



单位代码 10006

学 号 16711097

分 类 号 N533

**北京航空航天大学**  
B E I H A N G U N I V E R S I T Y

# 毕业设计(论文)

## 资源受限的移动边缘计算系统计算卸载问题研究

学院名称 自动化科学与电气工程学院

专业名称 自动化

学生姓名 李子建

指导教师 王岩

2020 年 5 月

资源受限的移动边缘计算系统卸载问题研究

李子建

北京航空航天大学

# 北京航空航天大学

## 本科生毕业设计（论文）任务书

### I、毕业设计（论文）题目：

资源受限的移动边缘计算系统计算卸载问题研究

### II、毕业设计（论文）使用的原始资料（数据）及设计技术要求：

仿真模型中有单一服务器多个基站，每个基站若干增益不同的通频带。移动设备 1 到 80 个，任务到来密度 0.001 到 0.1 个每秒。以非线性激活函数作为划分，决策网络深度为三。

设计时要求考虑多种资源限制和环境的随机性，决策算法

### III、毕业设计（论文）工作内容：

1. 搭建了一个考虑多种资源限制、随机性较大贴近真实场景的多基站单服务器的移动边缘计算仿真模型。
2. 将卸载决策任务规范化为部分可观马尔科夫决策问题，定义相应的状态、动作、回报
3. 设计演员、批评家网络结构，利用 pytorch 开源框架搭建基于 DDPG 的决策神经网络，编写学习算法。
4. 训练网络，进行实验。试验多组参数取值，确定合理的参数设定，重复实验，对比分析结果。

### IV、主要参考资料：

1. Mach P, Becvar Z. Mobile edge computing: A survey on architecture and computation offloading[J]. IEEE Communications Surveys & Tutorials, 2017, 19(3): 1628-1656.
2. Chen Z, Wang X. Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach[J]. arXiv preprint arXiv:1812.07394,

2018.

3. Zhang K, Mao Y, Leng S, et al. Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks[J]. IEEE access, 2016, 4: 5896-5907.
4. Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. arXiv preprint arXiv:1509.02971, 2015.

自动化科学与电气工程 学院 自动化专业类 1603124 班

学生 李子建

毕业设计（论文）时间：2020年2月19日至2020年5月26日

答辩时间: 2020 年 6 月 4 日

成 绩：良

指导教师: 王岩

兼职教师或答疑教师（并指出所负责部分）

无

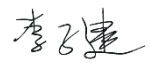
自动化系（教研室） 主任（签字） \_\_\_\_\_

注：任务书应该附在已完成的毕业设计（论文）的首页。

## 本人声明

我声明，本论文及其研究工作是由本人在导师指导下独立完成的，  
在 完成论文时所利用的一切资料均已在参考文献中列出。

作者：李子建

签字：

时间：2020 年 6 月



## 资源受限的移动边缘计算系统计算卸载问题研究

学 生：李子建

指导教师：王 岩

### 摘 要

近来人们对移动应用服务质量的要求变得越来越高和 5G 技术的日渐普及，移动边缘计算正受到越来越多的关注。计算卸载是移动边缘计算的关键技术，但当前相关的研究仍处于起步阶段，以仿真为主要研究手段，追求更合理的仿真模型和更有效的决策算法。

本次毕业设计在融合主流边缘计算建模方法的同时，考虑了包括 UE 本地算力和通信能力、MECS 算力、UE 与 MECS 间通信信道各种资源有限制的情况，首先搭建了一个随机性较大贴近真实场景的多基站单服务器的移动边缘计算仿真模型。综合考虑 5G 技术应用场景中的宏基站与多个微基站等传输通道同时存在的局面，简化考虑不同通频带的信号增益可能的差异，提出了同时进行计算卸载决策与信号传输通道和边缘服务器算力分配的决策优化问题。然后结合深度确定策略梯度下降这一深度强化学习方法，设计了一种可行的变周期卸载决策算法，把任务失败比率、平均能耗、平均时延作为优化指标，并通过在不同的仿真测试条件下与全本地和全卸载组的对比实验，说明所提算法的有效性和可扩展性，又分析了本研究的一些不足。在最后，列出了未来可以在目前的这项工作基础上所做的一些改进。

本文的主要优点在于所建立的仿真环境因其随机性较大和考虑限制较多贴合实际、变周期决策学习算法克服了任务等待决策时隙所耽误的时间。

**关键词：**移动边缘计算，强化学习，卸载决策



# **Research on Computation Offloading Decision of Mobile Edge Computing System with limited Resource**

Author : LI Zi-jian

Tutor : WANG Yan

## **Abstract**

This graduation design studies and learns from many popular modeling methods when build its own one-server-multiple-station mobile edge computing simulation environment with high random factors, taking into account various resources limitation, such as one task most per time executed in any User Equipment, simplifying the consideration of the possible difference in signal gain of different channels, and simultaneously decide weather to offload or not and server's resource allocation, the signal transmission channels if offloading.

Using deep deterministic policy gradient descent, a deep reinforcement learning algorithm, a feasible variable-cycle offloading decision method is provided, which takes task failure rate, energy consumption, and delay as its optimization indicators. Experiment via different testing conditions shows the effectiveness and scalability of the proposed algorithm, and analyzes some of the deficiencies of this study. At the end, some improvements that can be made based on the current work are listed.

The main advantage of this work is that the simulation environment built is practical due to its randomness and resources limitations, and the variable-cycle decision algorithm has unique advantages shown below.

**Key words:** Mobile edge computing, Reinforcement Learning, Offloading Decision



# 目 录

1	绪论.....	1
1.1	课题背景及研究意义 .....	1
1.2	国内外研究现状 .....	3
1.2.1	建模仿真的研究方式.....	3
1.2.2	最小化时间延迟 .....	4
1.2.3	时间约束下最小化能耗.....	4
1.2.4	综合优化时延和能耗.....	5
1.2.5	深度强化学习优化算法的应用 .....	5
1.2.6	存在的问题 .....	6
1.3	研究目标和研究内容 .....	7
1.3.1	研究目标.....	7
1.3.2	研究内容.....	7
1.4	论文组织结构.....	8
2	边缘计算模型与计算卸载决策.....	9
2.1	边缘计算仿真模型概述.....	9
2.2	本地处理模型.....	9
2.3	卸载处理模型.....	10
2.4	建立优化问题的模型 .....	12
2.4.1	MDP 动作 .....	13
2.4.2	MDP 状态 .....	13
2.4.3	MDP 回报 .....	14
2.5	本章小结.....	14
3	卸载决策算法分析与实验参数选取.....	17
3.1	深度强化学习 .....	17
3.2	确定策略梯度下降学习算法.....	18
3.3	网络结构及其接口 .....	19





---

3.4	实验参数的选取 .....	22
3.4	本章小结 .....	24
4	实验结果与结论 .....	25
4.1	仿真结果与分析 .....	25
4.1.1	实验补充说明 .....	25
4.1.2	结果可视化与分析 .....	26
4.2	工作总结与展望 .....	28
4.2.1	工作总结 .....	28
4.2.2	未来展望 .....	29
	致谢 .....	31
	参考文献 .....	32

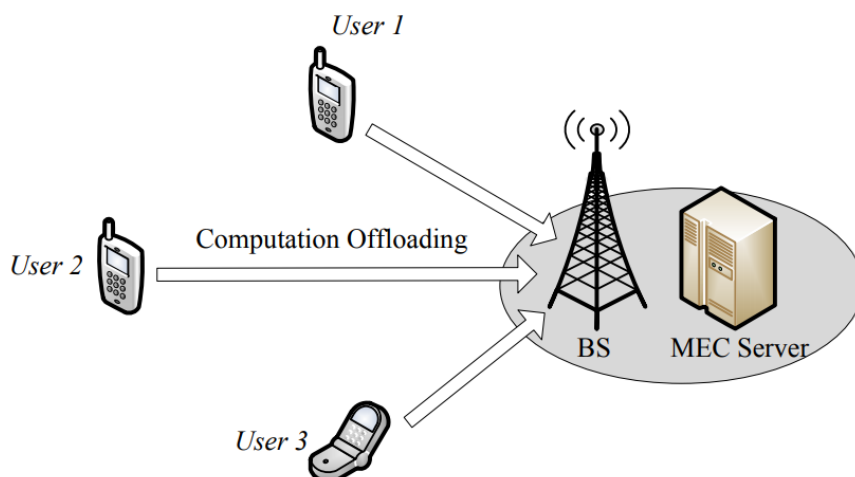


## 1 绪论

### 1.1 课题背景及研究意义

近年来,智能手机、笔记本电脑和平板设备等不断普及;虚拟现实、自然语言处理、复杂交互式软件等各种计算密集型应用服务得到越来越广泛的使用;用户对服务质量(Quality of Service, QoS)的需求不断提高。尽管电子设备在之前的很长时间中以近似以摩尔定律的描述得到高速的发展,然而人们还是总能遇到算力跟不上需求的情况,并且通常来讲用户所用的移动设备所能完成的计算任务还受其电池容量的限制。这就诞生了云计算(Cloud Computing)这种计算范式。云计算不只能够允许用户设备(User Equipment, UE)利用远程集中云(Central Cloud)的大量存储资源,还能够将部分计算任务从本地卸载到中央云服务器去完成。通过将输入数据和算法程序传输到云端,利用高性能的云服务器来完成计算任务,再将云端的计算运行结果传回本地,本地的能耗和计算时延均能得到降低,从而提升了 QoS。<sup>[1]</sup>然而,云计算有时也有着诸多缺点不足。因为云服务器一般都集中于一地或少数几地,很多用户要想进行云计算需要很高的传输通信成本,将计算卸载到云和将结果传回的时延和能耗均比较可观。

移动边缘计算(Mobile Edge Computing, MEC)被广泛认为是对云计算补充的一种新的范式,最早的边缘计算的概念(cloudlet)是在 2009 年被提出的<sup>[2]</sup>,其背后的思路是本着高计算性能的计算机策略性地布置到网络中靠近用户设备的位置,但是 cloudlet 只允许用户通过无线局域网(WiFi)与之通信,每次用户需要使用 cloudlet 服务时都常常需要在移动网络和 WiFi 之间切换,同时也限制了使用此服务的移动性。边缘计算把服务器部署到网络边缘,移动设备的用户可以通过无线网络(Radio Access Networks, RANs)将计算卸载到与其相近的边缘服务器,以期解决云计算传输距离过长的问题,并与传统的云计算模式形成互补。计算卸载(computation offloading)是边缘计算的一个关键技术,其大致可以分为节点发现、程序切割、卸载决策、程序传输、计算结果回传这几步。其中卸载决策又是研究者们最关心的还有待进一步讨论的问题。

图 1.1 移动边缘计算示意图<sup>[3]</sup>

总体上讲, 计算卸载决策时首先要考虑的是要不要卸载。如果决定卸载, 才需要讨论究竟是卸载多少计算任务、卸载哪些计算任务<sup>[4]</sup>。卸载决策的结果通常会是在本地执行计算任务( Local Execution)、全部卸载( Full Offloading)、部分卸载( Partial offloading)三种, 其中部分卸载是一个受多种因素影响的高度复杂化的过程, 甚至于有些计算不能拆分成在本地执行和卸载到服务器完成的两部分, 使用时有时还要考虑任务的可拆分性, 子任务的依赖关系等待, 因此种种, 最新的许多研究都是只在“是否卸载”这一问题的层面进行讨论的, 给出的决策也只有本地执行计算或全部卸载到边缘服务器两种情况。UE 在判断某一计算任务是否卸载时, 往往会比较卸载与否的情况下所需要的完成计算消耗的时间和能耗, 以期给用户更高的 QoS。

边缘计算正在获取持续性的越来越多的关注, 许多研究者相信它将在下一代移动通信5G技术和物联网( Internet of Things, IoT)的应用普及下发挥出重要的作用。由于IoT的设备部署模式下, 各信息传感设备不能保证都具有满足其所要提供的服务的计算能力, 而许多设备的计算资源将会在大多数时间中处于空闲状态。考虑到机器对机器( M2M)的通信连接方式在IoT中的进一步普及的预测, 各种分布式设备将能很方便地卸载自己的计算任务到其他设备或服务器中, 因此边缘计算被广泛地认为将对IoT提供十分必要的技术支持。由于边缘网络中计算资源受限及无线传输中信道的复杂性, 计算卸载决策算法对于MEC能否发挥出降低UE所要进行的高计算密度任务的能耗延迟的潜力有决定性的作用, 不科学的卸载决策并不见得能够提高用户的使用体验( Quality of Experience, QoE)。



而计算卸载作为边缘计算的核心技术之一, 尽管在过去的十年内已经积累了大量的研究, 仍然显得不是很成熟, 相关工作往往不能把系统中的各种关键要素考虑全面, 所提的优化指标也往往只从整体入手, 而忽略了用户需求的差异性, 不能真实的反应QoS。且 MEC的应用场景中边缘服务器服务的小区中用户数量是流动变化的, 应对可能出现的边缘计算需求高峰就需要卸载决策算法具有一定的可扩展能力, 这一点还鲜有模型满足, 需要更进一步的研究。

由于现阶段很多研究中对MEC的系统建模都是十分简单甚至粗糙的, 很多在其上检验能满足决策任务需求算法可能会显得非常简单, 而使得其所提出的卸载决策算法胜任真实场景下的卸载决策的能力容易遭人质疑。本毕设的意义在于, 面对通信资源和边缘服务器算力受限的情况下, 融合主流边缘计算建模方法, 搭建一个考虑多种资源限制的仿真模型, 同时综合考虑任务失效、时间延迟、能量消耗到损失函数(优化目标)中, 使用深度强化学习的优化方式, 提供一种可行的无模型(model-free)的动态卸载决策算法, 来提供给其他的建模场景甚至是未来的真实应用场景使用。相信这会为未来的相关研究提供思路 and 比较标准, 更好的满足预测中在未来会普及的在各种复杂多变的场景下的边缘计算服务计算卸载合理决策的需求。

## 1.2 国内外研究现状

### 1.2.1 建模仿真的研究方式

对于现阶段边缘计算卸载决策领域的研究情况特别需要注意的一点是, 尽管存在部分工作在真实环境中所获得的数据上展开, 比如王文文基于麻省理工学院校内 Youtube 网页服务器访问情况数据评价检验所提的算法<sup>[5]</sup>, 然而几乎所有的工作都是基于研究者自己对情景的建模仿真来着手尝试提出更优的决策算法的, 人们往往通过非常简单的情景模型或是理论分析来验证自己的理论, 比如假设信道的传输速率恒定, 任一时刻 MCES 覆盖区内用户数目恒定不变, 各用户设备性能完全相同等等, 模型中的一些人为设定的参数更是有不总可靠的嫌疑。这样, 相互之间会因模型的不统一无法直接的对照比较, 同时没有在真实场景下的测试或是更贴近现实情况的模型, 也使得许多研究欠缺了说服力。从 2009 年边缘计算这一概念的提出到现在<sup>[2]</sup>, 尽管其正得到越来越多研究者的关注, 然而它还是一项非常新的不成熟而尚待发掘的研究领域。现在对全部卸载决策



的研究工作都会把最小化任务完成时间、在时间约束下最小化能耗或综合优化时延和能耗的其中一个作为目标。

在边缘计算的相关研究中如果以能耗作为优化指标,表示每个待决策是否卸载的任务通常可用三个参数:  $\text{Task}(d, c, t)$ , 其中  $d$  为计算所用到的数据量, 包括程序和输入参数两部分;  $c$  代表执行任务的计算量, 用计算程序所需的 CPU cycle 来表示, 一般都会假设此值是设备不敏感的, 即在本地和在移动边缘计算服务器(MECS)上执行相同计算任务所需的 CPU 周期数被认为是一致的;  $t$  是该计算任务的最大允许时延约束项, 表示此任务必须要在规定的时间内得到解决, 否则计算任务失败, 如果是选择卸载到边缘服务器计算则包括数据上传时间、MECS 执行计算时间、结果回传时间及可能的等待时间甚至是做出决策所消耗的时间等。如果综合考虑能耗和时延作为优化指标, 那么只需两个参数  $\text{Task}(d, c)$  就足够了。也有的研究工作只用一个参数来描述任务, 可以假定完成一定量数据的计算所需的 CPU 周期是一定的而只用计算任务数据量来描述之<sup>[6]</sup>。

### 1.2.2 最小化时间延迟

最小化时延。边缘计算的一个主要优势就是有望能够减少计算任务执行的时间, 计算卸载到 MEC 服务器所引入的时延通常包括数据上传时间、MEC 服务器执行计算任务时间、结果回传时间三部分。一般来讲因为所需传输的数据量较少以及该方向的通讯速度较快之故结果回传用时较总时延来说可以忽略不计<sup>[7]</sup>。通过比较在本地执行的时间开销和卸载到服务器的总时间开销, 来进行卸载决策, 是一种可行的思路。Mahmoodi 等人提出了一个根据应用缓冲器排队状态、UE 与 MECS 的可用计算量与信道情况找到最优的卸载策略的一维搜索算法<sup>[8]</sup>, 对照全本地计算和全卸载两个基准策略, 仿真结果表明该搜索算法能够相对全本地减少最高 80% 的时间消耗, 而相对全卸载减少了最高可达 44% 的时间消耗。而只考虑时延这一指标的缺点也是显而易见的, 有时候用户会更在意边缘计算能带来的能耗上的优化。

### 1.2.3 时间约束下最小化能耗

在满足时间约束下最小化能量消耗。UE 将部分计算任务卸载到 MECS 去完成时,



节约了本地 CPU 完成计算的能耗,但也要付出传输数据的通信耗能。KE ZHANG 等人提出了一种此类算法(EECO, Energy-Efficient Computation Offloading)<sup>[9]</sup>,该算法考虑了 5G 应用场景中宏基站(Macro Base Station, MBS)与微基站(Small Base Station, SBS)多种传输通道同时存在的局面,假设各频道是正交的,同时进行卸载决策与信道通频带分配优化。值得一提的是,EECO 模型的优化指标是包括边缘服务器计算能耗在内的整个系统的所有能耗,而不是像大多数相关研究一样只考虑资源限制更明显的 UE 本地上的能耗。但由于通常从单位 CPU 周期能耗的角度来看 MECS 较之 UE 有更高的运算效率,且因上文所解释的回传时间常常被忽略相同的原因,MECS 向 UE 回传结果的耗能也总可以忽略不计,边缘计算仍然能大大的降低总能耗。

#### 1.2.4 综合优化时延和能耗

北京邮电大学的李季提出了将时延和能耗加权求和后的值作为优化指标的方法<sup>[9]</sup>,采用 Q-learning 和 Deep Q-learning 分别进行学习、优化检测。并且在其研究的情景中,不同用户奖励函数中对时延和能耗加权重允许不同,以考虑进不同用户服务对降低时延和能耗的偏好性与需求轻重程度,同时权重也有统一量纲的作用。Chen 等人同样以能耗和时延按一定权重所求的和的最小化作为优化目标<sup>[10]</sup>,并且其考虑了在边缘服务器不能很好的胜任卸载任务时,如某 MECS 服务区域内有太多用户同时请求边缘计算卸载服务,将计算任务进一步卸载到中央云服务器的情况。

#### 1.2.5 深度强化学习优化算法的应用

现有的传统算法对于解决 MEC 计算卸载决策与资源分配问题是可行的,但是 MEC 系统有较高的对响应时间的需求,故传统所需计算时间较大的迭代式优化算法往往不太适用。由于所建模型的简单性,一些基于基本逻辑判断的算法甚至都取得了不错的结果,但考虑真实场景下的应用就显得有些难以胜任。深度强化学习(Deep Reinforcement Learning, DRL)很适合解决决策问题,事实上到现在就有许多工作是基于强化学习(RL)技术来研究 MEC 计算卸载的<sup>[3,6,11]</sup>。

Jie Xu 等人考虑了偏远地区只依靠当地产生的可再生能源这种不太可靠甚至断续供应的能源作为唯一能量供应的边缘计算的应用场景<sup>[11]</sup>,基于马尔科夫决策过程(Markov



Decision Process, MDP)和强化学习算法,提出了一种基于决策后状态的在线学习(Online Learning)的算法——PDS based learning algorithm, (PDS: post-decision state), 并将之与作为基准(benchmark)的 Q-learning 算法, 短视(Myopic)算法和固定卸载量的算法(Fixed power)比较。文章所定义的短视算法忽略了系统状态和决策的时序上的关联性, 仅在每一个时隙(time slot)上根据当前的状态利用现有能获得的电池能量最小化损失函数; 固定卸载量的算法在任何剩余能量允许的条件下将固定的能量分给边缘计算。Jie Xu 等表示他们的算法的关键在于把离线的迭代值和在线的强化学习解耦, 仿真结果最终 PDS 在线学习算法和 Q-learning 算法的能耗分别为不采用边缘计算的 12.6%和 16.7%。

另外随着移动用户数目的增加, 采用去中心化计算卸载(Decentralized computation offloading)把决策任务交由每个 UE 去做会比由边缘服务器统一卸载决策来做会在减轻 MECS 与 UE 的整体任务量, 而且能提高系统的可扩展性, 这种去中心化方式的卸载决策也正吸引着越来越多的研究目光<sup>[3,12,13]</sup>。Zhao Chen 等人将边缘计算卸载决策的任务完全交由每一个 UE 自己做出<sup>[3]</sup>, 在他们研究的单服务器单基站模型中, UE 所能获知到的系统状态量有本地任务堆栈情况、信道信噪比 SINR (Signal to Interference plus Noise Ratio)和信道状态信息 CSI(Channel State Information)。UE 的决策输出是一个表示在本地或在 MECS 执行计算的二元组。然而从当前的研究现状中可以得知, 去中心化卸载决策能提高系统的扩展性, 但是 UE 如果不获知边缘服务器的使用情况, 不考虑其他 UE 的数目与卸载需求, 独立地做出卸载决策, 是不太可能做到全局指标的最优化的。

### 1.2.6 存在的问题

对于部分卸载技术的研究在此不做过多讨论, 实际上, 现在部分卸载计算的研究规模较之全部卸载来说是比较有限的, 部分原因可能是部分卸载要考虑的因素太多, 诸如程序切割, 任务的可卸载性, 太多可能的选择, 程序块之间的依赖关系等。总结来看, 目前大部分的卸载决策算法都是以在可接受的执行延迟下最小化 UE 能耗, 或者权衡考虑时延能耗两个指标的优化为目来进行的。而且基本上所有的研究都是用建模仿真的方式评估自己的算法的, 只有少部分给出了理论分析。

MEC 系统中的任务调度要受 MECS 算力、通信资源等等的限制, 在过去的诸多研究中, 所建立的过于理想化的系统模型往往会忽略这些或那些资源上的限制, 这方面还



期待着更进一步的考虑更为全面的研究。对于前人的有些相对简单的模型,传统的决策方法尚可胜任,但是在资源受限的复杂真实环境中规则过于简单的决策算法则可能无法给出合理的决策。本毕设认为 DRL 算法因其具有挖掘深层信息的能力,更能胜任资源受限条件下的 MEC 卸载决策问题。

### 1.3 研究目标和研究内容

这一节将概述此次研究的目标和内容,各种资源受限并且受随机因素影响大的仿真模型和将 DDPG 用于 MEC 卸载决策是主要的创新点,最终的实验结果也肯定了仿真模型和决策算法的有效性。

#### 1.3.1 研究目标

本毕设的研究目标是在移动边缘计算(Mobile Edge Computing, MEC)的应用背景下,针对有限的计算资源、通信资源与用户设备能量储备,首先搭建一个考虑了计算资源、电源能量限制的多基站单服务器的仿真模型,然后基于深度强化学习的方法设计有效的整体计算卸载策略,从而使得本毕设定义的考虑了能耗和时延的综合反应 QoS 的损失函数最小化。

#### 1.3.2 研究内容

##### 1. 建立资源受限的单服务器多基站 MEC 系统的仿真模型

考虑边缘计算系统中有一个算力资源有限的边缘服务器、同时存在一个宏基站多个微基站作为移动边缘服务器与 UE 之间的通信中介,并设定每个基站有多个信号增益不同的通频带。UE 的各项参数随机取值,每一个时隙服从二项分布随机出现待卸载决策的计算任务,计算任务的参数也是随机取值的,将分由本地计算模型和卸载计算模型两类进行讨论。

##### 2. 建立优化问题的模型

本毕设定义了考虑时延和能耗的全局优化指标,并将任务超出最大容许时延仍未解决的情况进行的惩罚体现在优化函数上。将 MEC 计算卸载决策构建为一个马尔科夫





(Markov)决策过程优化问题, 如果不考虑多个 UE 在同一个仿真时隙面临决策任务的情况, 基本上每个任务来临都会对应一个过程状态转移, 由卸载决策算法给出决策动作, 在下一个转移时收到回报。

### 3. 决策方法与训练

考虑为应对 DRL 的输入输出维数的固定性与 MEC 用户的时变性的矛盾, 本毕设采用深度确定策略梯度下降(Deep Deterministic Policy Gradient, DDPG)的深度强化学习优化算法来解决。收集到卸载请求 UE 的状态和各种资源占用程度后, 再由最终的决策网络给出对每个用户卸载请求的答复, 包括是否允许其卸载计算、分配的卸载通道、分配的 MECS 计算资源。并探讨用历史记忆学习的方式综合训练优化卸载决策网络。

## 1.4 论文组织结构

在本章中, 介绍了边缘计算计算卸载问题的研究背景与研究现状: 建模仿真的研究手段和追求低时延低能耗的卸载决策目标、吸引了越来越多的关注的基于深度强化学习决策算法、越来越受重视的决策算法可扩展性。阐述了本课题的意义所在, 说明了 MEC 仿真模型、卸载决策算法两方面的研究的目标。

第二章将讨论构建单服务器多基站的边缘计算仿真模型, 并详细介绍模型中各个参数的意义, 分别就卸载计算和本地计算两种任务执行模式给出分析, 给出时延能耗的构成。然后, 将卸载决策任务表示成了部分可观马尔可夫决策过程, 定义其状态、动作、回报, 使面临的卸载决策问题能得到规范化的表示。

第三章首先概述了深度强化学习的一些背景知识, 然后讨论了深度确定策略梯度下降的强化学习算法, 基于其解决连续动作空间的能力而被本项目采用, 并给出了所采用的决策算法中演员网络和批评家网络的网络结构, 对于决策算法和仿真环境的接口也给予了必要说明。最后又对实验中各种参数的取值列表给出。

第四章给出了实验测试的结果, 以全本地和全卸载为对照, 在 UE 数和任务密度不同的情况下多次进行实验, 随机又分析了时延、能耗、任务失败率的实验结果以说明所提决策算法的有效性。最后, 又总结了项目的优点与不足, 并对基于此毕设能做的工作做了展望。

## 2 边缘计算模型与计算卸载决策

### 2.1 边缘计算仿真模型概述

移动边缘计算中,对于随机到来的计算任务,决策结果可能是在本地执行,也可能是卸载到边缘服务器去执行,本章将分别讨论这两种情况。阐述用于刻画 QoS 的优化指标时延、能耗的组成和计算方法。并且,将给出此卸载决策问题可被视为一般的马尔科夫决策问题的分析论述,使得决策可以由通用的强化学习方法来解决。

本模型针对单小区多用户的场景下,单服务器多基站的边缘计算卸载问题建模。移动设备的本地算力取值范围设置的较边缘服务器来说小很多,行文中算力一词与表示运算速度的 CPU 周期同义。MECS 的全部算力可以同时分配给多个正在其上卸载执行的任务共享,而本地算力只能支持一个计算任务执行。

一般来讲,采用边缘计算卸载计算任务到 MECS 上去的目的是追求比仅仅依靠本地资源更优的延时和能耗。对于每一个计算任务,可以用计算量和数据量完备地刻画:用计算量代表完成此任务所需的 CPU 周期——一般不是设备敏感的;数据量表示此任务的输入和程序数据的数据大小,这也是当卸载执行该任务时 UE 需要上行传输地数据大小。

### 2.2 本地处理模型

本毕设用两个量来刻画任务  $T$ ,即前文所述的计算任务的数据量  $d$  和计算所需的 CPU cycle  $c$ ,同时设置宏超参数任务的最大容许时延  $t$ 。如果某一用户设备  $UE_n$  选择在本地执行其计算任务  $T_n(d_n, c_n)$ 。在此定义  $t_n^l$  为其本地执行延时,他只包括本地 CPU 的执行时间。在此表示  $f_n^l$  为其本地每秒 CPU cycle 数,则

$$t_n^l = c_n / f_n^l \quad (2.1)$$

再用  $z_n^l$  表示  $UE_n$  的能量效率,与 CPU 频率的平方成正比,比例系数称为能量密度,因设备而异,即完成单位 CPU 周期的耗能,则选择本地执行的能耗  $e_n^l$  可以被解出来:

$$e_n^l = c_n z_n^l \quad (2.2)$$

本毕设假设每个 UE 的本地 CPU 和上行数据通信在一定时刻只能服务于一个任务,

这就导致当有早先到来的任务在本地执行时，新的待卸载决策任务不可以在本地执行。如果决策算法重复分配被占用的本地算力和上行数据通信能力，则此决策无效，且将待解决且为失败的任务建立 **buffer** 来留待资源空闲条件允许时完成之，并将 **buffer** 中自其产生以来超过最大容许时延的任务判为失败，并从 **buffer** 中删除之，因 MEC 应用场景所面向的一般是高 QoS 要求的任务，这样做是有其合理性的。对于决定在本地计算的任务，若本地电源不足以支撑计算任务的完成时，则任务失败。

这样，本地执行计算的能耗、时延就都可以表示出来了。

### 2.3 卸载处理模型

本毕设考虑的模型中有唯一的边缘服务器，其用两个指标来完备地表示，即代表其计算能力的每秒最大 CPU 周期数  $F$  和最小有效可分配 CPU 频率，服务器不会划拨小于此最小频率的算力给卸载任务。而在系统的通信网络上，则假设采取频分多址的方式，由多个距离 MECS 在网络中远近不同的微基站( Small Base Station, SBS)和一个与 MECS 部署在同一位置的宏基站( Macro Base Station, MBS)共同组成了在 MECS 与 UE 间可选的通信链路，在此考虑每个基站都有相同数目的通带，每个通带的信号增益均不同。

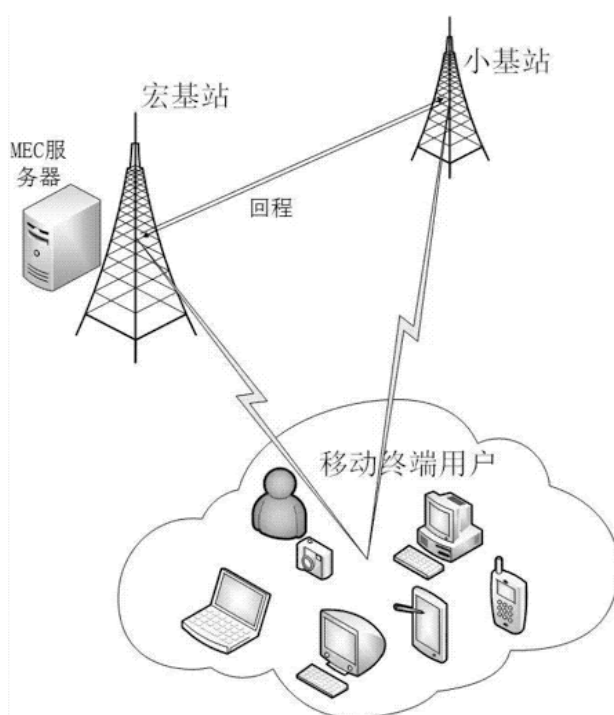


图 2.1 边缘计算任务卸载示意图

如果经过决策 $UE_n$ 选择卸载计算来执行任务 $T_n(d_n, c_n)$ ，则整个卸载计算将分为三个步骤，即数据上传到 MECS、MECS 完成计算任务、MECS 执行结果回传到 UE。

数据上传到 MECS 的过程可分为从 UE 到基站和从基站到 MECS 两步。由公式

$$r_n = W \log_2(1 + \frac{P_n g_n}{\sigma^2}) \quad (2.3)$$

可求出 $UE_n$ 到基站的数据上传速率，式中 $W$ 为 $UE_n$ 被分到的无线信道带宽， $P_n$ 为上传数据时 $UE_n$ 的传输功率， $g_n$ 为信道的增益，每个基站都有多个增益不同的信道供选择， $\sigma^2$ 为信道的复杂高斯白噪声方差。则计算任务数据从 $UE_n$ 到基站上传的时间花销可表示为：

$$t_n^{o,t1} = \frac{d_n}{r_n} \quad (2.4)$$

而数据从基站到服务器的传输过程中将花费的时间为

$$t_n^{o,t2} = d_n \varphi_n \quad (2.5)$$

其中 $\varphi_n$ 是衡量 $UE_n$ 传输数据所经的中继通讯基站到 MECS 的传输速度的参数，与网络中基站到 MECS 的距离有关，因宏基站与 MECS 部署在一起，如果该基站是 MBS， $\varphi_n = 0$ 。

作为执行延迟的另一个组成的 MECS 处理完计算任务的时间消耗为

$$t_n^{o,p} = \frac{c_n}{f_n} \quad (2.6)$$

其中 $f_n$ 为 MECS 分配给 $UE_n$ 的计算资源数，单位是每秒 CPU cycle 数。对所有 UE 分配的计算资源求和，不得大于 F。考虑到运算结果回传的数据一般会远小于任务卸载时上传的数据，在此处本毕设忽略掉数据回传的过程中所花的时间。从而选择卸载计算的总时间消耗就可以表示为：

$$t_n^o = t_n^{o,t1} + t_n^{o,t2} + t_n^{o,p} \quad (2.7)$$

再来讨论卸载过程中 UE 的能耗，首先注意到的是任务上传到基站会需要 UE 发射无线信号，用 $P_n^{send}$ 表示 UE 发送数据时的功率，此步骤的能量开销大概为：

$$e_n^{o,t} = P_n^{send} t_n^{o,t1} \quad (2.8)$$

而从基站到 MECS 的传输过程和 MECS 执行计算任务的过程中，UE 均处于等待状态，也需消耗一小部分能量。定义等待状态 $UE_n$ 的功率为 $P_n^l$ ，则因这段等待的时间而引入的 UE 的能量消耗为：

$$e_n^{o,p} = P_n^l(t_n^{o,t2} + t_n^{o,p}) \quad (2.9)$$

从而选择卸载计算的总开销也知道了:

$$e_n^o = e_n^{o,t} + e_n^{o,p} \quad (2.10)$$

但由于未收到可卸载任务时 UE 亦会消耗待机能量, 故而在计算损失时只考虑相对多出的通信所需能量, 即:

$$e_n^o = e_n^{o,t} \quad (2.11)$$

这是一个与之前的研究均不相同的设定。不考虑卸载决策时等待 MECS 执行完任务后的回传结果这段时间的待机能耗, 这么做会使得能量与时间的耦合性变小, 这部分能量一般相对来说可以忽略, 计算时加入与否对结果影响应该不大。

## 2.4 建立优化问题的模型

表示出本地和卸载两个计算模式的模型后, 现在就可以讨论将计算卸载问题表示成一个马尔可夫决策过程。马尔可夫决策过程是在具有马尔可夫性质的环境中模拟智能体可实现的随机性策略与回报的序贯决策(sequential decision)的数学模型<sup>[14]</sup>, MDP基于一组交互对象, 即智能体和环境进行构建, 所具有的要素包括状态、动作、策略和回报。即使是非马尔可夫或者说是部分可观马尔可夫决策过程都可以用MDP来建模<sup>[14]</sup>, 再应用相关决策算法来得到相当好的解决。

MDP问题具有马尔可夫性, 给定当前状态未来的状态与过去无关, 即满足:

$$P(s_{t+1}|s_t, s_{t-1}, s_{t-2}, \dots, s_0, a_t, a_{t-1}, a_{t-2}, \dots, a_0) = P(s_{t+1}|s_t, a_t) \quad (2.12)$$

下围棋就是一个很好的例子, 棋局上的子的出现次序并不是弈棋者所关心的, 如果不考虑对“下棋风格”的分析, 若干子前的棋局并不该影响当前棋手的选择。而消砖块的破坏者游戏(breaker)就是一个部分MDP的例子, 如果只根据当前帧做出决策的话, 决策者就无法判断球的运动方向。在此, 转移概率与历史状态有关。但是, 如果选取连续的三帧作为系统状态, 则此问题又是一个严格的马尔科夫决策问题了, 可见, 状态的选择能决定决策问题是MDP还是部分可观MDP的。



图 2.2 消方块问题屏幕截图

本文的 MEC 仿真模型卸载决策问题在接下来的讨论中将按 MEP 来考虑, 虽然就下文的定义的其实际上是部分可观 MDP 的。

#### 2.4.1 MDP 动作

决策时隙所有决策行为都发生在决策时隙上, 定义MECS对 $UE_n$ 计算卸载请求的答复为

$$y_n = (R_n, B_n, g_n, f_n) \in \mathcal{Y} = \{0,1\} \times \mathcal{N} \times \mathcal{G} \times (0, F_n] \quad (2.13)$$

下面一一解释四维卸载决策向量 $y_n$ 的各维度的含义。第一个维度上的变量 $R_n$ 在 $\{0,1\}$ 中取值, 表示是否允许 $UE_n$ 进行卸载计算, 当 $R_n = 0$ 时表示决策结果是 $T_n$ 在 $UE_n$ 本地进行计算;  $R_n = 1$ 时表示卸载决策的结果是 $T_n$ 卸载到MECS在边缘服务器进行计算。当 $R_n = 0$ 时, 任务在本地完成, 不需要MEC服务,  $B_n, g_n, f_n$ 的取值无意义。这里 $\mathcal{N}$ 是所有可用的通信基站的编号, 从第零号的MBS到第一号及以后的SBS,  $B_n$ 在 $\mathcal{N}$ 中取值时, 表示将通过编号为 $B_n$ 的基站向MECS卸载此计算任务。第三个维度上的变量 $g_n$ 表示为其分配的该基站上增益为 $g_n$ 的通频带, 第四个维度上的变量 $f_n$ 表示为此计算任务分配的MECS计算资源, 单位为 CPU cycle per seconds。MDP动作的做出应考虑到MECS资源占用情况、通信资源、本地电量等多方面。

#### 2.4.2 MDP 状态

卸载决策算法基于所观察到的状态向量进行决策, 经过多次的改动, 状态向量被确定为

$$x_n = (d_n, c_n, f_n^l, z_n^l, P_n^{send}, C_n^s, C_n^l, O_{N,M}, F^{left}) \in \mathcal{X}^l \quad (2.14)$$

其中 $d_n, c_n, f_n^l, z_n^l, P_n^{send}$ 这些量在本地计算模型中已经讨论过了，值得注意的是对同一个UE，其 $f_n^l, z_n^l, P_n^{send}$ 值不会随时间改变，但不同的UE其值可能不同。

$$C_n^s = \mathbf{1}(UE_n \text{ 数据发送被占用}) \quad (2.15)$$

$$C_n^l = \mathbf{1}(UE_n \text{ 本地算力被占用}) \quad (2.16)$$

这里 $\mathbf{1}(\cdot)$ 表示真值函数，括号内的事件为真其值为一，否则为零。用 $F^{left}$ 表示当前MECS空余的可分配的计算资源，矩阵 $O_{N,M}$ 表示信道占用情况，这里的N行表示编号从0到N-1的所有基站，M列表示每一个基站都有的M个通频带，信道占用矩阵 $O_{N,M}$ 的任意一个元素都有0、1两种可能的取值，元素 $(i,j)$ 取零时表示编号为i的基站的第j个通频带被占用了，元素 $(i,j)$ 取一时表示编号为i的基站的第j个通频带没有被占用，允许分配给请求边缘计算的UE进行数据传输，在送入状态向量时， $O_{N,M}$ 将被拉伸成一维长向量。

### 2.4.3 MDP 回报

对于优化指标，选取总损失函数为系统中所有UE的损失函数之和： $Cost^{all} = \sum_{n=1}^N Cost_n$ 。每一个任务在其完成时或确定失败时引入的损失定义为

$$Cost_n = \mathbf{1}(T_n \text{ 任务完成})(I^t t_n + I^e e_n + I^{finish}) + I^{fail} \mathbf{1}(T_n \text{ 任务失败}) \quad (2.17)$$

$\mathbf{1}(\cdot)$ 与上面出现时的含义相同，表示真值函数。 $I^t$ 、 $I^e$ 、 $I^{fail}$ 、 $I^{finish}$ 为人为选择的权重系数，还起到了统一量纲的作用。每一个动作( action)的回报( reward)为这个动作做出后到下一个动作做出为止这段时间内所有损失的和，当然这期间也可能不会有任何任务完成或失败，那么回报就是0。

## 2.5 本章小结

在本章中，针对实际的应用场景，分别就本地计算和卸载计算讨论了边缘计算仿真环境的数学模型。考虑了UE本地算力和MECS的算力、基站和通频带等多种资源限制，并且在模型中加入了大量随机因素。通过对计算卸载问题的设计将其转化为一个

马尔可夫决策问题，讨论了此部分可观MDP问题的状态观测量、决策动作量、何时状态转移、以及状态转移的回报。

本次研究设置了离散的时隙，假定所有事件都发生在一个个时隙上，这种做法被广泛采用。我们采用了变周期决策的方式，在前文关于 MDP 模型的讨论中有详细说明，而以往的相关研究中，虽然一般都没有明确说明，大都会采用另一个处理方式，即对决策行为的发生设置固定的决策时隙，并取其为仿真时隙的整数倍，即固定周期决策的方法。由下面的两幅示意图可以清楚的观察到，任务到来后立即进行决策的变周期决策较任务到来后会面临可能的等待时间的固定周期决策有先天的优势。

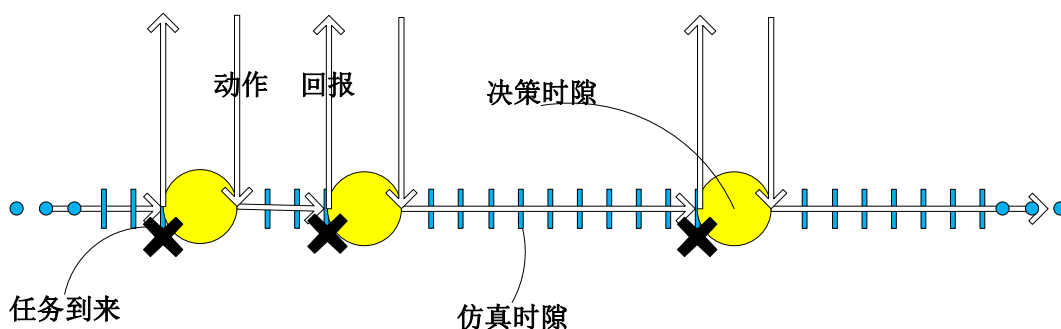


图 2.3 变周期决策

如果是固定周期决策算法，对于决策时隙与仿真时隙的选取，是需要谨慎考虑的。仿真时隙自然是越密越好，而决策时隙不宜取的过密，否则会有大量决策时隙中无待卸载决策任务，也不宜取的过于稀疏，这样会使得从任务到来的时刻到对其卸载决策安排解决的时刻空闲太大，严重影响QoS。由下面的示意图可见，在每个仿真时隙都有可能到来的待卸载决策任务会因算法的先天不足被“耽误”该仿真时隙到下一个决策时隙到来的时间：



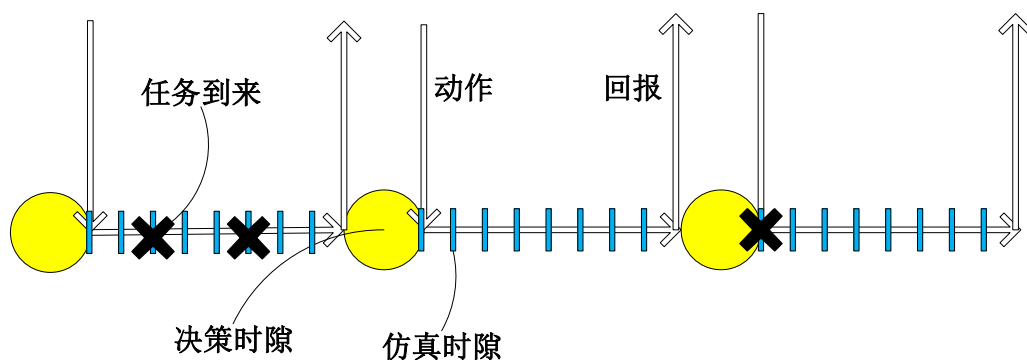


图 2.4 固定周期决策

此次研究的一个目标就是让所建立的仿真环境尽量贴近实际，出于此考虑，本毕设对每个 UE 的 CPU 频率、能量密度( Power Density)、数据传输功率、待机功率、和每个随机到来的任务的数据量和完成此任务所需 CPU 周期这些个超参数，均通过在一定范围内均匀分布( Unified Distribution)的随机值来生成。相信这样的仿真参数的设定较之大多数的相关研究设定所有 UE 所有任务大部分参数均相同的通常做法更能令人信服。笔者自己评价仿真环境考虑的全面性为研究的一大亮点，变周期决策的方式更是具有其先天优势。独立搭建的仿真环境程序也已公开，网页放于致谢部分，期望这能为未来的研究提供便利。



### 3 卸载决策算法分析与实验参数选取

#### 3.1 深度强化学习

无论是在围棋上战胜世界冠军的Alpha Go<sup>[15]</sup>, 还是波士顿动力那蹒跚学步的仿生机器人, 得益于相关研究技术的逐渐主流化和日益精进的学习理论及计算机性能, 强化学习正越来越体现出其能力的强大。早期的强化学习算法诸如Q-learning和与之类似的Sarsa等, 其中Q-learning是一种非常著名的解决MDP问题的无模型算法, 已经证明对于任意有限的马尔科夫决策问题, 理论上Q-learning最终都能找到最优解。Q-learning需建立Q表, 表格中的每一格的Q值对应每一个状态下每一种可能的动作, 这里Q表示质量( Quality), 故而其只能应付动作有限、状态有限的问题。训练学习算法的过程就是更新Q表的过程, Q表要根据对应状态下所选动作的回报基由贝尔曼方程来更新。测试时查阅Q表, 找到对于现阶段的状态观测来说Q值最大的动作, 并采取它。

近几年来, 基于神经网络的深度学习技术发展突飞猛进, 其应用在我们生活中比比皆是: 刷脸支付、二维码识别、语音助手、机器翻译、偏好推荐系统, 我们的生活已离不开深度学习技术。学界和工业界都呈现出了无与伦比的研究热性和应用活力, 神经网络中利用激活函数将线性变换非线性化, 通过多层非线性变换的叠加, 赋予了网络以拟合“抽象”映射的能力。训练时, 根据大量训练样本, 计算出网络输出与目标输出的差距, 即损失函数——例如分类问题的交叉熵( Cross Entropy)和回归问题的均方差( Mean Square Error), 然后利用梯度下降反向传播更新网络参数, 以期得到想要的网络输入到输出的映射。

深度强化学习将神经网络应用于强化学习算法中, DeepMind团队2013年提出了深度Q学习( Deep Q-learning)算法<sup>[16]</sup>, 这种无模型的算法, 通过观察屏幕图像来做出操作动作, 在雅达利游戏机Atari 2600的六款动画游戏中的三款上取得了超越人类专家的成绩。不再构建Q表储存( 状态, 动作)对的Q值, 而是构建神经网络( Deep Q Network, DQN), 网络输入( 状态, 动作)对, 输出Q值, 此法结构类似于本章下两节中的批评家网络, 训练Q网络的目标是能够评估不同动作的好坏。

### 3.2 确定策略梯度下降学习算法

在上文中已经对于由仿真模型卸载决策提炼出的部分可观马尔科夫决策问题的状态、动作、回报三要素一一作了说明，本章所讨论的基于深度强化学习的决策算法可以解决此决策问题。

由于本次所需要决策的动作空间和状态空间中都有连续变量，虽然深度Q网络学习算法(DQN)能够应对系统状态连续取值的情况，但是其只能在有限的可能动作中抉择，无法适应动作空间也连续的情况。并且因为离散参数的可选组合太多，DQN的遍历成本不可接受，会陷入维度灾难。故而这里拟采用DDPG(Deep Deterministic Policy Gradient Descent)这一策略梯度下降的学习算法<sup>[17]</sup>。

由于在做出行为 $y_n$ 后，该决策涉及到的UE的总损失不是即刻就能观察到的，需要在若干个 $f$ 仿真时隙之后，所有涉及到的UE的计算任务完成或失败了，才能计算出回报，将这若干仿真时隙划分为一个决策时隙，对应一个训练epoch。由著名的动态规划方程(贝尔曼方程, Bellman equation)定义这一具有递归关系的Q函数

$$Q^\mu(x, y) = E_{p, x'}[p(x, y) + \gamma Q^\mu(x', \mu(x'))] \quad (3.1)$$

代表了在状态 $x$ 处采取动作 $y$ 的质量(Quality)， $E$ 代表期望，取期望是因为 $p, x'$ 不是确定的。表达式中的 $p(x, y)$ 为采取该动作的时隙产生的所有UE所有计算任务的回报，注意这里的 $p(x, y)$ 并不是 $x$ 与 $y$ 的确定函数。这里采用参数为 $\theta^Q$ 的批评家网络来估计行为的Q值<sup>[18]</sup>，输入状态和动作，输出其Q值：即 $Q(x, y) \approx Q(x, y; \theta^Q)$ 。强化学习注重长期的回报最大化，对于网络参数的更新算法采用反向传播、梯度下降方式。在训练时都设置一组动作网络参数 $\theta^\mu$ 和一组评价网络参数 $\theta^Q$ 的两个网络，测验时只需用到 $\theta^\mu$ 动作网络。参数为 $\theta^\mu$ 的演员网络 $\mu$ 用于根据状态 $x$ 直接输出决策的动作 $y$ ；参数为 $\theta^Q$ 的批评家网络 $Q$ 用于估计状态 $x$ 下做出动作 $y$ 的Q值，来评价该动作的好坏。

定义批评家网络的损失：

$$L(\theta^Q) = E_{(x, y, p(x, y), x') \in \tilde{\mathcal{O}}^k} [(p(x, y) + \gamma Q(x', \mu(x'; \hat{\theta}^\mu); \hat{\theta}^Q) - Q(x, y; \theta^Q))^2] \quad (3.2)$$

上式中 $\tilde{\mathcal{O}}^k$ 为记忆库，下文将介绍之。 $\hat{\theta}^Q$ 、 $\hat{\theta}^\mu$ 分别为批评家目标网络参数和演员目标网络参数，这两个目标网络的结构分别是对批评家网络和演员网络的拷贝，初始化

时目标网络与原网络的参数相同，只是参数更新方法上略有不同。根据最小化梯度来更新批评家网络参数

$$\theta^Q \leftarrow \theta^Q - \alpha_Q \nabla_{\theta^Q} L(\theta^Q) \quad (3.3)$$

每epoch开始，进行赋值 $\hat{\theta}^Q \leftarrow \tau \theta^Q + (1 - \tau) \hat{\theta}^Q$ 和 $\hat{\theta}^\mu \leftarrow \tau \theta^\mu + (1 - \tau) \hat{\theta}^\mu$ 软更新批评家目标网络，此法设置目标网络的目的是避免训练目标变化过于激烈，演员目标网络也采用之。

第k个epoch的转移过程表示为 $m^k = (x^k, y^k, p(x^k, y^k), x^{k+1})$ ，存储最近的U个转移过程到记忆库中 $\mathcal{O}^k = (m^{k-U+1}, \dots, m^k)$ ，每epoch决策网络都会从记忆库中随机采样出一个大小为S的子集( mini-batch) $\tilde{\mathcal{O}}^k \subseteq \mathcal{O}^k$ 来在 $L(\theta^{k+1})$ 降低的方向训练批评家网络，这样随机地在历史记忆库中取出训练数据的方式加上前文对批评家目标网络和演员目标网络的软更新方式可以很好的防止训练发散。而演员网络的训练则是要根据梯度

$$\nabla_{\theta^\mu} J = E_{(x,y,p(x,y),x') \in \tilde{\mathcal{O}}^k} [\nabla_y Q(x, y; \theta) \nabla_{\theta^\mu} \mu(x; \theta^\mu)] \quad (3.4)$$

来更新参数

$$\theta^\mu \leftarrow \theta^\mu + \alpha_\mu \nabla_{\theta^\mu} J \quad (3.5)$$

考虑解决所有强化学习模型都面临的探索资源( Exploration)和利用资源( Exploitation)之间的权衡( trade-off)，在训练时增加由奥恩斯坦-乌伦贝克过程(Ornstein-Uhlenbeck process)所决定的随机噪声到决策动作上，得到有一定探索性的输出：

$$\mu(x; \theta^\mu) + \Delta \mu \quad (3.6)$$

对于决策网络，状态输入并没有包括各基站到MECS的传输速度和各通频带的增益，这是可以接受的，原因在于这两个量在系统中不会发生变化。在每一个仿真时隙，如果任意一个UE的待卸载任务队列不为空，都要对其中的第一个任务进行卸载决策。将上文论及的 $x_n = (d_n, c_n, f_n^l, z_n^l, p_n^{send}, C_n^s, C_n^l, O_{N,M}, F^{left})$ 作为网络输入，输出卸载决策动作。

### 3.3 网络结构及其接口

在本节将描述演员网络和批评家网络各自的结构，以及更重要的，如何将输出的动作向量转化为仿真系统可执行的决策动作。



演员网络采用三层网络结构，如段落末尾的图所示，注意示意图中的全连接层只代表线性变换。考虑到状态输入向量 $x_n$ 的各个元素的取值范围迥异，这给网络的训练带来了困难，因为不同尺度的特征适配于不同大小的学习率，为应对这一问题，采用批标准化(Batch Normalization, BN)这一常用于神经网络中的特征放缩算法<sup>[19]</sup>。所谓批标准化，在训练时传入的数据以一定数量的批一批一批的处理，对数据的每个特征独立地做线性放缩平移，减去批的均值再除以批的标准差，得到中间数据：

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}} \quad (3.7)$$

上式中的 $Var[\cdot]$ 代表求方差， $E[\cdot]$ 代表求均值，上角标 $k$ 代表网络层数。然后再乘上可训权重(weight)参数，加上可训偏差(bias)参数，得到BN最终输出：

$$y^{(k)} = \gamma^{(k)} \hat{x}^{(k)} + \beta^{(k)} \quad (3.8)$$

通过BN层来规范化各特征值的尺度，还有一定的防止过拟合的作用。隐藏层采用线性整流函数(Rectified linear unit, ReLU)作为激活函数，输出层采用双曲正切(tanh)激活函数，

$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \quad (3.9)$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3.10)$$

输出动作向量每一个元素的范围本应为-1到1。但是因为训练时在原始输出上加入了探索噪声，从而使得输出的范围不确定，可能位于 $[-1, 1]$ 区间之外。

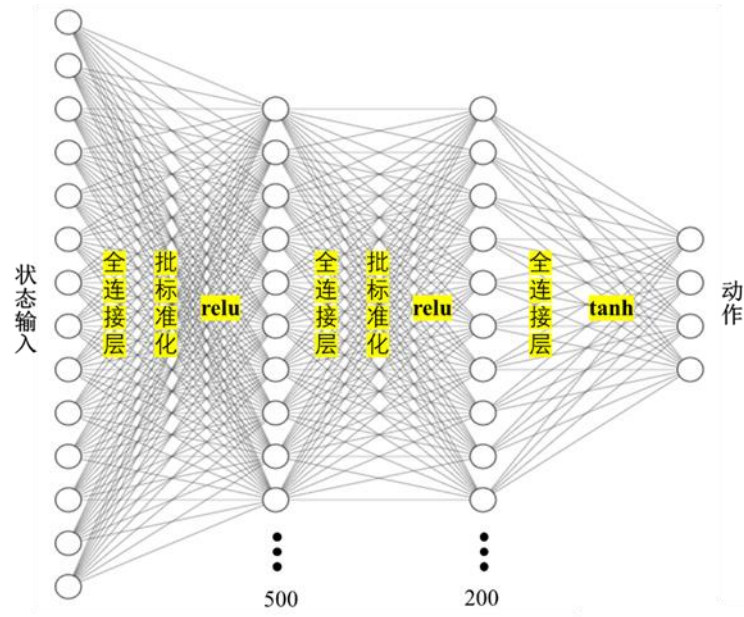


图 3.1 演员网络

演员网络的输出需要经过处理才能和前文所讨论的动作向量  $y_n = (R_n, B_n, g_n, f_n) \in \mathcal{Y} = \{0,1\} \times \mathcal{N} \times \mathcal{G} \times (0, F_n]$  供仿真系统执行：首先为应对噪声的加入，先进行界约束，对于大于0.9999999的元素和小于-0.9999999的元素，分别取这两个边缘值。然后进行放缩，把原来范围为  $(-1, 1)$  的元素分别放缩为范围在  $(0, 2)$ 、 $(0, M)$ 、 $(0, N)$ 、 $(0, F_n)$ ，并且设定  $f_n$  在小于MECS最小可分配单元时修改  $f_n$  为此值。最后对前三个元素向下取整，得到最终的动作向量，范围刚好与前文动作空间  $\{0,1\} \times \mathcal{N} \times \mathcal{G} \times (0, F_n]$  相契合。如前文所述，基站数目为  $N$ ，每个基站通频带数目为  $M$ 。

批评家网络的功能是评价演员网络，输入状态和动作，输出  $Q$  值，其结构图见于段末。构建网络时首先分别提炼动作输入和状态输入中的深层特征信息，因为状态的模式略显复杂，故其所经过的提取特征的网络也较深。因为这里的动作输入是加入了探索噪声的演员网络的输出，基本上在  $(-1, 1)$  之间，从而对于动作输入没必要做批标准化处理。对于两组张量流 (tensor flow) 在其维数相等时相加，经过后半段网络输出  $Q$  值。

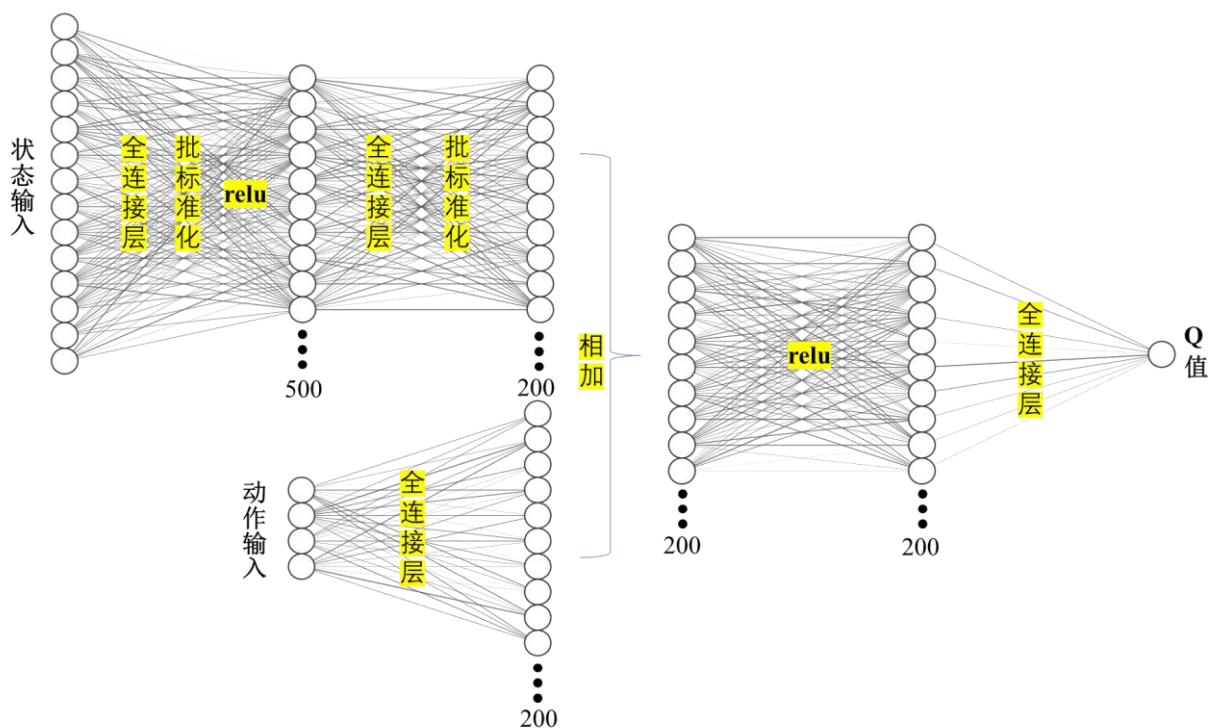


图 3.2 批评家网络

需要指出，并不是所有的决策算法给出的动作都可被执行。当决定卸载时可能面临 MECS 空闲算力不足、信道资源已被占用的情况，决定本地执行时也可能面临本地 CPU 被更早的任务占用的情况。若动作无效，不会被执行，任务保存在对应 UE 的任务缓冲区(Buffer)中。若所有 UE 的缓冲区不都为空，重新做出卸载决定，但是如果缓冲区中的任务再超过最大容许时延后，将被清理，任务判为失败。

### 3.4 实验参数的选取

上一章节和本章之前内容讨论了环仿真境卸载决策问题和强化学习决策算法，在下一章介绍实验结果前，在此先给出各类参数值的设置，参数值的选取如下表，其中数组类型的参数量代表每一个列出的参数取值都对应一个物理实例。

表 3.1 仿真参数

	参数量	类型	取值	量纲
时间	仿真时隙(slot)	标量	$10^{-2}$	秒
	训练总仿真时间	标量	10000	秒
	测试总仿真时间	标量	3600	秒



基站	BS2MECS 传输速率	数组	[0, 0.0002, 0.0004]	秒/千字节(秒/kB)
	信道增益	数组	[-4, -12]	分贝(dB)
	信道带宽	标量	2	兆赫兹(MHz)
	信道噪声	标量	$4 \times 10^{-8}$	瓦特(W)
MECS	CPU 频率	标量	20	GHz
	最小 CPU 单元	标量	0.1	GHz
UE	CPU 频率	范围	0.4~2	GHz
	能量密度	范围	1~4	$W/GHz^{-2}$
	数据传输功率	范围	3~8	W
	个数(基础取值)	标量	30	
任务	到来概率(基础取值)	标量	$0.01 \times slot$	--
	数据大小	范围	0.2~10	MB
	计算量	范围	1~50	GHz
	最大容许时延	标量	50	秒
回报	能量系数	标量	-0.0005	$J^{-1}$
	时间系数	标量	-0.1	$秒^{-1}$
	任务失败处罚	标量	-50	--
	任务完成奖励	标量	10	--
网络训练	演员网络学习率	标量	$1 \times 10^{-8}$	--
	批评家网络学习率	标量	$1 \times 10^{-7}$	--
	回报衰减系数	标量	0.99	--
	参数软更新系数	标量	$1 \times 10^{-5}$	--
	BN 处理批大小	标量	64	--
	记忆库大小	标量	1000000	--

注：任务到来概率为每一个仿真时隙每一个 UE 收到任务的概率。

由上面的数据可以算得，本地执行的最大时延(不包括等待时间)为 100 秒，最小时延为 0.25 秒。本地执行的最大能量消耗为 800 焦耳，最小能耗为 0.5 焦耳。选择卸载执





行并且调用边缘服务器的全部算力的最大时延约为 8.195 秒,。最小时延约为 0.082 秒。

### 3.4 本章小结

本章介绍了卸载决策所使用的深度强化学习算法,详细讨论了决策网络的训练方式和网络结构,并给出了仿真参数的取值。因为问题的数据形式不是太复杂,网络结构并不需太深就足以应付。对于决策网络和仿真模型这两者之间的接口,神经网络的输出是连续取值的,对其分元素进行放缩或者离散化这样的处理,以适配仿真环境的要求。



## 4 实验结果与结论

### 4.1 仿真结果与分析

#### 4.1.1 实验补充说明

仿真参数的选取对于实验结果影响十分之大，在取定仿真的参数值时，参考了其他的研究工作仿真值的设定、也通过网络搜索了相关电子设备的参数，最后通过大量实验敲定了下来。而参考相关研究后，发现一般 MEC 移动边缘计算所面临的应用场景中秒级的时延就已经较为可观了，这在 VR 或 NLP 等 MEC 的潜在服务任务中已经能够使得用户收到很差的体验质量。在如此短的时间内电量的消耗是非常有限的，不太可能出现因电量不足而导致决定在本地执行的任务无法完成的情况。

在此对于一些对于仿真结果可能有影响的一些设置加以补充说明，首先是任务卸载申请机制：在每个仿真时隙中，已随机的顺序循环遍历所有 UE。对每个 UE 都检查其是否有新任务生成，如果有，就将新任务压入任务缓冲区队列中。当任务缓冲区不为空，则将其中的第一个任务视为本时隙的申请卸载任务，并不予理会此后本时隙的其他 UE 的任务缓冲区中的任务。如此，每个仿真时隙最多只能做出一个卸载决策。

对于任务缓冲区，任务到来时会为之添加新元素，当为其中的任务分配了有效的决策方案后，会删除缓冲区中的该任务。无效的决策方案包括本地已被分配执行中的任务后的本地决策、UE 通信资源被占用或信道被占用或所分配的 MECS 算力超出其空闲算力的卸载决策，在进行有效的决策后，如果计算出的任务完成时间超过其最大容许时间限制，本算法不重新给此任务在之后的时隙重新决策，而是直接删除该任务，认定任务失败，并立刻结算奖励惩罚。

本章将用实验结果说明所提出的算法的有效性，在 UE 数和任务到来密度取多种值的情况下，采用训练完的决策网络 and 全卸载、全本地两个对照组各自进行决策，分别给出时延、能耗、任务失败概率的曲线。决策网络已在上一章中讨论过了，训练决策网络时设置 UE 数 30，每用户每秒任务到来的期望为 0.01 个。作为对照组的全本地和全卸载两个简单决策算法，如果有多个待卸载决策的任务，实验时仍在每个仿真时隙只对其

一个做出在本地执行或卸载执行的决策。全卸载决策的通信渠道为宏基站的衰减最小的频带，并分配 MECS 全部算力于此任务。

值得说明的一点是，因为每个 UE 的 CPU 频率、通信频率、能量密度等参数在环境初始化时是随机生成的，并在每次仿真中不在改变，故而即使是相同参数的仿真环境都会有所差距，使仿真结果具有较大的随机性，这在 UE 数目较少时的影响尤其大，鉴于此，可视化图中的每一个数据点都是由三次独立重复实验的结果取平均得来的。

#### 4.1.2 结果可视化与分析

下面所有三组图中实线为神经网络所做卸载决策所得曲线、点状虚线代表全本地决策、段状虚线代表全卸载决策。基于控制变量法的原则，左侧的图每个 UE 的任务到来期望是 0.01 个每秒、右侧的图 UE 数是 30。时延的单位是秒，能耗的单位是焦耳。

##### 1. 时延曲线与分析

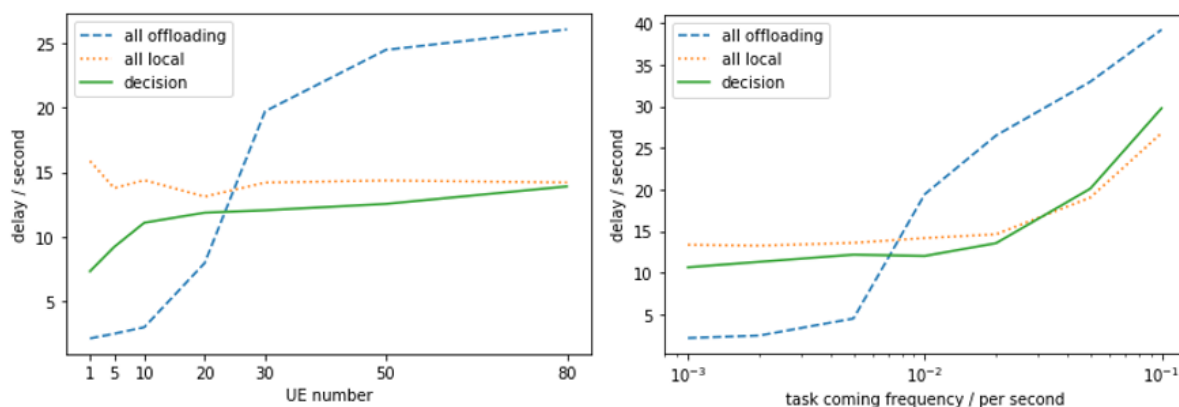


图 4.1 时延曲线

理论上，全本地卸载的实验结果与 UE 数目无关，上面左图也基本反映了这一点。随着仿真 UE 数的增加，全卸载决策会因拥堵造成的等待时间而使时延增大。和任务到来密度的增加对卸载和本地都会造成拥堵。可以看到，决策算法在大多数情况下是可以降低时延的。

##### 2. 能耗曲线与分析

在最初的设计中，还加入了对设备电量的仿真，并设置电量归零的奖励惩罚。但通过前期对 UE 充放电的仿真模拟，发现在所构建的边缘计算仿真环境中，因完成计算卸载任务的电量的消耗并不太能够使得 UE 电池达到无电量，这可能部分是由小时级的仿

真时间不算太长所导致。实际上,在本文所取得仿真参数值下,无论是决定卸载还是本地执行所引入的能耗都并不是很大,因此训练决策网络时奖励函数对于能耗的权重所取值较小,从而下图可见算法降低能耗的效果也并不是太显著。

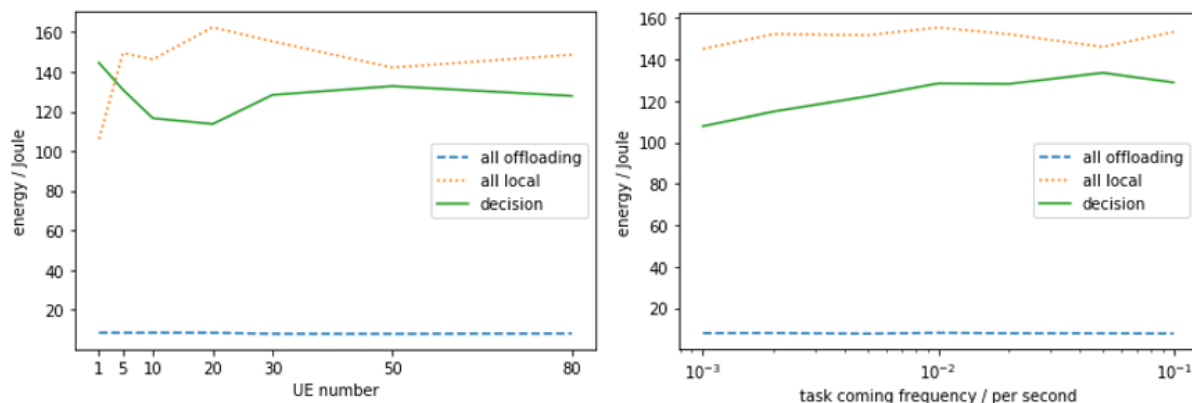


图 4.2 能耗曲线

从实验结果可以看出,选择本地完成计算任务较之卸载执行的 UE 能耗差距巨大,毕设所提算法的能耗基本上在二者之间。

### 3. 任务失败率曲线与分析

从下图可以看出,UE 数增加,全卸载决策的任务失败率会激增,全本地决策失败率基本恒定在较低值。任务到来密度增加,所有决策算法任务失败概率均会增大。而在训练网络时就赋予了任务失败 reward 惩罚以较大值,下图显示在 UE 数为 30、任务到来密度为 0.01 的位置上,本地决策的任务失败率明显好于卸载决策,故而在 UE 数为 30、任务到来密度为 0.01 的参数取值下训练出来的决策网络会更倾向于在本地执行计算任务。

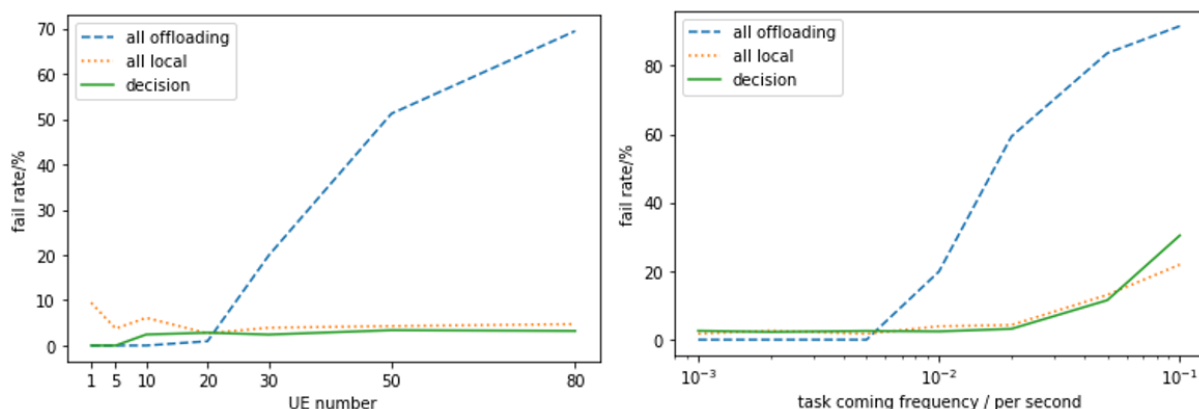


图 4.3 任务失败率曲线



## 4.2 工作总结与展望

### 4.2.1 工作总结

项目搭建了一个单服务器多基站的边缘计算仿真环境,综合考虑了多种资源限制,并赋予了其以极大的随机变化性。将降低时延、能耗、任务失败概率作为目标,利用 DDPG 强化学习算法训练了变周期卸载决策网络,并通过实验说明了其可行性。开题最初既定的目标基本完成。

#### 1. 项目亮点

本次研究采用了变周期决策的方式,在任务队列不拥挤时任务到来立刻进行卸载决策,解决了以往固定周期决策方法引入额外决策等待时间的问题。基于深度强化学习的决策算法,可以通过在训练时改变回报函数的组成,而很方便的调整对能耗、时延、任务失败率的偏好。通过实验还表明,决策算法具有一定的可扩展性。

正如现实中所有移动设备的性能不会完全相同,实际应用场景充满了变化性与不确定性。出于让所建立的仿真环境贴合实际的目的,从环境的初始化到随机到来的任务的参数,均被赋予了很大的随机性。模型还赋予了 MECS 以在最低阈值以上灵活分配给各卸载任务不同 CPU 周期的能力,考虑了诸如 BS 和通频道、本地通信于算力等多种资源的限制。

#### 2. 不足之处

通过上面的可视化结果和分析,基本可以看到所提出的卸载决策算法确实行之有效,但可能因为强化学习算法训练时间较短和参数选择的问题,结果还有一定的提升空间,深度学习之所以行之有效,得益于海量的训练数据,训练时间略显紧张也可能导致网络决策算法效果欠佳。不过本毕设方法性的工作已经达到了。其实,用在固定 UE 数和任务密度条件下训练出的决策网络来应对 UE 数和任务密度千变万化的测试环境,说服力还是欠佳的,这也是本毕设的一个缺点。

项目目前还存在其他许多值得改进的地方,随机任务生成概率各个 UE 相同,这一设置是出于模型的简单,并没有什么道理。0.01 秒的仿真时隙可能过于粗糙,最快 0.01 秒才能做出一个决策,这在用户数和任务到来概率比较大时显现出了一些缺陷。



#### 4.2.2 未来展望

本小节主要讨论针对此研究的一些改进想法。

鉴于：1. 仿真环境中设置不同的 UE 数和任务到来概率可以使得哪怕同样的观测到的状态量，对其最优的动作都不尽相同。2. 现实中要面对的应用场景中，UE 的行为随日升日落和社会活动具有时变性。可以：1. 设置 UE 一定数目范围内随机的变化；在仿真环境中设置由诸如 OU 随机过程决定或者正弦变化的时变的 UE 任务密度，即参数漂移(Covariate Shift)。2. 做出每一个决策都最好考虑之前和计划之后的处境，采用 LSTM+深度学习的序贯决策算法。

有一些比较值得去做的小改动：对待解决的任务打上优先级别，每执行一次无效决策此任务优先级降低一等，每时隙遍历 UE 查阅各任务缓冲区时优先对优先级高的任务进行卸载决策，这样做可以避免大量连续的无效决策。

另外，对于微基站、宏基站共存的通信架构，目前使所有用户都能通过任何一个基站的任何增益的通频带与 MECS 进行通讯的设定不是太科学，在一个 MECS 的服务区域内下一步可以设置微基站覆盖局部，宏基站覆盖全局，并且 UE 在各微基站的服务区内依一定的转移概率发生移动。

未来的对于本课题的改进工作中还可以将reward衰减设为由时间决定的，而不是现在在每一步状态转移都乘上衰减因子，即将贝尔曼方程改写如下：

$$Q^{\mu}(x, y) = E_{p, x'}[p(x, y) + \gamma^n Q^{\mu}(x', \mu(x'))] \quad (4.1)$$

这在强化学习训练中是笔者所不曾见的，但是似乎更合理一些，考虑到本仿真环境中每一步转移的时间可能不同。另外，对于探索噪声的设置，将引入位置从动作空间中加入替换为参数空间中加入，根据M Plappert等的研究可以提供更有效的训练学习<sup>[20]</sup>，下图直观展示了网络参数噪声(右)与传统探索噪声的对比，引自

<https://openai.com/blog/better-exploration-with-parameter-noise/>。

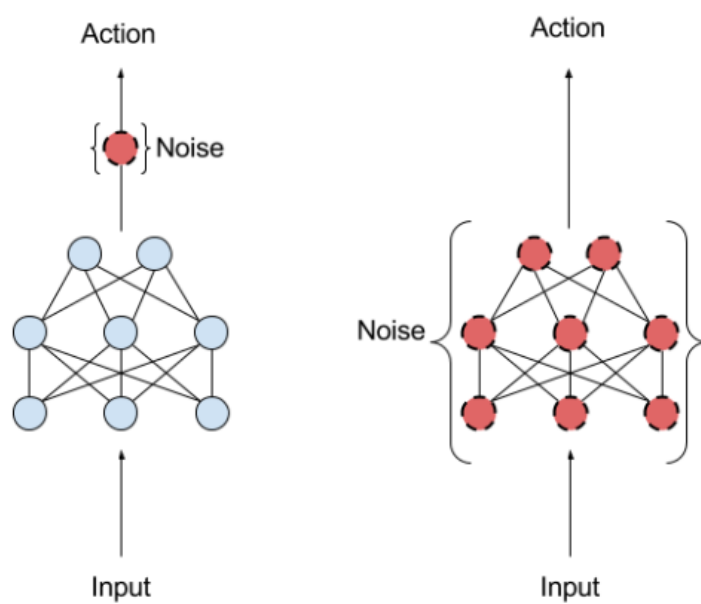


图 4.4 网络噪声加入位置对比

对于边缘计算卸载决策技术的未来，工业界的应用需求应该会越来越大。而学界也期待着统一的仿真环境，但现在边缘计算应用很多具体范式细节多有不定的情况下，一段时间内是达不到的。



## 致谢

首先我想感谢我的导师王岩和实验室的同学雷泽宇、杨钧尧，我总是会由衷地感激王老师的耐心和指导，每一次组会都会督促着我加紧完成任务，实验室已经毕业的黄锐师兄和徐珣璠师姐也曾给我留下了很好的回忆。然后我想感谢教授我本科课程的老师们，秦曾昌老师的课程和网络上李飞飞在斯坦福的开放课程将我引入了模式识别专业的大门，王涛同学在我刚接触专业知识时也给了我很多帮助。

在大三暑假我在南洋理工大学 CIL 实验室作为助理研究员的访问实习经历开启了我科研的钥匙，Marhadika 教授和中文名为火神龙的 Marcos 学长的耐心与治学态度给了我很大的动力，谢任春子学姐帮助了我很多。其间和回国后徐家兴、张琛煜、陈禹帆和邵卫等人给了我很多欢乐的时光。

最后，感谢每一个支持开源精神的研究者，对于 DDPG 强化学习算法的程序，我参考了 Phil 的复现。由于 本文所述的实现程序见于 [https://github.com/lizijian-buaa/my\\_MEC\\_program](https://github.com/lizijian-buaa/my_MEC_program)，环境接口与被广泛用来测试强化学习算法的 openAI gym 的接口格式相同。

人生匆忙而易逝，在北航的大学四年时光历历在目，恍如昨日，纵有诸多遗憾，也已事毕难追。毕业设计消磨了我 2020 年小半年的时光，这正是新冠肺炎防疫工作部署下的半年，收获颇多，这样的日子，也没有了。本科阶段迁于高中时的一些人事，蹉跎了很多年少的岁月。未来有朝重拾动力，且行且坚，无愧初心。





## 参考文献

- [1] Mach P, Becvar Z. Mobile edge computing: A survey on architecture and computation offloading[J]. IEEE Communications Surveys & Tutorials, 2017, 19(3): 1628-1656.
- [2] M. Satyanarayanan P B, R. Caceres and N. Davies. The Case for VM-Based Cloudlets in Mobile Computing[J]. IEEE Pervasive Computing, Oct.-Dec. 2009, 8: 14-23
- [3] Chen Z, Wang X. Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach[J]. arXiv preprint arXiv:1812.07394, 2018.
- [4] Y. Zhang H L, L. Jiao, and X. Fu. To offload or not to offload: an efficient code partition algorithm for mobile cloud computing[C]. 1st International Conference on Cloud Networking (CLOUDNET), 2012: 80-86.
- [5] 王文文. 基于深度强化学习的边缘服务动态部署策略研究[D]. 浙江大学, 2019.
- [6] Chen X, Zhang H, Wu C, et al. Performance optimization in mobile-edge computing via deep reinforcement learning[C]. 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall), 2018: 1-6.
- [7] 李季. 基于深度强化学习的移动边缘计算中的计算卸载与资源分配算法研究与实现[D]. 北京邮电大学, 2019.
- [8] Liu J, Mao Y, Zhang J, et al. Delay-optimal computation task scheduling for mobile-edge computing systems[C]. 2016 IEEE International Symposium on Information Theory (ISIT), 2016: 1451-1455.
- [9] Zhang K, Mao Y, Leng S, et al. Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks[J]. IEEE access, 2016, 4: 5896-5907.
- [10] Chen M-H, Liang B, Dong M. A semidefinite relaxation approach to mobile cloud offloading with computing access point[C]. 2015 IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), 2015: 186-190.
- [11] Xu J, Chen L, Ren S. Online learning for offloading and autoscaling in energy harvesting



- mobile edge computing[J]. IEEE Transactions on Cognitive Communications and Networking, 2017, 3(3): 361-373.
- [12] Jošilo S, Dán G. Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks[J]. IEEE Transactions on Mobile Computing, 2018, 18(1): 207-220.
- [13] Pham Q-V, Leanh T, Tran N H, et al. Decentralized computation offloading and resource allocation for mobile-edge computing: A matching game approach[J]. IEEE Access, 2018, 6: 75868-75885.
- [14] Sutton R S a B. Reinforcement learning: An introduction (Second edition)[M]. 2. MIT press, 2018.
- [15] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of go without human knowledge[J]. Nature, 2017, 550(7676): 354-359.
- [16] Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[J]. arXiv preprint arXiv:1312.5602, 2013.
- [17] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. arXiv preprint arXiv:1509.02971, 2015.
- [18] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.
- [19] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[J]. arXiv preprint arXiv:1502.03167, 2015.
- [20] Plappert M, Houthoofd R, Dhariwal P, et al. Parameter space noise for exploration[J]. arXiv preprint arXiv:1706.01905, 2017.