DOI:10. 20079/j. issn. 1001-893x. 20220818003

基于元强化学习的自适应卸载方法*

郑会吉[a,b],余思聪[a,b],邱鑫源[a,b],崔翛龙[a,b]

(武警工程大学 a. 信息工程学院;b. 反恐指挥信息工程联合实验室,西安 710086)

摘　要:计算卸载是移动边缘网络中的一个关键问题,基于深度学习的算法为高效生成卸载策略提供了一种解决方法。但考虑到移动终端设备的动态性以及不同任务场景之间的转换,需要大量的训练数据和较长的训练时间重新训练神经网络模型,即这些方法对新环境的适应能力较弱。针对这些不足,提出了一种基于元强化学习(Meta Reinforcement Learning,MRL)的自适应卸载方法,先对外部模型进行预训练,处理具体任务时再基于外部模型训练内部模型。该方法能快速适应具有少量梯度更新的样本的新环境。仿真实验表明,该算法能够适应新的任务场景,效果良好。

关键词:移动边缘计算(MEC);自适应卸载;元强化学习

开放科学(资源服务)标识码(OSID):

中图分类号:TN929　　**文献标志码**:A　　**文章编号**:1001-893X(2024)02-0177-07

An Adaptive Offloading Method Based on Meta Reinforcement Learning

ZHENG Huiji[a,b], YU Sicong[a,b], QIU Xinyuan[a,b], CUI Xiaolong[a,b]

(a. College of Information Engineering; b. Counter-terrorism Command Information Engineering Research team, Engineering University of PAP, Xi'an 710086, China)

Abstract: Computing offloading is a key problem in mobile edge networks. Deep learning-based algorithms provide a solution to efficiently generate offloading strategies. However, considering the dynamic characteristic of mobile terminal devices and the transformation between different task scenarios, a large amount of training data and a long training time are needed to retrain the neural network model, that is, these methods are weak in adapting to the new environment. In order to solve these problems, an adaptive offloading method based on meta reinforcement learning is proposed. Firstly, the outer model is pretrained, and then the inner model is trained based on the outer model when dealing with specific tasks. The method can quickly adapt to the new environment with a small number of gradient updated samples. Simulation results show that the algorithm can adapt to new task scenarios and has good effect.

Key words: mobile edge computing(MEC); adaptive offloading; meta reinforcement learning

0　引　言

近年来,随着新的计算和通信技术快速发展,增强现实、无人驾驶以及移动医疗等创新型移动应用和服务日益涌现。这些移动应用对计算和存储资源具有大量需求,从而在云和终端用户之间产生较大的网络流量,给传输链路造成沉重负担,影响服务传输时延。新兴的移动边缘计算技术[1]为解决这一问题提供了一种解决方法,其核心思想是将云计算

*　**收稿日期**:2022-08-18;**修回日期**:2022-09-10
基金项目:国家自然科学基金资助项目(U1603261)
通信作者:郑会吉　　Email:1344261395@qq.com

的强大能力扩展至靠近终端用户一侧,以缓解网络拥塞和降低服务时延。

计算卸载是移动边缘计算中的一项关键技术,它能将移动应用的密集型计算任务从终端用户卸载到合适的边缘服务器。一般来说,计算卸载和资源分配共同构成一个混合整数非线性规划问题,是一种 NP-hard 问题[2],现有的许多方法都是基于启发式或近似算法[3-5],但对于移动边缘网络,这些方法都十分依赖先验知识和精确的模型,当环境发生变化时,则需要对应地去更新先验知识和模型。因此,特定的启发式或近似算法很难完全适应动态的移动边缘环境。

深度强化学习(Deep Reinforcement Learning, DRL)将强化学习和深度神经网络相结合,通过试错学习来解决游戏、机器人等复杂问题。深度强化学习在各种任务卸载问题中的应用也被越来越多研究者关注,将终端用户、无线信道以及边缘服务器看作环境,通过与环境的交互学习卸载策略。此过程通常被建模为一个马尔可夫决策过程(Markov Decision Process, MDP),主要元素包括(S,A,R,γ),分别表示环境的状态、智能体执行的动作、环境反馈的奖励以及折扣因子。文献[6-12]提出的算法由于样本效率比较低,需要进行充分的二次训练以更新策略,因此比较耗时[13]。

元学习是以一种系统的、数据驱动的方式从先前经验中去学习。收集描述先前学习模型的元数据,然后从元数据中学习,以提取和传递用于指导搜索用在新任务上的最佳模型的知识。本文中,元学习通过对不同任务场景先训练一个通用元策略,明显加快新策略的学习。同时,结合强化学习,元强化学习通过借鉴历史任务,与环境的少量交互中学习新策略。基于此,本文提出一种基于元强化学习的自适应卸载模型,将卸载过程建模为一个马尔可夫决策过程(Markov Decision Process, MDP)。该模型包含两个子模型,一个外部模型利用历史任务数据训练得到元策略;基于元策略,内部模型通过少量梯度更新快速学习新策略,适应新环境[14]。

1 系统模型

图 1 所示为本文的移动边缘场景,由边缘服务器和 N 个移动终端组成,表示为 $\mathcal{N}=\{1,2,\cdots,N\}$。每个移动终端在时间 t 时刻需要处理计算任务,不同任务权重值不同。计算任务遵循二值卸载策略,即移动终端要么在本地执行,要么将任务卸载到边缘服务器执行。假设边缘服务器的计算能力远大于移动终端,定义 t 时刻的计算策略为 $x_t=\{x_n(t)\in\{0,1\}\mid n\in\mathcal{N}\}$,$x_n(t)=0$ 表示本地计算,反之表示卸载计算。

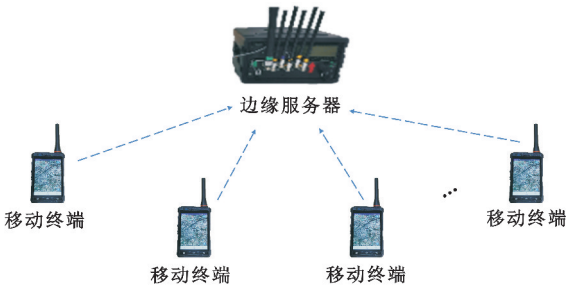


图 1 移动边缘场景

Fig. 1 Mobile edge scenario

1.1 时延模型

主要研究移动边缘网络中考虑时变无线信道因素的时延优化问题。计算任务表示为一个列表 $\text{task}_n=[i_n,o_n,\zeta_n]$,分别表示数据大小、回传结果大小以及完成任务所需的 CPU 周期数。移动终端 n 和边缘服务器之间传输速率为

$$r_n(t)=B_n\text{lb}\left(1+\frac{P_n\cdot h_n(t)}{\omega_0}\right) \tag{1}$$

式中:B_n 是传输信道带宽;P_n 是传输功率;ω_0 是环境噪声功率;$h_n(t)\in h_t$ 是对应的信道增益。则移动终端 n 的总通信时延为

$$t^o_{\text{com}}=\frac{i_n+o_n}{r_n(t)} \tag{2}$$

边缘服务器和移动终端的 CPU 周期数分别为 f_e 和 f_0,根据前面的假设,满足 $f_0\ll f_e$。因此,边缘服务器的处理时延为

$$t^o_{\text{exe}}=\frac{\zeta_n}{f_e} \tag{3}$$

同理,移动终端本地执行的时延为

$$t^l_{\text{exe}}=\frac{\zeta_n}{f_0} \tag{4}$$

则移动终端 n 的时延为

$$T_n(t)=(t^o_{\text{com}}+t^o_{\text{exe}})x_n(t)+t^l_{\text{exe}}[1-x_n(t)] \tag{5}$$

移动边缘网络的加权时延和为

$$Q(x_t,f_t,h_t)=\sum_{n\in\mathcal{N}}T_n(t)\cdot w_n(t) \tag{6}$$

式中:$w_n(t)$ 表示移动终端 n 的权重。

1.2　问题描述

本文以最小化加权时延和 $Q(x_t,f_t,h_t)$ 为目标,通过求解策略函数 π,得到相应的卸载策略 x_t^*,表示为

$$\pi:h_t\mapsto x_t^* \tag{7}$$

2　基于元强化学习的自适应卸载算法

MRL 利用移动终端和边缘服务器的计算资源进行训练。训练包括两个模型:一个是针对具体任务的内部模型;另一个是针对元策略的外部模型。内部模型在移动终端训练(内部模型往往只需较少训练步骤和训练数据,因此假设移动终端能支持训练);外部模型在边缘服务器上训练。

MRL 的详细训练过程如图 2 所示,包括 4 个步骤:第一步,移动终端从边缘服务器下载元策略;第二步,移动终端基于元策略和本地数据训练内部模型,以获得特定任务策略;第三步,移动终端将特定任务策略的参数上传到边缘服务器;第四步,边缘服务器根据接收到的参数训练外部模型,生成新的元策略,并开始新一轮训练。

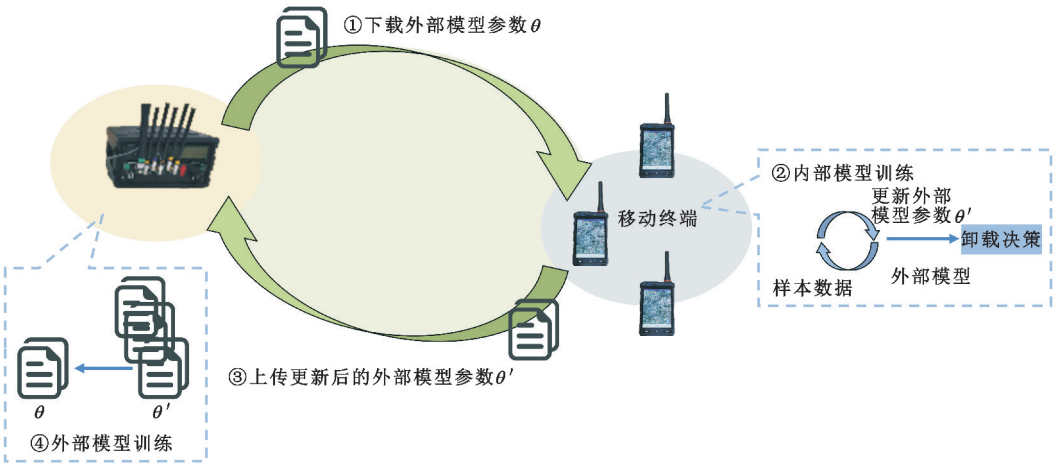


图 2　MRL 训练过程

Fig. 2 MRL training process

2.1　MDP 建模

将计算卸载过程建模为一个 MDP 过程,其基本要素包括:

1)状态空间:环境的状态是 t 时刻信道增益,则状态空间表示为 state = $\{h_t\}$;

2)动作空间:由于采用二值卸载模式,因此动作空间表示为 action = $\{0,1\}$;

3)奖励函数:若动作为最优解的动作值,奖励为最小优化函数值的相反数,反之为最大优化函数值的相反数。

MRL 采用 AC(Actor-Critic)算法,其中包括 actor 和 critic 两个模块。actor 模块会根据输入产生卸载动作,而 critic 则会对产生的动作进行评价打分,最后输出打分最高的卸载动作。

2.2　内部模型

训练集包含 Ω 个不同任务场景 $I=\{\psi_i\mid i=1,2,\cdots,\Omega\}$,一个任务场景包含 K 个样本数据,表示为 $\psi=\{(h_k,x_k)\mid k\in K\}$。采用强化学习方法对元策略进行训练,算法具体如下:

输入:样本数据(h_i,x_i)

输出:最优卸载动作 x_i^*

1 从边缘服务器下载元策略,初始化参数 $\theta_i=\theta$

2 设定迭代数 M

3 for $i\in\{1,2,\cdots,M\}$ do:

4　从 I 中采样样本数据 I_b

5　for $\psi_i\in I_b$ do:

6　　从 ψ_i 中采样样本数据 D_i

7　　输入无线信道增益 h_i

8　　利用保序量化算法产生 ϕ 个卸载动作$\{x_i\}$

9　　计算 $Q(x_i,h_i,w_i)$,并选出最优动作 $x_i^*=\underset{\{x_i\}}{\operatorname{argmax}}\,Q(x_i,h_i,w_i)$

10　　根据公式$\sum_{D_i}\|f_{\theta_i}(h_k)-x_k\|_2^2$计算 $L(f_{\theta_i})$

11　　根据公式 $\theta'_i=\theta_i-\alpha\nabla_{\theta_i}L(f_{\theta_i})$更新内部模型

12　　从 ψ_i 中重新采样样本数据 D'_i

13　　计算 $L(f_{\theta'_i})$

14　end

15　　更新外部模型:$\theta\leftarrow\theta-\beta\nabla_\theta\sum_{i\in I_b}L(f_{\theta'_i})$

16 end

神经网络输入信道增益 h_i，actor 模块采用一种保序量化方法[15]产生卸载动作，而后由 critic 模块进行评价根据 $Q(\cdot)$ 值选择最优动作 x_i^*，即神经网络的输出，如图 3 所示。对某个具体的任务场景 ψ_i，按照图 2 过程，移动终端先从边缘服务器下载元策略参数，其结构和参数与外部模型相同，即 $\theta_i=\theta$，然后从 ψ_i 中采样样本数据 D_i，通过最小化神经网络的均方误差损失训练内部模型：

$$L(f_{\theta_i})=\sum_{D_i}\|f_{\theta_i}(h_k)-x_k\|_2^2 \tag{8}$$

式中：f_{θ_i} 是内部模型的函数。最后，通过梯度下降的方法对模型的参数进行更新，即

$$\theta'_i=\theta_i-\alpha\nabla_{\theta_i}L(f_{\theta_i}) \tag{9}$$

式中：α 是一种超参数。

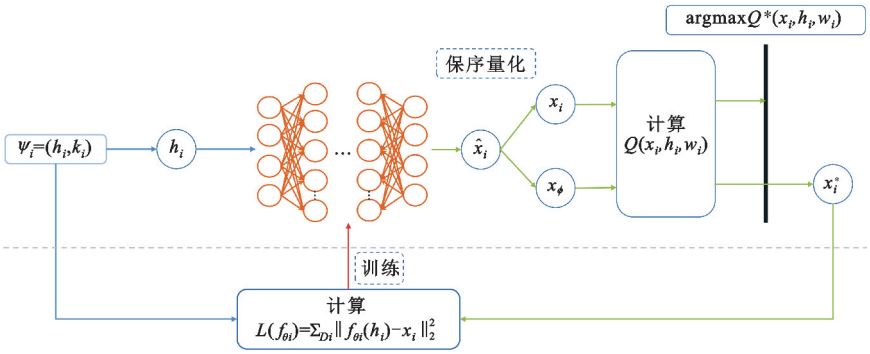


图 3 内部模型

Fig. 3 Internal model

2.3 外部模型

如图 4 所示，基于随机采样的任务场景 $I_b=\{\psi_i\,|\,i\in\Omega\}$，为训练外部模型，移动终端按照之前的方式训练内部模型后，重新从 ψ_i 中采样样本数据 D'_i 并根据公式计算损失值 $L(f_{\theta'_i})$。

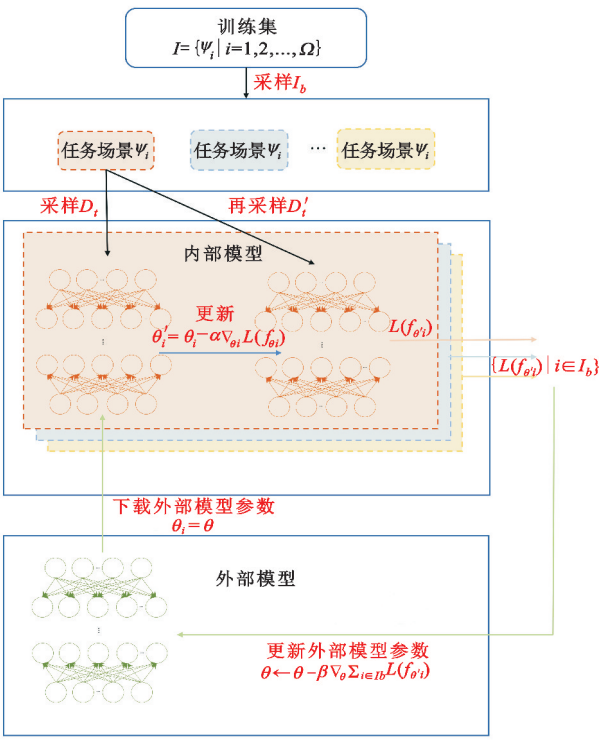


图 4 外部模型

Fig. 4 External model

当迭代完所有采样的任务场景后，合并所有损失函数并计算外部模型的梯度 $\nabla_\theta\sum_{i\in I_b}L(f_{\theta'_i})$。最后，按下面公式更新外部模型：

$$\theta\leftarrow\theta-\beta\nabla_\theta\sum_{i\in I_b}L(f_{\theta'_i}) \tag{10}$$

式中：β 是步长。基于更新的外部模型，将采样下一批任务场景训练直至收敛。

3 仿真实验

3.1 参数设置

本文通过 Python 编程语言进行实验仿真，实验运行在 Intel Core i7-9700H 3.6 GHz CPU，内存 8 GB 的服务器上，虚拟环境采用框架 Tensorflow-gpu 2.3。假设所有移动终端随机分布在指定区域，服从概率为 3×10^{-4} 的泊松分布，其计算所需的计算周期与输入大小有关，表示为 γ_n = 165 cycle[16]与边缘服务器之间的信道功率增益服从路径损耗模型 $H[\text{dB}]=103.8+20.9\lg d_{[\text{km}]}$，$d$ 表示移动终端与服务器之间的距离。部分实验参数如表 1 所示。

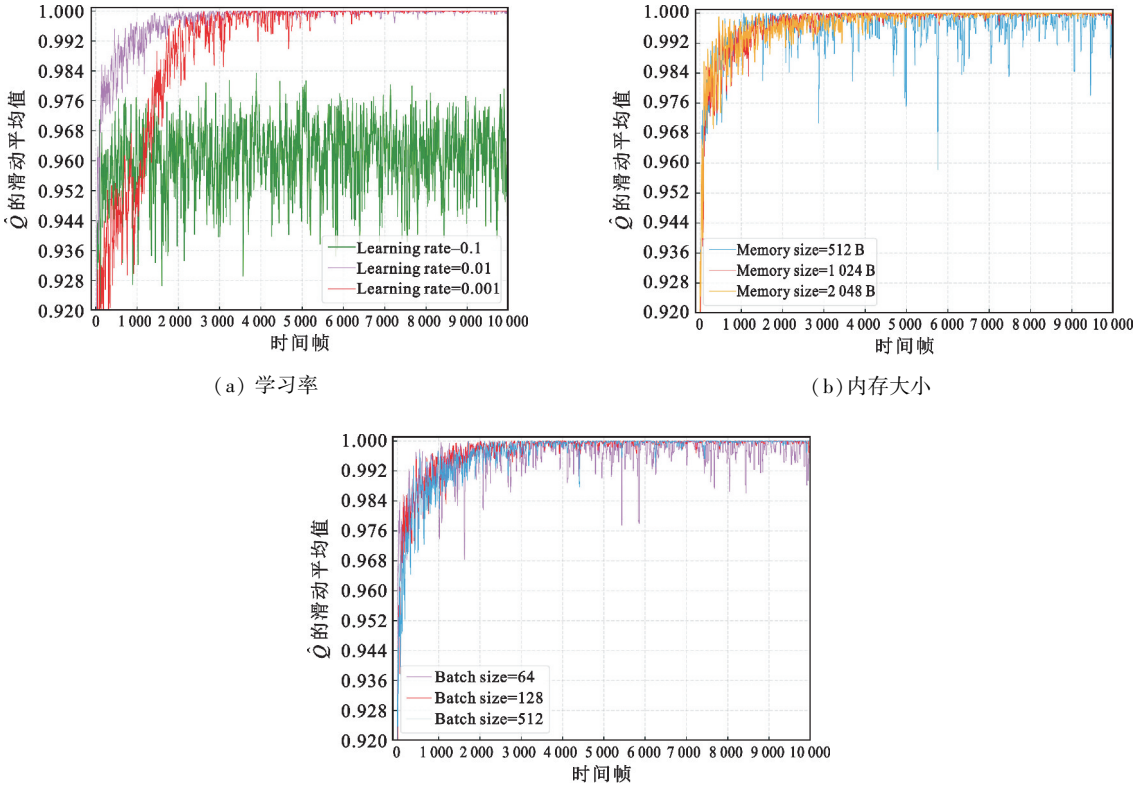
表 1　实验参数

Tab. 1 Experimental parameters

参数	值
f_0/(cycle/s)	2.84×10^9
f_e/(cycle/s)	1×10^{11}
B_n/MHz	10
P_n/dBm	23
ω_0/dBm	−95

3.2　参数分析

图 5 给出了 MRL 在不同参数下的收敛性能,包括学习率、内存大小和批量大小。图 5(a)为学习率在 Adam 优化器中的影响,从中容易得出,过大的学习率(0.1)会导致算法难以收敛,过小的学习率(0.001)收敛速度较慢,所以,在仿真实验中,将学习率设置为 0.01。在图 5(b)中,较小的内存(512 B)导致在收敛上较大的波动,但较大的内存(2 048 B)需要更多的训练数据收敛至最优,因此将内存大小设置为 1 024 B。如图 5(c)所示,较小的批量(64)并没有充分利用保存在内存中的训练数据,但较大的批量(512)则会频繁采样旧的数据从而降低收敛性能,因此,将批量大小设置为 128。



(a) 学习率　　(b)内存大小

(c)批量大小

图 5　MRL 在不同参数下的收敛性能

Fig. 5 Convergence performance of MRL under different parameters

3.3　性能分析

在图 6 中,横坐标为微调步数,纵坐标为归一化计算速率(它是枚举的最优卸载策略和评估策略之间的比值)。为更好说明 MRL 的性能,将一般强化学习算法 RL[13]作为对比,可以看到,在 MRL 中,内部模型的参数是从预训练的外部模型复制而来,它能在 20 步微调内适应新的任务场景,并使归一化计算速率达到 0.99 以上;相反,一般的强化学习算法则需要更多的步数来收敛,这说明了 MRL 能更快更高效地适应新任务场景。

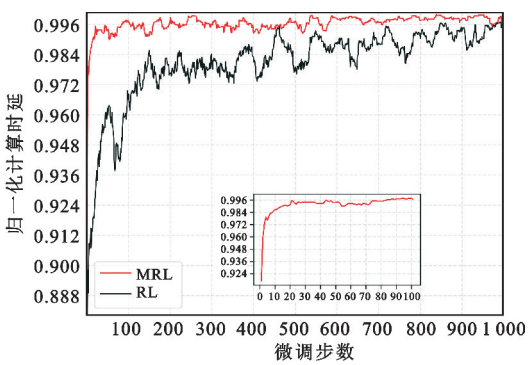


图 6　MRL 算法的性能

Fig. 6 Performance of MRL algorithm

图 7 给出了 3 个不同任务场景 ψ_1,ψ_2 和 ψ_3 下的 MRL。一旦任务场景变换,例如从场景 ψ_1 变到 ψ_2,借助外部模型快速训练一个新的内部模型。如图 7 所示,MRL 快速适应新的场景,并在 100 步微调都能达到归一化计算速率超过 0.99。同时,如图 8 所示,对比不同算法下 3 个场景的平均时延可以看出,MRL 相比一般的 RL、贪婪算法以及深度神经网络[17](Deep Neural Network,DNN)具有最优的性能。

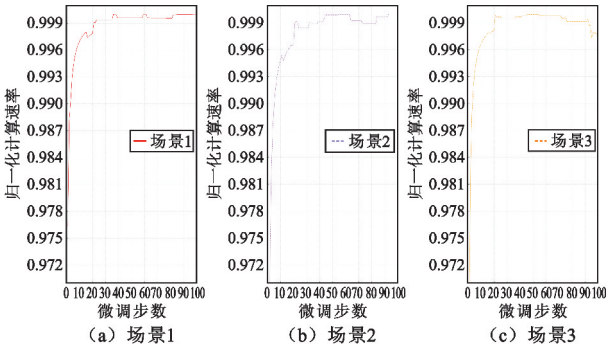


图 7 动态场景下的 MRL

Fig. 7 MRL in dynamic scenarios

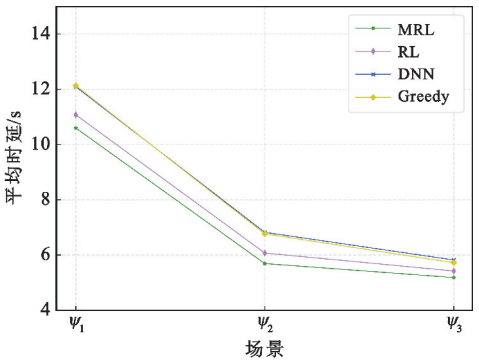


图 8 平均时延对比

Fig. 8 Comparison of average delay

表 2 从平均 $\hat{Q}$ 对比了不同算法性能,包括贪婪算法、DNN 以及不同微调步数的 MRL。得益于预训练,MRL 的性能优于另外两种算法,而且在仅仅 10 步微调后,归一化计算时延能达到 0.99 以上。

表 2 不同算法性能对比

Tab. 2 Performance comparison of different algorithms

算法		平均 $\hat{Q}$
Greedy		0.915 9
DNN		0.978 6
MRL	10 步微调	0.982 9
	20 步微调	0.992 3
	50 步微调	0.994 2
	100 步微调	0.995 7

4 结 论

本文针对以往深度学习算法的不足,提出了一种基于元强化学习的自适应卸载方法,它能适应动态 MEC 任务场景。移动终端基于元策略和本地数据训练内部模型,以获得特定任务策略,用更少的训练数据可以快速训练一个内部模型以适应新的任务。仿真实验表明,MRL 在少量的微调步数内能达到 0.99 以上的准确率。因此,该算法在未来移动边缘网络中的快速部署是可能的。

参考文献:

[1] 张依琳,梁玉珠,尹沐君,等. 移动边缘计算中计算卸载方案研究综述[J]. 计算机学报,2021,44(12):2406-2430.

[2] 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.

[3] LIN X, WANG Y, XIE Q, et al. Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment [J]. IEEE Transactions on Services Computing, 2014, 8 (2):175-186.

[4] 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.

[5] WU H, KNOTTENBELT W J, WOLTER K. An efficient application partitioning algorithm in mobile environments [J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 30(7):1464-1480.

[6] LIU N, LI Z, XU J, et al. A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning[C]//Proceedings of 2017 IEEE 37th International Conference on Distributed Computing Systems. Atlanta: IEEE, 2017:372-382.

[7] CHEN X, ZHANG H, WU C, et al. Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning[J]. IEEE Internet of Things Journal, 2018, 6(3):4005-4018.

[8] WANG J, HU J, MIN G, et al. Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning [J]. IEEE Communications Magazine, 2019, 57(5):64-69.

[9] LI J, GAN H, LV T, et al. Deep reinforcement learning based computation offloading and resource allocation for MEC [C]//Proceedings of 2018 IEEE Wireless Communications and Networking Conference. Barcelona:

IEEE,2018:1-6.

[10] LU H, GU C, LUO F, et al. Optimization of task offloading strategy for mobile edge computing based on multi-agent deep reinforcement learning [J]. IEEE Access,2020,8:202573-202584.

[11] ZHAN W, LUO C, WANG J, et al. Deep reinforcement learning-based computation offloading in vehicular edge computing [C]//Proceedings of 2019 IEEE Global Communications Conference. Hawaii:IEEE,2019:1-6.

[12] ZHAN W, LUO C, WANG J, et al. Deep reinforcement learning based offloading scheduling for vehicular edge computing[J]. IEEE Internet of Things Journal,2020,7(6):5449-5465.

[13] 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,2019,19(11):2581-2593.

[14] BOTVINICK M, RITTER S, WANG J X, et al. Reinforcement learning, fast and slow [J]. Trends in Cognitive Sciences,2019,23(5):408-422.

[15] 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,2019,19(11):2581-2593.

[16] MIETTINEN A P,NURMINEN J K. Energy efficiency of mobile clients in cloud computing[C]//Proceedings of 2010 IEEE International Conference on Cloud Computing Technology and Science. Boston:IEEE,2010:1-7.

[17] HUANG L,FENG X,FENG A Q,et al. Distributed deep learning based offloading for mobile edge computing networks[J]. Mobile Networks and Applications,2022,27(3):1123-1130.

作者简介:

郑会吉　男,1997 年生于重庆,2020 年获学士学位,现为硕士研究生,主要研究方向为边缘计算。

余思聪　男,1999 年生于湖南岳阳,2020 年获学士学位,现为硕士研究生,主要研究方向为边缘计算。

邱鑫源　女,1999 年生于江西南昌,2020 年获学士学位,现为硕士研究生,主要研究方向为联邦学习。

崔翛龙　男,1973 年生于安徽长丰,1995 年获学士学位,现为教授,主要研究方向为作战数据与人工智能。