南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

学 士 学 位 论 文

基于深度强化学习的移动边缘计算卸载方法研究

周天钰

（学生姓名）

920103860442

（学号）

指 导 教 师	孟顺梅　　副教授
学 生 学 院	计算机科学与工程学院
专　　业	计算机科学与技术
研 究 方 向	移动边缘计算卸载方法
论文提交时间	2024 年　6 月

声　明

我声明，本毕业设计说明书及其研究工作和所取得的成果是本人在导师的指导下独立完成的。研究过程中利用的所有资料均已在参考文献中列出，其他人员或机构对本毕业设计工作做出的贡献也已在致谢部分说明。

本毕业设计说明书不涉及任何秘密，南京理工大学有权保存其电子和纸质文档，可以借阅或网上公布其部分或全部内容，可以向有关部门或机构送交并授权保存、借阅或网上公布其部分或全部内容。

学生签名：

年　月　日

指导教师签名：

年　月　日

毕业设计说明书中文摘要

移动边缘计算技术在近来展现出其巨大潜力，这种方法通过无线接入网络和边缘服务器，为终端设备提供计算卸载服务，允许这些设备将工作负载转移至附近的边缘服务器。其中，与深度强化学习相结合的计算卸载方法，更是成为了解决动态边缘环境中计算卸载问题的重点研究方向。本文聚焦于单个边缘服务器，多个终端用户的计算卸载场景，为了最小化终端用户的长期平均任务计算成本，提出了一个基于深度强化学习的计算卸载算法，来联合优化终端用户的计算和传输功率。在该算法中，每个终端用户被建模为一个智能体，并通过与移动边缘计算环境的不断交互，来累积卸载经验，逐步提升其卸载策略。实验结果表明，相较于基线算法而言，所提出的算法能够更加有效地权衡终端用户的能耗和任务缓存时延，降低其任务计算成本。

关键词　　移动边缘计算　深度强化学习　计算卸载　多用户 MIMO

毕业设计说明书外文摘要

Title Research on Mobile edge computing Method Based on Deep Reinforcement Learning

Abstract

Mobile Edge Computing (MEC) has shown tremendous potential recently. This approach leverages wireless access networks and edge servers to provide computation offloading services for terminal devices, allowing these devices to offload workloads to nearby MEC servers. Among these, the integration of deep reinforcement learning has emerged as a key research direction to address computation offloading in dynamic edge environments. This paper focuses on the scenario of computation offloading with multiple terminal users to a single edge server. To minimize the long-term average task computation cost for terminal users, a deep reinforcement learning-based computation offloading algorithm is proposed to jointly optimize terminal users' computation and transmission power. In this algorithm, each terminal user is modeled as an agent and accumulates offloading experience through continuous interaction with the mobile edge computing environment to gradually improve its offloading strategy. Experimental results demonstrate that the proposed algorithm can effectively trade off terminal users' energy consumption and task caching latency compared to baseline algorithms, thereby reducing their task computation costs.

Keywords Mobile edge computing, Deep reinforcement learning, Computation offloading, Multi-user MIMO

目 次

1 绪论

本文研究的课题内容是在单个边缘服务器，多个终端用户的计算卸载场景中，研究基于连续动作空间的深度确定性策略梯度(deep deterministic policy gradient, DDPG)深度强化学习方法的移动边缘计算卸载方法，运用分布式的计算卸载框架，以实现终端用户的计算成本多目标优化。本章内容主要介绍了课题的研究背景，介绍了移动边缘计算卸载方法的研究现状，并且对本课题的主要研究内容和研究意义进行了说明。

1.1 研究背景及意义

随着科技的迅速发展，特别是即将到来的 5G 时代智能移动设备的普及，接入互联网的移动设备数量越来越多，产生的数据流量的规模也前所未有的庞大，这不可避免地给网络基础设施的设计带来了新的严峻挑战[1]。同时，用于服务新兴物联网[2]的大量低功耗、资源受限的无线终端，也已经难以满足物联网终端日益增长的服务质量等需求。

移动云计算（Mobile Cloud Computing，MCC）[3]是一种结合了移动计算和云计算的技术，通过利用云计算资源和服务来增强移动设备的功能和处理能力。它有效地利用云基础架构的强大特性，来减轻移动设备的压力，为用户提供了丰富的计算资源和强大的网络服务。移动云计算的核心优势就是可以将应用程序任务转移到云中心的服务器，然后在云计算中心的高效计算平台实现数据的处理和分析，以此解决计算密集型任务。

但是，物联网的高速发展更使得数据的增长远远甩开网络带宽的增速，传统的云计算模式遇到了带宽与延迟的瓶颈，这使得云计算无法适应对延迟有严格要求的适应场景[4]。因此，要满足质量要求，移动边缘计算(Mobile Edge Computing, MEC)[5]于 2013 年被首次提出，IBM 和诺基亚西门子通信联合推出了一个计算平台，可以在无线基站内运行应用程序，为移动用户提供服务[6]。移动边缘计算这一种支持移动用户设备实现对带宽与时延有着极度的敏感性的应用的新兴技术[7]，被认为是一种很有前途的解决方案，以弥补移动设备上有限的资源与日益增长的移动应用计算需求之间的差距[8]。

移动边缘计算的主要思想是将云计算资源带到终端用户附近，并在本地为终端用户的请求提供计算卸载（Computation Offloading，CO）服务[9]。计算卸载实际上就是把用户的计算任务从网络核心侧卸载到靠近用户的边缘设备来进行，边缘服务器则可以运用其丰富的计算资源高效完成分配给自己的任务，再将任务结果返还给终端。这样能够显著降低功耗、减少时延，使得用户计算和传输需求在很大程度上靠近用户的网络边缘而得到满足，弥补了用户计算能力不足、能量供给受限和大量数据远距离传输带来的高服务时延以及用户能耗的缺陷，

从而为设备提供高质量的服务[10]。

移动设备可将计算任务转移到与基站相连的 MEC 服务器，以此大幅减少延迟和功耗，进而提升移动应用的运行效能。然而，计算卸载的效率在很大程度上依赖于无线数据传输的有效性，这就要求支持 MEC 的系统能够管理无线电资源和计算资源，并有效地完成计算任务[11]。因此，为了降低移动设备的时延和能量消耗，提高用户的服务质量以满足日常的生产生活需要，有必要为终端设备制定一个良好的卸载决策。

近年来，深度强化学习（Deep Reinforcement Learning, DRL）[12]已经发展成为解决复杂环境下任务优化的重要技术。移动边缘计算环境复杂多变，涉及众多设备和多样化的应用需求，同时网络条件也在不断变动。传统资源分配和任务调度方法基于固定模型优化，难以应对环境的迅速变化。而 DRL 通过与环境的实时互动学习最佳策略，可在未知和变化的环境中持续优化，为决策提供更灵活高效的支持。在 MEC 领域，采用深度强化学习方法进行资源优化和任务调度，被视为解决传统方法难以克服的问题的有力工具，在一定程度上突破了传统强化学习中存在的瓶颈，受到了广泛的关注。如何利用 DRL 为终端设备做出更加合理有效的卸载决策，已经成为一个值得深入探究的问题。

1.2 国内外研究现状

近年来，随着移动边缘计算技术的迅速发展和广泛应用，提升任务卸载效率成为学术界和工业界研究的热点。在不同的 MEC 应用场景中，如何实现任务卸载的高性能运行非常关键。为此，很多国内外的研究者投入了大量的精力来探索和研究高效的任务卸载策略及资源配置方案。研究者主要集中在三个优化目标上：最小化任务执行的时延、最小化系统的能耗以及找到一个在时延和能耗之间的最优平衡点。每一个优化目标都对应一系列独特的挑战和解决方案，需要从不同的技术角度进行探索。本节内容将从这三个方面对移动边缘技术的国内外研究现状进行介绍。

1.2.1 以最小化时延为目标的计算卸载技术

减少时延是用户极为重视的性能指标，学者们主要通过智能的任务调度和算法优化，减少在任务处理和数据传输过程中的总体时延。这包括算法的优化，以及硬件和软件层面上的协同工作，以确保任务可以在最短的时间内完成。Chen 等人[13]研究了在软件定义的超密集网络环境中，移动边缘计算的任务卸载时延最小化问题。他们详细分析了影响任务卸载性能的关键因素，包括网络延时、数据传输速率和能耗等，并提出了一种优化算法来寻找最佳的任务卸载和资源分配策略。通过模拟实验验证了所提算法的有效性，并实现了系统延迟最小化。文献[14]主要研究了车辆网络中云辅助的移动边缘计算（MEC）的计算卸载和资源分配问题。

文章提出了一种名为协同计算分流与资源分配优化（Collaborative Computation Offloading and Resource Allocation Optimization, CCORAO）的算法，该算法针对车辆网络中的计算密集型任务，通过协同利用边缘计算和云计算资源，提高了整个系统的计算资源利用率并降低了计算时延。

1.2.2 以最小化能耗为目标的计算卸载技术

在此目标下，研究的重点是如何降低 MEC 系统在执行任务过程中的能耗。这涉及到优化硬件设备的能源效率与设计低功耗的运行策略。 随着移动设备和云计算的普及，如何在保障通信与计算性能的前提下，有效降低能源消耗成为了一个重要的研究领域。Zhang 等人[15]提出了一种考虑无线信道随机性的能源优化策略，通过动态调节移动设备与云服务器之间的数据传输和处理策略，来适应不断变化的信道条件，从而最小化整个系统的能源消耗。该方法基于随机编程理论，通过建立数学模型和算法来解决实际应用中的能源优化问题,能够有效降低在不确定的无线环境下进行移动云计算时的能源消耗，进而延长移动设备的电池寿命，提高云计算服务的可持续性。在协同边缘计算环境中，文献[16]提出了一个两级交替方法框架，在满足延迟约束的前提下，最小化所有移动用户的能耗。上层使用启发式算法来处理协作决策和卸载决策, 在多用户环境中实现公平和高效的资源共享；下层则利用深度强化学习来优化通信与计算资源的具体分配，通过不断地学习和调整策略，DRL 能够精细调控资源分配，以适应网络状态的实时变化，从而进一步减少能耗，提高系统的整体效率。在移动边缘计算中，通过边缘和云的协作，可以有效地处理大规模的数据和复杂的计算需求。于是 Chen 等人[17]提出了一个依赖感知的卸载策略，以确保在边缘和云资源之间有效地分配和执行任务。考虑到任务之间的依赖性，这种策略将整个卸载决策问题划分为两个不同的合作模式即边云协作和边边协作。并设计了两种不同的贪婪算法实现了最小化网络通信消耗与最大化计算资源的使用效率。

1.2.3 时延和能耗间的权衡的计算卸载技术

时延和能耗都是用户关注的目标，在不同的使用场合，用户对时延和能耗的需求各不相同。在这种情况下，需要找到一种在时延和能耗之间进行有效权衡的方法，确保系统的整体消耗维持在一个较低的水准。因此，在移动边缘计算（MEC）系统中实现时延与能耗之间的平衡具有实际的意义。Tran 等人[18]在研究中以降低任务执行延迟和减少用户能耗为目标，探讨了联合任务卸载与资源分配的问题。他们将面临的主要问题拆分为任务卸载与资源分配两个部分，并分别采用相应的优化策略进行处理。这些方法的应用显著缩短了任务执行时间，并有效降低了能源消耗。在移动边缘计算环境中，由于各种应用对于延迟和带宽具有不同的

要求，如何合理分配计算和网络资源变得尤为重要。文献[19]探讨了在核心网络、边缘层和设备层之间有效分配资源的问题，提出了一个三层的资源分配框架，旨在实现网络资源的高效利用和流量的合理调度，优化整体网络性能并减少延迟。

在移动边缘计算（MEC）系统中，实现计算卸载的时延与能耗之间的权衡优化是一项复杂任务，因为需要在多目标优化的框架下考虑众多环境约束。深度强化学习提供了一种有效的解决方案来应对这种复杂性。Chen 等人[20]通过深度强化学习技术，建立了一个智能的决策框架以实现计算任务的有效卸载。该研究提出的模型能够自动学习环境中的动态变化，并据此做出最佳的卸载决策，以达到降低延迟和节省能耗的双重目标。Meng 等人[21]使用机器学习技术来提高移动边缘计算中的任务卸载效率。他们采用改进的 Q 学习和深度学习算法，用随机性策略来应对无线网络环境的不确定性，通过机器学习模型预测最优的卸载决策，实现了能耗和时延的加权和最低。文献[22]对超密集网络环境中的异构任务卸载问题进行了深入探讨，并提出了一种基于 Double Deep Q-Network (DDQN) 的在线卸载方法。通过这种方式，系统不仅能决定哪些任务应在本地执行，哪些应卸载到边缘服务器，还能决定在什么条件下调整设备的 CPU 频率和发射功率，以实现整体能耗和延迟的最小化。

1.3 总体技术方案

在移动边缘计算的应用场景中，计算任务的卸载决策尤为关键。传统优化方法在这里通常只能提供近似解，并且往往需要预先获得环境的统计信息。深度强化学习方法能在缺乏环境统计信息的未知环境中仅依靠从环境中获取的实时观察不断调整其策略，并通过长期的动态探索和训练，有效地学习到实现最佳长期目标的策略。本课题构建了一个输入多输出（Multiple Input Multiple Output，MIMO）的多用户 MEC 系统，建立起长期平均计算成本最小化问题的计算模型，通过深度强化学习方法来优化每个用户的本地执行和计算卸载能力。引入了基于连续动作空间的深度确定性策略梯度的深度强化学习方法，构建了基于 DDPG 算法的分散式动态计算卸载框架，使每个用户仅从局部观测中独立学习有效的卸载策略，以实现连续域内的动态功率分配。通过数值仿真说明了本文所提出的策略学习的分散策略和功率延迟权衡的性能，证明了该策略的连续功率控制优于其他基线算法。

1.4 技术方案的经济因素分析

本文采用了基于深度确定性策略梯度的深度强化学习方法，通过这种方法，可以在 MIMO 支持的多用户 MEC 系统中更好地处理长期平均计算成本最小化问题，从长远角度考虑了计算卸载的成本效益，可以帮助减少整体系统的能耗，提高资源利用率，降低运营成本，具有重要的经济效益。

1.5 论文章节安排

本文共分为四章，各章节的内容如下：

第一章为绪论，系统介绍了移动边缘计算技术的发展趋势，从三个角度介绍了国内外移动边缘计算技术研究现状，探讨了基于深度强化学习的移动边缘计算技术的研究价值。

第二章为相关理论基础，先介绍了移动边缘计算技术方面的相关理论，包括基本框架以及计算卸载方面的理论基础。然后围绕深度强化学习，依次介绍了相关算法内容，为下面章节的研究内容做好准备。

第三章为基于 DDPG 框架的移动边缘计算系统的方法研究。模拟了一个支持多输入多输出（MIMO）的 MEC 系统，给出了相关系统模型，并制定了系统中每个用户的长期平均计算成本最小化问题。然后构建了基于 DDPG 算法的分散式动态计算卸载框架以及算法伪代码。

第四章为实验与分析，进行了仿真设置，通过实验数值模拟验证了所提出的计算卸载策略优于传统的深度强化学习方案以及其他基线算法。

2 相关技术与理论

本章节将详细介绍与本文研究工作相关的基础理论知识，主要内容为移动边缘计算与深度强化学习两个领域的一些理论基础，为下文算法的设计与实现打好基础。首先介绍了移动边缘计算和任务卸载技术，并对实现任务卸载的相关理论和方法做了介绍。然后，介绍了深度强化学习的相关知识以及在边缘计算领域中常用的深度强化学习方法的理论和实现。

2.1 移动边缘计算技术

移动边缘计算技术是一种开放、分散的网络架构优化手段，它在无线接入网中融合了网络、计算、存储和应用的核心部分。该技术旨在将这些功能部署在靠近移动终端或数据源的地方，以便更高效地在下行和上行链路上处理云服务与移动数据源之间的数据传输。这样，边缘网络能变得更智能、更高效，从而确保实时通信的顺畅。简单来说，移动边缘计算通过在网络边缘提供强大的计算和智能网络服务，大幅降低了从终端设备到服务平台的处理和传输延迟。这种技术满足了物联网时代多种新兴数字化行业在实时通信、敏捷连接、性能优化、智能服务、数据安全和隐私保护等多方面的关键需求[23]。以下是一些移动边缘技术的优势特点：

（1） 低延迟: 通过在数据产生地点近处处理数据，MEC 可以显著降低通讯延迟。

（2） 带宽节约: 大量数据在本地处理和存储，减少了跨网络传输的数据量，从而节省带宽。

（3） 隐私和安全增强: 数据在本地处理，可以减少敏感数据的远程传输，增强数据安全和隐私保护。

（4） 高效的资源利用: 根据地理位置和需求动态调配边缘资源，从而优化资源利用。

作为一种新兴的计算模式，移动边缘计算有效解决了终端设备资源有限的问题。与传统的云计算相比，MEC 把计算和存储资源从云端中心服务器迁移到接近移动设备的边缘网络，显著缩短了资源中心与终端设备间的距离。这种架构配置使得边缘服务器具备云端相似的资源能力，包括计算、通信和存储能力。这不仅极大地降低了应用的响应延时，提升了实时性和移动设备使用时长，还提高了终端的计算性能和用户体验，向终端设备提供了更迅速、可靠且高质量的服务。同时，它也减轻了云端核心网络的负担。对于那些对延迟和能耗有特定需求的新兴应用，MEC 能够实施传统云计算难以支持的应用程序。具体而言，移动设备能够通过无线网络，把任务转移到附近的边缘节点服务器上执行，从而获得所需的计算资源。这种方式不仅能满足高资源需求的复杂计算任务，还能有效延长设备的电池寿命。与使用云端资源相比，移动边缘计算的架构可以缩短数据传输距离，减少时间延迟，并降低数据在长途传输中泄露的风险。MEC 的系统架构从上到下依次分为云端、边缘端和终端，其具体结构在图

2-1 中展示。

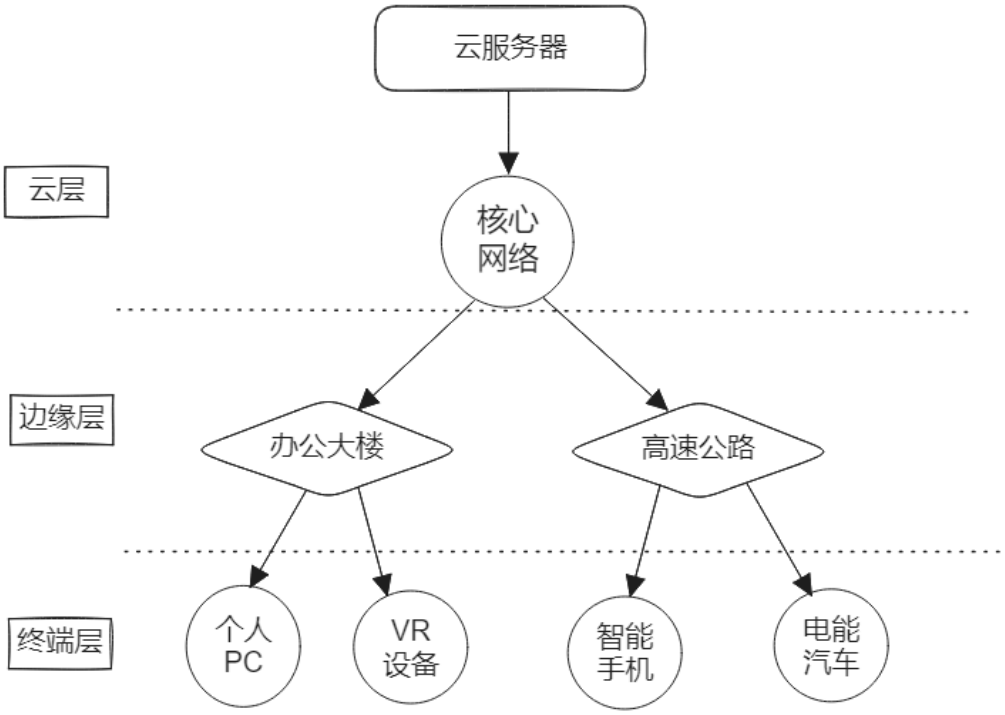

图 2-1 移动边缘计算体系架构

（1）云层：云服务器配备了丰富的计算能力，能够满足各类应用程序对于资源的高要求。此外，通过网络互联，云层能够支持边缘层节点之间的互动，实现数据共享及任务协同处理。然而，由于云端的计算资源通常位于离终端用户较远的位置，处理对时间反应极为敏感的任务时，可能无法确保服务的即时性，这往往导致较高的延时成本。

（2）边缘层：边缘层主要构成了配备了丰富资源的服务器，这些服务器位于用户设备的周边地区，能够满足用户对延迟、能耗和可靠性等方面的要求。由于 MEC 服务器物理位置上更靠近用户端，它能有效减轻主干网络的拥堵状况，减少数据处理和任务计算的延迟，同时也有助于降低移动设备的能耗。这种布置加强了网络的整体效能，提升了用户体验。

（3）终端层：终端层主要由具备一定计算能力的移动设备组成，这包括智能手机、电动汽车、个人电脑等各类智能设备。虽然这些设备搭载了一定的计算资源，但对于一些资源需求较高的新兴应用而言，它们的计算能力还是不够的。在这种情况下，MEC 服务器就能够提供所需的额外计算资源，以支持这些应用的运行。

2.2 计算卸载技术

2.2.1 计算卸载基本介绍

移动边缘计算技术是否能发挥出应有的作用，首先要解决的难题便是决定计算任务卸载的问题。计算卸载是移动边缘计算的一项关键技术，它可以把终端设备的计算任务转移至

MEC 服务器进行处理[24]。在卸载决策过程中，用户需在考虑了时延、能量消耗等多个因素后，评估是否该将待办的计算任务转移至边缘服务器上完成，或是保留在本地 CPU 中处理。任务卸载的精髓就在于评估并决定哪些任务应该被卸载以及卸载的程度，旨在优化终端设备与 MEC 服务器间的计算资源使用效率。具体来说，计算卸载机制使得移动设备能够将计算任务转移到其通信范围内的 MEC 服务器上。通过利用上行无线链路，这些任务被成功转移至 MEC 服务器进行处理，不仅加速了计算过程，同时也减少了移动设备运行应用时所需的能量消耗。而其中卸载决策则直接影响了 MEC 是否能达到最佳的服务质量和能源利用效率。计算卸载具体流程一般可分为六步[25]：1）节点感知 2）任务划分 3）卸载决策 4）程序传输 5）执行计算 6）计算结果返回。其中计算卸载具体流程图如图 2-2 所示：

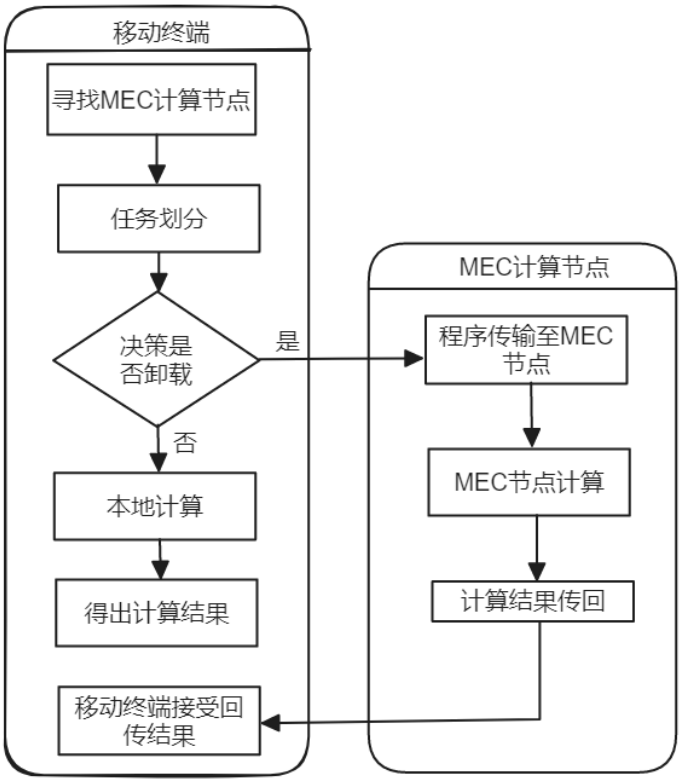

图 2-2 计算卸载流程图

2.2.2 计算任务卸载类型

对于移动边缘计算系统中的计算卸载决策问题，需要考虑多方面的因素，如延迟敏感性、计算需求与资源可用性、网络状况、整体能耗、安全与隐私等。而进行合理和高效的计算卸载决策对于实现移动边缘计算系统的高效运行、提升用户体验、节能降耗和确保数据的安全性等方面具有重要意义。对于上述问题，我们可以根据几个标准对应用程序进行分类，如图 2-3 所示。

（1）0-1 卸载：0-1 计算卸载涉及两个方面：一是完全依赖本地设备处理器来完成任务，不进行任何卸载；二是将全部计算任务通过无线网络传送至移动边缘计算服务器进行处理。

这种卸载决策可以用二进制变量 0 和 1 来表示，0 意味着任务在本地完成，而 1 则表示任务被卸载到服务器。当基站端的 MEC 服务器计算资源不足，或者卸载任务到服务器会消耗过多能源和时间时，任务就只能在本地设备上处理。

（2）部分卸载：在任务处理过程中，若代码或数据可被细分为更小的子任务，部分卸载策略常被采用。这种策略允许用户根据自身所持有的计算资源情况，灵活地决定将哪些子任务交给移动边缘计算服务器处理，而哪些任务则保留在本地设备上执行。通过这种方式，本地设备和服务器能够并行地处理各自分配的任务，从而有效提升整体处理效率和响应速度。

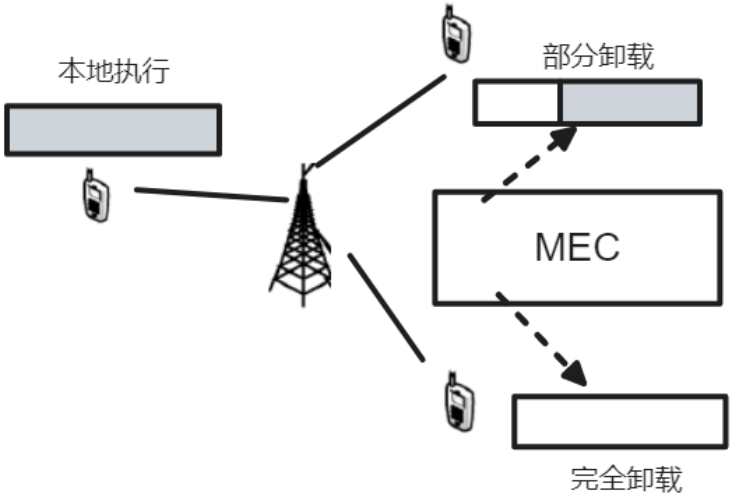

图 2-3 计算卸载方式

2.3 深度学习

深度学习(Deep Learning, DL)，由 Hinton 等人[26]于 2006 年提出，是机器学习(Machine Learning, ML)的一个新领域，它通过使用类似于人脑的大规模神经网络结构处理和模型化大量数据。深度学习以神经网络为基础，网络中包含多个隐藏层。深度神经网络的核心概念在于通过多层次的网络结构对目标进行层次化表征，通过这些多层网络提取出的高级特征来捕捉数据的抽象语义，从而增强特征的鲁棒性。深度学习的实现涉及到许多技术和方法，主要包括下面概的一些基础和关键步骤：

（1） 选择网络架构：这涉及模型的设计，包括输入、隐藏和输出层的设定，以及层与层之间的连接。

（2） 数据预处理：这是深度学习流程中的关键环节，包括规范化、标准化和数据增强等技术，以优化数据，增强其适应性，进而提升模型的性能和稳定性。

（3） 选择优化算法：这些算法用于调整网络参数，旨在减少模型的损失。通过迭代更新网络权重，达到最小化预测误差的目标。

（4） 定义损失函数：它用于衡量模型预测与真实值之间的差距，为模型学习提供方向指引。

（5） 模型训练和评估：模通常将数据划分为训练集和验证集，通过多轮训练来精细调整网络参数。

（6） 调参和正则化：这涉及选择超参数，如学习率、批大小和层数等。同时，利用正则化技术来降低过拟合风险，增强模型的泛化能力。

（7） 测试和部署：模型训练和验证后，需在独立测试集上评估其性能。之后，可将训练好的模型部署到生产环境，执行实际的预测或分类任务。

2.4 强化学习

强化学习[27]是人工智能领域中的一种学习方式，其核心思想是通过一系列的试错过程，让智能体逐步学习如何在一个复杂的环境中进行最优的决策。智能体与环境互动，根据获得的奖励信号调整行动策略，以在长期内获得最大的累积奖励。

马尔可夫决策过程（MDP）[28]是一种重要的数学框架，用于刻画智能体在与环境交互时进行连续决策的情景。为了简化描述，智能体与环境的交互可以视作一系列离散的时间步骤。智能体首先感知到环境的初始状态s_0，并据此选择一个动作a_0执行。环境随之变化到新的状态s_1，并即时给予智能体一个奖励信号r_1。接着，智能体再根据新的环境状态s_1选择下一个动作a_1，环境又会产生相应的变化s_2，并再次给出奖励r_2。这个过程会不断重复进行：

$$s_0, a_0, s_1, r_1, a_1, ..., s_{t-1}, r_{t-1}, a_{t-1}, s_t, r_t, ...,$$

其中$r_t = r(s_{t-1}, a_{t-1}, s_t)$ 第 t 时刻的即时奖励。这个交互过程就可以被视为一个马尔可夫决策过程。这种模型因其在机器学习，特别是强化学习中的基础性地位而备受瞩目。基于马尔可夫决策过程，强化学习的基本框架可以如图 2-4 所示。

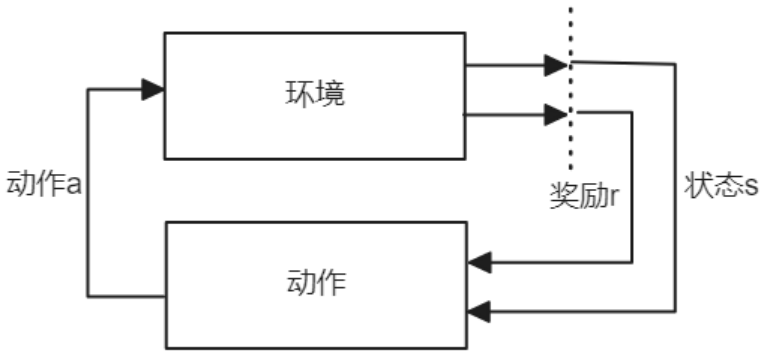

图 2-4 强化学习基本框架

强化学习的基本框架包括以下几个要素[29]：

（1） 智能体：智能体是强化学习系统中决策的主体。它观察环境的状态，并根据策略来选择行动，以实现特定目标。

（2） 环境：环境是智能体所处并与之交互的系统或问题域。环境接收智能体的行动，并反馈新的状态及奖励。

（3） 状态：状态是对环境在某一时刻的描述。它是智能体决策的基础，智能体根据状态来选择最合适的行动。

（4） 行动：行动是智能体在特定状态下可以选择进行的操作。智能体的行动会影响环境，进而改变未来的状态和奖励。

（5） 奖励：奖励是环境对智能体行动的即时反馈，用于评价该行动的效果好坏。智能体的目标是通过其行动累积尽可能多的奖励。

（6） 策略：策略是从状态到行动的映射，指导智能体在给定状态下应选择哪个行动。

2.5 深度强化学习

2.5.1 背景介绍

深度强化学习融合了强化学习与深度学习的优势，形成了一种创新技术。在这种技术中，强化学习被用来明确问题和设定优化目标，而深度学习则被用于处理状态表征、策略表达以及值函数建模等任务。强化学习的核心在于训练智能体通过与环境互动来最大化其长期奖惩值，在近来大数据和深度学习快速发展的时代下，针对传统强化学习无法解决高维数据输入的问题，2013 年 Mnih V 等人首次将深度学习中的卷积神经网络[30]引入强化学习中，提出了 DQN（Deep Q Learning Network）[31]算法，至此对深度强化学习的研究开始涌现。深度强化学习利用深度神经网络训练算法，使其在复杂环境中做出决策，并通过与环境的交互来自我优化其行为，达到预定目标。这些网络能够处理高维的、结构化的数据，使得智能体能够在更复杂、更接近现实的环境中学习和做出决策。深度强化学习已经在多个领域显示其巨大潜力，包括但不限于自动驾驶、机器人导航、游戏、以及资源管理等。

2.5.2 详细介绍

DRL 的标准设置与传统的 RL 一致，它由一个代理、一个环境 E、一组可能状态 S、一组可用动作 A 和一个奖励函数 r: S ×A→r 组成。在离散的时间步长中，代理通过与环境的持续交互来学习和制定决策。在每个步骤 t 中，代理观察到环境的当前状态 $s_t \in S$，并根据策略 π 在动作 A 中选择并执行一个动作。然后，代理将收到一个标量奖励 $r_t = r(s_t, a_t) \in R \subseteq ¡$，并根据环境的转移概率 $p(s_{t+1} \mid s_t, a_t)$ 过渡到下一个状态 s_{t+1}。由此可知，环境 E 的状态转移取决于代理在当前状态 s_t 下执行的动作。为了找到最优策略，我们将某一状态的收益定义为未来的

折现奖励之和，记为$R_t=\sum_{i=t}^{T}\gamma^{i-t}r(s_i,a_i)$。其中 t→∞表示总时间步数，$\gamma\in[0,1]$表示折现因子。RL 的目标是找到从开始分配中最大化长期预期折现奖励的策略，即:

$$J=\mathbf{E}_{s_i\sim E,a_i\sim\pi}[R_1] \tag{2.1}$$

在策略 π 下，也将状态-行为函数$Q^{\pi}(\mathrm{s}_t,a_t):S\times A\longrightarrow R$称为批评函数，定义为从状态$s_t$开始，选择动作$a_t$ 的期望折现收益，即

$$Q^{\pi}(\mathrm{s}_t,a_t)=\mathbf{E}_{s_{i>t}\sim E,a_{i>t}\sim\pi}[R_t(s_t,a_t)] \tag{2.2}$$

其中期望大于转移概率$p(s_{t+1}\mid s_t,a_t)$和策略 π。

环境 E 的动态变化没有一个明确的模型，也就是说，潜在的转移概率$p(s_{t+1}\mid s_t,a_t)$是未知的，甚至是非平稳的。但是根据证明[32]，在使预期折现奖励最大化的策略π^*下，状态-行为函数满足以下 Bellman 最优性方程:

$$Q^{*}(\mathrm{s}_t,a_t)=\mathbf{E}_{s_{t+1}\sim E}[r(s_t,a_t)+\gamma\max_{a_{t+1}}Q^{*}(s_{t+1},a_{t+1})] \tag{2.3}$$

其中，最优策略π^*可以通过在任意状态$s\in S$下选择相应行动而得到，如式（2.4）:

$$\pi^{*}(s)=\arg\max_{a\in A}Q^{*}(s,a) \tag{2.4}$$

为了从原始经验中学习状态-动作函数的值，可以利用时间差分(temporal-difference, TD)方法在每个时间步长 t 上从对应代理的经验元组(s_t,a_t,r_t,s_{t+1})中更新状态-动作函数，即

$$Q(s_t,a_t)\longleftarrow Q(s_t,a_t)+\alpha[r(s_t,a_t)+\gamma\max_{a_{t+1}}Q(s_{t+1},a_{t+1})-Q(s_t,a_t)] \tag{2.5}$$

其中 α 为学习率，$\delta_t=r(s_t,a_t)+\gamma\max_{a_{t+1}}Q(s_{t+1},a_{t+1})-Q(s_t,a_t)$称为 TD 误差。熟知的 Q-learning 算法[33]基于式(2.5)，因此状态-作用函数也称为 Q 值。此外，Q-learning 算法以 1[32]的概率收敛，这表明最终将得到一个预估计的$Q^{*}(\mathrm{s}_t,a_t)$。

为了增强学习的稳定性，它还引入了一个经验重放缓冲区 B 来存储代理在每个时间步长 t 的经验元组$e_t=(s_t,a_t,r_t,s_{t+1})$。具体而言，对于每个时隙，将从 B 中均匀随机抽取一小批样本$(s,a,r,s')\sim U(\beta)$，来计算以下损失函数:

$$L(\theta^{Q})=\mathbf{E}_{(S,A,R,S')\sim U(\beta)}[(r+\gamma\max_{\hat{a}\in A}Q(s',\hat{a}\mid\theta^{Q'})-Q(s,a\mid\theta^{Q})^2] \tag{2.6}$$

这种损失函数将用于更新网络参数，即 $\theta^{Q}\longleftarrow\theta^{Q}\longleftarrow\alpha_Q\cdot\nabla_{\theta^{Q}}L(\theta^{Q})$。为了进一步提高稳定性，DQN 采用了软更新策略，利用式(2.6)中的目标网络θ^{Q}来推导代理的 TD 误差。目标网络通过$\theta^{Q'}\longleftarrow\tau\theta^{Q}+(1-\tau)\theta^{Q'}$跟踪所学网络的权重，其中$\tau=1$。此外，根据式(2.4)，

代理在每个时间步 t 所采取的动作是通过 $a_t = \arg\max_{a \in A} Q^*(s_t, a \mid \theta^Q)$ 获得的。

2.5.3 DQN 算法

DQN 算法巧妙地结合了深度学习的神经网络技术与基于马尔可夫决策过程的 Q 学习理念。在此算法框架下，环境的状态信息被输入到神经网络中，而网络的输出则对应着各个可能行动的 Q 值。借助深度神经网络的强大表征能力，智能体得以对不同行动的效果进行评估，并据此选择出最佳行动策略。这种融合方法不仅提升了智能体的决策能力，也为解决复杂环境下的学习问题提供了新的思路。DQN 算法的具体步骤如图 2-5 所示，这些过程共同作用，允许智能体通过与环境的互动来学习最优行动策略：

（1） 初始化经验回放池：DQN 算法采用经验回放池来储存智能体的经历，涵盖状态、行动、奖励及后续状态信息。此举旨在训练时从池中随机抽取以往经验，以降低训练样本间的关联性，并增强学习的稳固性。

（2） 初始化深度神经网络与目标网络：构建两个结构相同但参数独立的神经网络，分别用于预测 Q 值和计算目标 Q 值。目标网络的参数会定期或平滑地更新为评估网络的参数。

（3） 观察初始状态：智能体观察环境的初始状态，并作为神经网络的输入。

（4） 选择并执行行动：根据当前策略，智能体选择并执行行动。

（5） 收集样本并存入经验回放池：智能体行动执行后，环境会反馈新状态与奖励等信息。此信息将与当前状态、选定行动及新状态共同作为样本存入经验回放池。

（6） 随机抽取经验批次进行学习：从经验回放池中随机选取一批经验，利用这些经验来更新评估网络的参数。此过程中，会通过目标网络为下一个状态预估 Q 值，并结合奖励与折扣因子 γ 进行调整，从而设定评估网络的更新目标。

（7） 更新评估网络：通过损失函数衡量评估网络输出的 Q 值与计算得出的目标 Q 值之间的差距，并利用优化算法更新评估网络的权重，以缩小这一差距。

（8） 定期更新目标网络：在设定的时间间隔后，将评估网络的参数同步至目标网络，以确保学习过程的稳定性。

（9） 重复步骤 4 至 8，直至学习过程结束。

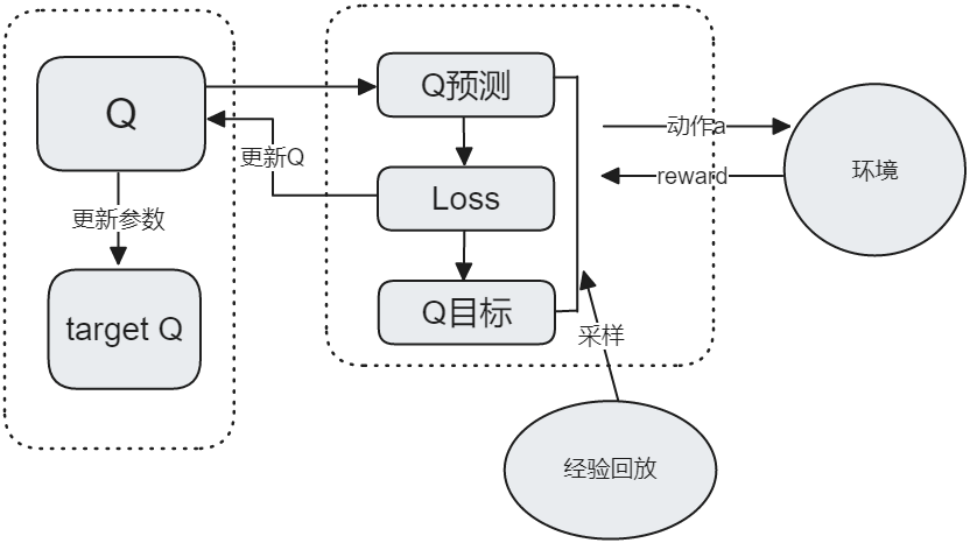

图 2-5 DQN 算法框架图

其中，DQN 算法的成功之处主要在于引入了 3 项核心技术：

（1） 经验回放：经验回放是 DQN 算法中的一项创新技术，它的主要作用是消除样本间的相关性并提升学习效率。在智能体与环境交互时，所获得的经验样本会被保存在经验回放池中。而当训练网络时，将从这个池中随机选择一小批转移样本。

（2） 固定的目标网络：在深度学习训练中，模型参数不断更新，如果在 Q 学习中使用相同的网络来即时更新目标值，学习过程会非常不稳定。DQN 算法通过引入一个目标网络来解决这一问题。目标网络的结构与主训练网络相同但参数更新较慢，目标 Q 值是由一个相对静态的网络生成的。这样可以提供一个稳定的目标，从而使训练过程更加稳定。

（3） ϵ-贪婪策略：在 DQN 中，ϵ-贪婪策略被用来平衡探索和利用。这意味着智能体在大部分时间内选择当前估计最优的行动，但也有一定比例的时间会随机选择行动，以探索可能的未知的有利行动。这有助于智能体在学习初期探索环境，在学习后期则更多地利用已学到的策略。

2.5.4 DDPG 算法

DDPG 是一种融合了深度学习和策略梯度技术的深度强化学习算法，专门针对连续动作空间的问题。该算法的核心是基于确定性策略梯度定理，这个定理指出在连续动作空间的马尔可夫决策过程中，对于每个状态只对应一个确定动作的确定性策略，我们可以精确地计算出其策略梯度，并且这个梯度的更新方向与最优策略是一致的。

DDPG 算法采用了演员-评论家（Actor-Critic）的架构，其中策略网络和动作值函数都是通过深度神经网络来近似的。策略网络和价值网络的参数则是通过随机梯度下降法来进行更新的。

演员网络负责从给定的状态出发，映射到一个连续的动作空间，即输出一个具体的动作，这个动作是基于当前策略网络的输出，策略网络通常实现为一个或多个深度神经网络；评论家网络则评估当前演员网络输出的动作的价值。它通过学习状态-动作对的值函数来实现，这个值函数也是通过深度神经网络来近似的。

演员网络通过策略梯度方法更新。策略梯度是根据评论家网络提供的状态-动作对的值函数的梯度来计算的。也就是说更新的方向是增加获得高奖励动作的概率的方向。

评论家网络的更新则是根据均方误差损失函数来进行，即试图缩小预测的值函数和通过 Bellman 方程计算得到的目标值之间的差距。

和之前介绍的 DQN 算法一样，DDPG 算法也引入了三个关键技术:

（1） 经验回放：DDPG 也引入了 DQN 中的经验回放机制。此机制保存之前的转移数据，并在训练时从这些存储的经验中随机抽取样本来更新网络。这样做可以消除数据之间的相关性，提升学习的稳定性。

（2） 目标网络：DDPG 为演员和评论家网络都设置了对应的目标网络。这些目标网络的参数并非即时更新，而是逐渐追踪主网络的参数。这种渐进的更新方式有助于防止学习过程中的波动和不稳定。

（3） 噪声探索：由于确定性策略产生的动作是确定的，缺乏对环境的探索性。因此，在训练时，我们向演员网络的输出动作中加入噪声，以增强智能体的探索能力。

3 基于深度强化学习的移动边缘计算方法研究

本章将在第二章的相关理论基础上提出去中心化动态计算卸载策略。在本章中将先给出总体模型架构；进而给出系统架构具体模型，包括系统模型，网络模型，计算模型；然后，将构建基于深度强化学习的去中心化动态计算卸载框架，给出相应状态空间、动作空间和奖励函数。最后，还给出了该框架具体算法实现。

3.1 模型架构

系统模型架构由三部分组成：MEC 网络、MEC 环境和 DRL 代理，如图 3-1 所示。智能体首先会基于环境观察来决定行动。环境随后会给出反馈，并显示下一个状态给 DRL 代理。代理会把当前的交互数据，包括当前状态、动作、奖励和接下来的状态，存入经验回放缓冲区以供模型训练。通过与环境的持续交互，DRL 代理不断生成训练数据集。接着，智能体会从这个缓冲区中提取数据，用以训练 DRL 模型内部的学习网络。最终，DRL 代理会指导用户完成计算卸载过程。

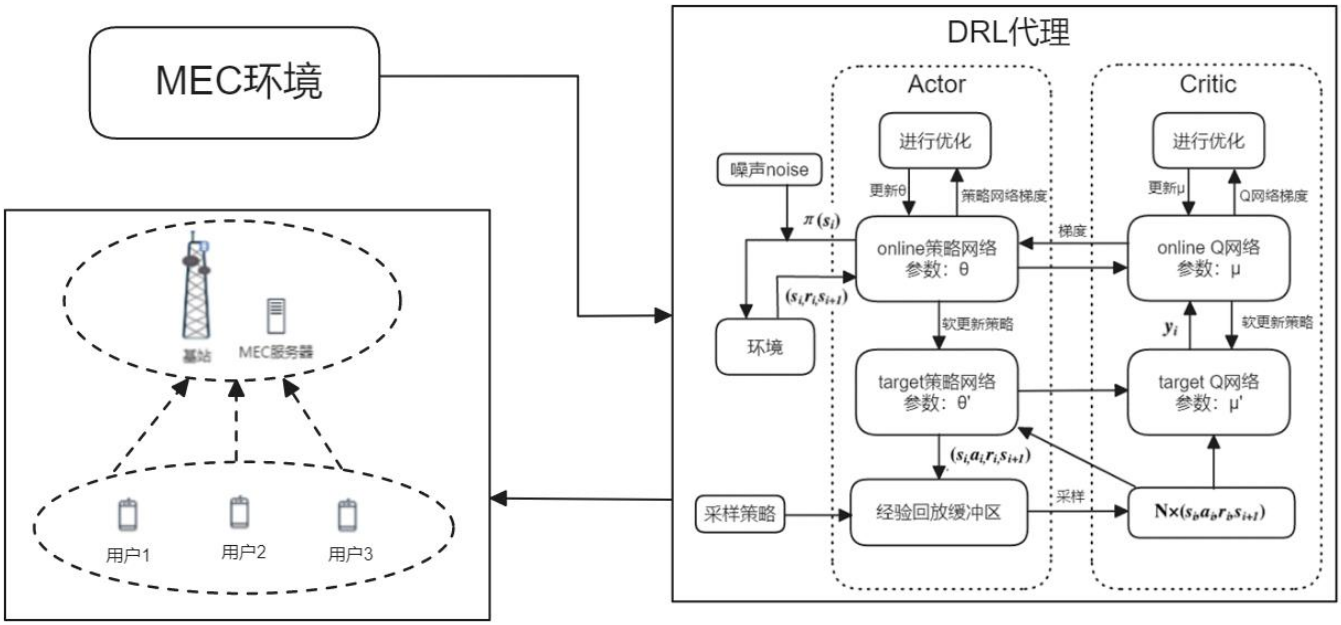

图 3-1 模型架构图

3.2 系统模型

本文将构建一个支持多输入多输出（MIMO）的多用户 MEC 系统，该系统由 n 天线基站（BS）, MEC 服务器和一组单天线移动用户 M ={1,2，…，m}组成。给定移动设备上有限的计算资源，每个用户 m∈M 都有计算密集型的任务需要完成。为了提高用户的计算体验，MEC 服务器部署在基站附近，用户可以通过无线链路[36]将部分计算任务卸载到 MEC 服务器。本

系统采用离散时间模型，操作周期被划分为等长时间槽，并以 T ={0,1，…}索引。对于每个插槽 t∈T，每个用户的信道条件和任务到达都是不同的，因此需要恰当选择本地执行与计算卸载的比例，以平衡平均能耗和任务处理延迟。此外，随着移动用户数量的增加，采用分散的任务调度能够降低用户与 MEC 服务器间的系统开销，提升系统的可扩展性。在接下来的部分中，将详细介绍该系统的组网建模和计算模型。

3.2.1 网络模型

对于每个槽位 t，若每个移动用户$m \in M$的信道向量用$h_m(t) \in C^{N\times 1}$表示，则上行链路中所有用户接收到的基站信号为

$$y(t)=\sum_{m=1}^{M} h_m(t)\sqrt{p_{o,m}(t)}s_m(t)+n(t) \tag{3.1}$$

其中，$p_{o,m}(t)\in[0,p_{o,m}]$为用户 m 以 $p_{o,m}$ 卸载任务数据位的传输功率，m 为最大值，$s_m(t)$为单位方差的复数据符号，$n(t)\sim CN(0,\sigma_R^2 I_N)$为方差为$\sigma_R^2$的加性高斯白噪声(AWGN)向量。$I_N$ 表示一个 N×N 单位矩阵。为了表征每个移动用户 m 的时隙之间的时间相关性，采用以下高斯-马尔可夫块衰落自回归模型[37]:

$$h_m(t)=\rho_m h_m(t-1)+\sqrt{1-\rho_m^2}\,e(t) \tag{3.2}$$

其中 ρ_m 为槽 t 与 t−1 之间的归一化信道相关系数，误差向量 e(t)为复高斯，与 hm(t)不相关。根据 Jake 的衰落谱，$\rho_m=J_0(2\pi f_{d,m}\tau_0)$，其中 $f_{d,m}$ 为用户 m 的多普勒频率，t_0 为槽长度，$J_0(\cdot)$为第一类的贝塞尔函数[38]。

$H(t)=[h_1(t),...,h_M(t)]$是 BS 与所有用户之间的 N×M 通道矩阵，BS 处 H 的线性零强迫(ZF)探测器可以用其伪逆表示为$H^*(t)=(H^H(t)H(t))^{-1}H^H(t)$。将$H^*(t)$的第 m 行记为$g_m^H(t)$用来 BS 检测用户 m 的信号。因此检测到的信号就可以被计算为

$$g_m^H(t)y(t)=\sqrt{p_{o,m}(t)}s_m(t)+g_m^H(t)n(t) \tag{3.3}$$

ZF 通过定义[39] $g_i^H(t)h_j(t)=\delta_{ij}$来检测。在这里计算时，当 i = j 时 $\delta_{ij}=1$，否则 $\delta_{ij}=0$。接着，就可以推导出 BS 处对应的信噪比(SINR)

$$\gamma_m(t)=\frac{p_{o,m}(t)}{\sigma_R^2\,\|\,g_m(t)\,\|^2} \tag{3.4}$$

$$=\frac{p_{o,m}(t)}{\sigma_R^2[(H^H(t)H(t))^{-1}]_{mm}} \tag{3.5}$$

上式中$[A]_{mn}$为矩阵 A 的第(m,n)个元素。由式(2.4)可以验证，随着用户 m 数量的增加，每个用户的 SINR 会降低，这通常会使每个用户在任务卸载过程中消耗更多的能源。在下文会展示用户如何从 SINR 反馈中学习和适应环境。

3.2.2 计算模型

假设槽位 t 在所有槽位上是独立且同分布的(i.i.d)，定义$a_m(t)$来量化用户 m 在槽位 t 期间的任务到达量(以 bit 为单位)。每个任务到达后，将首先存储在用户的排队缓冲区中，然后再从 t + 1 开始的即将到来的插槽中进行处理。假设计算应用是细粒度的[40]，那么在槽位 t 期间，一部分缓冲任务位(记为$d_{l,m}(t)$)将在移动设备上本地处理，另一部分缓冲任务位(记为$d_{o,m}(t)$)将被卸载到 MEC 服务器并由其执行。其中，所有被执行的任务位的数量，即$d_{l,m}(t)+d_{o,m}(t)$，不能超过任务缓冲区的队列长度。设$B_m(t)$为用户 m 的任务缓冲区槽位 t 开始时所处的队列长度，那么它的迭代过程如下

$$B_m(t+1)=[B_m(t)-(d_{l,m}(t)+d_{o,m}(t))]^+ + a_m(t), \forall t \in \mathrm{T} \tag{3.6}$$

其中$B_m(0)=0$且$[x]^+=\max(x,0)$ $[\mathrm{x}]^+=\max(\mathrm{x},0)$。

3.2.2.1 本地执行

对于用户 m，如果$p_{l,m}(t)\in[0,p_{l,m}(t)]$ 是在槽 t 上进行本地执行的能耗，则本地处理的任务比特数可以计算为

$$d_{l,m}(t)=\tau_0 f_m(t) L_m^{-1} \tag{3.7}$$

其中$p_{l,m}$ 为本地最大执行能力，$f_m(t)=\sqrt[3]{p_{l,m}(t)/k}$为使用 DVFS 技术[24]调节芯片电压调度的 CPU 频率。L_m表示处理一个任务位所需要的 CPU 周期数，可以通过离线测量[41]来估算。

3.2.2.2 边缘计算

MEC 服务器通常拥有充足的计算资源（例如高性能的多核 CPU），这使得多种应用的任务能够并行处理，几乎无需担忧处理延迟问题。这在处理像视频流分析或联网车辆服务这类应用时，小规模计算的输出反馈延迟也可以忽略。因此，所有经过基站转发到 MEC 服务器的任务数据都可以得到及时处理。在这种情况下，给定上行传输功率$p_{o,m}(t)$，用户 m 在槽位 t 期间的卸载数据位数可由下方公式得出

$$d_{o,m}(t)=\tau_0 w \log_2(1+\gamma_m(t)) \tag{3.8}$$

其中 W 为系统带宽，$\gamma_m(t)$ 为由式(3.4)得到的信噪比。

3.2.2.3 计算费用

为了平衡能源成本和任务延迟，每个用户代理的总体计算成本可以量化为能源消耗和任务缓冲延迟的加权总和。根据定理[42]，可以知道任务缓冲区的平均队列长度与缓冲延迟成正比。因此，每个用户 m 的长期平均计算成本可以定义为

$$C_m = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T} w_{m,1}(p_{l,m}(t)+p_{o,m}(t)) + w_{m,2}B_m(t) \tag{3.9}$$

其中 $w_{m,1}$ 和 $w_{m,2}$ 都是非负加权因子。通过为这些加权因子设置不同的值，就可以在能量消耗和动态计算卸载存在缓冲延迟之间进行权衡。

相对于在基站进行集中式决策，这种决策需要全面了解环境，采用去中心化决策能够有效利用局部信息来最小化每个用户的计算成本。这种方式通过在每个时隙 t 动态分配本地执行和计算卸载的任务，便可根据局部观察来独立优化计算策略，如下所示:

$$\min_{p_{l,m}(t),p_{o,m}(t)} C_m$$

$$\text{s. t.}$$

$$p_{l,m}(t)\in[0,p_{l,m}],\forall t\in T; \tag{3.10}$$

$$p_{o,m}(t)\in[0,p_{o,m}],\forall t\in T; \tag{3.11}$$

但是，在基站上收集所有用户的上行信道矢量、任务缓冲区队列长度等全局信息和每个用户在基站点的接收信噪比 SINR，然后分发给每个用户，会带来很高的系统开销和相当大的延迟，甚至会随着用户数量的增加而呈指数级增长。为此，可以假设每个用户 m 仅根据其对环境的局部观察做出去中心化决策。即在每个时隙 t 开始时，用户 m 将只获得到达任务位 $a_m(t)$ 来更新任务缓冲区，并以基站处的先前接收 SINR 即 $\gamma_m(t-1)$ 进行确认，并且能够利用信道交换信息估计当前信道向量 $h_m(t)$。这样，每个用户处的学习策略就可以独立于其他用户代理来自己决定 $p_{l,m}(t)$ 和 $p_{o,m}(t)$。

3.3 基于 DDPG 的动态计算卸载框架

传统的深度强化学习方法如 DQN，虽然可以有效地解决高维状态空间中的问题，但它们仅限于处理离散和低维的动作空间。为了克服这一限制，本文引入了 DDPG 算法，旨在将深度强化学习的应用范围扩展至连续的动作空间。DDPG 采用行动者-批判者框架，其中包括两

个独立的深度神经网络：一是批判者网络$Q(s,a|\theta^Q)$用来逼近 Q 值函数，类似于 DQN 中的价值网络；二是行动者网络$\mu(s|\theta^\mu)$，它负责将状态映射到一个特定的连续动作。具体来说，批判者网络$Q(s,a|\theta^Q)$与 DQN 相似，可以根据式(2.6)进行更新。另一方面，行动者网络$\mu(s|\theta^\mu)$确定性地将状态 s 映射到特定的连续动作。根据文献[43]的推导，行动者的策略梯度可以计算如下:

$$\nabla_{\theta^\mu} J \approx \mathbf{E}_{(s,a,r,s')\sim U(\beta)}[\nabla_a Q(s,a|\theta^Q)|_{a=\mu(s|\theta^\mu} \cdot \nabla_{\theta^\mu}\mu(s|\theta^\mu)] \tag{3.12}$$

它实际上是开始分布 J 相对于参与者参数θ^μ在抽样的小批量数据$U(\beta)$上的期望收益的平均梯度。这里应用了链式法则，因为动作$a_t=\mu(s|\theta^\mu)$被用作批判者函数的输入。因此，只要根据式(3.6)和式(3.7)，行动者和批判者的网络参数就可以分别通过$\theta^Q \longleftarrow\!\!- \theta^Q \longleftarrow\!\!- \alpha_Q \cdot \nabla_{\theta^Q} L(\theta^Q)$和$\theta^\mu \longleftarrow\!\!- \theta^\mu \longleftarrow\!\!- \alpha_\mu \cdot \nabla_{\theta^\mu} J$来更新。这里，$\alpha_Q$和$\alpha_\mu$是学习率。此外，DDPG 也应用了软更新策略，即使用θ_μ和θ_Q参数化的目标网络来计算与式(3.6)不同的损失函数：

$$L(\theta^Q)=\mathbf{E}_{(S,A,R,S')\sim U(\beta)}[(r+\gamma Q(s',\mu(s|\theta^{\mu'})|\theta^{Q'})-Q(s,a|\theta^Q)^2] \tag{3.13}$$

目标网络分别通过$\theta^{\mu'} \longleftarrow\!\!- \tau\theta^\mu+(1-\tau)\theta^{\mu'}$和$\theta^{Q'} \longleftarrow\!\!- \tau\theta^Q+(1-\tau)\theta^{Q'}$来跟踪所学习网络的权重，其中$\tau=1$。

为了最大限度地减少每个用户的计算成本，本文将提出一个基于 DDPG 的计算卸载框架来训练分散策略，以精细地控制本地执行和计算卸载的功率分配。利用这一框架，可以在一个去中心化的设置中，针对每个用户独立地优化本地执行和计算卸载的决策。该策略依赖于对环境的局部观察，允许用户在连续域内精确调整电力分配，从而最大限度地降低计算成本。下面会详细介绍框架中状态空间、动作空间和奖励函数的定义。

3.3.1 状态空间

在框架中，状态空间从每个用户 m 的角度定义了环境的表示，然后将它用于做出权力分配决策。对于去中心化的策略，每个用户的状态空间只依赖于局部观察。具体来说，在时隙 t 开始时，其数据缓冲区的队列长度$B_m(t)$将按照式(2.6)进行更新。同时，用户可以根据其最后一次接收 SINR $\gamma_m(t-1)$时收到 BS 的反馈。此外，还可以利用信道互易性来估算上行传输的信道向量$h_m(t)$ 。

为了根据上面提到的本地观察定义状态空间，首先可以利用$B_m(t)$来指示等待执行的任务数据位的数量。同时为了反映在 ZF 检测下其他移动用户上行信号干扰的影响，槽位 t 的归一

化 SINR 可以表示为:

$$\phi_m(t)=\frac{\gamma_m(t)\sigma_R^2}{p_{o,m}(t)\|h_m(t)\|^2} \tag{3.14}$$

$$=\frac{1}{\|h_m(t)\|^2[(H^H(t)H(t))^{-1}]_{mm}} \tag{3.15}$$

这可以表示在 BS 点 ZF 检测后的投影功率比。实际上，为了在没有流间干扰的情况下对用户 m 的符号进行解码，ZF 检测将接收到的信号 y(t)投影到与其他用户的信道向量[44]所张成的子空间正交的子空间中，这样就能消除其他用户对用户 m 的干扰。$\phi_m(t)$ 可以理解为用户 m 的归一化 SINR 表示，因为它排除了用户 m 的发射功率 $p_{o,m}(t)$ 和信道矢量 $h_m(t)$ 的影响，能够准确反映其他用户的干扰对用户 m 的单位接收功率的影响。最后的归一化 SINR $\phi_m(t-1)$ 可以由 $\gamma_m(t-1)$ 估计，然后包含在状态空间中。此外，信道质量可以通过信道向量 $h_m(t)$ 来量化。综上所述，从每个用户 m 的角度出发，槽位 t 的状态空间可以定义为

$$s_{m,t}=[B_m(t),\phi_m(t-1),h_m(t)] \tag{3.16}$$

3.3.2 动作空间

基于当前状态 $s_{m,t}$，每个用户代理 m 会根据观察到的系统状态，在每个时间槽 $t\in T$ 中选择一个包括本地执行和计算卸载的功率分配的动作 $a_{m,t}$，如下所示:

$$a_{m,t}=[p_{l,m}(t),p_{o,m}(t)] \tag{3.17}$$

DDPG 算法的一个显著优势是它允许在连续动作空间（即 $p_{l,m}(t)\in[0,p_{l,m}]$ 和 $p_{o,m}(t)\in[0,p_{o,m}]$）内进行精细的功率调整，而非局限于几个预设的离散功率水平。这种方法能有效地降低因高维离散行动空间带来的复杂性，从而实现对平均计算成本的最小化。通过这种方式，DDPG 不仅提升了优化的灵活性，也增强了策略的应对能力，能在动态环境中为用户提供更为精确的控制。

3.3.3 奖励函数

在 DDPG 框架中，每个用户代理的行动是由奖励机制驱动的[45]，这使得奖励函数成为驱动 DRL 算法性能的关键组成部分。为了使式(3.9)中定义的长期计算成本最小化，可以将每个用户在时间步长 t 之后收到的奖励函数 $r_{m,t}$ 定义为

$$r_{m,t}=-w_{m,1}\cdot(p_{l,m}(t)+p_{o,m}(t))-w_{m,2}\cdot B_m(t) \tag{3.18}$$

即总功耗的瞬时值与任务缓冲区队列长度的加权负和。

3.4 算法设计与实现

为了学习和评估学习到的动态计算卸载策略，本文构建了一个包含多组用户代理的模拟环境，以便进行系统的训练和测试。通过用户代理与模拟环境间的持续交互，我们能够有效地生成所需的训练和测试数据。在此过程中，模拟环境会接收并处理每个用户代理做出的决策，随后返回相应的局部观测值，以供进一步的分析和学习。其中交互将以随机初始状态 $s_{m,1}$ (每个用户 m)手动启动，并在每个集的预定义最大步数 T_{MAX} 处终止。具体算法流程如下所示：

算法 3.1 基于 DDPG 的动态计算卸载算法

1：**for each user agent** $m \in M$ **do:**

2： θ^Q 和 θ^μ 随机初始化 Critic 网络 $Q(s,a|\theta^Q)$ 和 Actor 网络 $\mu(s|\theta^\mu)$

3： $\theta^{Q'} \longleftarrow \theta^Q, \theta^{\mu'} \longleftarrow \theta^\mu$ 初始化目标网络权重参数 Q' 和 θ'

4： 初始化经验回放区 R

5：**end for**

6：**for episode=1,M do:**

7： 行动探索，随机噪声 N 初始化

8： 获得初始观察状态 $s_{m,1}$

9： **for t=1,T do:**

10： $a_t = \mu(s_t|\theta^\mu) + N_t$

11: 执行动作 a_t，得到奖励 r_t 和环境状态 s_{t+1}

数据 (s_t, a_t, r_t, s_{t+1}) 存入 R

12： 从 R 中随机采样批量数目值 N 的多维数组 (s_i, a_i, r_i, s_{i+1})

13: $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$

14: 最小化损失函数 L 来更新 Critic 网络：

15： $Loss = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$

16： 采样策略梯度更新 Actor 策略网络：

17： $\nabla_{\theta^\mu} J(\theta^\mu) \cong \frac{1}{N}\sum_i \nabla_{\theta^\mu}\mu(s_i \mid \theta^\mu)\nabla_a Q(s_i, a \mid \theta^Q)|_{a=\mu(s_i)}$

18： 更新目标网络：

19： $\theta^{Q'} \longleftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$;

20: $\theta^{\mu'} \longleftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$;

21: **end for**

22: **end for**

在本方法中，在一个事件的每个时间步 t，每个智能体的经验元组 (s_t, a_t, r_t, s_{t+1}) 将被存储在它自己的经验缓冲区 B_m 中。同时，智能体的行动者和评论家网络会利用从经验回放缓冲区 B_m 随机采样的一小批经验元组 (s_i, a_i, r_i, s_{i+1}) 来更新相应网络。过这种方式，在 K_{MAX} 集训练完成后，每个用户代理就能逐步学习如何独立执行动态计算卸载策略。为了平衡探索和利用，DDPG 在探索过程中向行动者网络加入了从随机噪声过程中采样的噪声 μ。在测试阶段，用户代理首先会加载其在训练阶段学到的行动者网络参数。接着，用户代理会从一个空的数据缓冲区开始，与一个随机初始化的环境交互。在这个特定的环境中，用户代理会依据行动者网络的输出来选定动作，同时将对环境的即时观察作为当前的状态信息。

4 实验与分析

本章说明了实验的工具及运行平台，给出了数值模拟来说明所提出的深度强化学习框架在系统中用于分散动态计算卸载的性能。在单用户和多用户场景下，分别演示了基于 DDPG 的框架与基于传统深度强化学习算法 DQN 的框架的 MEC 系统性能，具体展示了实验的数据和实验结果，并对实验结果进行了分析比较，证明了本文提出的方案可行性。

4.1 实验环境

4.1.1 硬件环境

CPU：8 核 Intel(R) Core(TM) i5-8500U CPU @ 1.60GHz 1.80 GHz

内存：8.00 GB

4.1.2 软件环境

操作系统：64 位 Windows 10

编程语言：Python

集成开发环境：Pycharm

4.1.3 仿真设置

在本文系统中，设置时间间隔为$\tau_0 = 1\text{ms}$。每个用户 m 处的任务到达过程遵循泊松分布$\mathrm{a}_m(t) \sim Pois(\lambda_m)$，其中$\lambda_m$的平均值称为任务到达率，即$\lambda_m = \mathrm{E}[a_m(t)]$。在每一轮次的开始，每个用户 m 的信道矢量初始化为$\mathrm{h}_m(t) \sim \mathrm{CN}(0, h_0(d_0/d_m)^{\alpha} I_N)$，其中路径损耗常数$h_0 = -30dB$，参考距离$d_0 = 1m$，路径损耗指数$\alpha = 3$，$d_m$为用户 m 到基站的距离，单位为米。系统的带宽固定为 1MHz，最大发射功率$P_{o,m} = 2W$，噪声功率$\sigma_R^2 = 10^{-9}W$。对于本地执行，设置$\kappa = 10^{-27}$，每比特所需 CPU 周期设为$L_m = 500$周期/比特，最大允许 CPU 周期频率$F_m = 1.26GHz$，这样本地执行所需的最大功率就为$P_{l,m} = 2W$。

为了应用 DDPG 算法，每个用户代理 m 的行动者和评论家网络均为一个具有两个隐藏层的四层全连接神经网络，隐藏层的神经元数量分别为 400 和 300。所有隐藏层使用 Relu 激活函数，即 $\mathrm{f(x)} = \max(0, \mathrm{x})$，而行动者的最终输出层采用 sigmoid 函数以界定动作，评论家网络在第二个隐藏层引入动作。本文将自适应矩估计方法应用于学习神经网络参数，演员和评论家的学习率分别为 0.0001 和 0.001。目标网络的软更新率为 $\tau = 0.001$。为了加强探索效果，设置了 $\theta = 0.15$，$\sigma = 0.12$ 两个变量来提供时间相关噪声。经验回放缓冲区大小设置为 $|B_m| = 2.5\times10^5$，一个批次大小设为 M = 16。为了更好地平衡探索，引入因子 $w_m \in [0,1]$，以

此将两个非负加权因子重新定义为 $w_{m,1}=10w_m$ 和 $w_{m,2}=1-w_m$。这样，式(3.18)中的奖励函数可以更改为：

$$r_{m,t}=-10w_m\cdot(p_{l,m}(t)+p_{o,m}(t))-(1-w_m)\cdot B_m(t) \tag{4.1}$$

于是可以通过设置单个因子 w_m 的值来进行能耗和缓冲延迟之间的权衡。训练阶段的集数设置为 $K_{max}=2000$，每集的最大步数为 $T_{\max}=200$。

为了比较，本文引入了四种策略：

（1） 基于 DQN 的动态卸载: 采用传统的基于离散动作空间的 DQN 算法。

（2） 基于 DDPG 的动态卸载：采用基于连续动作空间的 DDPG 算法。

（3） 本地执行优先(Local): 对于每个槽位，用户代理首先在本地执行尽可能多的任务数据位，然后卸载剩余的缓冲位。

（4） 计算卸载优先(Offload): 每个用户代理首先以最大的努力卸载数据位，然后在本地执行剩余的缓冲数据位。

为了公平比较 DDPG 和 DQN，本文使用相同的神经网络配置，并选择了相同数量的离散功率电平，以确保两种方法都能达到相似的收敛行为。在这种情况下，本地执行和计算卸载的功率级别被定义为 $P_{l,m}=\{0,\frac{P_{l,m}}{L-1},...,P_{l,m}\}$ 和 $P_{o,m}=\{0,\frac{P_{o,m}}{L-1},...,P_{o,m}\}$，其中功率级数为 L=8。可供每个用户代理选择的操作空间为 $P_{o,m}\times P_{l,m}$。

4.2 单用户场景

用户被设置为随机定位在距离基站 $d_1=100m$ 的范围内。

4.2.1 算法奖励收敛过程

单用户动态计算卸载的训练过程分别设置 $w_1=0.5$ 和 0.8，如图 4-1 和图 4-2 所示。其中每条曲线均为 10 次数值模拟的平均值。图中还标明了任务到达率 λ_1= 2.0Mbps 和 3.0Mbps 两种不同的情况。可以观察到，对于 DDPG 和 DQN 学习的策略，每集的平均奖励呈增加趋势，这表明用户代理和系统环境之间的交互仍在继续，逐渐学习到有效的计算卸载策略。另一方面，每个学习到的策略在大约 1500 集后变得稳定。尤其在不同场景下，DDPG 学习到的策略稳定性能总是优于 DQN，这表明基于 DDPG 的策略在连续控制问题上比基于 DQN 的策略更有效地探索动作空间。

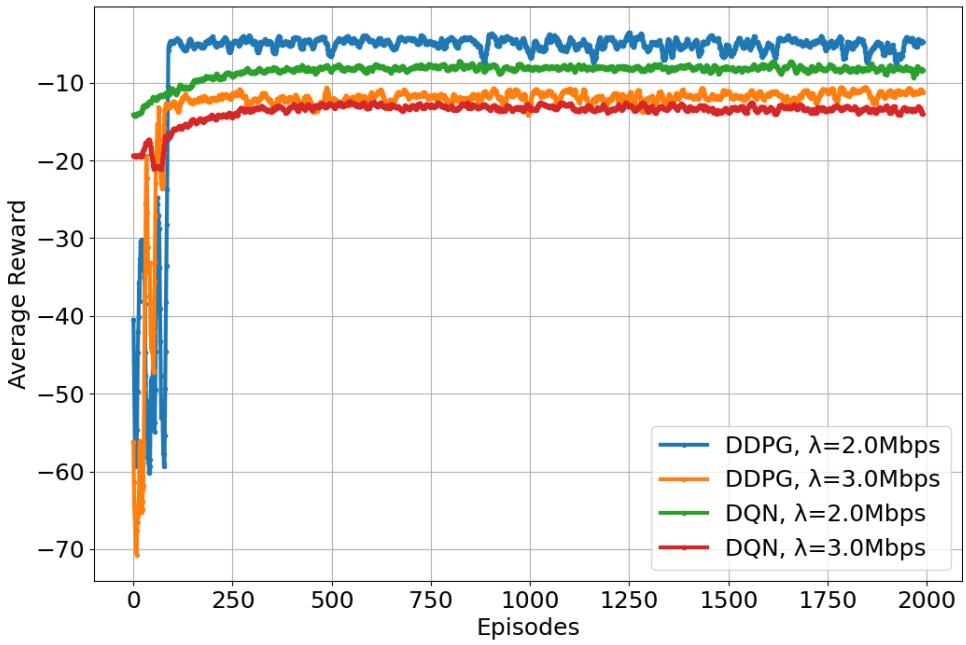

图 4-1 $w_1 = 0.5$ 的单用户代理在训练过程中每集的平均奖励

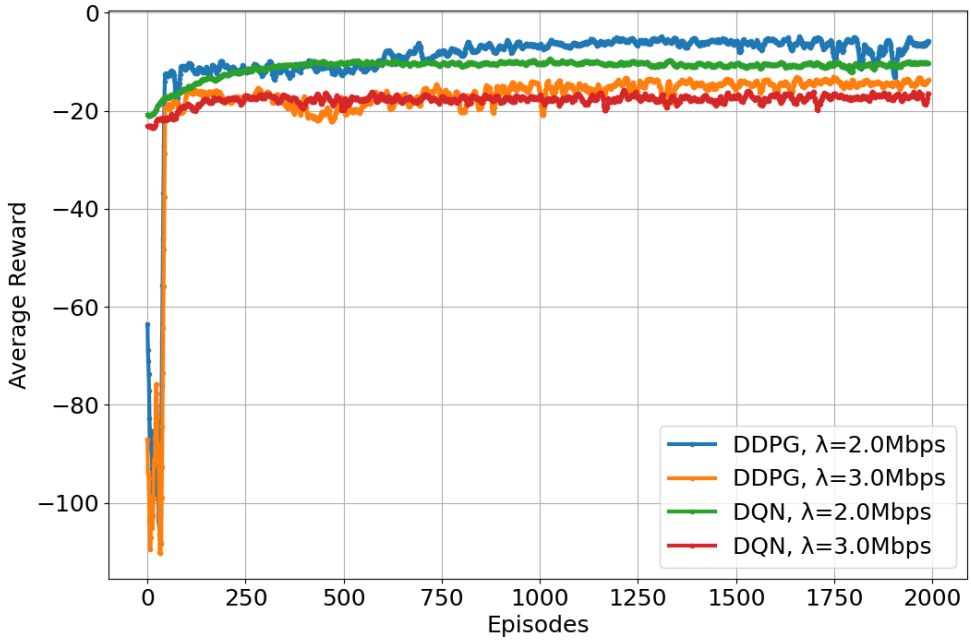

图 4-2 $w_1 = 0.8$ 的单用户代理在训练过程中每集的平均奖励

4.2.2 算法测试比较结果

在经过 K_{max} =2000 集的训练后，我们得到了分别由 DDPG 和 DQN 策略学习到的动态计算卸载策略。为了比较不同策略的性能，我们在 100 次数值模拟下进行测试，其中每次运行由

10000 步组成。如图 4-3 和图 4-4 所示，分别给出 $w_1=0.5$ 和 0.8 时的平均测试结果，每个测试结果包括平均奖励、功耗和缓冲延迟的性能。从图 4-3 中可以观察到，平均奖励随着任务到达率的增加而降低，这说明在面对更大的计算需求时，计算成本会变得更高。也就是说，计算成本的增加来自于更高的功耗和更长的缓冲延迟。并且可以从图中看到尽管基于 DDPG 的策略优于基于 DQN 的策略，但为了实现最低能耗，它稍微牺牲了缓冲延迟。此外，当任务到达率高于 3.5Mbps 时，基于 DQN 的策略的平均奖励低于贪婪策略，这是由于基于 DQN 的策略的离散功率水平有限。同时，两种贪心策略，即 Local 和 Offload，在图 4-3 中具有几乎相同的平均缓冲延迟，这是由于这两种策略都试图在每个时隙中尽最大努力执行缓冲的任务位，而不考虑因子 w_1。实际上，对于较低的任务到达率，例如 $\lambda_1=1.5\sim3.0$Mbps，对于任意一种贪婪策略，到达一个插槽的任务位很可能在下一个插槽中完全执行。这两种策略的唯一区别是本地执行和任务卸载的优先级，这导致了不同的平均功耗，如图 4-3 所示。因此，两种平均缓冲延迟都非常接近于到达一个插槽的任务比特的平均数量，随着任务到达率的增长，它会略有增加。

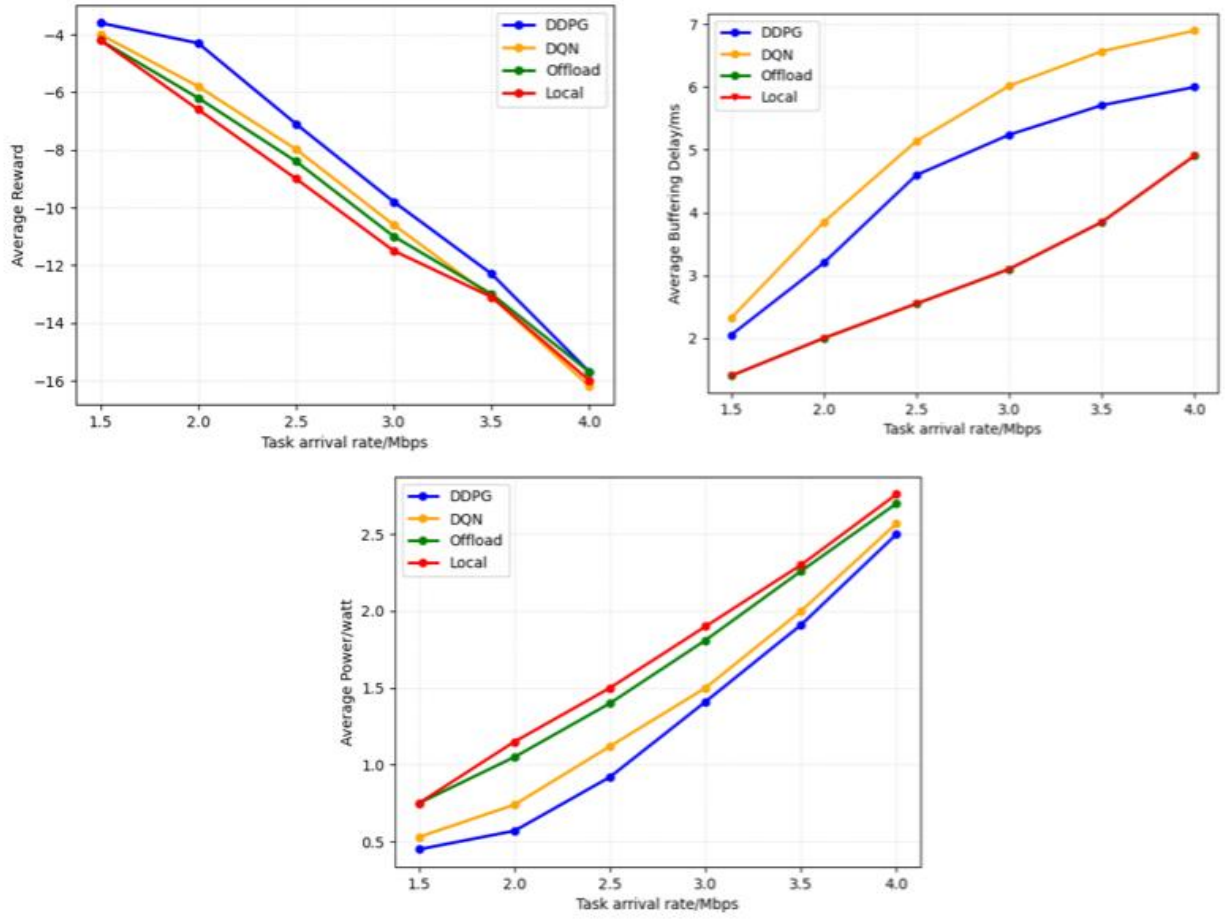

图 4-3 $w_1=0.5$ 的单用户代理的测试结果对比

在图 4-4 中，本文还提供了较大权衡因子 $w_1=0.8$ 的测试结果。从式（4.1）中，我们知道较大的 w_1 将在奖励函数中对功耗给予更大的惩罚。在这种情况下，基于 DDPG 的策略的平均奖励再次优于 DQN 策略，并且差距比 $w_1=0.5$ 的情况大得多。具体来说，这是通过更低的功耗和增加的缓冲延迟实现的。图 4-4 中 Local 和 Offload 的平均缓冲延迟与图 4-3 相同，因为这两种贪婪策略不受 w_1 影响，其行为也不会改变。

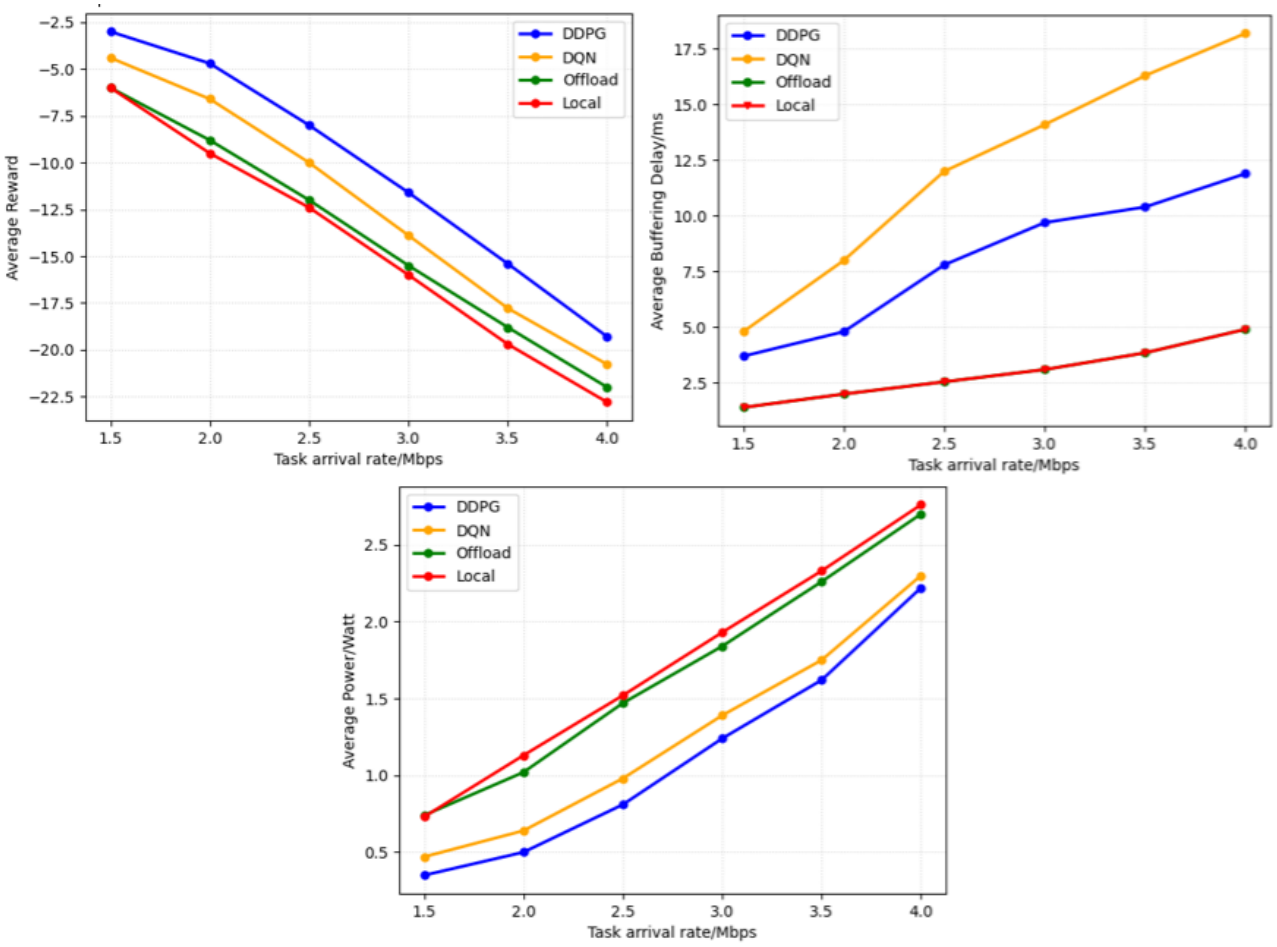

图 4-4 $w_1=0.8$ 的单用户代理的测试结果对比

4.3 多用户场景

本文系统中设置 M = 3 个移动用户，每个移动用户随机分布在距离基站 $d_m=100$ 米的范围内，任务到达率为 $\lambda_m=m\times 1.0Mbps$ 。

4.3.1 算法奖励收敛过程

通过对所有用户设置 $w_m=0.5$，训练过程如图 4-5 所示。可以观察到，移动用户在与系统环境进行多次交互后，其获得的平均奖励会呈现上升趋势。这表明无论是基于 DDPG 的策略还是基于 DQN 的策略，都可以在每个移动用户上学习到高效的分散动态计算卸载策略，特别

是对于具有不同计算需求的异构用户。因此，可以合理推测，那些计算需求更高的用户将承担更大的计算成本。此外，在相同的任务到达率下，多用户场景相较于单用户场景多，其平均奖励显著降低。这是因为随着更多用户使用基站服务，数据传输的频谱效率会受到影响而下降，从而导致多用户场景下计算卸载的能耗增加。

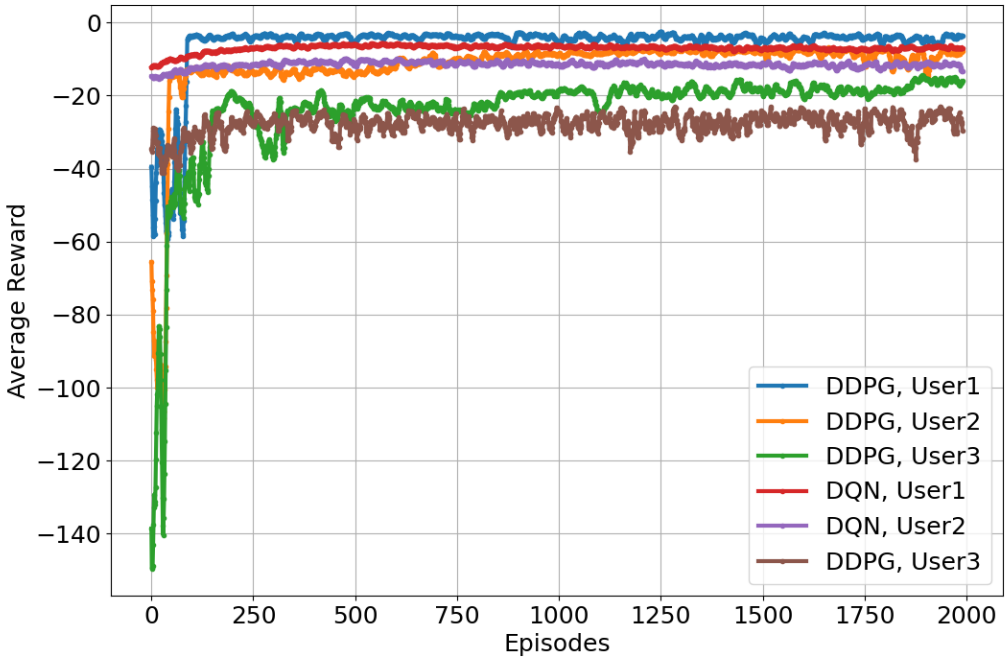

图 4-5 所有移动用户在条件 $w_m = 0.5$ 下训练过程中的平均奖励示意图

4.3.2 算法测试比较结果

在 $K_{\max} = 2000$ 集后加载基于 DDPG 和 DQN 算法学习到的神经网络参数，比较不同动态计算卸载策略的测试结果如表 4-1 和表 4-2 所示。由表 4-1 可知，在 $w_m = 0.5$ 的情况下，采用 DDPG 略的用户 2 和用户 3 的平均回报优于其他所有策略。然而，对于用户 1 来说，基于 DDPG 策略略差于 DQN 策略，这表明对分配功率小的 DDPG 需要进一步改进。此外，可以观察到，基于 DDPG 和基于 DQN 的策略都可以在稍微损害缓冲延迟的情况下实现更低的功耗。

如表 4-2 所示，通过设置权衡因子 $w_m = 0.8$，可以发现基于 DDPG 的策略在每个用户代理上产生的平均奖励均高于基于 DQN 的策略。同时，随着时间的推移，DDPG 与 DQN 之间的性能差异逐渐扩大。另外，由于对功耗给予了更多的惩罚权重，每个用户的消耗功率远低于

$w_m = 0.5$的情况，

然而，这种调整也带来了缓冲延迟的适度增长。因此，对于多用户场景，通过选择适当的w_m值，以确保在满足平均缓冲延迟要求的同时，实现功耗的最小化。最后，无论在何种情况下，基于 DDPG 的策略在所有用户的平均奖励上都展现出了优于基于 DQN 策略的表现。

表 4-1 对所有移动用户$w_m = 0.5$的测试结果对比

	Average Reward			**Average Power/watt**			**Average Delay/ms**		
	User 1	User 2	User 3	User 1	User 2	User 3	User 1	User 2	User 3
DDPG	-1.923	-6.453	-16.193	0.166	0.614	1.685	3.144	7.798	13.912
DQN	-2.790	-8.954	-18.698	0.288	0.955	1.976	2.543	7.989	15.281
Offload	-3.423	-10.921	-26.347	0.461	1.322	3.153	1.011	2.210	5.981
Local	-1.967	-13.912	-28.675	0.224	1.687	3.434	1.108	2.235	6.101

表 4-2 对所有移动用户$w_m = 0.8$的测试结果对比

	Average Reward			**Average Power/watt**			**Average Delay/ms**		
	User 1	User 2	User 3	User 1	User 2	User 3	User 1	User 2	User 3
DDPG	-1.765	-5.671	-12.799	0.211	0.769	1.934	1.489	3.654	6.155
DQN	-2.159	-7.688	-14.456	0.288	1.167	2.368	1.436	3.761	6.178
Offload	-2.551	-7.605	-18.785	0.461	1.322	3.153	1.011	2.210	5.981
Local	-1.655	-9.691	-20.091	0.224	1.687	3.434	1.108	2.235	6.101

结 论

在本文中，构建了一个支持 MIMO 的多用户 MEC 系统，在随机任务到达和无线信道条件下，核心目标是均衡功耗和缓冲延迟实现最小化长期平均计算成本，以优化每个用户的本地执行和计算卸载能力。从算法的设计、算法的实现、算法的比较三个方面入手，实现了分散动态计算卸载算法的设计。通过采用基于连续动作空间的深度强化学习算法，在每个用户处分别学习有效的计算卸载策略，再利用学习到的策略根据其对所提出的系统环境的局部观察分配本地执行和任务卸载的权力。最后通过仿真设置数值模拟，将实验结果对比可以得出：基于 DDPG 的策略可以更有效地帮助每个用户学习有效的动态卸载策略，并且验证了其性能优于传统的基于离散动作空间的强化学习方法 DQN 学习的策略和其他计算成本降低的贪心策略。

如何对移动边缘计算技术进行更一步优化实现更好的性能，在未来仍有很大的研究价值。未来也许可以从以下几个方面进行研究：1）研究不可分割任务的分散二进制计算卸载策略，即任务只能卸载到基或作为一个整体在本地执行。这可能需要一个分层 DRL 框架来给出二进制卸载决策，然后再给出连续的功率分配决策。2）因为目前的策略是在每个用户上独立训练的，而不考虑其他用户代理的行为，所以未来可以就将移动用户之间的协作也引入到基于的 DDPG 框架中，以提高学习策略的性能。

致 谢

四年的大学生活在不知不觉中就要结束了，在这段难忘的生活中，有我许多美好的回忆。回首走过的四年岁月，所获点滴都足以让我在诸多方面有所成长，心存感激。

在这份大学的最后一页里，我要感谢的人很多。首先要感谢我的导师孟老师，从科研训练的初识，到如今在她的指导下完成了毕业论文，这一路走来孟老师都给予了我极大的鼓励与支持。孟老师的平易近人、耐心教导，使我深感关怀与温暖；孟老师渊博的知识、一丝不苟的态度，为完成我的论文提供了极大帮助。我向她表示衷心的感谢。同时也非常感谢我的师姐，在毕业设计的开题到结题整个过程中，她解答了我研究过程中的一个个问题，耐心指导我完成了实验，在她的帮助下我才能顺利完成论文。

然后，我要感谢大学四年陪我学习、玩乐、奋斗的朋友们，没有他们我的大学生活就不会如此丰富多彩。

最后，特别感谢我的家人，他们对我的支持与关怀永远是我前进的动力。

人生的路还很长，希望我能长风破浪。

参 考 文 献

[1] 周悦芝, 张迪. 近端云计算: 后云计算时代的机遇与挑战[J]. 计算机学报, 2019, 42(4): 677-700.

[2] Sun Xiang, Ansari Nirwan. EdgeIoT: Mobile Edge Computing for the Internet of Things[J]. IEEE Communications Magazine. 2016,54,(12):22-29.

[3] 王妍, 葛海波, 冯安琪. 云辅助移动边缘计算中的计算卸载策略[J].计算机工程, 2020, 46(8): 27-34.

[4] 巩宸宇,舒洪峰,张昕.多层次算力网络集中式不可分割任务调度算法[J].中兴通讯技术, 2021, 27(3):7.

[5]Satyanarayanan M .The Emergence of Edge Computing[J].Computer, 2017, 50(1):30-39.

[6] Abbas N, Zhang Y, Taherkordi A, et al. Mobile edge computing: A survey[J]. IEEE Internet of Things Journal, 2017, 5(1):450-465.

[7] Zhuang Y, Li X, Ji H, et al. Optimization of mobile MEC offloading with energy harvesting and dynamic voltage scaling[C]//2019 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2019: 1-6.

[8] 顾冰雪. 面向深度学习的云边任务迁移与调度研究[D]. 中国电子科技集团公司电子科学研究院, 2022.

[9] 吕洁娜,张家波,张祖凡,甘臣权.移动边缘计算卸载策略综述[J].小型微型计算机系统,2020,41(09):1866-1877

[10] B. Liu, C. Liu and M. Peng. Resource Allocation for Energy-Efficient MEC in NOMA-Enabled Massive IoT Networks. IEEE Journal on Selected Areas in Communications, 2021,39(4):1015-1027.

[11] P. Mach, Z. Becvar. Mobile edge computing: A survey on architecture and computation offloading[J]. IEEE Communications Surveys and Tutorials, 2017, 19(3): 1628-1656.

[12] Li Y. Deep reinforcement learning: An overview[J]. arXiv preprint arXiv:1701.07274, 2017.

[13] Chen M, Hao Y X. Task Offloading for Mobile Edge Computing in Software Defined Ultra-dense Network[J]. IEEE Journal on Selected Areas in Communications,2018,36(3):587-597.

[14] Zhao J, Li Q, Gong Y, et al. Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks[J]. IEEE Transactions on Vehicular Technology, 2019, 68(8): 7944-7956.

[15] Zhang W, Wen Y, Guan K, et al. Energy-optimal mobile cloud computing under stochastic wireless channel[J]. IEEE Transactions on Wireless Communications, 2013, 12(9): 4569-4581.

[16] Tan L, Kuang Z, Zhao L, et al. Energy-efficient joint task offloading and resource allocation in OFDMA-based collaborative edge computing[J]. IEEE Transactions on Wireless Communications, 2021, 21(3): 1960-1972.

[17] Chen L, Wu J, Zhang J, et al. Dependency-aware computation offloading for mobile edge computing with edge-cloud cooperation[J]. IEEE Transactions on Cloud Computing, 2020, 10(4): 2451-2468.

[18] Tran T X, Pompili D. Joint task offloading and resource allocation for multi-server mobile-edge computing networks[J]. IEEE Transactions on Vehicular Technology, 2018, 68(1): 856-868.

[19] Lin Y D, Lai Y C, Huang J X, et al. Three-Tier Capacity and Traffic Allocation for Core, Edges, and Devices for Mobile Edge Computing[J]. IEEE Transactions on Network and Service Management, 2018, 15(3): 923-933

[20] Chen Z, Wang X. Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach[J]. EURASIP Journal on Wireless Communications and Networking, 2020, 2020(1): 1-21.

[21] MENG H, HUO R, GUO Q, et al. Machine Learning-Based Stochastic Task Offloading Algorithm in Mobile-Edge Computing[J]. Journal of Beijing University of Posts and Telecommunications, 2019, 42(2): 25.

[22] Zhang Y, Liu T, Zhu Y, et al. A deep reinforcement learning approach for online computation offloading in mobile edge computing[C]//2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS). IEEE, 2020: 1-10.

[23] 王浩翔. 基于博弈论的边缘计算资源分配算法研究[J]. 邯郸: 河北工程大学, 2019.

[24] 陈琳,董甲东,潘凯,等.云边端和 D2D 边缘架构下的计算卸载策略综述[J].计算机工程与应用, 2023, 59(15):55-67.

[25] 董思岐,李海龙,屈毓锛,等.移动边缘计算中的计算卸载策略研究综述[J].计算机科学, 2019, 46(11):9.

[26] Hinton G E , Osindero S , Teh Y W .A Fast Learning Algorithm for Deep Belief Nets[J].Neural Computation, 2006, 18(7):1527-1554.

[27] 陈学松, 杨宜民. 强化学习研究综述[J]. 计算机应用研究 2024, (8):40-44.

[28] 王蓁蓁,王智钢.逻辑半马尔可夫决策过程及 Q 学习[J].金陵科技学院学报, 2013, 29(2):7.

[29] 高阳, 陈世福, 陆鑫. 强化学习研究综述[J]. 自动化学报, 2004, 30(001): 86-100.

[30] Kim P, Kim P. Convolutional neural network[J]. MATLAB deep learning: with machine learning, neural networks and artificial intelligence, 2017: 121-147.

[31] MINH V, KAVUKCUOGLU K, SILVER D, et al.Playing atari with deep reinforcement learning[J].Computer Science, 2013: 1-9.

[32] Sutton R S , Barto A G .Reinforcement Learning: An Introduction[J].IEEE Transactions on Neural Networks, 1998, 9(5):1054-1054.

[33] Dayan P, Watkins C. Q-learning[J]. Machine learning, 1992, 8(3): 279-292.

[34] Kelvin M J, Schneiders D. Learning to Play Computer Games with Deep Learning and Reinforcement Learning[J]. 2018.

[35] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. arXiv preprint arXiv:1509.02971, 2015.

[36] Mach P, Becvar Z. Mobile edge computing: A survey on architecture and computation offloading[J]. IEEE communications surveys & tutorials, 2017, 19(3): 1628-1656.

[37] 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.

[38] Abramowitz M , Stegun I A , Miller D .Handbook of Mathematical Functions With Formulas, Graphs and Mathematical Tables (National Bureau of Standards Applied Mathematics Series No. 55)[J].Journal of Applied Mechanics, 1965, 32(1):239.

[39] Smith G S .A direct derivation of a single-antenna reciprocity relation for the time domain[J].IEEE Transactions on Antennas & Propagation, 2004, 52(6):1568-1577.

[40] 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.

[41] 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): 203-221.

[42] Miettinen A P, Nurminen J K. Energy efficiency of mobile clients in cloud computing[C]//2nd USENIX workshop on hot topics in cloud computing (HotCloud 10). 2010.

[43] Ghosh G S, Kwasnica A M, Shortle J S. A laboratory experiment to compare two market institutions for emissions trading[J]. 2010.

[44] Silver D, Lever G, Heess N, et al. Deterministic policy gradient algorithms[C]//International conference on machine learning. Pmlr, 2014: 387-395.

[45] Chen H H, Guizani M. Next generation wireless systems and networks[M]. John Wiley & Sons, 2006.