计算机科学
COMPUTER SCIENCE

基于深度强化学习和无线充电技术的D2D-MEC网络边缘卸载框架

张乃心, 陈霄睿, 李安, 杨乐瑶, 吴华明

引用本文

张乃心, 陈霄睿, 李安, 杨乐瑶, 吴华明基于深度强化学习和无线充电技术的D2D-MEC网络边缘卸载框架[J]. 计算机科学, 2023, 50(8): 233-242.

ZHANG Naixin, CHEN Xiaorui, LI An, YANG Leyao, WU Huaming. Edge Offloading Framework for D2D-MEC Networks Based on Deep Reinforcement Learningand Wireless Charging Technology [J]. Computer Science, 2023, 50(8): 233-242.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220900181

基于深度强化学习和无线充电技术的D2D-MEC网络边缘卸载框架

张乃心[1]　陈霄睿[1]　李　安[1]　杨乐瑶[1]　吴华明[2]
1 天津大学数学学院　天津 300192
2 天津大学应用数学中心　天津 300072
(15598555618@163.com)

摘　要　物联网设备中大量未被充分利用的计算资源,正是移动边缘计算所需要的。一种基于设备对设备通信技术和无线充电技术的边缘卸载框架,可以最大化利用闲置物联网设备的计算资源,提升用户体验。在此基础上,可以建立物联网设备的D2D-MEC网络模型。在该模型中,主设备根据当前环境信息和估计的设备状态信息,选择向多个边缘设备卸载不同数量的任务,并应用无线充电技术提升传输的成功率和计算的稳定性。运用强化学习方法解决任务分配和资源分配的联合优化问题,也就是最小化计算延迟、能量消耗和任务丢弃损失,最大化边缘设备利用率和任务卸载比例的优化问题。除此之外,为了适应状态空间更大的情况,提高学习速度,提出了一种基于深度强化学习的卸载方案。基于以上理论和模型,使用数学推导计算出了D2D-MEC系统的最优解及性能上限。仿真实验证明了D2D-MEC卸载模型及其卸载策略的综合性能更好,更能充分利用物联网设备的计算资源。

关键词:移动边缘计算;D2D;强化学习;物联网;计算卸载;无线能量传输
中图法分类号　TP181

Edge Offloading Framework for D2D-MEC Networks Based on Deep Reinforcement Learning and Wireless Charging Technology

ZHANG Naixin[1],CHEN Xiaorui[1],LI An[1],YANG Leyao[1] and WU Huaming[2]
1 School of Mathematics,Tianjin University,Tianjin 300192,China
2 Center for Applied Mathematics,Tianjin University,Tianjin 300072,China

Abstract　A large amount of underutilized computing resources in IoT devices is what mobile edge computing requires. An edge offloading framework based on device-to-device communication technology and wireless charging technology can maximize the utilization of computing resources of idle IoT devices and improve user experience. The D2D-MEC network model of IoT devices can be established on this basis. In this model,the device chooses to offload multiple tasks to multiple edge devices according to the current environment information and the estimated device state. It applies wireless charging technology to increase the success rate of transmission and computation stability. The reinforcement learning method is used to solve the joint optimization allocation problem,which aims to minimize the computation delay,energy consumption,and task dropping loss as well as maximize the utilization of edge devices and the proportion of task offloading. In addition,to adapt to larger state space and improve learning speed,an offloading scheme based on deep reinforcement learning is proposed. Based on the above theory and model,the optimal solution and upper limit of performance of the D2D-MEC system are calculated by mathematical derivation. Simulation results show that the D2D-MEC offloading model and its offloading strategy have better all-around performance and can make full use of the computing resources of IoT devices.

Keywords　Mobile edge computing,Device-to-device(D2D),Reinforcement learning,Internet of things(IoT),Computation offloading,Wireless energy transmission

1　引言

随着第五代移动通信技术(5G)和人工智能技术的飞速发展,大量新颖的应用程序和网络服务不断涌现,改变了我们的日常生活。但是由于智能移动设备的计算资源及计算能力有限,其往往难以处理计算密集型和时间敏感型任务[1]。为了解决这样的问题,有学者基于计算卸载技术提出了移动云计算(Mobile Cloud Computing,MCC)的计算模型,设备用户

到稿日期:2022-09-19　返修日期:2023-02-06
基金项目:国家自然科学基金(62071327)
This work was supported by the National Natural Science Foundation of China(62071327).
通信作者:吴华明(whming@tju.edu.cn)

可以将任务传输到具有强大计算能力的远端云服务器进行计算，从而突破计算能力限制和存储限制，延长设备电池寿命。但是，移动云计算同样面临着时延风险未知和传输距离远等问题。

相比之下，移动边缘计算系统(Mobile Edge Computing，MEC)在网络节点(即基站)部署 MEC 服务器，使得移动设备能够将难以处理的任务卸载到 MEC 服务器上，从而节省时间和能源。相比移动云计算，移动边缘计算能够更快、更高效地为移动终端提供计算服务，同时缓解核心网络的压力。

移动边缘计算尽管可以提升计算效果，但仍面临着资源受限和卸载选择单一等问题。随着物联网设备间通信技术的日益成熟，设备对设备的移动边缘计算模型(D2D-MEC)应运而生。这种模型允许移动设备将计算任务卸载到其他移动设备或 MEC 服务器，拥有本地处理、空闲用户设备(User Equipment，UE)辅助处理和 MEC 辅助处理这 3 种模式。相比之下，D2D-MEC 系统丰富了用户选择，优化了计算决策，进一步降低了系统的延迟和能源的消耗。

但是 D2D-MEC 模型还面临着许多挑战：

(1)续航能力受限。许多物联网设备属于移动充电设备，其电池大小受到硬件成本和尺寸的约束，电量有限，随时面临着计算能力下降乃至计算程序终止的风险。同时，处于移动状态中的设备无法长期充电。因此，主设备的续航能力成为了限制卸载效果的重要因素。

无线电能传输技术的出现，为解决移动状态设备充电问题提供了思路。磁耦合谐振机构和双向 DC-DC 核心拓扑电路可以实现设备间双向无线充电技术，突破固定充电桩传统拔插式充电方式的限制，最大程度实现现有移动设备电量的最优分配，并提高电能利用率。

(2)卸载策略亟待优化。动态环境中，设备的位置、信道衰弱与干扰及计算资源余量等状态时刻变化，导致主设备任务分配的决策问题也进一步复杂化。传统的优化方法并不能解决复杂条件中边缘设备的卸载决策问题。

因此，一些研究考虑运用深度强化学习的方法寻找更优卸载策略。相对于传统优化方法，强化学习有 3 方面优点：1)强化学习模型能更快适应复杂的状态空间；2)智能体无需了解历史信息；3)强化学习不依赖对环境的精确建模，无需存储计算大量的模型信息。以上 3 点加快了模型获取最优决策的速度。

(3)空闲设备可被利用。随着物联网技术的发展，越来越多的电子设备智能化，实际环境中将会有更多的空闲智能设备，若能开发利用其中丰富的计算资源，整个系统的计算潜能将会大大提高。多数研究中未提及这一点，而本文将着力于提高空闲设备的利用率。

针对以上挑战，本文研究基于无线电能传输技术的 D2D-MEC 移动边缘卸载模型。本文的主要工作如下：

(1)建立 D2D-MEC 传输模型，兼顾计算性能和设备利用率提出了评估系统性能的优化公式，基于此公式建立优化问题。为求解该优化问题，基于深度强化学习方法提出了卸载方案。

(2)通过数学理论推导，证明了在动态环境中 D2D-MEC 卸载模型最优解存在。

(3)进行实验仿真，实验结果显示，与基础方法的卸载策略相比，D2D-MEC 卸载策略既保证了较低的延迟和能耗，又提高了设备的利用率。

2 相关研究

在有关 D2D-MEC 的研究中，Ko 等[2]提出了一种分布式 D2D 卸载系统，任务所有者可以广播卸载请求及完成要求，附近的移动设备以分布式方式决定是否接受卸载。Li 等[3]针对具有能量收集(Energy Harvesting，EH)功能的 MEC 系统从环境中收集能量不稳定，以及通信延迟和无线信道衰弱与干扰问题，建立 D2D 辅助 MEC 计算分流模型，并提出了一种基于 Lyapunov 优化的动态计算分流(Lyapunov Optimization-Based Dynamic Computation Offloading，LODCO)低复杂度在线算法[4]以解决成本最小化问题。Pu 等[5]提出了新型的 D2D Fogging 算法，利用 Lyapunov 在线任务卸载算法构建出高效的任务调动策略，在考虑到防止过度开发及搭便车行为对用户合作的激励损失约束下，最小化了平均时间能源消耗。D2D-MEC 系统的结构较为多样，本文采用一种全新的 MEC 辅助 D2D 计算分流模型，意为优先向附近的移动设备进行卸载，由附近的基站辅助完成其余的卸载任务，这样可以充分利用空闲设备的计算资源。Fang 等[6]研究了基于 D2D 辅助的 MEC 网络下的任务卸载问题，利用博弈论，通过优化各个用户的卸载决策，最终实现全网用户累积时延的最小化。

在关于深度学习与强化学习在 MEC 系统中的应用的研究中，Qiao 等[7]将预期的系统运行成本和所需的任务执行时间建模为受约束的马尔可夫决策过程，利用强化学习框架来获得最佳决策。Li 等[8]将 MDP 问题分解为显式成本最小化问题和长期成本最小化问题，采用基于神经网络逼近的强化学习算法进行求解。Huang 等[9]提出了一种基于深度强化学习的方法来解决 MEC 的任务分配和资源分配问题，仿真结果表明所提出的基于深度 Q 学习的算法可以达到最优的性能。Liang 等[10]将类似的模型应用于一个飞行移动边缘计算平台，提出了深度强化学习的轨迹控制算法，并应用了体验重放技术。Min 等[11]根据设备当前电量和到每个 MEC 服务器的无线电传输原速率及预测速率，选择边缘卸载设备。仿真结果表明，与原卸载方案相比，所提出的 RL 卸载方案降低了能耗、计算延迟和任务丢失率。针对无线衰落环境中的 MEC 系统，Liang 等[12]建立了一个遵循二进制任务卸载策略，基于深度强化学习在线卸载(Deep Reinforcement Online Offloading，DROO)框架，解决了计算复杂度高及需要大量迭代的问题。Kang 等[13]基于 DNN 算法神经网络层的粒度在移动设备和数据中心之间的自动化分计算，优化了云处理的移动能耗与延迟，数据吞吐量得到提高。Wang 等[14]针对任务调度分配问题，提出了一种基于 DRL 的任务卸载算法对任务进行合理的划分并制定相应的任务卸载策略。

除此之外，还有其他的计算卸载方法。比如，Liang 等[15]考虑到动态 MEC 设备卸载任务时的大规模与长时间数据训练，提出了基于元学习的计算卸载(Meta-learning-based Computation Offloading，MELO)算法，从历史 MEC 任务场景中

进行微调，得到适应新任务场景的 MEC。Deng 等[16]建立了基于 MEC 网络的多目标任务卸载问题，考虑了 MD 的移动性和任务间的顺序依赖关系；然后，分析应用和 MD 相关信息，设计 MDP 模型，并提出了基于 DQN 的多目标任务卸载算法 MTOA-DQN 来同时优化所关注的两个目标。针对移动用户的不同请求在云雾资源节点上的分配问题，Liu 等[17]提出了基于云雾协作的任务分配算法；Cheng 等[18]设计了一种基于 MEC 的车联网模型架构，构建了卸载决策机制，并在此基础上提出了 KMGA 算法对多目标问题进行优化。

在性能估计方面，Dinh 等[19]考虑了固定和弹性 CPU 频率两种情况，分别使用线性、半定松弛的方法和基于穷举搜索与 SDR 的方法，对从单个移动设备卸载到多个边缘移动设备的框架进行优化，证明了移动设备的 CPU 范围会对任务分配产生影响。

在有关双向无线功率传输（Bidirectional Wireless Power Transfer，BWPT）的研究中，Zhao 等[20]提出了一种用于 BWPT 系统的数字相位同步法，在没有物理接触的情况下共享相位信息。实验结果表明，该系统可以在 2ms 内快速切换功率传输方向，效率高达 96.84%，且不易受参数扰动影响，为移动状态下设备与设备间充电提供参数支持。本文中的实验参数以此为基础。

基于上述研究工作可以发现，在一般的 MEC 系统中有大量使用深度强化学习模型的研究，而在引入 D2D 辅助的 MEC 系统的研究中则较少使用到深度强化学习方法，并且多数的 D2D-MEC 模型并没有考虑到设备耗电与充电的问题。此外，关于 D2D-MEC 在动态环境下的卸载选择与卸载能力上限的理论研究也很缺乏。因此，本文先通过数学推导对不同环境状态下的运行状态和性能进行分析，再尝试将强化学习中的 Q 学习与 DQN 算法应用于 D2D-MEC 卸载决策中，并引入无线充电模块，对不同卸载框架下的卸载性能进行全面评估。

3 系统模型和优化问题

本文研究包含多个 IoT 设备和一个基站的 D2D-MEC 网络，如图 1 所示。

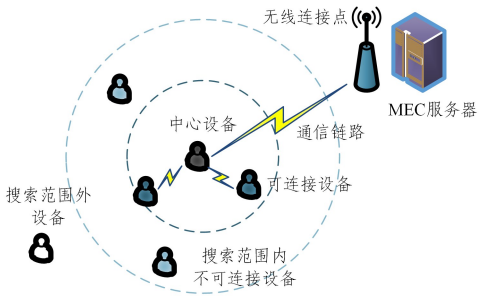


图 1 D2D-MEC 网络模型

Fig. 1 Connection network model of Device-to-Device and MEC server

图 1 中模型包含了一个无线网络覆盖区域内的所有移动设备（包括主设备与边缘设备），以及在此区域之外的一个基站。该模型可以模拟表示当前一些常见物联网场景，例如一间正在进行清扫的房屋，扫地机器人为主设备，其他家电作为边缘设备，局域网外稍远的屋顶处有一基站。将有无线充电需求和协助计算需求的设备称为主设备 I；其余设备称为边缘设备，记作 $I_i(i\in\{1,2,\cdots,M\})$。

如图 1 所示，主设备 I 可以与其他设备建立无线连接，也可与基站通过蜂窝网络无线连接，这两种连接可以同时存在。但是，仅当处于空闲状态的边缘设备 I_i 电量充足（大于等于 80%），且与主设备 I 之间的距离小于 d 时，主设备与 I 才可与之连接；而 I 在任何条件下都可以与基站连接。

将连续时间划分为相同长度的小时间段，并将其称为时隙，记作 $t^{(k)}(k\in\{1,2,\cdots\})$。在时隙 $t^{(k)}$ 内，主设备 I 产生的数据量为 $C^{(k)}$（字节）的计算任务。本文主要研究延迟敏感型任务，因此假设每个时隙产生的任务必须在该时隙处理完后，否则任务失败。主设备 I 会将任务分成相互独立的 n 份，每份任务的数据量为 $C^{(k)}/n$。对于每一份任务，主设备 I 都有 3 种选择：1）在本地计算它；2）将其卸载至邻近的某个移动设备 I_i 进行计算；3）将其卸载至基站进行处理。

时隙 $t^{(k)}$ 内 I 与 I_i 的连接状态用 $[B_i^{(k)},c_i^{(k)}]$ 表示，其中，$B_i^{(k)}$ 是 I 和 I_i 之间的上行无线传输速率，$c_i^{(k)}$ 为 I 与 I_i 产生连续连接的时长，主设备 I 可根据 $c_i^{(k)}$ 预测下一时隙的连接状态。$c_i^{(k)}$ 可表示为：

$$c_i^{(k)}=\begin{cases}c_i'^{(k)}+1, & c_i^{(k)}=1\\ 0, & c_i^{(k)}=0\end{cases}\tag{1}$$

其中，$c'^{(k)}$ 表示当前时刻 I 与 I_i 间是否产生连接。

$$c_i'^{(k)}=\begin{cases}1, & \text{时刻 } I \text{ 和 } I_i \text{ 间有连接}\\ 0, & \text{其他}\end{cases}\tag{2}$$

主设备 I 在时隙 $t^{(k)}$ 的卸载选择用 $\boldsymbol{a}^{(k)}=[x_1^{(k)},x_2^{(k)},\cdots,x_M^{(k)},y^{(k)}]$ 表示，其中 $x_1^{(k)},x_2^{(k)},\cdots,x_M^{(k)}$ 分别表示 I 向边缘设备 I_i 卸载的任务比例，$y^{(k)}$ 表示 I 向基站卸载的任务比例，满足 $x_1^{(k)}+\cdots+x_M^{(k)}+y^{(k)}\leqslant 1$。

3.1 本地计算

主设备 I 在 $t^{(k)}$ 需要处理的任务量为 $(1-\sum_1^M x^{(k)}-y^{(k)})C^{(k)}$ 字节。N 为 I 处理每个字节任务所需要的 CPU 周期数，若 f_m 为周期 m 对应的 CPU 时钟频率，则本地计算时间为：

$$T_l^{(k)}=\sum_{m=1}^{(1-\sum_1^M x^{(k)}-y^{(k)})C^{(k)}N}\frac{1}{f_m}\tag{3}$$

本地计算耗能为：

$$E_l^{(k)}=\sum_{m=1}^{(1-\sum_1^M x^{(k)}-y^{(k)})C^{(k)}N}\zeta(f_m)^2\tag{4}$$

其中，ζ 是芯片的有效电容系数，它取决于芯片结构。

3.2 边缘计算

卸载计算的时延由 3 部分构成：主设备向基站或边缘设备传输任务的时间、基站或边缘设备处理任务的时间，以及基站或边缘设备将结果传回主设备的时间。三者中，将结果回传的时间很短，可以忽略不计。若主设备将任务卸载到基站，基站的计算速度很快，其处理任务的时间也可以不计。设 $B^{(k)}$ 是时隙 $t^{(k)}$ 内主设备 I 与基站之间的上行无线传输速率，则主设备向基站传输任务的时间 $T_e^{(k)}$ 为：

$$T_e^{(k)}=\frac{y^{(k)}C^{(k)}}{B^{(k)}}\tag{5}$$

因此，时隙 $t^{(k)}$ 内任务处理过程的时延 $T^{(k)}$ 为：

$$T^{(k)}=\max\{T_l^{(k)},T_{d_i}^{(k)},T_e^{(k)}\} \tag{6}$$

设卸载过程中 I 的传输功率为 P，则由 I 向基站进行卸载时消耗的能量为：

$$E_e^{(k)}=T_e^{(k)}P \tag{7}$$

由 I 向 I_i 进行卸载时消耗的能量为：

$$E_{d_i}^{(k)}=T_{d_i}^{(k)}P \tag{8}$$

综上，时隙 $t^{(k)}$ 内主设备 I 的总能量消耗为本地处理与卸载传输消耗的能量之和，计算公式为：

$$E^{(k)}=E_l^{(k)}+\sum_{i=1}^{M}E_{d_i}^{(k)}+E_e^{(k)} \tag{9}$$

3.3 无线充电

主设备 I 配有可以无线充电的电池，能储存能量，也能通过无线充电的方式从附近的边缘设备 I_i 获取能量。I 利用电池中的电量进行本地计算和边缘卸载。$b^{(k)}$ 是 $t^{(k)}$ 时的电量，则：

$$b^{(k+1)}=\max\{0,b^{(k)}-E^{(k)}+\rho^{(k)}\} \tag{10}$$

其中，$\rho^{(k)}$ 表示 $t^{(k)}$ 是边缘设备向主设备 I 所充电量的总和。

$$\rho^{(k)}=\sum_{i=1}^{M}\rho_i^{(k)} \tag{11}$$

$\rho_i^{(k)}$ 表示主设备 I 从边缘设备 I_i 处获得的电量，它的计算公式为 $\rho_i^{(k)}=\upsilon\eta^{(k)}(d_i^{(k)})^{-\tau}G_i$。其中，$\upsilon\in(0,1)$ 表示能量转换效率，$\eta^{(k)}$ 表示能量传输功率，$d_i^{(k)}$ 表示 $t^{(k)}$ 时隙内 I 到 I_i 的距离，$\tau\geqslant 2$ 表示路径衰减指数。一个时间段内边缘设备能够向主设备传输的电量用 G 表示。I 可对 $\rho^{(k)}$ 进行估计，估计的误差为：

$$\delta^{(k)}=\rho^{(k)}-\hat{\rho}^{(k)}\sim\xi U(-1,1) \tag{12}$$

这表示 $\delta^{(k)}$ 遵循最大值为 ξ 的均匀分布。

若 $b^{(k)}=0$，则说明电量耗尽，会导致当前时隙的全部任务丢失。因此，维持主设备的电量充足是 D2D-MEC 网络的关键一环。

3.4 优化问题

本节建立衡量卸载效果的优化公式。对于执行时间敏感型任务的移动设备（如体感游戏的计算设备、动态规划路线的扫地机器人），执行任务的时延与设备的工作效果高度相关；而移动设备的能量储备通常较少，因此需要密切关注执行任务消耗的能量与当前电量（本文也正是因此将无线充电技术引入边缘卸载过程，若移动设备可以通过无线充电的方式保证续航，则可进一步缩小电池体积，提升设备的灵活性和计算能力）。这样的设备通常对连接稳定性的要求也很高，因此需考虑传输过程中的丢包率 D。当这样的设备周围有大量空闲计算资源可利用（如有联网能力的家电）时，为了提高对空闲计算资源的利用率，需将主设备向边缘设备及基站卸载任务的比例 F 和边缘设备利用率 U 设置为优化问题中的两个指标。

因此综合来看，本文从以下 5 个方面来衡量卸载效果的好坏：时延 T、能量消耗 E、主设备向边缘设备及基站卸载任务的比例 F、边缘设备利用率 U，以及丢包率 D。其中，T 与 E 的计算公式前文已经给出。$F^{(k)}$ 指的是时隙 $t^{(k)}$ 内主设备向边缘设备及基站卸载的任务在总任务中的比例，计算公式为：

$$F^{(k)}=\sum_{1}^{M}x^{(k)}+y^{(k)} \tag{13}$$

类似地，边缘设备利用率 $U^{(k)}$ 是时隙 $t^{(k)}$ 内接受卸载任务的边缘设备数量占所有边缘设备数量的比例：

$$U^{(k)}=\frac{\sum_{1}^{M}I(x_i^{(k)}\neq 0)}{M} \tag{14}$$

其中，I 为示性函数。若时隙 $t^{(k)}$ 内边缘设备 I_i 接受主设备的卸载任务，辅助主设备进行计算，则 $I(x_i^{(k)}\neq 0)=1$，否则 $I(x_i^{(k)}\neq 0)=0$。因此，$\sum_{1}^{M}I(x_i^{(k)}\neq 0)$ 为时隙 $t^{(k)}$ 内接受卸载任务的边缘设备数量。边缘设备利用率 $U^{(k)}$ 为参与资源分配的设备数量在所有边缘设备中的比例。

卸载计算过程中，有两种情况会导致丢包：1）主设备电量不足，这会导致全部任务的丢失；2）主设备将部分任务卸载到未连接的边缘设备，这会导致向该设备卸载的这部分任务丢失。因此 D 的计算公式为：

$$D^{(k)}=\max\left(I(b^{(k)}=0),\frac{\sum I(C_i^{(k)}=0)I(x_i^{(k)}\neq 0)}{n}\right) \tag{15}$$

为这 5 项指标填上正数权重 $q_i(i=1,2,3,4,5)$，得到表示卸载效果的优化公式：

$$r^{(k)}=-q_1E^{(k)}-q_2T^{(k)}+q_3F^{(k)}+q_4U^{(k)}+q_5D^{(k)}(q_i\geqslant 0) \tag{16}$$

求解该优化问题，相当于在保证计算内容完整的基础上寻求低时延、低能耗且充分利用边缘设备的计算资源的卸载策略。

优化问题的可行域为主设备的动作空间 $A=\{\boldsymbol{a}^{(k)}=[x_1^{(k)},\cdots,x_M^{(k)},y^{(k)}]\mid x_1^{(k)}+\cdots+x_M^{(k)}+y^{(k)}\leqslant 1\}$。$A$ 中的元素为动作向量 $\boldsymbol{a}^{(k)}=[x_1^{(k)},\cdots,x_M^{(k)},y^{(k)}]$，$x_1^{(k)},\cdots,x_M^{(k)}$ 分别表示 I 向边缘设备 I_i 卸载的任务比例，$y^{(k)}$ 表示 I 向基站卸载的任务比例。任意的动作向量 $\boldsymbol{a}^{(k)}$ 满足约束 $x_1^{(k)}+\cdots+x_M^{(k)}+y^{(k)}\leqslant 1$，这是因为卸载的任务比例不可能超过百分之百。例如，若 $\boldsymbol{a}^{(k)}=[1,0,\cdots,0]$，则表示主设备 I 选择将所有任务卸载到边缘设备 I_i。

模型中涉及的主要符号及含义如表 1 所列。

表 1 主要符号及其含义

Table 1 Main notations and their descriptions

符号	含义
M	边缘设备数
C	主设备产生任务数据量
n	任务分成的份数
B_i	I 与 I_i 之间的上行无线传输速率
B	I 与基站之间的上行无线传输速率
$c_i^{(k)}$	$t^{(k)}$ 结束前 I 和 I_i 连续连接的时长
x_i	I 向 I_i 卸载的任务比例
y	I 向基站卸载的任务比例
N	处理每字节输入任务所需的 CPU 周期数
f	CPU 时钟频率
ζ	有效电容系数
P	主设备传输功率
b	主设备电量
ρ	主设备获取的电量
η	边缘设备传输功率
E	能量消耗
T	时延
F	主设备向边缘设备卸载任务比例
U	边缘设备利用率
D	丢包率
r	卸载效果

4 算法设计

4.1 马尔可夫模型

计算卸载的过程可以被看作一个马尔可夫过程。标准的马尔可夫决策过程（Markov Decision Process，MDP）包含以下 5 个元素：离散有界的状态空间 P、离散的动作空间 S、状态转移概率函数 P、回报函数 r，以及折扣因子 γ。MDP 的目标是找到一个策略，使得回报函数 r 的长期累计值的数学期望最大。

状态空间 S 包含了时隙 $t^{(k)}$ 时的状态向量 $\boldsymbol{s}^{(k)}=[B^{(k-1)}, B_1^{(k-1)},\cdots,B_M^{(k-1)},c_1^{(k-1)},\cdots,c_M^{(k-1)},b^{(k-1)},\rho^{(k)}]$，其中 $B^{(k-1)}, B_1^{(k-1)},\cdots,B_M^{(k-1)},c_1^{(k-1)},\cdots,c_M^{(k)},b^{(k)}$ 可由主设备对环境状态和自身状态的观察得到，$\rho^{(k)}$ 可根据式(11)估计得到。动作空间 A 中的元素为动作 $\boldsymbol{a}^{(k)}=[x_1^{(k)},x_2^{(k)},\cdots,x_M^{(k)},y^{(k)}]$。主设备每次依策略执行结束都会得到一个即时的回报，回报函数 r 由当前状态和选取的动作决定，采用模型中的优化公式(16)。

4.2 基于 Q 学习的计算卸载策略

Q 学习[21-22]基于马尔可夫决策过程，是一种与模型无关的强化学习算法。智能体学习从环境到行为的映射，使得奖励信号函数值最大。Q 学习每次通过选择结果更新奖励信号 $Q(\boldsymbol{s},\boldsymbol{a})$，使得它是对状态动作价值函数 $Q*(\boldsymbol{s},\boldsymbol{a})$ 的近似。所有状态和动作对应的奖励信号存储在一个叫做 Q-table 的表格中。

传统优化方法每次工作需要输入最新环境信息，然后重新计算最优策略。而 Q 学习每次与环境交互后，只需更新 Q-table 中对应本次交互的一个奖励信号的值。因此，相比于传统的优化方法，Q 学习的计算速度更快，且能更好地应用于不断变化的环境。

ε-贪心策略（$0<\varepsilon\leqslant 1$）指的是以某个较大的概率选取最优的动作 $\boldsymbol{a}^*$，并以很小的概率从全部动作中随机选取进行探索的一种决策方式。它可以被应用于强化学习的训练过程中，以防止网络收敛到局部极小值点。

基于上节的 MDP 模型，构建基于 Q 学习的 D2D 卸载算法 D2D-Q。算法的伪代码如算法 1 所示。

算法 1 D2D-Q 卸载算法

1. 初始化 $\mathbf{s}^{(0)}$，C，f，M，N，q_i(i=1,2,3,4,5)；
2. 通过迁移学习初始化 Q(S,A)的值；
3. for k=1,2,…,do
4. 依式(11)估计 $\rho^{(k)}$；
5. 观察 $B,B_1^{(k-1)},\cdots,B_M^{(k-1)},c_1^{(k-1)},\cdots,c_M^{(k-1)},b^{(k)}$；
6. $\mathbf{s}^{(k)}=[B,B_1^{(k-1)},\cdots,B_M^{(k-1)},c_1^{(k-1)},\cdots,c_M^{(k-1)},b^{(k)},\rho^{(k)}]$；
7. 选择 $\mathbf{a}^{(k)}=[x_1^{(k)},x_2^{(k)},\cdots,x_M^{(k)},\mathbf{y}_{(k)}]$；
8. 分别向 I_i 卸载数据量为 $x_i^{(k)}C$ 的数据，向 MEC 服务器卸载数据量为 $y^{(k)}C$ 的数据；
9. 在整个时隙内，从符合无线充电条件的边缘设备 I_i 处持续获取能量；
10. 依式(16)计算回报函数 r；
11. 依式(17)更新奖励信号 Q(s,a).

首先，初始化模型参数及环境信息，并通过转移学习初始化 $Q(S,A)$。$Q(S,A)$是一个表，记录着每一个“状态-动作对”的价值。这个表格称为 Q 表（Q-table），表中的每一项 $Q(S,A)$表示主设备在状态 $\boldsymbol{s}$ 下选择动作 $\boldsymbol{a}$ 的价值。

在每个时隙 $t^{(k)}$ 内，主设备 I 首先得到状态向量 $\boldsymbol{s}^{(k)}$（如 4.1 节所述），再根据 $\boldsymbol{s}^{(k)}$ 和 ε-贪心算法（$0<\varepsilon\leqslant 1$）从动作空间 A 中选择一个动作 $\boldsymbol{a}^{(k)}=[x_1^{(k)},x_2^{(k)},\cdots,x_M^{(k)},y^{(k)}]$。任务完成卸载后，主设备 I 会计算 $r^{(k)}$，并将其用于更新 $Q(\boldsymbol{s},\boldsymbol{a})$，更新公式如下：

$$Q(s^{(k)},a^{(k)})\leftarrow(1-\alpha)Q(\boldsymbol{s}^{(k)},\boldsymbol{a}^{(k)})+\alpha(r^{(k)}+\gamma\max_{a'\in A}Q(\boldsymbol{s}^{(k+1)},\boldsymbol{a}')) \tag{17}$$

其中，$\max_{a'\in A}Q(\boldsymbol{s}^{(k+1)},\boldsymbol{a}')$表示递归地得到下一状态 $\boldsymbol{s}^{(t+1)}$ 所对应的最优策略的 Q 值，即之后状态的 Q 值会对之前状态的 Q 值产生影响。折扣因子 $\gamma\in[0,1]$为算法注重长远利益的程度。$\gamma=0$ 时，算法只关注当前获得的奖励；$\gamma=1$ 时，算法只关注后续的奖励。$\alpha\in(0,1)$为学习率，学习率越大，回报函数值 r 对 $Q(\boldsymbol{s},\boldsymbol{a})$的修正影响越大。

算法 1 的时间复杂度与状态空间的大小有关，为 $O(M^M)$（M 为边缘设备的数量）。空间复杂度与 Q-table 的大小有关，为 $O(M^{2M})$。因此，算法 1 仅适用于边缘设备较少的情况。

4.3 基于 DQN 的计算卸载策略

本节用 DQN 算法代替 Q 学习来训练模型，从而提出了 D2D-DQN 卸载方案。

DQN 算法使用深度神经网络来代替 Q-table，这使得该算法可以适应更大的状态空间。该神经网络的参数设为 $\boldsymbol{\theta}$。为了给该神经网络提供训练集，引入经验回放（Experience Replay）机制。设置一个经验池（Memory Pool）来储存训练所需的信息，训练时随机从中取出一部分。

此外，网络更新与强化学习同时进行，这使得每次估计的 Q 值与目标 Q 值产生自同一个神经网络，二者具有一定的相关性，所以并不能很好地起到更新网络的效果。为了解决这个问题，设置两个结构和初始参数完全一样但后续参数不同的深度神经网络，称为 Main Net Q 和 Target Net $\bar{Q}$。Main Net Q 用于在强化学习中估计“状态-动作对”的 Q 值，始终采用的是最新的参数。而 Target Net $\bar{Q}$ 中的参数是一段时间以前的参数，用于在神经网络训练中借助式(17)估计目标Q值，并根据损失函数更新神经网络的参数 $\boldsymbol{\theta}$。每隔一段时间，Main Net Q 将参数 $\boldsymbol{\theta}$ 复制给 Target Net $\bar{Q}$ 的参数 $\bar{\boldsymbol{\theta}}$，这样可以在保证估计准确性的同时尽量减小两个网络的相关性。

图 2 给出了 DQN 对 D2D-MEC 模型的具体的训练过程。首先，初始化环境变量，初始化两个网络 Q 和 $\bar{Q}$ 以及经验池 $\mathcal{D}$。主设备在每个时隙之初进行决策，时隙结束后更新参数。决策时，Main Net Q 从环境获取 $\boldsymbol{s}^{(k)}$，得到 $Q(\boldsymbol{s}^{(k)},\boldsymbol{a};\boldsymbol{\theta}^{(k)})$，这是状态 $\boldsymbol{s}$ 关于所有动作 $\boldsymbol{a}$ 的 Q 值函数。主设备依 ε-贪婪策略选取动作 $\boldsymbol{a}$。卸载完成后，主设备 I 计算 $r^{(k)}$。再将这一步的状态、选择、回报及结果（下一步状态）作为经验池 $\mathcal{D}$ 的一个元素 $(\boldsymbol{s}^{(k)},\boldsymbol{a}^{(k)},r^{(k)},\boldsymbol{s}^{(k+1)})$存入。

神经网络的训练采用随机小批量梯度下降的方式。首先从经验池 $\mathcal{D}$ 中随机取出一小批训练经验 $(\boldsymbol{s}^{(k)},\boldsymbol{a}^{(k)},r^{(k)},\boldsymbol{s}^{(k+1)})$，利用 Target Net $\bar{Q}$ 和卸载所得 $r^{(k)}$ 估计目标 Q 值

Q^*。估计公式为：

$$Q^*(s^{(k)},a^{(k)})=(1-\alpha)\bar{Q}(s^{(k)},a^{(k)},\bar{\theta})+\alpha(r^{(k)}+\gamma\max_{a'\in A}\bar{Q}(s^{(k+1)},s',\bar{\theta}) \quad (18)$$

并依据均方误差损失函数进行梯度下降，更新参数 θ。损失函数为：

$$L(\theta^{(k)})=E_{s^{(k)},a^{(k)},r^{(k)},s^{(k+1)}}[(Q^*-Q(s^{(k)},a^{(k)},\theta^{(k)})^2] \quad (19)$$

每一幕结束后，将 Q 的参数全部复制给 $\bar{Q}$，再进行新一幕的训练。

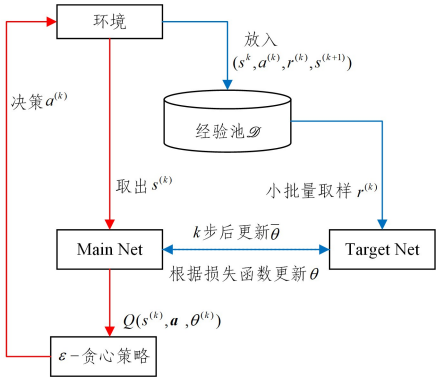


图 2 DQN 计算卸载过程

Fig. 2 Deep Q-network computing offloading process

相比于 Q 学习，DQN 更适合 D2D 模型。DQN 算法的优点在于具有更快的收敛速度，且在更高维度的训练集与动作集上表现出色。这是因为 Q 学习中使用一个 Q-table 来储存所有“状态-动作对”对应的 Q 值；而 DQN 中将其更换成了深度神经网络拟合的函数，因此可应用于连续而非离散的状态空间和动作空间。D2D 模型的应用场景多为延迟敏感型，因此采用 DQN 算法可以显著优化卸载体验。D2D 模型将多粒度任务卸载到多个边缘设备，环境状态更丰富，拥有更多可选动作，是高维度空间的模型。此外，DQN 支持任务粒度进一步细分，从而提升卸载效果。

具体训练过程的伪代码如算法 2 所示。

算法 2 D2D-DQN 卸载算法

1. 初始化 $\mathbf{s}^{(0)}$, C, f, M, N, q_i(i=1,2,3,4,5)和经验池；
2. 初始化 Q 和 $\bar{Q}$ 的所有参数；
3. for episode=1,2…, M do
4. for t=1,2,…T do
5. 估计 $\rho^{(k)}$；
6. 观察 B, $B_1^{(k-1)}$, …, $B_M^{(k-1)}$, $c_1^{(k-1)}$, …, $c_M^{(k-1)}$, $b^{(k)}$；
7. $\mathbf{s}^{(k)}=[B,B_1^{(k-1)},\cdots,B_M^{(k-1)},c_1^{(k-1)},\cdots,c_M^{(k-1)},b^{(k)},\rho^{(k)}]$；
8. 将 $\mathbf{s}^{(k)}$ 输入 Q 的神经网络，得到该状态对应所有动作的 Q 值 $Q(\mathbf{s}^{(k)},\mathbf{a};\boldsymbol{\theta}^{(k)})$；
9. 选择 $a^{(k)}=[x_1^{(k)},x_2^{(k)},\cdots,x_M^{(k)},y^{(k)}]\in A$；
10. 分别向 $\mathbf{I}_i$ 卸载数据量为 $x_i^{(k)}C$ 的数据，向 MEC 服务器卸载数据量为 $y^{(k)}C$ 的数据；
11. 在整个时隙内，从符合无线充电条件的边缘设备 $\mathbf{I}_i$ 处获取能量；
12. 依式(16)计算回报函数 r；
13. 将四元组$(\mathbf{s}^{(k)},\mathbf{a}^{(k)},r^{(k)},\mathbf{s}^{(k+1)})$放入经验池 $\mathcal{D}$；
14. 从 $\mathcal{D}$ 中小批量随机采样 batchsize 个样本；
15. if t+1=T,
16. $Q^*(\mathbf{s}^{(k)},\mathbf{a}^{(k)})=\bar{Q}(\mathbf{s}^{(k)},\mathbf{a}^{(k)})$；
17. else
18. $Q*(\mathbf{s}^{(k)},\mathbf{a}^{(k)})=(1-\alpha)\bar{Q}(\mathbf{s}^{(k)},\mathbf{a}^{(k)},\bar{\boldsymbol{\theta}})+\alpha(R^{(k)}+\gamma\max_{a'\in A}\bar{Q}(\mathbf{s}^{(k+1)},\mathbf{a}',\bar{\boldsymbol{\theta}}))$；
19. 依式(19)更新 $\boldsymbol{\theta}$；
20. end for
21. $\bar{Q}=Q$；
22. end for

算法 2 的时间和空间复杂度分析如下。设 DQN 中每个隐含层包含 N 个神经元(输入层和输出层的神经元数小于或等于 N)，隐含层的数量为 D。在一次训练中，算法的主要计算过程由前向传播、误差计算(反向传播)和权值更新 3 部分组成。设每次训练时，从经验池中随机取出的样本量大小为 a，则前向传播的时间复杂度为 $O(a(D+2)N^2)$，误差计算的时间复杂度为 $O(aDN^3)$，权值更新复杂度为 $O(aDN(N+1))$。因此，一次训练中，算法的时间复杂度为 $O(a(D+2)N^2+aDN^3+DN(N+1))=O(aDN^3)$。由于 a 和 D 是常数，可以忽略，因此算法 2 最终的时间复杂度为 $O(MTN^3)$。算法 2 的神经网络中参数数量为 $O((N^2+N)(D-1))=O(N^2)$(省略输入层和输出层的参数数量)。

5 性能估计

本节分析了普遍场景中的最优卸载策略，可被用作性能上限指标来评估动态过程中充足时隙下的强化学习卸载方案。为研究性能上限，不妨假设主设备一直在边缘设备的无线充电支持下保持着充足的电量，也无需考虑能耗问题。

在动态的物联网环境中，边缘设备的位置及边缘设备与主设备之间的连接状态等处于不断变化中，下面分别对这两点进行详细分析。

5.1 动态问题

边缘设备的位置不断变化，会影响边缘设备与主设备的连接状态，进而影响丢包率。因此，需要分析不同移动状况对边缘设备卸载策略的影响。

为表述清晰，本节的符号和算式中省略了时间上标 k。

假设处于某个特定的状态时，主设备选择 $\boldsymbol{a}^*=[x_1^*,x_2^*,\cdots,x_n^*,y^*](x_i^*,y^*\in\{l/n\},0<l<n)$作为最优卸载策略。其中，$x_i^*$ 表示主设备向 I_i 卸载的任务比例，y^* 表示主设备向 MEC 服务器卸载的任务比例。

状态 1 假设所有边缘设备静止，则主设备与边缘设备 I_i 的连接状态 c_i' 取值恒定，即始终为连接状态或未连接状态。称系统的这种状态为全静止状态。

那么可以计算全静止状态下的断连丢包率：

$$I(x_i\neq 0)\cdot I(c_i=0)=\begin{cases}1, & c_i'=0 \text{ 且 } x_i\neq 0\\ 0, & \text{其他}\end{cases} \quad (20)$$

若全静止状态持续足够长的时间，则设备会在学习决策的过程中减少断联丢包现象的发生，最终断连丢包率会收敛到 0。

状态 1 的推广：边缘设备不一定静止，但与主设备的连接状态 c_i' 仍恒定。若所有边缘设备均如此，则称为系统的稳定

连接状态。断连丢包率的计算与状态 1 完全相同，训练后断连丢包率仍会收敛到 0。

状态 2(全随机状态) 边缘设备全部进行两个位置之间的随机往返运动，且这两个位置上边缘设备与主设备的连接状态不同。这一状态可以抽象为：设备 c_i' 的状态满足期望为 p 的二项分布 $b(p)$，则在全随机状态下，任一时隙的断连丢包率为：

$$E[I(x_i\neq 0)\cdot I(c_i'=0)]=p \tag{21}$$

由于 c_i' 在下一时隙的状态与上一时隙的状态完全无关，因此无论主设备做出何种卸载选择，断连丢包率都为 p。

由于每个时隙足够短，边缘设备在连接状态改变的一个极小时间邻域 $(t^{(k-j)},t^{(k+j)})$ 内可视作全随机状态，其余时间为稳定连接状态。因此，只需分析这两种状态的最优策略及卸载表现。由式(16)知，二者的卸载效果 r 只相差断连丢包率，由上文的计算可知，这个差值就是常数 p。因此，只需分析稳定连接状态，即可将分析结果以完全一致的方法推广到全随机状态。而稳定连接状态与全静止状态的断连丢包率完全相等，r 值也完全相等，因此只需对全静止状态的 MEC-D2D 系统进行分析，即可完成对任意状态的分析。

5.2 分配问题

本节的分析建立于 D2D-MEC 系统的全静止状态，最终结论为：对于延迟敏感型任务，设备与设备间的上行传输速率 $[B_1,B_2,\cdots,B_n,B]$ 是决定任务配比 $\boldsymbol{a}=[x_1,x_2,\cdots,x_n,y]$ 的关键因素。假设 CPU 时钟频率为 f，且恒定，则卸载延迟 T 为：

$$T=\max\left\{\frac{CN(1-\sum x_i-y)}{f},\frac{x_iC}{B_i},\frac{yC}{B},(i=1,\cdots,M)\right\} \tag{22}$$

其中，C,N,f 都是定值。

下面的一系列命题说明了几种不同的上行传输速率对应的最优卸载策略。

命题 1 令 $V=\dfrac{q_2Cf}{nq_2CN+nf(q_3+q_4)}$，当 $B\leqslant V$ 且 $\forall B_i\leqslant V$ 时，$x_i=y=0$，即 $\boldsymbol{a}^*=[0,0,\cdots,0,0]$。

证明：在全静止状态，电量充足的假设下，断连丢包率与断电丢包率在训练后都可趋于 0，因此计算最优性能时，将丢包率 D 置 0。同时，因为无需考虑能耗问题，因此能耗 E 也为 0，则优化公式变为：

$$r^{(k)}=-q_2T^{(k)}+q_3F^{(k)}+q_4U^{(k)}\ (q_i\geqslant 0) \tag{23}$$

(1)若 $x_i\neq 0$，由 $T\geqslant T_{d_i}$，有：

$$\begin{aligned}r(x_1,x_2,\cdots,x_n,y)&\leqslant -q_2T_{d_i}+q_3F+q_4U\\&\leqslant -q_2\frac{x_iC}{V}+q_3+q_4\\&=-q_2\frac{CNnx_i}{f}-(nx_i-1)(q_3+q_4)\\&\leqslant -q_2\frac{CN}{f}=r(0,\cdots,0,0)\end{aligned}$$

(2)若 $y\neq 0$，由 $T\geqslant T_e$，有：

$$\begin{aligned}r(x_1,x_2,\cdots,x_n,y)&\leqslant -q_2T_e+q_3F+q_4U\\&\leqslant -q_2\frac{yC}{V}+q_3+q_4\\&=-q_2\frac{CNny}{f}-(ny-1)(q_3+q_4)\\&\leqslant -q_2\frac{CN}{f}=r(0,\cdots,0,0)\end{aligned}$$

因此 $x_i=y=0$。

由证明过程可知，在命题 1 的条件下，r 最终会收敛到 $-q_2\dfrac{CN}{f}$。

命题 2 若存在 i^* 使得 $B_{i^*}\geqslant\dfrac{nf}{N}$，且 $B\leqslant\dfrac{f}{N}$，则 $\sum x_i=1$。

证明：(1) 若 $y=0$，由 $T\geqslant T_e$，有：

$$\begin{aligned}r(x_1,x_2,\cdots,x_n,y)&\leqslant -q_2T_e+q_3F+q_4U\\&\leqslant -q_2\frac{yCN}{f}+q_3+q_4\\&=-q_2\frac{nyC}{B_{i^*}}+q_3+q_4\\&\leqslant -q_2\frac{Cx_{i^*}}{B_{i^*}}+q_3+q_4\\&=r(\textstyle\sum x_i=1)\end{aligned}$$

(2)若 $y=0,\sum x_i\neq 1$，则 $1-\sum x_i-y\neq 0$。由 $T\geqslant T_l$，有：

$$\begin{aligned}r(x_1,x_2,\cdots,x_n,y)&\leqslant -q_2\frac{(1-\sum x_i-y)CN}{f}+q_3F+q_4U\\&\leqslant -q_2\frac{(1-\sum x_i-y)nC}{B_{i^*}}+q_3+q_4\\&\leqslant -q_2\frac{Cx_{i^*}}{B_{i^*}}+q_3+q_4\\&=r(\textstyle\sum x_i=1)\end{aligned}$$

因此 $\sum x_i=1$。

由证明过程可知，在命题 2 的条件下，r 最终会收敛到 $r(\sum x_i=1)$。

命题 3 令 $W=\dfrac{q_2Cf}{fnq_4+q_2CN}$，若 $B\geqslant\dfrac{nf}{N}$，且 $\forall B_i\leqslant W$，则 $y=1$，即 $\boldsymbol{a}^*=[0,0,\cdots,0,1]$。

证明：(1)若 $y\neq 1,\exists x_i\neq 0$，由 $T\geqslant T_{d_i}$，有：

$$\begin{aligned}r(x_1,x_2,\cdots,x_n,y)&\leqslant -q_2T_{d_i}+q_3F+q_4U\\&\leqslant -q_2\frac{x_i(fnq_4+q_2CN)}{f}+q_3+q_4\\&=-q_2\frac{x_iCN}{f}+q_3+(1-nx_i)q_4\\&\leqslant -q_2\frac{nx_iC}{B}+q_3\\&\leqslant -q_2\frac{C}{B}+q_3\\&=r(0,0,\cdots,0,1)\end{aligned}$$

(2)若 $y\neq 1,\forall x_i=0$，$T\geqslant T_l$，有：

$$\begin{aligned}r(x_1,x_2,\cdots,x_n,y)&\leqslant -q_2\frac{(1-\sum x_i-y)CN}{f}+q_3F\\&\leqslant -q_2\frac{x_i(1-\sum x_i-y)nC}{B}+q_3\\&\leqslant -q_2\frac{C}{B}+q_3\\&=r(0,0,\cdots,0,1)\end{aligned}$$

因此 $y=1$。

由证明过程可知，在命题 3 的条件下，r 最终会收敛到 $-q_2\dfrac{C}{B}+q_3$。

命题 4 若 $B \geqslant \frac{nf}{N}$，且 $\forall B_i \geqslant \frac{nf}{N}$，则 $\sum x_i + y = 1$。

证明：该命题证明与前几个证明思路相同，可由命题 2 和命题 3 的证明加以修改得到，此处暂不赘述。

6 实验结果与分析

6.1 参数设置

为了验证所提出的 D2D-MEC 模型下的两种计算卸载策略（以下简称为 D2D-Q 和 D2D-DQN），在 python 环境中调用 TensorFlow 来评估它们的性能。

将边缘设备的数量设置为 $M=3$，主设备每个时隙生成的计算任务为 120 kb，每处理一位数据需要 100 个 CPU 周期。主设备向每个边缘设备卸载的传输速率 B_i 遵从有 5 个状态的马尔可夫转移矩阵。例如，B_1 的状态转移矩阵为[4，5，6，7，10](Mb/s)，B_2 的状态转移矩阵为[3，4，5，6，7](Mb/s)。主设备向基站卸载的传输速率数量 B 遵从的转移矩阵为[4，5，6，7，10](10^7 Mb/s)。连接状态 c_i 为随机生成的 1 或 0。表 2 列出了主设备的性能参数。

表 2 训练参数

Table 2 Training parameters

参数	取值	参数	取值
C/kb	120	υ	0.9
n	10	P/W	0.5
N	100	η/W	7.5
f/GHz	1	τ	2

本文选取另外两种卸载方法与本文的两组方法进行比较分析。其中，D2D-R[23]为不包含无线充电的 D2D-MEC 网络，用 DQN 进行训练；RLO[24]模型不同，它由一个主设备 I 和 3 个基站组成，不包含边缘设备，且同一时隙最多卸载到一个基站。表 3 为具体的实验方案对比。

表 3 实验方案对比

Table 3 Comparison of experimental schemes

序号	实验方案	有无无线充电	算法	模型
1	D2D-R	无	DQN	D2D+server
2	D2D-Q	有	Q-learning	D2D+server
3	D2D-DQN	有	DQN	D2D+server
4	RLO	有	Q-learning	只有 server

训练过程中，对 4 组实验分别进行了 80 幕，每一幕的步长为 500 的训练。在 D2D-DRLO 中，DQN 是一个有两个隐藏层的全连接神经网络。网络采用的参数是 *learning rate*=0.001，*reward decay*=0.9，*greedy*=0.95。在 D2D-RLO 中，折扣因子 $\gamma=0.9$。在 D2D-R 和 RLO 中，$\gamma=0.99$。在 D2D-RLO，D2D-R 和 RLO 中，将 learning rate 和 greedy 的值设置为随训练过程动态下降。

6.2 训练结果

本小节详细分析了 4 组方案在相同实验背景下的训练结果。能量消耗、时延、设备利用率、卸载回报 r 和训练收敛速度是评价卸载方法性能的主要指标。影响实验结果的主要因素为实验方案和任务的产生速度。

6.2.1 能量消耗比较

本次实验比较了不同方案训练过程中能量消耗的变化。如图 3 所示，RLO 方案的能量消耗最低，其他 3 种方案能量消耗都略高。这是由于 RLO 方案无 D2D 系统，只向一个设备卸载任务，卸载消耗较小。在采用 D2D 系统的方案中，D2D-Q 方案能量消耗最小，只比 RLO 方案高 0.004 J/s。

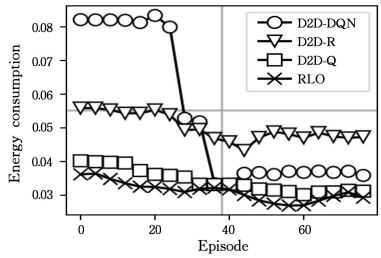


图 3 能量消耗

Fig. 3 Energy consumption

6.2.2 时延比较

图 4 为不同方案训练过程中的时延变化。图中，D2D-Q 方案与 D2D-DQN 方案的时延低于另外两种方案，这说明能够无线充电的 D2D-MEC 系统处理任务的速度更快，有着在时间敏感型设备上应用的前景。

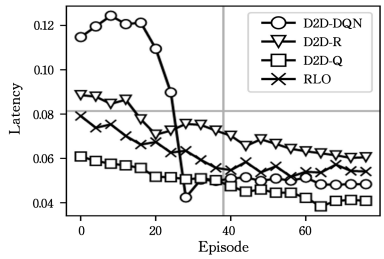


图 4 时间延迟

Fig. 4 Latency

6.2.3 设备利用率比较

提高设备利用率和利用边缘设备能源是构建 D2D-MEC 系统的关键目的。从图 5 可以看出，D2D-DQN 和 D2D-Q 的设备利用率都达到了 0.7 左右，显著高于另外两组方案。因此，D2D-MEC 系统成功地做到了鼓励卸载。D2D-R 的设备利用率很低，且在训练过程中持续下降，这是因为 D2D-R 没有无线充电模块，卸载导致的耗电量增加会提高丢包率，降低卸载性能。因此，D2D-R 方案不鼓励卸载。

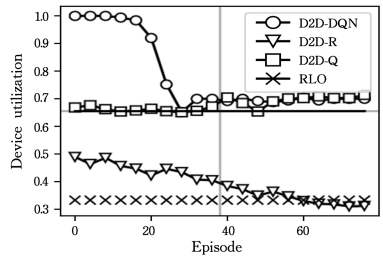


图 5 设备利用率

Fig. 5 Device utilization

6.2.4 卸载回报比较

图 6 给出了 4 种方案在训练过程中的卸载回报。D2D-DQN 与 D2D-Q 的回报显著高于其他两组，体现了具有无线充电的 D2D-MEC 卸载系统的性能优越性。D2D-DQN 的卸载回报最高，卸载效果最好。

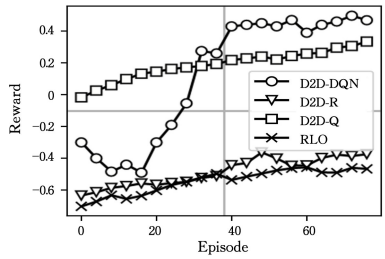


图 6 卸载奖励

Fig. 6 Off-loading reward

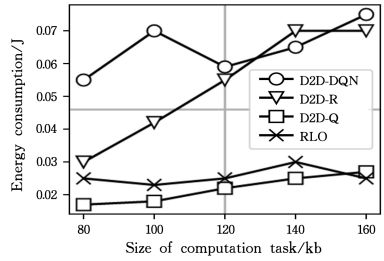
6.2.5 收敛速度比较

综合图 3－图 6，D2D-DQN 卸载策略的收敛时间不到其他卸载策略的一半。这是由于 DQN 网络中采用神经网络代替 Q-table，大大加快了收敛速度。

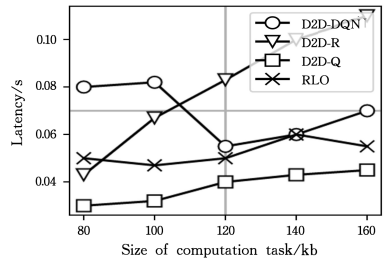
比较之下，本文提出的两种模型综合性能显著优于其他两种。其中，D2D-DQN 卸载策略的收敛速度更快，在综合性能相近的情况下，有着比 D2D-Q 更广泛的应用前景。

6.2.6 任务产生速度比较

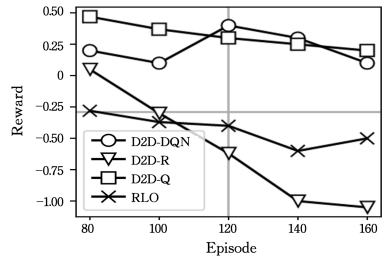
图 7 描述了各组实验在每秒生成任务量从 80kb/s 到 160kb/s 变化时训练结果的变化。



(a)能量消耗



(b)时间延迟



(c)卸载奖励

图 7 不同任务生成速度下的卸载表现

Fig. 7 Unloading performance at different task generation speeds

从图 7 可以看出：

(1)D2D-DRLO 在 120kb/s 时表现突出，甚至能量消耗和计算延迟远低于任务量更少的时候。

(2)随着任务量的增加，能量消耗和计算延迟本应成比例增加，但只有 D2D-R 策略有相应的增长幅度，其他 3 组带有无线充电模块的实验增长幅度都偏小很多。因此，带有无线充电模块的模型在每秒钟产生任务量更多时的表现更突出。

结束语 D2D 网络下的计算卸载是一种应用场景非常丰富的卸载方案，而设备间的无线充电技术为该网络的续航提供了保证。本文提出了一个包含 D2D 网络以及无线充电的移动物联网设备计算卸载模型，来计算最优卸载策略；应用了两种计算卸载的训练方案，一是利用强化学习的 D2D-Q 方案，二是利用深度强化学习的 D2D-DQN 方案。该方案使用 DQN 学习，用深度神经网络解决状态空间扩大的问题并加快收敛速度。本文对该决策框架下的卸载性能进行了全面估计，即不同的条件范围对应的卸载情况。最后对基于射频信号的无线功率传输的物联网设备进行了仿真，验证了理论结果。D2D 网络以及无线充电模块的加入，都使计算卸载的性能得到了显著提高。D2D-DRLO 方案在卸载性能相近的情况下，收敛速度更快，卸载率更高，因此优于 D2D-RLO 方案。我们将在未来的工作中实施上述卸载框架及两种卸载方案，并通过实验评估其性能。

参 考 文 献

[1] BOYD B, SHILPA T, REZA A, et al. Networks and devices for the 5G era[J]. IEEE Communications Magazine, 2014, 52(2): 90-96.

[2] KO H, PACK S. Distributed Device-to-Device Offloading System: Design and Performance Optimization[J]. IEEE Transactions on Mobile Computing, 2021, 20(10): 2949-2960.

[3] MOLIN L, CHEN T, ZENG J, et al. D2D-Assisted Computation Offloading for Mobile Edge Computing Systems with Energy Harvesting[C]// 2019 20th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT). Gold Coast, QLD, Australia: 2019: 90-95.

[4] 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.

[5] PU L, CHEN X, XU J, et al. D2D Fogging: An Energy-Efficient and Incentive-Aware Task Offloading Framework via Network-assisted D2D Collaboration[J]. IEEE Journal on Selected Areas in Communications, 2016, 34(12): 3887-3901.

[6] FANG T, YANG Y, CHEN J X. Optimization of offloading strategy under assisted mobile edge computing[J]. Computer Science, 2022, 49(S1): 601-605.

[7] QIAO G H, LENG S P, ZHANG Y. Online Learning and Optimization for Computation Offloading in D2D Edge Computing and Networks[J]. Mobile Networks & Applications, 2019, 27: 1111-1122.

[8] GAIBIN L, CHEN M, WEI X, et al. Computation Offloading With Reinforcement Learning in D2D-MEC Network [C]// 2020 International Wireless Communications and Mobile Computing (IWCMC). Limassol, Cyprus, 2020: 69-74.

[9] HUANG L, FENG X, QIAN L, et al. Deep Reinforcement Learning-Based Task Offloading and Resource Allocation for

Mobile Edge Computing[C]// Machine Learning and Intelligent Communications(MLICOM). Cham: Springer, 2018: 33-42.
[10] LIANG W, WANG K, PAN C, et al. Deep Reinforcement Learning Based Dynamic Trajectory Control for UAV-assisted Mobile Edge Computing[J]. arXiv: 1911.03887v2, 2019.
[11] MINGHUI M, XIAO L, CHEN Y, et al. Learning-Based Computation Offloading for IoT Devices With Energy Harvesting [J]. IEEE Transactions on Vehicular Technology, 2019, 68(2): 1930-1941.
[12] 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.
[13] KAN L P G, HAUSWALD J, GAO C, et al. Collaborative Intelligence Between the Cloud and Mobile Edge[J]. Acm Sigplan Notices A Monthly Publication of the Special Interest Group on Programming Languages, 2017, 45(1): 615-629.
[14] WANG Z C, HUA J W, ZHU J Q, et al. Vehicle task offloading based on deep reinforcement learning for parking cooperation [J]. Small Microcomputer System, 2023, 44(3): 658-664.
[15] LIANG H, ZHANG L, YANG S, et al. Meta-Learning Based Dynamic Computation Task Offloading for Mobile Edge Computing Networks [J]. IEEE Communications Letters, 2021, 25(5): 1568-1572.
[16] DENG S Q, YE X G. Multi-objective task offloading algorithm based on deep Q network [J]. Computer Application, 2022, 42(6): 1668-1674.
[17] LIU P F, MAO Y C, WANG B L. Task allocation method based on cloud-fog collaboration model [J]. Computer Application, 2019, 39(1): 8-14.
[18] CHENG P, ZHANG W Z, XIE S H, et al. Research on multi-objective balanced task offloading method for edge computing of Internet of Vehicles [J]. Small Microcomputer System, 2022, 43(9): 1992-1997.
[19] DINH Q, TANG J, LA Q, et al. Offloading in Mobile Edge Computing: Task Allocation and Computational Frequency Scaling [J]. IEEE Transactions on Communications, 2017, 65(8): 3571-3584.
[20] ZHAO S, LI Y, WU D, et al. Current-Decomposition-Based Digital Phase Synchronization Method for BWPT System [J]. IEEE Transactions on Power Electronics, 2021, 36(11): 12183-12188.
[21] WATKINS C. Learning from delayed rewards [D]. Cambridge: King's College of University of Cambridge, 1989.
[22] CHRISTOPHER J, PETER D. Q-learning [J]. Machine Learning, 1992, 8(3/4): 279-292.
[23] XIAO L, LI Y, HUANG X, et al. Cloud-based malware detection game for mobile devices with offloading[J]. IEEE Trans Mobile Computing, 2017, 16(10): 2742-2750.
[24] LI J, GAO H, LV T. Deep reinforcement learning based computation offloading and resource allocation for MEC[C]// IEEE Wireless Communication and Networking Conf(WCNC). 2018: 1-6.

ZHANG Naixin, born in 2000, master candidate. Her main research interests include mobile edge computing and machine learning.

WU Huaming, born in 1986, Ph.D, associate professor, Ph. D supervisor, is a member of China Computer Federation. His main research interests include Internet of things(IoT), mobile edge computing and cloud computing.

(责任编辑:柯颖)