: 国家自然科学基金项目(11461038)资助; 甘肃省科技支撑计划项目(144NKCA040)资助. 作者简介: 张文献,男,1995 年生,硕士研究生,研究方向为移动边缘计算、人工智能; 杜永文,男,1974 年生,博士,副教授,研究方向为嵌入式系统; 张希权,男,1996 年生,硕士研究生,研究方向为移动边缘计算、嵌入式系统.

采用了分支定界算法[2]和动态规划[3]，但它们的计算复杂度过高，不适合应用在大规模 MEC 网络. 为了降低计算复杂度，一些新的研究提出了启发式局部搜索和凸松弛方法. 然而，它们都需要大量的迭代才能达到局部最优值. 因此，不适合在快速衰落信道中进行实时卸载决策，一旦信道衰落变化的速度变快，这些方法就需要重新解决优化问题.

随着智能体数量的增加，状态空间维度的爆炸将使传统的表格方法变得不可行[4]，传统的强化学习(RL)算法将无法很好的解决维数问题. 最近，深度强化学习(DRL)已经被证明可以通过利用深度神经网络(DNN)有效的逼近 RL 的 Q 值[5]. 深度强化学习与移动边缘计算结合可以使移动设备基于任务队列状态、能量最大化长期效用、队列状态以及信道质量学习到最优任务卸载决策和能量分配方案.

在本文中，考虑了如何更有效的判别计算任务是否需要卸载到边缘节点上. 提出具有多用户的 MEC 网络，其中每个用户都遵循二进制卸载策略. 本文的目标是根据时变无线信道共同优化用户任务卸载决策. 为此我们提出了基于深度强化学习的计算卸载框架，以实现最低时延与能耗的卸载. 与现有基于深度强化学习的算法相比，本文的工作如下:

1) 在现有的基于离散动作空间做出决策的基础上，提出了一种基于候选网络的连续动作空间的算法，以获得更好的本地执行和任务卸载的功率控制.

2) 基于多用户的 MEC 系统，针对每个具有任务随机到达和时变无线信道的移动用户独立地学习动态卸载策略. 依此策略实现 ECOO(Edge Computing Optimize Offloading)算法，实现功耗和计算成本降低，并减少时延.

3) 通过实验仿真，说明通过 ECOO 与传统的 DQN 和 DDPG 算法的分散策略中学习的性能相比，在能耗与时延方面有更好的仿真结果，并分析了每个用户的功率延迟权衡.

2　相关工作

现有的大量工作研究了移动边缘计算的优化问题. 一些工作为了降低时延与能耗，将卸载决策的问题建模成 MIP 问题，并考虑怎样应对随机任务到达和信道快速衰落.

卸载决策作为 MIP 问题，有许多相关的工作将 MEC 网络中的计算模式决策问题和资源分配问题联合建模. 例如，文献[6]提出了一种坐标下降(CD)方法，它每次沿着一个变量维度搜索. 文献[7]研究了一种类似于多服务器 MEC 网络的启发式搜索方法，它可以动态地调整二进制卸载决策. 另一种广泛采用的启发式方法是凸松弛法，例如，通过将整数变量放宽到 0 到 1 之间的连续变化[8]，或者通过用二次约束逼近优化[9]. 尽管如此，一方面，降低复杂性的启发式算法解决方案质量无法保证. 另一方面，基于搜索和凸松弛方法都需要相当多的迭代才能达到令人满意的局部最优值，并且不适用于快速衰落信道.

为了应对随机任务到达和信道快速衰落，MEC 系统中无线电和计算资源的动态联合控制策略变得更具挑战性[10-13]. 文献[10]，作者考虑了多用户的部分计算卸载，并研究了基于时分多址和正交频分多址的资源分配，目的是最大限度地减少用户能耗的加权总和. 在文献[11]中，作者设计了多输入多输出系统，通过形成的多输入多输出波束和计算资源分配的联合优化，解决任务卸载的能耗问题. 文献[12]研究了用于能量获取的绿色 MEC 系统，其中利用延迟成本解决了执行延迟和任务失败的问题. 对于多用户场景，作者主要讨论功率延迟权衡[13].

最近，许多工作在无线网络的资源管理研究上已经取得进展，在文献[14]中，作者提出了一种深度强化学习算法，用于研究在时变的实际无线环境中的最佳缓存和干扰抗性. 在文献[15]中，Chen 等人在具有多个基站的超密集切片无线接入网络中，为移动用户判断哪些 MEC 可用于任务卸载. 作者提出了一种基于深度 Q 网络(DQN)的计算卸载算法策略，以获得最佳策略，进而最大化长期效用性能. 在现有的工作中，只有基于集中式 DRL 算法来解决 MEC 系统中的最优计算卸载，而用于多用户 MEC 系统动态任务卸载控制的 DRL 算法设计研究数量仍然是欠缺的.

3　问题模型

如图 1 所示为一个多用户 MEC 系统，它由一个 MEC 服务器，一个基站(BS)和一组移动用户组成 $N=\{1,2,\cdots,N\}$. 其中每个用户都需要完成计算密集型任务. 由于需要解决每个移动设备计算能力有限的问题，将 MEC 服务器部署在 BS 的附近，这样可以通过不同的用户需求来改善用户的 QoS. 此外，随着移动用户的增加，联合用户共同处理问题使每个用户的分散任务卸载问题解决起来更加便利，因为这可以减少用户与 MEC 服务器之间的系统开销，提高 MEC 系统的扩展性. 在下文中将详细介绍建模过程.

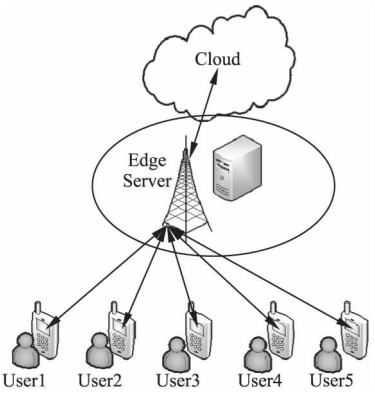

图 1　多用户移动边缘计算模型

Fig. 1　Multi-user MEC system

3.1　网络模型

考虑到一个 5G MBS 或 BS 作为一个 MEC 系统，并通过采用线性检测算法来管理多个移动用户上行链路传输(ZF). 对于每个时隙 $t\in T$，BS 的接收信号可以写为[14]:

$$x(t)=\sum_{n=1}^{N}g_n(t)\sqrt{p_{0,n}(t)}s_n(t)+n(t) \tag{1}$$

其中 $p_{0,n}(t)$ 是用户 n 卸载任务数据位的传输功率，$s_m(t)$ 是具有单位方差的复数数据符号，$n(t)\in(0,\sigma^2)$ 是加性高斯(AWGN)的矢量，方差为 σ^2. 为了表征每个移动用户时隙的相关性，本文采用以下的高斯马尔可夫块衰落自回归模型[17]:

$$g_n(t)=\rho_n g_n(t-1)+\sqrt{1-\rho_n^2}e(t) \tag{2}$$

其中 ρ_m 是时隙 t 和 t − 1 之间的归一化信道相关系数，$e(t)$ 是误差矢量.

$H(t) = [h_1(t), \cdots, h_n(t)]$ 是 BS 和 N 个用户之间的 $N \times M$ 的信道矩阵，BS 处的线性 ZF 检测器可以由信道矩阵写入 $H^{\dagger}(t) = (H^H(t)H(t))^{-1}H^H(t)$，如果 M 行的 $H^{\dagger}(t)$ 由 $z_n^H(t)$ 表示，则用户 n 的接收信号为 $z_n^H(t)x(t) = \sqrt{p_{0,n}(t)}s_n(t) + z_n^H(t)n(t)$. 因此，相应的信号与干扰加噪声(SINR)为:

$$\Upsilon_n(t) = \frac{1}{\|z_n(t)\|^2\sigma^2} = \frac{1}{[H^H(t)H(t))^{-1}]_{nn}\sigma^2} \quad (3)$$

从公式(3)中可以验证每个用户的信噪比 SINR 随着用户数 N 的增加而变差，为了解决这一问题，需要为每个用户分配更多的卸载功率.

3.2 计算模型

在这一部分中，我们将讨论每个移动用户是如何利用本地执行或者计算卸载来满足其运行应用程序. 假设所有的应用程序是细粒度的[18]，$d_{1,m}(t)$ 表示计算任务在本地移动设备上，$d_{0,n}(t)$ 表示将计算任务卸载到边缘服务器上执行. 在时隙 t 开始时，用户 n 的任务缓冲区的队列长度为:

$$B_n(t+1) = [B_n(t) - (d_{1,n}(t) + d_{0,n}(t))]^+ + a_n(t), \forall t \in T \quad (4)$$

其中 $a_n(t)$ 表示时隙 t 期间任务到达的数量.

1) 本地计算: $p_{1,n}(t) \in [0, p_{1,n}]$ 为本地执行分配功率. 首先，假设用户 n 处理一个任务所需要的 CPU 周期数为 L_n，周期数可以通过离线测量来估算[19]. 使用 DVFS 技术调整芯片电压[20]，κ 为在时隙 t 时写入有效开关电容的 CPU 频率. 因此，在 t 时隙的本地处理可以通过以下方式导出:

$$\mu_n(t) = \sqrt[3]{p_{1,n}(t)/\kappa} \quad (5)$$

$$d_{1,n}(t) = \tau_0\mu_n(t)L_n^{-1} \quad (6)$$

2) 边缘计算: 为了利用边缘计算解决问题，MEC 服务器通常配备有足够的计算资源，通过 BS 卸载到 MEC 服务器的所有任务都将被处理. 因此，根据公式(3)可以得出用户 n 的卸载数据的比特量.

$$d_{0,n} = \tau_0 W \log_2(1 + \gamma_n(t)) \quad (7)$$

其中 W 是系统带宽.

3.3 能耗模型

智能手机，传感器和远程服务器在内的所有计算设备在一定执行时间内的总能耗主要由两部分组成: 计算能耗 $Energy^{comp}$ 以及用于卸载的移动设备能耗 $Energy^{off}$. 首先，能耗模型 i 计算如下:

$$\begin{cases} K * P_i^{\max} + (1-K) * P_i^{\max} * o \\ 0 \end{cases} \quad (8)$$

其中 K 表示计算设备空闲状态与满载状态的百分比，$P_i^{\max}$ 表示设备满负荷状态下消耗的能量，o 表示 CPU 利用率. 此外，计算设备的负载随时间的变化不断变化. 假设 $o(t)$ 是用于计算每单位时间设备的 CPU 利用率函数. 单位时间 t 内设备的计算能耗为:

$$\begin{cases} Energy_i^{comp}(t_0) = \int_{t_0}^{t_{0+t}} P_i(o(t))\,dt \\ o(t) = \min\left\{1, \frac{\sum_{j=1}^{alltask_i} M_j}{DR_i}\right\} \end{cases} \quad (9)$$

其中 $alltask_i$ 表示任务数为 i 的计算设备，M_j 表示所需的 CPU 资源，DR_i 表示 CPU 总资源.

3.4 成本模型

移动用户需要为远程服务器提供的计算资源支付相应的费用. 基于剩余资源量的动态价格模型. 剩余资源量越少，资源价格越高. 此时，用户更愿意选择单价较低的服务节点作为卸载目标，以降低用户成本，提高资源利用率. 单位时间 t 中剩余资源量的动态价格模型[21]:

$$Cost_i(t_0) = CC + UT * RPM * TM * \int_{t_0}^{t_0+t} LU(t)\,dt \quad (10)$$

CC 表示当前设备的成本，UT 表示计算费用的间隔时间，RPM 表示计算资源的单价，TM 表示当前设备的总计算资源，$LU(t)$ 表示当前设备每单位时间使用的计算资源比率. 同时，由于本地设备的计算资源属于用户自身，不需要计算成本，因此，远程设备的总成本:

$$Cost_{sum}(t_0) = \sum_{i=1}^{1+m} Cost_i(t_0) \quad (11)$$

4 基于 DRL 的动态计算卸载

强化学习可以通过特定场景中的自学能力来做出最佳决策，它通过将所有问题抽象为智能体与环境之间的交互过程来进行建模. 在交互过程的每个时间步骤，智能体接收环境的状态并选择相应的响应动作. 然后在下一个时间步骤中，智能体根据环境的反馈获得奖励值和新的状态. 基于不断的学习，强化学习能够适应环境. 虽然强化学习具有很多优势，但是它缺乏可扩展性，并且仅限于相当低的维度问题. 为了解决强化学习中决策困难的问题，深度强化学习将深度学习的感知能力与强化学习的决策能力相结合. 依靠强大的函数逼近和深度神经网络的表达学习特性来解决高维状态空间和动作空间的环境问题[22].

在本节中，采用改进的 DDPG 算法[23]，在每个用户处能独立地学习分散的动态计算卸载策略，为本地执行和任务卸载分配功率. 特别的，每个用户都没有 MEC 系统的先验知识，所以每个用户都不知道用户 N 的数量. 下面引入分散动态计算卸载的 DRL 框架，定义状态空间，动作空间和奖励函数. 最后介绍 ECOO 算法.

4.1 DRL 框架

状态空间: 为了全面研究 MEC 中子任务和服务器资源之间的关系，对系统的全面观察包括所有用户的信道向量和任务缓冲区的队列长度. 但是，现实中在 BS 上收集这些信息后将它们分发给每个用户的系统开销是巨大的. 为了减少开销并使 MEC 系统更具可扩展性，假设每个用户根据其独立的状态来选择动作.

在时隙 t 开始时，每个用户 n 的数据缓冲区 $B_n(t)$ 的队列长度将根据式(4)更新. 同时，其将接收一个来自 BS 的反馈，该反馈在 BS 处传给用户 n 最后接收到的 SINR. 此外，还可以通过信道互异性来估计用于即将到来的上行链路传输的信道矢量 $h_n(t)$，定义的状态为:

$$S_{n,t} = [B_n(t), \phi_n(t-1), h_n(t)] \quad (12)$$

$$\phi_n(t) = \frac{1}{\|h_n(t)\|^2 [H^H(t)H(t))^{-1}]_{nn}} \quad (13)$$

为了确保卸载决策能在本地移动设备或者远程服务器上执行子任务,子任务的卸载决策只需要考虑数量为 N + M + 1 的计算设备,其中包括一个云数据中心,N 个本地移动设备,M 个边缘服务器.

动作空间: 根据每个用户智能体观察到的系统的当前状态 $S_{n,t}$,为每个时隙 t 选择本地执行或是任务卸载的分配功率动作 $a_{n,t}$:

$$a_{n,t} = [P_{1,n}(t), P_{0,n}(t)] \tag{14}$$

不同于其他传统 DRL 算法,从若干预定义的离散功率电平中进行选择,本文通过应用改进的 DDPG 算法可以从连续动作空间中优化分配功率,并且显著减少离散动作空间的高维度缺陷.

为了将子任务卸载到合适的计算设备,动作空间与深度强化学习中的计算设备集一一对应,$(0/1)_i^j$ 表示是否将子任务 i 卸载到计算设备 j 上. 此外,通过对动作空间的预处理,确保深度强化学习算法可以在迭代过程中学习有效的动作,避免为无效动作设置惩罚值,这可以降低奖励函数的复杂度和计算复杂度.

奖励函数: 每个智能体的行为都是通过奖励驱动的. 为了学习能量感知动态计算卸载策略的 MEC 模型,研究在可接受的缓冲延迟内花费最少的能耗完成任务. 根据 Little 定理[24],任务缓冲区的平均队列长度与缓冲延迟成比例. 我们认为奖励值最低的值是最佳的,因此定义每个用户 n 在时隙 t 之后接收的奖励函数 $r_{n,t}$:

$$r_{n,t} = -\omega_{n,1}P_{1,n}(t) - \omega_{n,2}P_{0,n}(t) - \omega_{n,3}B_n(t) \tag{15}$$

其中 $\omega_{n,1}$,$\omega_{n,2}$和 $\omega_{n,3}$都是非负加权因子,通过设置不同的值,可以动态权衡在任务卸载时的能量消耗和缓冲延迟. 改进的 DDPG 在策略 o_n 下从初始状态开始最大化用户 n 的值函数:

$$V^{o_n}(s_{n,1}) = \mathrm{E}\left[\sum_{t=1}^{\infty}\gamma^{t-1}r_{n,t} \mid s_{n,1}\right] \tag{16}$$

当 $\gamma \to 1$ 时,它可用于近似每个用户在无线范围内未损失奖励[25],平均计算成本为:

$$C_n(s_{n,t}) = E\left[\lim_{T\to\infty}\frac{1}{T}\sum_{i=t}^{T} -\omega_{n,1}P_{1,n}(i) - \omega_{n,2}P_{0,n}(i) - \omega_{n,3}B_n(i) \mid s_{n,t}\right] \tag{17}$$

4.2　基于候选网络的优化

ECOO 算法的伪代码描述如下:

输入: 批次大小 δ,当前神经网络 MainNet,候选网络集 Net

输出: 训练完成的神经网络

1. 随机选择大小为 δ 的数据样本 miniBatch
2. 初始化状态列表 stateList 和奖励列表 rewardList
3. 初始化数组 NetTotalArray 来存储的所有奖励值 Net_2,$Net_2 \in Net$
4. totalValue = 0
5. N = Net 的大小
6. for i = 0, δ − 1 do
7.　targetValue = 0
8.　根据 miniBatch 中的第 i 个样本获取当前状态 S_t,动作 a_t,奖励 r_t 和下一个状态 S_{t+1}
9.　根据状态 S_t 计算网络 MainNet 的奖励列表 Q_1
10.　根据状态 S_{t+1} 计算网络 MainNet 的奖励列表 Q_2
11.　在 Q_2 中选择与最大奖励值相对应的动作 a_{t+1}
12.　初始化奖励值列表 NetValueList 以存储候选人计算的奖励值
13.　for j = 0, N − 1 do
14.　　基于状态 S_{t+1} 和动作 a_{t+1} 计算第 j 个网络的奖励值 Q_j
15.　　NetValueList [i] = Q_j
16.　end for
17.　targetValue = Max(NetValueList)
18.　Q_1 [a_t] = targetValue
19.　totalValue = totalValue + r_t
20.　stateList [i] = s_t
21.　rewardList [i] = Q_j
22.　通过主网络中的每步替换 *Net* 中最早的目标网络
23. end for
24. if totalValue > Min(NetTotalArray) do
25. 用 NetTotalArray 中总奖励值最小的网络替换主网络,并将对应的总奖励值更新为 totalValue
26. end if
27. 根据 stateList 和 rewardList 训练主网络
28. Return MainNet

图 2 表示了基于 DRL 的移动边缘计算卸载的模型,其包括: 高维状态空间表示、神经网络结构设计和长期奖励的最大化. 如图 2 所示,智能体观察环境并获得原始服务信号,例如信噪比和无线信道信息. 这些信号可以组合成高维状态输入,然后进入到深度神经网络中. 需要特定结构的深度神经网络,例如卷积神经网络或递归神经网络,它们能够挖掘有用的信息并输出值函数或策略. 根据输出,智能体可以选择出一个动作,该动作代表下一个卸载动作. 然后观察所得的卸载性能,并将其作为奖励返回给控制器,来决定是否卸载. 卸载智能体使用奖励来训练和改进深度神经网络模型,以最大化累积奖励.

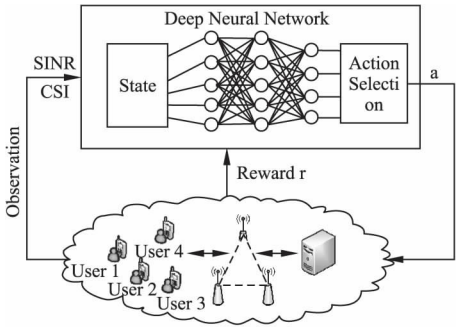

图 2　深度强化学习计算卸载过程

Fig. 2　Deep reinforcement learning computing offloading process

假设所有的计算任务具有等大的规模,并且假定边缘节点的计算能力足以进行任务的计算. 目标是找到最佳的缓存策略,以最大程度地减少卸载时造成的时延和能耗. DQN 算法通过延迟更新,以保证当前网络与目标网络之间的参数差异,从而提高训练过程的稳定性. 但是当噪声或误差而过估训练过程的动作值时,在后面的参数更新过程中,相应动作的值也将不可避免地被高估. 综合考虑多个候选网络的结果,将行动选择网络与行动价值网络分离,以确保最优学习策略[26]. 这些网络的详细功能如图 3 所示,具体将在下文介绍这些网络.

输入: 每次请求 T,智能体接收的状态输入到它的神经网络. 仅从索引为 0 的当前请求任务到索引为 1 的请求任务中

提取特征. 实际上,除了此处所述的请求信息外,状态输入中还可以包含有关上下文和边缘网络的更多原始观察结果.

策略: 收到状态后,智能体需要判断边缘节点是否有该任务的计算资源,如果有,则智能体将进行计算卸载动作. 智能体根据策略选择操作,由评论者网络表示. 应用每个动作后,移动边缘卸载环境为智能体返回了奖励,它定义了每个请求中的卸载流量. 执行者网络参数的每次更新 θ 遵循策略梯度[27]:

$$\theta \leftarrow \theta + \alpha \sum_t \nabla_\theta log_\pi(a_t | s_t; \theta) A(s_t, a_t; \theta, \theta_v) \quad (18)$$

评论者网络参数遵循时差法[28]:

$$\theta_v \leftarrow \theta_v - \alpha' \sum t \nabla_\theta (r_t + \gamma V(ls_{t+1}; \theta_v) - V(s_t; \theta_v))^2 \quad (19)$$

这里 α 和 α' 是学习率. 为了权衡与环境之间的交互,将熵正则化(其定义为每个时间步长下策略的熵的加权梯度)添加到公式(18). 训练过程可以离线或在线进行. 通过脱机的方式,卸载策略是先验生成的(在训练阶段),然后在部署后保持不变. 通过在线方式,缓存策略直接在边缘节点上进行训练,并在新数据到达时定期进行更新.

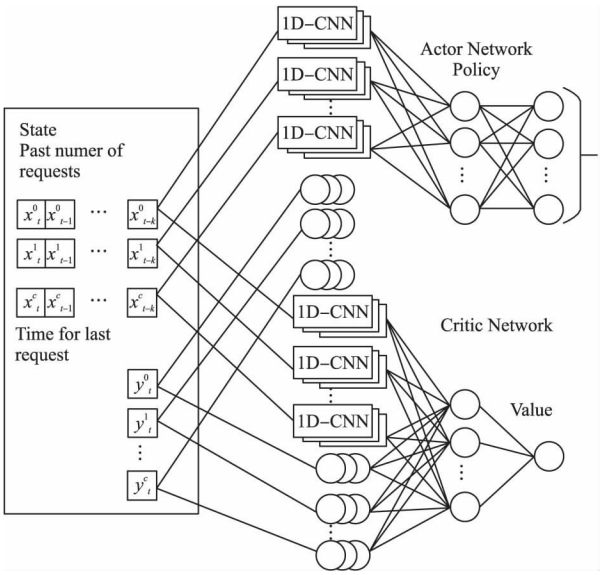

图 3 神经网络架构

Fig. 3 Neural network architecture

鉴于 MEC 中资源随时间的逐渐变化以及 LSTM 网络长期状态的记忆能力,本文提出将 LSTM 与 DDPG 结合起来处理随时间变化的计算卸载问题. 循环结构用于集成长期的历史数据,通过用 LSTM 层替换 DDPG 网络的最后一个完全连接层来更准确地估计当前状态. 并且基于候选网络的优化来选择最优的学习策略.

如图 4 所示,假设候选网络集合 $Net = (net_1, net_2, \cdots, net_i, \cdots, net_n)$ 可以存储总数为 n' 的网络,有 m' 个网络集中的网络 Net_1,在满足固定迭代次数 C 后更新它们. 网络设置 Net_2 具有 $(n'-m')$ 个通过比较奖励值来选择进行更新的网络. 当网络设置的数量 Net_2 小于 $(n'-m')$ 时,每次迭代生成的当前网络都会被添加到 Net_2 作为候选者网络. Net_2 等于 $(n'-m')$,当前网络和 Net_2 将训练当前选定的状态-动作对. 如果大于,具有最小奖励值的候选网络被当前网络替换,否则继续训练.

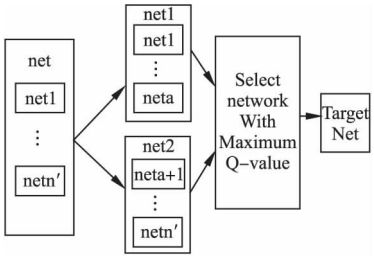

图 4 候选网络更新流程图

Fig. 4 Candidate network update flow chart

5 实验仿真

为验证本文提出的 ECOO 算法在多用户移动边缘计算模型中的有效性,本文挑选出效果较好的几种算法与本文算法进行比较. 实验环境在 ubantu16.1, python3.6.8, tensorflow1.8 下仿真. 通过比较成本,能耗,服务延迟,反映出卸载决策的优缺点. 在大规模异构集群中,实现的算法包括: 贪婪本地执行优先(GD-Local),贪婪计算优先卸载(GF-Offload),基于 DQN 的动态卸载(DQN),基于 DDPG 的动态卸载(DDPG)以及 ECOO.

5.1 模拟设置

在 MEC 系统中,时间间隔 $\tau_0 = 1ms$. 每一次迭代开始的时候,每个用户 n 的信道向量初始化为 $h_n(0) \sim CN(0, h_0(d_0/d_n)^\alpha I_N)$,其中路径损耗常数 $d_0 = 1m$,路径损耗指数 $\alpha = 3$,信道相关指数 $\rho_n = 0.95$,误差矢量 $e(t) \sim CN(0, h_0(d_0/d)^\alpha I_N)$,$f_{d,n} = 70$Hz. 将系统带宽设置为 1MHz,最大传输功率 $P_{0,n} = 2$W,噪声功率 $\sigma^2 = 10^{-9}$W. 对于本地执行,假设 $\kappa = 10^{-27}$,每比特所需的 CPU 周期 $L_n = 500$,并且最大允许 CPU 周期频率 $Fm = 1.26$GHz. 本地执行所需的最大功率 $P_{1,n} = 2$W.

为了实现 DDPG 算法,对于每个用户,动作网络和评估网络都具有两个隐藏式的四层完全连接的神经网络. 两个隐藏层的神经元数量分别为 400 和 300. 神经网络使用 Relu 激活函数. 对于实现 ECOO 算法,在上文基础上设置了一个大小为 10000 的经验重播缓冲区,这样可以在查询时返回随机选择的小批量经验. 将小批量设置为 64,实现候选网络的优化. 使用自适应矩估计(Adam)方法[29],学习率分别为 0.0001 和 0.001. 目标网络的软更新速率 $\tau = 0.001$. 为了初始化网络层权重,采用[23]实验中的设置. 为了更好的找到更好的卸载决策,使用 $\theta = 0.15, \sigma = 0.12$ 的 Ornstein-Uhlenbeck 过程[30]来提供相关噪声. 经验重放缓冲区大小 $|B_n| = 2.5 * 10^5$.

在训练阶段,对于 1-5Mbps 的不同任务到达率,将使用相同的网络架构. 为了比较不同的策略性能,测试结果是 100 次训练结果的平均值.

如图 5 所示为用户动态计算卸载的训练过程. 结果是从 10 次数值模拟中的得到的平均值,其中任务到达率设置为 $\lambda = 3.0$Mbps. 可以观察到,对于 DDPG 和 ECOO 两个学习策略,迭代的平均奖励随着用户代理和 MEC 环境之间的交互的增加而增加,这表明 ECOO 算法可以在没有任何先验知识的情况下成功学习有效的计算策略. 另外,从 ECOO 学到策

略的性能总是优于不同场景的 DDPG,这表明对于连续控制问题,基于 ECOO 的策略可以比基于 DDPG 的策略更有效地探索动作空间.

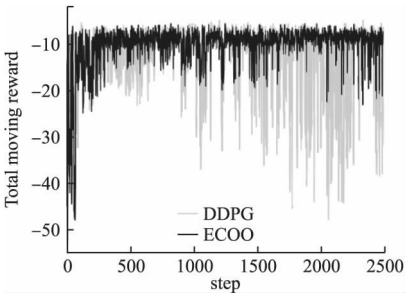

图 5　训练图

Fig. 5　Training

5.2　多用户卸载仿真结果

在 MEC 系统中,有 N = 5 个移动用户,每个移动用户随机位于距离 BS 的 100m 距离之内,任务到达率为 $\lambda_n = n * 1.0Mbps, n \in \{1,2,3,4,5\}$.

如图 6 所示,随着任务到达率的增加,平均奖励不断增加,这表明对于更大的计算需求,计算成本也会更高. 基于全部在本地执行计算任务的卸载策略(GD-Local) 可以在延迟上具有良好的效果,但是成本和能耗方面的性能一般. 这主要是因为 GD-Local 算法更喜欢将子任务卸载到本地设备以供执行. 当本地设备的资源不足时,子任务逐渐卸载到上层设备. 由于某些子任务可以在本地执行而无需网络传输,因此该算法具有较低的网络延迟. 此外,如图 6 所示,基于全部在边缘服务器执行计算任务的卸载策略(GD-offload) 与 GD-Local 算法相仿,都花费了很大的能耗. 主要原因是 GD-offload 算法

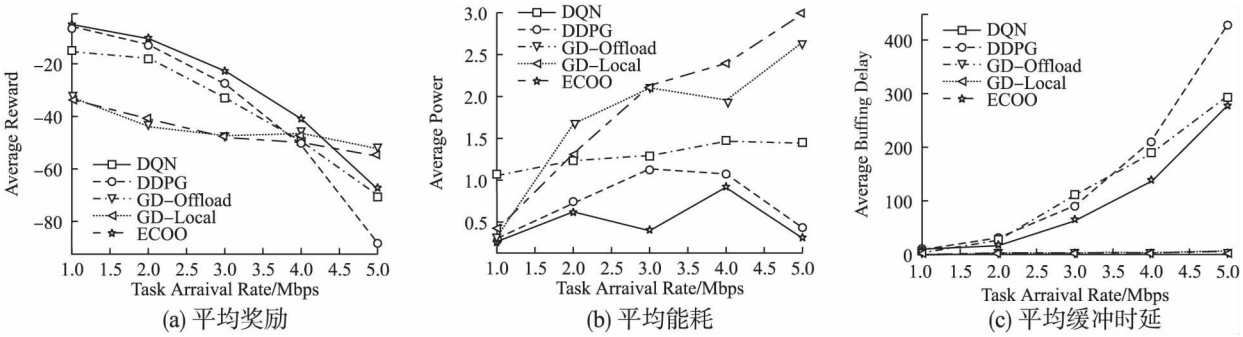

图 6　实验结果

Fig. 6　Experimental result

倾向于将子任务卸载到边缘服务器集群,这使得在传输的过程中耗费了许多能量. 同时,边缘服务器的性能可以满足更多子任务的处理需求,提高了整个集群的网络使用率.

DQN 算法,DDPG 算法和 ECOO 算法都使用深度强化学习从值中不断迭代生成相应的卸载策略. 从图 6 中结果可以看出,随着任务到达率的不断增加,ECOO 算法无论在成本,能耗还是时延上都优于前面两者. 这是因为 ECOO 算法全面考虑了目标网络的历史参数,并通过不断迭代实时的更新网络参数,用最小的奖励值替换网络,以此保持结果始终最佳,但在获得最低的能耗的同时,总会不断提高缓冲延迟.

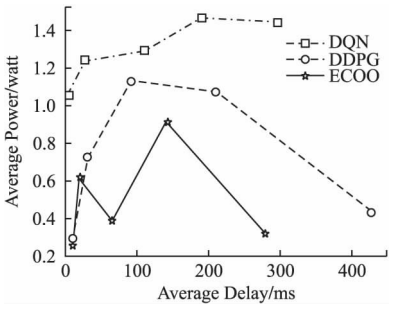

图 7　平衡能耗与时延

Fig. 7　Power-delay tradeoff

通过设置图 7 中 在 0.3-0.7 的不同值来研究功耗延迟之间平衡的训练结果. 从曲线可以推断出,平均功耗与平均缓冲延迟之间存在权衡. 具体而言,随着 w_1 的增大,功耗将通过牺牲延迟性能而降低,这表明实际上可以调整 为具有给定延迟约束的最小功耗. 还值得注意的是,对于每个值,从 ECOO 学习的策略在功耗和缓冲延迟方面始终具有更好的性能,这证明了基于 ECOO 候选网络策略的优越性.

6　结　论

在本文中,针对一个多用户 MEC 系统,这其中设置了任务随机到达,无线信道在每个用户中随时间变化的条件. 为了最小化功耗和缓冲延迟,本文设计了基于 DRL 的分散动态计算卸载算法. 并且应用 ECOO 算法成功让每个移动用户学习卸载策略,该策略能够根据从 MEC 系统本地的观察得到的结果自适应地本地执行或任务卸载. 通过模拟实验证明该策略优于传统的基于 DQN 的离散网络策略 DDPG 以及一些其他贪婪策略. 在进一步工作中,我们希望基于慢衰落参数和信道统计信息来进行资源管理,以解决因无法跟踪快速变化的无线信道而带来的问题.

References:

[1] You C, Huang K, Chae H. Energy efficient mobile cloud computing powered by wireless energy transfer [J]. IEEE Journal on Selected Areas in Communications, 2016, 34(5): 1757-1771.

[2] Narendra P M, Fukunaga K. A branch and bound algorithm for feature subset selection [J]. IEEE Transactions on Computational Imaging, 1977, C-26(9): 917-922.

[3] Bertsekas D P. Dynamic programming and optimal control [M]. A-

thena Scientific Belmont,MA,1995.

[4] Richard S Sutton,Andrew G Barto. Reinforcement learning: an introduction [M]. The MIT Press,1998: 51-85.

[5] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518 (7540): 529-529.

[6] Bi S,Zhang Y J A. Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading [J]. IEEE Transactions on Wireless Communications,2018,17 (6): 4177-4190.

[7] Tran T X,Pompili D. Joint task offloading and resource allocation for multi-server mobile-edge computing networks [J]. arXiv preprint arXiv: 1705.00704,2017.

[8] Guo S,Xiao B,Yang Y,et al. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing [C] //IEEE International Conference on Computer Communications (INFOCOM),2016: 1-9.

[9] Dinh T Q,Tang J,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.

[10] You C,Huang K,Chae H,et al. Energy-efficient resource allocation for mobile-edge computation offloading [J]. IEEE Transactions on Wireless Communications,2017,16(3): 1397-1411.

[11] Sardellitti S,Scutari G,Barbarossa S. Joint optimization of radio and computational resources for multicell mobile edge computing [J]. IEEE Trans,Signal Inf. Process,Over Netw,2015,1(2): 89-103.

[12] Mao 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.

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

[14] Wang C,Liang C,Yu F,et al. Computation offloading and resource allocation in wireless cellular networks with mobile edge computing [J]. IEEE Transactions on Wireless Communications, 2017, 16 (8): 4924-4938.

[15] Chen X,Jiao L,Li W,et al. Efficient multi-user computation offloading for mobile-edge cloud computing [J]. IEEE/ACM Transactions on Networking,2016,24(5): 2795-2808.

[16] Girshick R,Donahue J,Darrell T. Rich feature hierarchies for accurate object detection and semantic segmentation [J]. arXiv preprint arXiv: 1311.2524G,2013.

[17] Suraweera H A,Tsiftsis T A,Karagiannidis G K,et al. Effect of feedback delay on amplify-and-forward relay networks with beamforming [J]. IEEE Transactions on Vehicular Technology,2011,60 (3): 1265-1271.

[18] Kwak J,Kim Y,Lee J,et al. Dream: dynamic resource and task allocation for energy minimization in mobile cloud systems [J]. IEEE Journal on Selected Areas in Communications, 2015, 33 (12): 2510-2523.

[19] Miettinen A P,Nurminen J K. Energy efficiency of mobile clients in cloud computing [C] //2nd USENIX Workshop on Hot Topics in Cloud Computing,2010,10(1): 4-4.

[20] Burd T D,Brodersen R W. Processor design for portable systems [J]. Journal of VLSI Signal Processing Systems for Signal,Image and Video Technology,1996,13(2-3): 203-221.

[21] Jian Li,Sen Su,Xiang Cheng,et al. Costefficient coordinated scheduling for leasing cloud resources on hybrid workloads [J]. Parallel Computing,2015,44: 1-17,doi: 10.1016/j.parco.2015.02.003.

[22] Xu Z X,Cao L,Chen X L,et al. Deep reinforcement learning with sarsa and q-learning: a hybrid approach [J]. IEICE Transactions on Information and Systems,2018,E101d(9): 2315-2322.

[23] Lillicrap T P,Hunt J J,Pritzel A,et al. Continuous control with deep reinforcement learning [C] //International Conference on Learning Representations(ICLR),2016: 36-42.

[24] Shortle J F,Thompson J M,Gross D,et al. Fundamentals of queueing theory [J]. John Wiley & Sons,2018,399(2307): 41-76.

[25] Adelman D,Mersereau A J. Relaxations of weakly coupled stochastic dynamic programs [J]. Operations Research,2008,56(3): 712-727.

[26] Lepetit L,Strobel F. Bank insolvency risk and time-varying Z-score measures [M]. Journal of International Financial Markets,Institutions and Money,2013,25: 73-87.

[27] Richard S Sutton,David McAllester,Satinder Singh,et al. Policy gradient methods for reinforcement learning with function approximation [M]. Advances in Neural Information Processing Systems, 2000.

[28] Mishkin D,Sergievskiy Nikolay,Matas Jiri. Systematic evaluation of CNN advances on the ImageNet [J]. arXiv e-prints,2016,Jun arXiv: 1606.02228.

[29] Kingma D P,Ba J. Adam: a method for stochastic optimization [J]. arXiv preprint arXiv: 1412.6980,2014.

[30] Uhlenbeck G E,Ornstein L S. On the theory of the brownian motion [J]. Physical Review,1930,36(5): 823-823.