

中图分类号: TP306

单位代号: 10280

密 级: 公开

学 号: 19721829

上海大学



硕士学位论文

SHANGHAI UNIVERSITY
MASTER'S DISSERTATION

题 目	有限资源下的移动边缘计 算系统卸载决策研究
--------	--------------------------

作 者 柳宗云

学科专业 检测技术与自动化装置

导 师 付敬奇

完成日期 2022 年 4 月 14 日

姓 名：柳宗云

学号：19721829

论文题目：有限资源下的移动边缘计算系统卸载决策研究

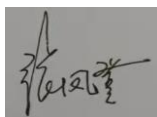
答辩委员会审查要求

上海大学

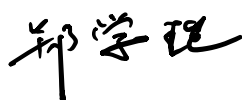
本论文经答辩委员会全体委员审查, 确
认符合上海大学硕士学位论文质量要求。

答辩委员会签名：

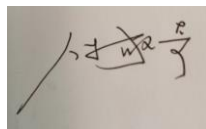
主任：



委员：



导 师：



答辩日期： 2022 年 4 月 14 日

姓 名：柳宗云

学号：19721829

论文题目：有限资源下的移动边缘计算系统卸载决策研究

原创性声明

本人声明：所呈交的论文是本人在导师指导下进行的研究工作。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已发表或撰写过的研究成果。参与同一工作的其他同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签 名：柳宗云 日 期：2022.04.14

本论文使用授权说明

本人完全了解上海大学有关保留、使用学位论文的规定，即：学校有权保留论文及送交论文复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容。

（保密的论文在解密后应遵守此规定）

签 名：柳宗云 导师签名：[Signature] 日期：2022.04.14

上海大学工学硕士学位论文

姓 名：柳宗云

导 师：付敬奇

学科专业：检测技术与自动化装置

上海大学机电工程与自动化学院

2022 年 04 月

A Dissertation Submitted to Shanghai University for the Degree
of Master in Engineering

Research on Offloading Decision for Mobile Edge Computing System with Limited Resources

MA Candidate: Liu Zongyun

Supervisor: Fu Jingqi

Major: Detection Technology and
Automatic Equipment

**School of Mechatronic Engineering and Automation,
Shanghai University**

April 2022

摘 要

随着物联网(Internet of Things, IoT)技术的迅猛发展,越来越多的移动设备需要运行计算密集型任务并接入互联网。然而,由于移动设备通常仅具有有限的电池工作寿命和计算资源,从而导致其无法胜任计算密集型任务的处理工作。如何解决设备终端资源受限与日益复杂的应用服务之间的矛盾已成为通信和互联网领域亟需面临的挑战。受到云计算、雾计算等服务计算框架的启发,一种新型的计算范式——移动边缘计算(Mobile Edge Computing, MEC)应运而生。移动边缘计算采用“在临近用户的网络边缘部署服务资源”的思路,具有低能耗、低延迟、规避核心网拥塞等优点。本文面向多用户-单边缘服务器场景、边云协作的多用户多边缘服务器场景,以能耗、延迟、效用收益等为目标,开展卸载数据、功率控制、资源分配、任务划分卸载决策等研究。取得的主要创新研究成果如下:

1. 针对移动设备忽视信道与边缘云计算资源而导致卸载性能低下乃至任务失败问题,提出了一种有限资源下的能源高效型计算卸载决策方法。首先,采用动态电压缩放和依据移动设备的延迟容忍构建了最佳卸载数据量和本地设备最优计算频率的模型;其次,通过信道资源分配建立了最佳卸载数据量和移动设备发射功率的模型,以及信道资源与最佳卸载数据量的模型;最后,将多变量卸载决策优化问题转换为单变量优化问题,并基于内罚函数法进行求解,提高了卸载策略的实际可行性以及资源配置的效率。仿真对比分析结果表明,提出的方法具有良好的扩展性,并在节约能耗方面具有明显的优势。

2. 针对现有定价策略平均分配边缘云计算资源而忽略用户需求差异问题,提出了一种边缘云卸载决策和 Stackelberg 博弈定价有机融合的一种新方法。首先,采用 Stackelberg 博弈理论建立了用户最佳卸载数据量和购买最佳计算资源块数量的模型,将用户的多变量卸载决策问题转换为单变量优化问题,使用户的卸载决策问题得到简化,并证明了纳什均衡的存在性;其次,运用 KKT 条件实现了用户购买最优计算资源块的卸载决策,建立了边缘云定价和购买最优计算资源块的决策模型,确立了边缘云定价的上下界;最后,基于动态规划的边缘云定价卸载算法将边缘云计算资源以适当粒度进行颗粒化,实现了边缘云效用最优定

价和各用户自身效用的最大化。仿真对比分析结果表明，提出的方法不但实现了边缘云效用和用户效用的均衡，而且具有良好的收敛性和可扩展性。

3. 针对边云协同中多边缘服务器的计算能力未得到充分利用问题，提出了一种基于蚁群算法的多边缘卸载决策优化策略。依据排队论思想并以能耗和时延性能最小化为目标，对移动设备的服务请求卸载速率、边缘云和中心云服务请求到达速率进行了协同建模，以及证明了边缘云服务请求存在最优到达速率；采用二分查找迭代方法对边缘云集群和中心云的服务请求到达速率进行决策，并利用蚁群算法实现了移动设备卸载速率的寻优和服务请求的合理分配，提高了资源的利用效率。仿真对比分析结果表明，提出的方法卸载决策不但能适应更复杂的工况，而且系统综合开销得到降低。

关键词：移动边缘计算，多场景，计算卸载决策，资源分配，任务成本优化

ABSTRACT

With the rapid development of Internet of Things (IoT) technology, more and more mobile devices need to run computation-intensive tasks and access the Internet. However, mobile devices usually have limited battery life and computational resources, which make them unable to handle computationally intensive tasks. How to solve the conflict between the limited resources of device terminals and the increasingly complex application services has become an urgent challenge for the communication and Internet fields. Inspired by service computing frameworks such as cloud computing and fog computing, a new computing paradigm, Mobile Edge Computing (MEC), has emerged. MEC adopts the idea of "deploying service resources at the edge of the network near the users", which has the advantages of low energy consumption, low latency, and avoiding core network congestion. In this paper, we conduct research on offload data, power control, resource allocation, and task division offload decision for multi-user-single-edge server scenario and multi-user-multi-edge server scenario of edge-cloud collaboration with the objectives of energy consumption, latency, and utility gain. The main innovative research results achieved are as follows:

1. To address the problem of low offloading performance and even task failure due to the neglect of channel and edge cloud computing resources by mobile devices, an energy efficient computing offloading decision method with limited resources is proposed. First, the model of optimal offload data amount and optimal computation frequency of local devices is constructed using dynamic voltage scaling and based on the delay tolerance of mobile devices; Second, the model of optimal offload data amount and mobile device transmit power, and the model of channel resources and optimal offload data amount are established by channel resource allocation; Finally, the multivariate offloading decision optimization problem is converted into a univariate optimization problem and solved based on the inner penalty function method, which improves the practical feasibility of the offloading strategy and the efficiency of

resource allocation. The results of the simulation and comparison analysis show that the proposed method has good scalability and has obvious advantages in energy saving.

2. To address the problem that existing pricing strategies allocate edge cloud computing resources equally and ignore user demand differences, a new method is proposed that organically integrates edge cloud offloading decision and Stackelberg game pricing. First, the Stackelberg game theory is used to model the optimal amount of data to be offloaded by users and the optimal number of computing resource blocks to be purchased, which converts the multivariate offloading decision problem of users into a univariate optimization problem, simplifies the offloading decision problem of users, and proves the existence of Nash equilibrium; Second, the KKT condition is applied to realize the offloading decision of users to purchase the optimal computing resource blocks. The decision models of edge cloud pricing and purchase of optimal computing resource blocks are established, and the upper and lower bounds of edge cloud pricing are established; Finally, the edge cloud pricing offloading algorithm based on dynamic programming granularizes edge cloud computing resources with appropriate granularity, and achieves optimal pricing of edge cloud utility and maximization of each user's own utility. The simulation comparison analysis results show that the proposed method not only achieves the balance of edge cloud utility and user utility, but also has good convergence and scalability.

3. To address the problem of underutilized computing capacity of multi-edge servers in edge cloud collaboration, an optimization strategy for multi-edge offloading decision based on ant colony algorithm is proposed. Based on the idea of queuing theory and with the objective of minimizing energy consumption and latency performance, the service request offload rate of mobile devices and the service request arrival rate of edge cloud and central cloud are modeled collaboratively, and it is proved that there is an optimal arrival rate of edge cloud service requests; The dichotomous lookup iterative method is used to make decisions on the service request arrival rates of edge cloud clusters and central clouds, and the ant colony algorithm is used to achieve the

optimization of offloading rate of mobile devices and reasonable allocation of service requests, which improves the efficiency of resource utilization. The simulation comparison analysis results show that the proposed method offloading decision can not only adapt to more complex working conditions, but also the comprehensive system overhead is reduced.

Keywords: Mobile Edge Computing, Multiple Scenarios, Computation Offloading Decision, Resource Allocation, Task Cost Optimization

目 录

第一章 绪论.....	1
1.1 研究背景及其意义.....	1
1.2 移动边缘计算与计算卸载.....	2
1.2.1 移动边缘计算的诞生.....	2
1.2.2 计算卸载技术.....	5
1.3 国内外研究现状.....	7
1.3.1 移动边缘计算的研究现状.....	7
1.3.2 计算卸载决策的研究现状.....	8
1.3.3 研究现状总结.....	10
1.4 本文的主要研究内容与章节安排.....	11
1.4.1 主要研究内容.....	11
1.4.2 本文组织结构.....	11
第二章 有限资源下的能源高效型计算卸载决策.....	13
2.1 引言.....	13
2.2 系统模型.....	13
2.2.1 多用户 MEC 场景描述.....	13
2.2.2 本地执行模型.....	14
2.2.3 数据传输模型.....	15
2.2.4 边缘云执行模型.....	16
2.2.5 能耗最小化问题.....	16
2.3 基于内罚函数的能耗最小化算法设计.....	17
2.3.1 本地计算频率决策优化.....	17
2.3.2 移动设备传输功率决策优化.....	18
2.3.3 移动设备卸载数据决策优化.....	18
2.3.4 基于内罚函数法的卸载决策设计.....	20
2.3.5 时间复杂度分析.....	23
2.4 MECOS 的仿真验证与分析.....	23

2.4.1 MECOS 的仿真实验设计	23
2.4.2 MECOS 的仿真结果分析	23
2.5 本章小结	28
第三章 基于 Stackelberg 博弈的资源定价与卸载决策	29
3.1 引言	29
3.2 系统模型	29
3.2.1 任务时延模型	29
3.2.2 博弈模型	30
3.3 基于 Stackelberg 博弈的资源定价与卸载决策设计	31
3.3.1 用户成本的优化	31
3.3.2 边缘云效用优化	34
3.4 算法设计	35
3.4.1 算法分析	35
3.4.2 时间复杂度分析	37
3.5 DPPO 的仿真验证与分析	37
3.5.1 DPPO 的仿真实验设计	37
3.5.2 DPPO 的性能验证	38
3.6 本章小结	44
第四章 基于边云协同的多目标卸载决策优化	45
4.1 引言	45
4.2 系统模型	45
4.2.1 MEC 场景描述	45
4.2.2 任务执行模型	47
4.3 基于蚁群算法的多目标卸载决策设计	52
4.3.1 服务请求卸载位置的优化	52
4.3.2 移动设备卸载速率的优化	55
4.3.3 时间复杂度分析	58
4.4 ACMODM 的仿真设计与分析	58
4.4.1 ACMODM 的仿真实验设计	58

4.4.2 ACMODM 的仿真结果分析	59
4.5 本章小结	64
第五章 总结与展望.....	65
5.1 全文总结	65
5.2 未来的展望	66
参考文献.....	68
作者在攻读硕士学位期间公开发表的论文及科研成果.....	73
致 谢.....	74

第一章 绪论

1.1 研究背景及其意义

随着物联网(Internet of Things, IoT)技术的蓬勃发展,大量对象将与互联网连接,例如智能手机,车辆,传感器和可穿戴设备^[1],促进了多种应用,包括智能家居,智能城市和智能交通系统的产生^[2],带来了大量计算密集型以及时延敏感性的业务请求。尽管当下智能手机、平板电脑、可穿戴设备等移动设备的软硬件技术发展迅速,但由于物理尺寸和工作场景的限制,大多数移动设备的计算能力和电池资源是有限的,在处理计算密集型业务时会占据移动设备本身大量的 CPU 和内存资源,导致消耗移动设备本身电池的大量电量和执行延迟过高等问题的出现,移动设备可能无法满足任务要求^[3]。

传统云计算范式可以从一定程度上缓解移动设备资源受限的压力,云服务器为移动设备提供了丰富的计算资源,移动设备可以将部分计算任务通过网络卸载到云服务器上执行,但传统云计算具有天然的缺陷,比如远程云中心到用户的距离远造成意外延迟,难以满足实时性要求;海量移动设备产生的大量数据对网络带宽造成的巨大压力,导致带宽资源不足。因此,传统云计算范式无法满足爆发式的网络请求^[4]。

5G 技术的问世催生了移动边缘计算(Mobile Edge Computing, MEC)这一全新的计算范式,多天线、毫米波和多小区三项 5G 的代表技术引发了相关配套基础设施建设的高潮,从而使移动边缘计算的实现成为可能^[5]。区别于传统云计算中服务器集中部署的方式,移动边缘计算将 IT 服务环境和云计算资源下沉至网络边缘^[6],美国韦恩州立大学的施巍松团队^[7]将“边缘”定义为数据源和远程云中心之间的路径上的任何计算和网络资源,包括智能网关、智能路由器、基站等。之后,边缘计算能力由最初的蜂窝网络边缘蔓延至各种接入方式(如 WiFi, 蓝牙)。移动边缘计算的思想使得业务程序可以运行在距离用户更近的地方,首先,可以避免请求任务进入核心网造成网络拥塞,边缘服务器可以直接提供服务从而保证较低的延迟,提高应用的响应速度,同时广泛分布的边缘服务器降低了云服务器

的能耗；其次，用户的某些敏感数据保留在附近的网络环境内，一定程度上降低泄露的风险，提高了安全性；另外，利用边缘的闲置资源处理用户请求，既能分担设备计算压力，又能提高整体资源利用率。可以说移动边缘计算是设备本地执行任务和远程云执行任务的一种折中，将网络连接功能和应用服务功能进行了深度融合，为“万物互联”时代提供了相当具有前景的解决方案。

尽管移动边缘计算具有广阔的前景，但仍面临诸多挑战。运营商在网络边缘部署各种边缘计算资源的同时势必会增加成本^[7]；而且，由于网络边缘无法像远程云那样部署高性能服务器，因而边缘服务器通常仅具有有限的计算能力^[8]。为了保证服务质量(Quality of Service, QoS)，计算任务需要在规定时延内完成^[9]。边缘服务器如何合理地分配资源，用户设备如何有效地将任务卸载到边缘服务器，以高效的计算卸载策略和资源分配充分挖掘移动边缘计算的潜能成为当下广受相关学者关注的研究热点。

1.2 移动边缘计算与计算卸载

1.2.1 移动边缘计算的诞生

边缘计算的概念最早可以追溯到 20 世纪 90 年代，Akamai 公司为解决网络带宽小、用户访问量大且不均匀等问题，提出了内容分发网络(Content Delivery Network, CDN)技术。CDN 通过在网络边缘增加缓存服务器来降低用户访问的延迟，用户的请求会被定向到离用户最近且负载低的节点上。CDN 关注的重点是内容转发，而边缘计算节点关注的重点则是数据处理。

移动边缘计算的标准化概念于 2014 年由欧洲电信标准化协会(European Telecommunications Standards Institute, ETSI)首次提出，这项技术旨在将远程云资源下沉到无线网络的边缘，在距离用户更近的地方提供计算、通信、存储能力^[10]。移动边缘计算是 IoT 技术发展到一定阶段的产物，在 IoT 技术发展的初期，业界主要采用云计算作为主要范式来解决个人设备的存储与计算能力不足的问题，云计算也是当下应用度最广、成熟度最高的计算范式。即便如此，对于云计算仍然没有一个标准的定义^[11]。目前，被普遍接受的说法为：对服务商而言，任何资源

均作为服务进行提供, 这些服务可以被虚拟化、池化、共享; 对于用户而言, 用户能够跨网络、动态弹性、按需而变地访问服务^[12]。

但随着 IoT 技术以及无线蜂窝网络技术的不断发展, “万物互联”的概念逐渐成为现实。根据预测, 到 2025 年, 平均每人拥有 6 部移动设备, 全球合计将有超过 500 亿部移动设备^[13]。网络带宽与远程云容量近乎枯竭, 所以急需探索新的计算范式及网络架构来应对这一问题, 比如微数据中心^[14, 15], cloudlet^[16]以及雾计算(Fog Computing)^[17]。

微数据中心是部署在不同地理区域的具有不同规模的服务器与存储集群。网络请求会首先被定向到区域临近的微数据中心, 微数据中心之间也会定时对某些数据(如会话缓存)进行同步以实现冗余和分区容错, 在某一区域的微数据中心宕机时, 其他数据中心仍然能提供临时服务。

Cloudlet 是云计算与移动云计算(Mobile Cloud Computing, MCC)相结合的产物, 最早由 Satyanarayanan 等人^[16]在 2009 年首次提出, cloudlet 是一种受信任且资源丰富的计算机集群, 不依赖于服务提供商而是用户自管理的, 它相当于在私人区域中维护一个计算机集群来代替远程云的某些功能, 从而提高移动用户的体验质量(Quality of Experience, QoE)。

雾计算作为移动边缘计算的前身, 于 2011 年某个关于车辆互联网的研讨会上被提出。时任思科副总裁 Bonomi 给出了雾计算的关键词——网络移动性、自组织网络、计算存储资源的随处可用性、接近网络边缘^[18]。实际上, 雾计算是一个概念而不是一个架构, 它通过在移动设备和云之间引入一个中间雾层来扩展云计算。可以使用许多相似技术实现雾计算。

移动边缘计算可以被视为雾计算概念的一种实现, 移动边缘计算的第一个平台在 2013 年由 IBM 和诺基亚西门子公布^[19]。移动边缘计算时常被混淆为由网络边缘的智能设备构成的 cloudlet, 实际上移动边缘计算是在网络边缘提供一个应用程序运行的平台。

回顾计算范式的发展历程可以发现, 微数据中心和 cloudlet 仍是传统云计算的思路, 由于请求被发送到服务器集群, 请求数据在复杂网络拓扑中的传输增加了很强的随机性, 当请求数超过一定负载时, 网络拥塞和服务器拒绝服务的概率

增加^[20]。而雾计算和 MEC 则呈现更强的分布式，在小区的无线接入点处即可提供计算资源，一定程度上能够缩短响应时间。

如图 1.1 所示，标准的移动边缘计算系统通常包含三层节点：移动设备、边缘云和远程云。

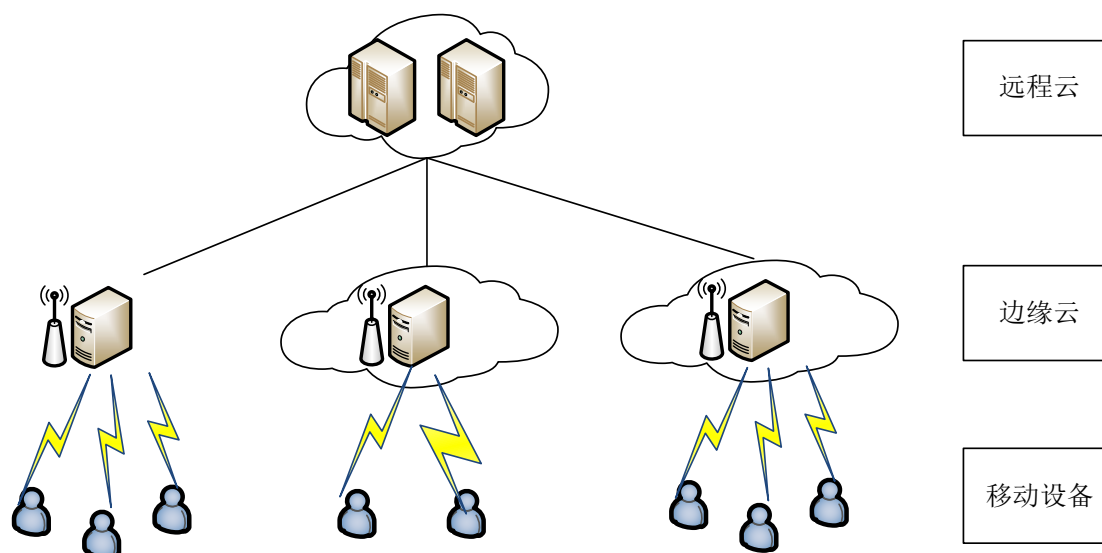


图 1.1 移动边缘计算系统

移动设备本身的资源（计算、存储、电池等）极为有限，但可以通过各类无线接入网直接与边缘云连接。移动设备运行计算密集型与时延敏感型应用时会产生不同数量的服务请求，移动设备可以选择在本地执行服务请求，或者通过各类无线接入网卸载到边缘云中进行计算。

边缘云通常位于蜂窝基站、小区路由器等无线接入点内，一个边缘服务节点可能配置一个或多个边缘云服务器，这些设备利用虚拟化、容器化等技术^[22,23]，依靠自身相对丰富的资源在边缘侧向用户提供服务。但受物理场景因素影响，计算资源相对于远程云具有一定的有限性，边缘云可以将部分计算任务卸载到远程云进行计算。

远程云是各种应用服务器组成的集群，有充足的计算、存储资源以及负载均衡等管理控制，在数据挖掘，信息整合，大数据处理，数据聚合和批处理等复杂数据计算中性能都优于移动边缘云服务。因而处理请求消耗的时间极短，延迟主要集中在数据在网络中的传输^[24]。

1.2.2 计算卸载技术

随着移动设备的快速增加,人们使用移动设备处理的服务越来越多,受限于自身的体积,移动设备不能拥有强大的计算资源。为了解决日益增长的用户体验质量、服务需求质量与移动设备资源受限之间的矛盾,技术人员将计算卸载引入到移动边缘计算系统中,计算卸载是资源短缺的移动设备的任务分配给资源充足的代理服务器执行,再把计算结果取回到终端设备的过程^[25]。计算卸载按照不同的分类标准可以分成不同的卸载方式。

(1) 按照卸载阶段

边缘计算系统三层架构中,主要有两个卸载阶段:端边卸载^[26-28]和边云卸载^[29,30]。端边卸载是移动设备终端连接到附近的无线接入点,将计算任务转移到无线接入点内的边缘云进行处理;边云卸载是当边缘云的资源不能够满足应用需求时,可以通过互联网将计算任务进一步转移到功能强大的远程云进行处理。

(2) 按照决策发出者

卸载策略可以由边缘云统一制定或者设备自身制定,分别称为集中式卸载控制^[31-34]和分布式卸载控制^[35-38]。集中式卸载控制,是以全局视角通过综合网络中的信道状态、资源数量、请求任务数及各任务需求做出的最优控制;分布式卸载控制,是各设备均以自身利益最大化为目标,根据自身的任务参数以及其他设备的卸载策略不断调整而达成的一种平衡。集中式控制可以得到最优或次优的卸载决策结果,但复杂度较大;分布式控制能根据网络和设备的实时状态,以较低的复杂度快速得到决策结果。

(3) 按照任务划分粒度

按照计算任务的分区粒度不同,可以分为二进制卸载(binary offloading)^[33,35,39]和部分卸载(partial offloading)^[40]。二进制卸载是将计算任务作为一个整体进行转移,二进制卸载方式通常需要对任务是否需要卸载、将任务卸载到哪个边缘云、设备以多大的发射功率进行数据传输进行决策;部分卸载是对任务数据按某种标准进行划分,一部分转移到边缘云执行而另一部分在设备端执行,部分卸载方式在上述问题基础上还需要对任务如何划分进行决策。二进制卸载可以说是部分卸载的一种特殊情况。

(4) 按照卸载场景

现实世界的网络环境是错综复杂的,在一片蜂窝网络区域内存在多个基站信号塔,它们的覆盖区域存在一定的重叠,因此会存在一个用户同一时间有多个基站可以作为接入点的情况。不失一般性,可以将其抽象成为单用户单边缘服务器、多用户单边缘服务器、单用户多边缘服务器和多用户多边缘服务器的场景进行研究^[41]。

针对不同的卸载方式面临的各种问题,研究人员对计算卸载过程的各个技术细节进行了深入探索,并给出一般性的计算卸载流程:边缘云节点感知、参数收集、卸载策略制定、任务数据上传、任务执行、结果回传。现对其进行详细阐述。

(1) 边缘云节点感知

边缘云节点感知是移动设备进行计算卸载决策之前的准备过程。当移动设备有计算任务时,将对网络中可以进行卸载的边缘云节点进行感知,获得边缘云节点的状态信息。

(2) 参数收集

边缘云获知计算任务的参数是做出卸载决策的前提。参数收集工作是由设备的系统分析器(device profilers)和移动边缘系统级管理共同完成的^[42-44],它们分别负责监控设备的运行状态和参数数据的分析,该过程也会进行少量的报文交换。

(3) 卸载策略制定

卸载策略制定是计算卸载过程的核心。高效的卸载策略能够充分利用边缘云有限的计算资源,降低任务执行的能耗、延迟、系统开销等。在本环节中将确定分配给移动设备的计算资源、信道资源等,确定需要由边缘云执行的任务。

(4) 任务数据上传

将卸载策略制定环节中确定上传的任务传输到边缘云进行执行,任务数据的传输可以通过 5G 等蜂窝网络进行,也可以通过 WiFi 等无线局域网进行。蜂窝网络的覆盖范围广,移动设备可以在宏小区之间进行切换,无需重复卸载,但蜂窝网络会对流量进行收费;局域网的覆盖范围有限,不同局域网之间存在复杂的认证鉴权机制,因此设备在不同局域网之间的移动性较差,但无需对产生的流量进行付费。

(5) 任务执行

移动设备将任务数据传输到边缘云以后,边缘云会根据卸载策略环节中制定的策略分配相应的资源执行该任务。由于设备移动性等原因,设备与边缘云的网络连接可能会中断,此时边缘云可以选择丢弃计算结果、缓存计算结果或者将计算结果转移至其他边缘云。丢弃计算结果适用于存储资源紧张并且转移成本较大的边缘云;缓存计算结果意味着边缘云认为短时间内设备有很大概率会重新连接;计算结果转移至其他边缘云适用于转移成本小于其他边缘云重新计算的场景。

(6) 结果回传

边缘云将执行完的计算结果传输到移动设备,设备与服务器之间的连接资源被释放,一次计算卸载过程完成。

1.3 国内外研究现状

关于移动边缘计算的研究仍处于起步阶段,目前,众多学者聚焦于系统标准化理论、关键使能技术、计算卸载决策以及 MEC 应用场景的研究。其中,计算卸载决策面临着计算任务分割、系统资源配置、设备的移动性管理等挑战。

1.3.1 移动边缘计算的研究现状

移动边缘计算展现出的巨大潜力使其一经提出就得到全世界相关标准化组织的积极响应,ETSI 率先推动了移动边缘计算的标准化工作,随后,3GPP、IEEE、中国通信标准化协会等多家机构均将移动边缘计算纳入 5G 技术的标准并作为关键技术之一^[45]。目前,移动边缘计算在创新学术研究、共建行业生态、推动商业落地三方面齐头并进。

在学术界,美国国家科学基金会和美国国家标准局先后将边缘计算列入项目申请指南,鼓励学者开展关于边缘计算的研究;关于移动边缘计算的国际会议也如雨后春笋一般,其中 IEEE/ACM Symposium on Edge Computing(SEC)是由 IEEE 和 ACM 联合举办的边缘计算专项顶会,International Conference on Distributed Computing Systems (ICDCS)、International Conference on Computer Communications (INFOCOM)等重大国际会议均增设了边缘计算的分会或专题研

讨会;通信、无线网络、云计算等领域的权威期刊(如 IEEE Wireless Communication、IEEE Transaction on Cloud Computing)陆续增加了关于移动边缘计算的专刊或研究方向。

在工业界,随着物联网、人工智能、5G 通信等技术的发展,边缘计算逐渐开始引起重视。2015 年 9 月,ETSI 发布了关于移动边缘计算的白皮书;2015 年 11 月,ARM、Cisco、Dell、Intel、Microsoft 以及 Princeton University Edge Computing Laboratory 宣布成立 OpenFog 联盟以解决带宽、延迟和通信挑战;2016 年,华为、沈自所、Intel、ARM、软通动力、中国通信技术研究院联合发起边缘计算产业联盟(Edge Computing Consortium, ECC),截至 2017 年底已有 154 家成员单位;中国移动在“‘5G+E’网边融合技术峰会”上公布了 105 个边缘计算项目,覆盖制造、教育、医疗等 15 个行业^[46]。2019 年 2 月,中国移动、中国电信、中国联通及产业链合作伙伴联合发布了 OTII 边缘定制服务器。

产品方面,亚马逊率先推出了 AWS Greengrass 软件,借助核心 Lambda 函数可以将云功能扩展至本地设备,在不连接到云的情况下完成消息交换、数据分析、服务响应等功能;另一个云计算巨头微软也发布了 Azure IoT Edge 托管服务组件,通过个性化定制组件源代码可以更加灵活开放地适应客户不同的应用场景;国内的公有云技术虽与前者尚有差距,但也在积极进行边缘计算的战略布局,2018 年 12 月,ABC Inspire 企业智能大会上,百度发布国内首个开源边缘计算平台 OpenEdge;阿里云的 Link Edge 和华为的 EC-IoT 也具有不可低估的潜力和市场^[47]。据工信部研究数据显示,2020 年我国国内边缘计算市场规模达到 180 亿元,同比增长 55.39%。

1.3.2 计算卸载决策的研究现状

在产学研同步驱动下,移动边缘计算在标准化和应用落地方面取得了不错的成效。当前,对于移动边缘计算的研究开始向计算卸载、内容缓存、用户偏好这些能够提高性能及用户体验的方向倾斜。移动边缘云服务器位于网络边缘,和中心云相比,移动边缘云服务器的计算资源以及信道资源也具有一定的限制性,因而不同的移动设备卸载到同一个移动边缘云服务器时也存在一定的竞争关系,移

动边缘云服务器需要根据移动设备卸载任务的情况合理分配资源。计算卸载决策就是以一项或多项指标为目标，在有限资源下达到更高的性能的过程。

到目前为止，国内外的研究主要集中在降低时延、能耗为目标的方向上。Liu 等人^[48]提出了一种双时间尺度机制，在长时间尺度上解决用户-服务器关联，在短时间尺度上执行动态任务卸载和资源分配策略，来实现最小化用户的能耗，同时调度分配的资源用于本地计算和任务卸载。Ding 等人^[49]提出了基于丁克尔巴赫法和牛顿法的两种迭代算法来实现非正交多址的移动边缘计算的卸载延迟的最小化。Li 等人^[50]将 MEC 系统表述为马尔可夫决策过程，然后引入状态和动作空间，设计了一种基于 DQN (Deep Q-Network) 的新型卸载策略，以确保以延迟和能耗为优化目标的系统性能。Elgendy 等人^[51]采用了深度学习的方法，提出了一种将应用程序和已完成任务的相关代码缓存在边缘服务器的缓存概念，以应用于以最小化延迟和能量开销为目的的计算任务。

此类研究数不胜数，本节不再赘述。然而目前计算卸载中关于经济因素的研究偏少，大多假设边缘云免费提供计算资源服务，但实际上边缘云的部署和维护是有成本的，因此需要考虑边缘云定价策略对用户卸载策略的影响。

Liu^[52]等人提出了一种基于 Stackelberg 博弈的卸载策略和价格控制算法。Moura^[53]等人提出经济激励在 MEC 中部署的重要作用，并引入 Stackelberg 博弈模型研究边缘服务器与多用户之间的交互作用。Xiong^[54]等人研究了区块链在 MEC 中的应用，每个用户都需要通过一个边缘云来进行挖掘服务，通过构建 Stackelberg 博弈模型，分别实现边缘云和用户的利润最大化。Hazra^[55]通过联合优化动态服务成本和用户需求，建立了雾和云资源分布式调度的非合作 Stackelberg 博弈模型。Liu^[56]等人将云服务提供商和边缘云之间的交互描述为 Stackelberg 博弈，并设计了两个算法，在低延迟和降低复杂性方面最大化双方的效用。Jie^[57]等人开发一个基于可重复 Stackelberg 博弈的任务调度模型，以优化响应时间和资源分配，并使用历史预测方法来求解该博弈。Li^[58]等人将边缘云和分配资源控制器之间的交互描述为 Stackelberg 博弈，并针对延迟敏感和计算密集型任务提出不同的卸载算法。Sun^[59]等人研究了 MEC 网络的 D2D 协同任务分流策略，提出了一种基于 Stackelberg 的方法来充分利用空闲设备资源。Wang^[60]

等人考虑一个由多个移动设备和边缘云组成的交互系统,其中每个移动设备可以选择服务请求是在本地计算还是卸载到边缘云完成,并构建一个嵌套的两阶段博弈模型,该模型在最大限度地发挥边缘云的资源的同时降低了用户的成本。

上述研究更多集中在边缘云与用户层面,没有引入远程云。针对边云协同的场景,Li^[61]等人研究了移动用户与公有云和边缘云之间的相互作用,分别用 $M/M/\infty$ 队列和 $M/M/1$ 队列模型模拟了公有云和边缘云,并用 Stackelberg 博弈分析了它们之间的相互作用。Chin^[62]等人以排队论的知识对边缘云建模,对边缘服务器放置和服务分配策略进行了研究,重点是最小化总交通负荷,以便实现移动网络中的绿色通信。Du^[63]等人针对边云协同的场景,通过联合优化计算资源,传输功率和无线电带宽,在混合雾/云系统中解决计算卸载问题,同时保证用户公平性和最大可容忍的延迟。Liu^[64]等人利用排队论对雾计算系统中的卸载过程的能耗,执行延迟和支付成本进行了研究。Ren^[65]等人为了在通信和计算能力有限的情况下提高边缘云的效率,提出了一个联合通信和计算资源分配问题,以最小化所有移动设备的加权和延迟。Guo^[66]等人研究了物联网边缘云计算系统中保证时延前提下的节能工作负载分配问题。

1.3.3 研究现状总结

经过以上调研可以发现,国内外关于 MEC 系统的计算卸载决策的研究大致集中在降低用户时延、能耗方面,对经济因素的影响有了一定的研究但并不深入。对边缘云的计算速率考虑过于理想,较少的考虑边云协同的问题。

目前,关于计算卸载决策的研究仍存在一定的局限性。在考虑降低用户时延、能耗方面,文献[48-50]认为边缘云的计算资源足够使所有用户在规定时延前完成任务,忽略了用户过多导致任务出现失败的情况,使得所得出的卸载决策在人群密集区域出现问题。文献[33,40]中,作者假设卸载任务在边缘服务器上独立执行或者数据传输和边缘执行是时间上互斥的,使得信道资源和边缘计算资源无法同时被移动设备利用,都与实际的 MEC 系统有所偏差。因此本文采用了动态电压缩放技术调整移动设备的本地计算频率,联合信道资源和边缘云计算资源对降低用户能耗问题进行研究。

在引入经济因素的相关研究中,大都是假设对边缘云资源进行平均分配^[52-60],然后根据移动设备占用计算资源的时间进行收费,但在实际中,不同移动设备生成的计算任务的延迟敏感度不同必然会导致计算资源需求出现差异。因此本文对于边缘云收益的研究中选择将不同计算能力的计算资源块出售给用户。

在引入边云协同的场景中,考虑的大都是多用户单边缘云的边云协同场景[63-65],忽视了实际应用中部署多个边缘服务器对计算卸载策略的影响,除此之外大部分研究没有考虑容器化虚拟化技术[61,66],只能处理一个服务请求,没有充分利用边缘云的计算能力。因此本文考虑了多用户多边缘云的边云协同场景,将边缘云计算模型建模为能同时处理多个服务请求的 M/M/c 模型。

1.4 本文的主要研究内容与章节安排

1.4.1 主要研究内容

基于以上对移动边缘计算背景及计算卸载研究现状的分析与总结,分别针对多用户静态环境下用户角度及运营商角度、多边缘服务器边云协同场景下的功率控制、通信资源分配、计算资源定价、负载均衡等问题进行分析与探索,涵盖不同的计算卸载模型。现对本文的主要研究内容总结为如下几个方面:

- (1) 针对移动设备通常忽视信道与边缘云计算资源而导致卸载决策性能比较低下乃至任务失败问题,提出了一种有限资源下的能源高效型计算卸载策略。
- (2) 针对现有定价策略平均分配边缘云计算资源而忽略用户需求差异问题,提出了一种边缘云卸载决策和 Stackelberg 博弈定价有机融合的一种新方法。
- (3) 针对边云协同的场景中多边缘服务器的计算能力未得到充分利用问题,优化了现有卸载模型,提出了一种基于蚁群算法的多边缘卸载决策优化,使卸载策略更加符合实际。

1.4.2 本文组织结构

第一章对移动边缘计算与计算卸载技术及其国内外研究现状进行了综述,分

析了计算卸载决策对充分发挥移动边缘计算优势的重要性,并简要介绍了本文研究的重点及难点。

第二章针对移动设备忽视信道与边缘云计算资源而导致卸载性能低下乃至任务失败问题,提出了一种有限资源下的能源高效型计算卸载决策方法。综合考虑了移动设备的计算频率,传输功率,卸载数据,边缘云的计算能力以及信道资源等变量,改进了卸载任务的能源模型。分别建立了最佳卸载数据量和本地设备最优计算频率的模型、最佳卸载数据量和移动设备发射功率的模型以及信道资源与最佳卸载数据量的模型,将问题简化为单变量优化问题,并基于内罚函数法进行求解,提高了卸载策略的实际可行性以及资源配置的效率。最后进行仿真对比实验,将仿真结果与其它基准算法进行比较。

第三章针对现有定价策略平均分配边缘云计算资源而忽略用户需求差异问题,提出了一种边缘云卸载决策和 Stackelberg 博弈定价有机融合的一种新方法。分别建立了边缘云定价和购买最优计算资源块的决策模型以及最佳卸载数据量和购买计算资源块数量之间的关系模型,实现了用户的最优卸载决策,确立了边缘云定价的上下界。最后,提出了一种基于动态规划的边缘云定价卸载算法,实现了边缘云效用最优定价和各用户自身效用的最大化。最后进行仿真对比实验,将仿真结果与其它基准算法进行比较。

第四章针对边云协同中多边缘服务器的计算能力未得到充分利用问题,提出了一种基于蚁群算法的多边缘卸载决策优化策略。依据排队论思想并以能耗和时延性能最小化为目标,对移动设备的服务请求卸载速率、边缘云和中心云服务请求到达速率进行了协同建模与优化。采用二分查找迭代方法对边缘云集群和中心云的服务请求到达速率进行决策,并利用蚁群算法实现了移动设备卸载速率的寻优和服务请求的合理分配。最后进行仿真对比实验,将仿真结果与其它基准算法进行比较。

第五章对全文进行了客观的总结,分析了文章中有待改进之处和未来工作的方向。

第二章 有限资源下的能源高效型计算卸载决策

2.1 引言

在 MEC 系统中,移动设备通常使用电池供电,开发一种节能效果显著的计算卸载策略能够延长系统的工作寿命。移动设备的计算任务一般都有其所能接受的最大容忍时延,如何平衡时延与能耗之间的关系一直是一个研究热点。由于边缘云的计算资源有限,计算任务在边缘云的调度是节约能耗与减少执行延迟的重要环节。在现有的卸载决策方法的系统模型中,通常忽视边缘云计算能力有限或者认为数据传输和边缘执行是时间上互斥的,忽视了信道资源与边缘云计算资源联合调度,使得所得出的卸载决策方法并不能符合移动设备的最大利益。

对于一种卸载策略来说,如果其能够在最大容忍时延范围和边缘云最大计算能力内完成计算任务,该策略就是可取的。在这种情况下,如何最小化完成计算任务的能耗就显得更加重要。如何设计一种策略来达到规定时延内的能效最优一直是一个研究热点。

针对以上问题,本章提出了一种有限资源下的能源高效型计算卸载决策,综合考虑了移动设备的计算频率,传输功率,卸载数据,边缘云的计算能力以及信道资源等变量,改进了卸载任务的能源模型。分别建立了最佳卸载数据量和本地设备最优计算频率的模型、最佳卸载数据量和移动设备发射功率的模型以及信道资源与最佳卸载数据量的模型,将问题简化为单变量优化问题,并基于内罚函数法进行求解,提高了卸载策略的实际可行性以及资源配置的效率。

2.2 系统模型

2.2.1 多用户 MEC 场景描述

本节主要考虑一个基于时分多址(Time division multiple access, TDMA)的多用户边缘计算系统,如图 2.1 所示。系统包含一个边缘云服务器和 n 个用户移动设备,用集合 $N = \{1, 2, \dots, n\}$ 表示。在 TDMA 系统中,时间被分成多个时隙,每个移动设备只能在分配的时隙内传输数据。此时边缘计算系统允许移动设备将

部分或全部计算任务卸载到边缘云侧以节省能源。

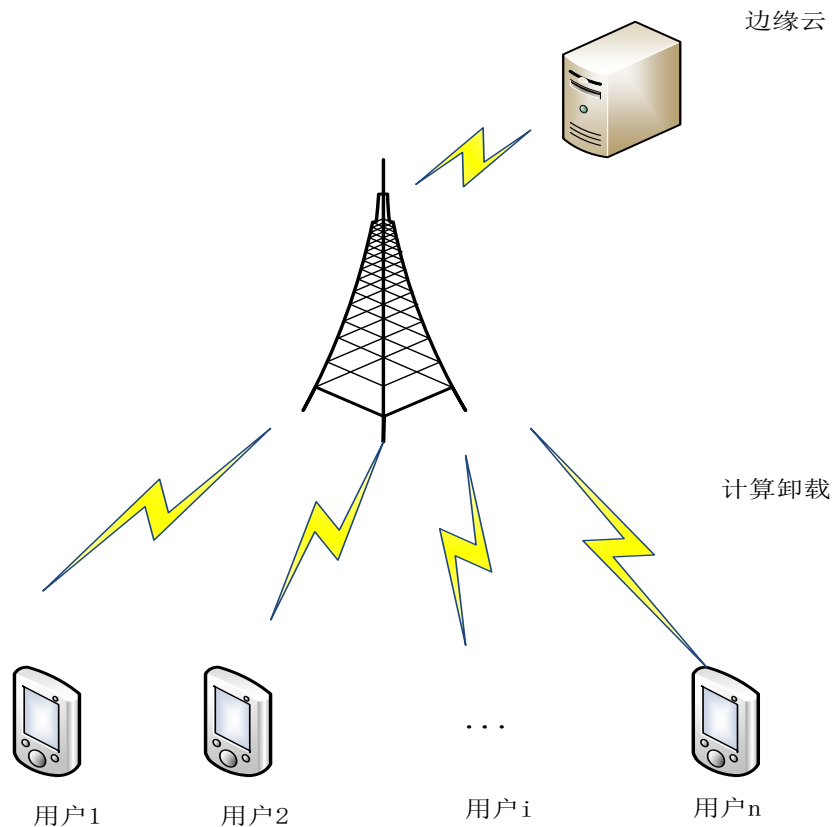


图 2.1 多用户计算系统

该边缘计算系统用于向用户设备提供某项服务，规定只有当 n 个设备的计算任务全部完成时，系统才能完成服务。为了保证 QoS，所有计算任务的完成时间应该小于该服务的延迟容忍，用 T 表示。与现有研究类似^[33,37]，将连续时间内的计算卸载决策问题简化为单个决策周期内的优化问题，即对于每个时间周期 T ，边缘云的调度方法应该是一致的，在特定的决策周期内系统状态(如信道状态、任务参数等)保持不变。

2.2.2 本地执行模型

移动设备本地执行计算任务需要电池提供能量，CPU 完成任务会有一定的延迟，本节对移动设备本地完成任务所需要耗费的电池电量及完成任务的延迟进行建模。

为了模拟系统的异构性，令移动设备的 CPU 频率、完成任务的数据量、任务的计算复杂度各不相同。设备处理器每个 CPU 周期的能耗为^[67]

$$e_i = \kappa \cdot f_i^2. \quad (2.1)$$

其中, κ 是与芯片结构相关的能耗系数, 当设备硬件结构确定时 κ 也就为一常数。 f_i 为移动设备 i 的 CPU 运行频率(Hz, 表征设备的计算能力)。令任务数据量为 o_i (bit), 需要卸载的数据量为 d_i (bit), 计算每 bit 数据需要的 CPU 周期数为 c_i (CPU cycle/bit, 表征任务的计算复杂度), 那么移动设备 i 执行任务所需的 CPU 周期数为

$$num_{CPU} = (o_i - d_i) \cdot c_i. \quad (2.2)$$

结合式(2.1)与(2.2), 可得由移动设备 CPU 执行任务需要耗费的电池能量为

$$E_i^l = \kappa \cdot f_i^2 \cdot (o_i - d_i) \cdot c_i, \quad (2.3)$$

由移动设备 CPU 执行任务所产生的延迟为

$$t_i^l = \frac{(o_i - d_i) \cdot c_i}{f_i}. \quad (2.4)$$

综上, 移动设备本地执行模型如式(2.3), (2.4)所示。

2.2.3 数据传输模型

移动设备将计算卸载的数据传输到边缘云进行计算, 数据上传过程需要电池提供能量, 数据传输过程会有一定的延迟, 因此本节对数据传输过程的能耗和时延进行建模。

计算卸载的数据传输过程分为任务数据上传和结果回传两个阶段。根据香农公式^[68], 给定设备的传输功率 p_i 及信道增益 g_i , 可以求得数据传输速率为

$$r_i = B \cdot \log_2 \left(1 + \frac{p_i \cdot g_i}{\omega_0} \right). \quad (2.5)$$

其中, B 和 ω_0 分别表示设备与边缘云之间的信道带宽和复杂高斯白噪声功率。

基于式(2.5), 可以得到移动设备进行计算卸载的数据传输延迟和能耗为

$$t_i^o = \frac{d_i}{r_i}, \quad (2.6)$$

$$E_i^o = p_i \cdot t_i^o. \quad (2.7)$$

实际上,计算卸载的通信过程还包括任务在边缘服务器执行完毕后结果的回传过程,但是本文基于以下几个原因将该过程忽略^[33-35]:

- (1) 结果的数据量不依赖于任务输入的数据量,具有随机特性;
- (2) 结果的数据量通常比任务输入的数据量小的多,一般为千分之一级;
- (3) 结果的回传由边缘服务器所在的基站等设施完成,这些设施通过电网供电,有充足的能源供应。

以上总结的本地执行模型与数据传输模型同样适用第三章中的计算卸载决策问题,仅需要做出细微的调整,因此后文不再赘述,只对调整部分进行说明。

2.2.4 边缘云执行模型

由于边缘云的计算资源是有限的,为了在延迟容忍 T 内完成卸载,卸载计算不能超过边缘云容量 F_{edge} (CPU 周期数)的限制。目前,存在两种最为研究人员广泛采用的对边缘云有限的计算能力进行描述模型^[69-71]:第一种是任务在边缘服务器的执行时间不可忽略;另一种是边缘服务器在一定时间内能够处理的计算量 (CPU 周期数)具有上界约束,将其表示为边缘服务器的计算容量。为了简单起见,相关研究同一时间只考虑不可忽略的边缘云执行时间或者边缘服务器的计算容量上界,而不是同时研究两种情况下的计算卸载决策问题。但值得注意的是,二者可以被认为是等价的。具体而言,限制边缘服务器的计算容量上界能够保证任务在要求的时间内完成。

本章节中考虑的是第二种情况,保证边缘服务器在一定时间内能够处理的计算量(CPU 周期数)具有上界约束即

$$\sum_{i=1}^n d_i c_i \leq F_{edge}. \quad (2.8)$$

只需满足式(2.8)即可保证卸载到边缘云上的任务能够在要求的时间内完成。

2.2.5 能耗最小化问题

本小节主要将能耗最小化的计算卸载决策问题描述为一个在服务请求延迟容忍约束下的多变量优化问题。移动设备 i 的设备能耗可以整合为本地执行时的

能耗 E_i^l 与计算卸载时向边缘云进行数据传输时的能耗 E_i^o 之和

$$E_i = E_i^l + E_i^o. \quad (2.9)$$

计算卸载决策问题可以描述如下：

$$\begin{aligned} \text{P2.1: } \min & \sum_{i=1}^n (E_i^l + E_i^o) \\ \text{s.t. } c2.1: & t_i^l \leq T, i \in N, \\ c2.2: & \sum_{i=1}^n t_i^o \leq T, \\ c2.3: & \sum_{i=1}^n d_i c_i \leq F_{edge}, \\ c2.4: & t_i^o \leq t_i^{slot}. \end{aligned} \quad (2.10)$$

在问题 P2.1 中， $c2.1$ 和 $c2.2$ 分别表示本地执行和卸载任务传输均需要满足服务的延迟容忍约束， $c2.3$ 表示边缘云的计算容量是有限的，传输的计算任务不能超过其上界。 $c2.4$ 表示移动设备的传输时延不能超过其分配的上界。

2.3 基于内罚函数的能耗最小化算法设计

由上节可知，能耗最小化问题需要考虑移动设备的计算频率，移动设备的传输功率，移动设备卸载数据以及分配给移动设备的时隙，是一个多变量优化问题。本节中通过研究分析各变量之间的联系，将能耗最小化问题转换为单变量优化问题。

2.3.1 本地计算频率决策优化

本小节研究了移动设备的卸载数据与计算频率之间的关系，结果如引理 2.1 所示。

引理 2.1：当移动设备 i 确定最佳卸载数据 d_i 时，移动设备 i 可以确定唯一的最优计算频率

$$f_i^* = \frac{(o_i - d_i)c_i}{T}. \quad (2.11)$$

证明：根据式(2.3)，对本地计算能耗 E_i^l 求关于本地计算频率 f_i 的偏导数可得

$$\frac{\partial E_i^l}{\partial f_i} = 2\kappa \cdot f_i \cdot (o_i - d_i) \cdot c_i. \quad (2.12)$$

可知 $(\partial E_i^l / \partial f_i) > 0$ 在定义域内恒成立。 E_i^l 是关于 f_i 的单调增函数, 因此 f_i 的值越小, 本地计算的能耗越低。根据式(2.4)可知, 本地计算时延 t_i 是关于 f_i 的单调减函数, 为了满足 c2.1, 可得 $f_i \geq ((o_i - d_i)c_i / T)$ 。综上所述本地计算频率的最小值为最佳计算频率, 即 $f_i^* = ((o_i - d_i)c_i / T)$ 。引理 2.1 得证。

从引理 2.1 中, 我们可以确定最佳的本地计算频率, 以使移动设备完成本地计算任务时的总能耗最小。

2.3.2 移动设备传输功率决策优化

本小节研究了分配给移动设备的时隙与卸载数据以及传输功率的关系, 结果如引理 2.2 所示。

引理 2.2: 对于移动设备 i 的最小化能耗 E_i 的问题, 当给移动设备 i 分配的时隙 t_i^{slot} 已确定时, 其最优卸载决策 d_i^* 和 p_i^* 总是满足如下条件:

$$d_i^* = r_i(p_i^*) \cdot t_i^{slot}. \quad (2.13)$$

证明: 引理 2.2 可以通过反证法进行证明。假设当给移动设备 i 分配的时隙 t_i^{slot} 确定时, 移动设备 i 的最小化能耗问题存在最优解 d_i^* 和 p_i^* , 使得 $d_i^* < r_i(p_i^*) \cdot t_i^{slot}$ 。由于 $r(p)$ 定义域内是关于 p 的单调增函数, 则该问题必存在另一个可行解 $\hat{p}_i (\hat{p}_i < p_i^*)$, 使得 $d_i^* = r_i(\hat{p}_i) \cdot t_i^{slot}$ 。取两个解之下的移动设备能耗的差, 得

$$\begin{aligned} \Delta E_i &= E_i(p_i^*) - E_i(\hat{p}_i) \\ &= \frac{p_i^* \cdot d_i^*}{r_i(p_i^*)} - \frac{\hat{p}_i \cdot d_i^*}{r_i(\hat{p}_i)}. \end{aligned} \quad (2.14)$$

通过求导可得函数 $f(p) = p/r(p)$ 是关于 p 的增函数, 所以有 $\Delta E_i > 0$, 可知 d_i^* 和 $p_i^* (d_i^* < r_i(p_i^*) \cdot t_i^{slot})$ 不是能耗最小化问题的最优解。引理 2.2 得证。

从引理 2.2 中可以发现, 移动设备 i 分配的时隙 t_i^{slot} 与传输时延 t_i^o 相等, 后文用 t_i^o 表示分配的时隙。

2.3.3 移动设备卸载数据决策优化

根据引理 2.1, 引理 2.2 可得当移动设备卸载数据决策确定后, 可以得到唯

一的最优计算功率和传输功率，可得

$$E_i = \frac{\kappa(o_i - d_i)^3 c_i^3}{T} + \frac{w_o}{g_i} \cdot (2^{\frac{d_i}{B \cdot t_i^o}} - 1) \cdot t_i^o, \quad (2.15)$$

因此 P2.1 可以转换为

$$\begin{aligned} \text{P2.2: } \min \sum_{d_i, t_i^o}^n & \left[\frac{\kappa(o_i - d_i)^3 c_i^3}{T} + \frac{w_o}{g_i} \cdot (2^{\frac{d_i}{B \cdot t_i^o}} - 1) \cdot t_i^o \right]. \\ \text{s. t. } & c2.2, \\ & c2.3. \end{aligned} \quad (2.16)$$

P2.2 的变量被简化为两个： d_i ， t_i^o 。

为了进一步降低问题难度，我们在定理 2.1 中证明了 d_i ， t_i^o 的对应关系。

定理 2.1: 对于移动设备 i 的最小化能耗 E_i 的问题，移动设备 i 分配的最优时隙 t_i^{o*} 与移动设备最优卸载决策 d_i^* 总是满足如下条件：

$$t_i^{o*} = \frac{d_i^*}{\frac{3c_i^3 \kappa(o_i - d_i^*)^2 B g_i}{T \ln 2 w_o}}. \quad (2.17)$$

证明: 当 t_i^{o*} 为 0 时，移动设备无法卸载数据，此时 d_i^* 必然为 0。当 $t_i^{o*} > 0$ 时，根据式(2.15)， E_i 关于 d_i 的一阶偏导数和二阶偏导数分别为

$$\frac{\partial E_i}{\partial d_i} = \frac{\ln(2) w_o \cdot 2^{\frac{d_i}{B t_i^o}}}{B g_i} - \frac{3c_i^3 \kappa(o_i - d_i)^2}{T}, \quad (2.18)$$

$$\frac{\partial^2 E_i}{\partial d_i^2} = \frac{\ln^2(2) w_o \cdot 2^{\frac{d_i}{B t_i^o}}}{B^2 g_i t_i^o} + \frac{6c_i^3 \kappa(o_i - d_i)}{T}. \quad (2.19)$$

由于系统的各项参数均大于 0，任务数据量 o_i 大于等于卸载数据 d_i ，因此 E_i 关于 d_i 的二阶偏导数 $(\partial^2 E_i / \partial d_i^2) > 0$ ，则 $(\partial E_i / \partial d_i)$ 是单调增函数。当 d_i 趋向于 0 时，可知

$$\lim_{d_i \rightarrow 0} \frac{\partial E_i}{\partial d_i} = \frac{\ln(2) w_o}{B g_i} - \frac{3c_i^3 \kappa o_i^2}{T}, \quad (2.20)$$

当 $\lim_{d_i \rightarrow 0} (\partial E_i / \partial d_i) \geq 0$ 时， $(\partial E_i / \partial d_i)$ 将恒大于 0， E_i 是关于 d_i 的单调增函数，此时出于能耗最小化的考虑， d_i^* 应为 0，相应的此时 t_i^{o*} 应为 0。当 $\lim_{d_i \rightarrow 0} (\partial E_i / \partial d_i) < 0$ 时，

E_i 是关于 d_i 的先减后增函数, 当 $(\partial E_i / \partial d_i) = 0$ 时, E_i 取最小值, 此时 t_i^o 的值为

$$t_i^o = \frac{d_i}{B \log_2 \frac{3c^3 \kappa (o_i - d_i)^2 B g_i}{T \ln 2 w_o}}. \quad (2.21)$$

当 d_i^* 为 0 时, t_i^{o*} 为 0 亦满足式(2.21), 综上所述

$$t_i^{o*} = \frac{d_i^*}{B \log_2 \frac{3c^3 \kappa (o_i - d_i^*)^2 B g_i}{T \ln 2 w_o}}. \quad (2.22)$$

定理 2.1 得证。

2.3.4 基于内罚函数法的卸载决策设计

根据定理 2.1 与引理 2.1, 只需要确立每个用户的最优卸载数据 d_i^* , 即可得到分配的最优时隙 t_i^* 和移动设备的最优计算频率 f_i^* , 再依据引理 2.2 可以得到移动设备的最优发射功率 p_i^* 以及最优传输速度 r_i^* , 可得

$$E_i = \left(\frac{\kappa(o_i - d_i)^3 c_i^3}{T} + \frac{w_o}{g_i} \cdot \left(\frac{3c^3 \kappa (o_i - d_i)^2 B g_i}{T \ln 2 w_o} - 1 \right) \cdot \left(\frac{d_i}{B \log_2 \frac{3c^3 \kappa (o_i - d_i)^2 B g_i}{T \ln 2 w_o}} \right) \right). \quad (2.23)$$

因此 P2.2 转换为一个单变量优化问题:

$$\begin{aligned} \text{P2.3: } \min_{d_i} & \sum_{i=1}^n E_i \\ \text{s.t. c2.2: } & \sum_{i=1}^n \frac{d_i}{B \log_2 \frac{3c^3 \kappa (o_i - d_i)^2 B g_i}{T \ln 2 w_o}} \leq T, \\ \text{c2.3: } & \sum_{i=1}^n d_i c_i \leq F_{edge}. \end{aligned} \quad (2.24)$$

P2.3 的约束条件是非线性的, 且是不等式约束。为了求解 P2.3 的最优解, 本节提出了一种基于内罚函数法的能耗最小化卸载策略 (Minimize Energy Consumption of Offloading Strategy, MECOS) 算法。MECOS 算法从可行域内部出发, 并保持在可行域内部进行最优解的搜索, 通过引入罚函数的方法将约束优化问题转换成无约束问题。将移动设备卸载的数据集合用 d 表示, 可得

$$d = \{d_1, d_2, \dots, d_n\}. \quad (2.25)$$

根据 c2.2 以及 c2.3, 将可行域记作

$$S = \{d \mid g_1(d) \geq 0, g_2(d) \geq 0\}. \quad (2.26)$$

其中

$$g_1(d) = T - \sum_{i=1}^n \frac{d_i}{\frac{3c^3 \kappa(o_i - d_i)^2 B g_i}{T \ln 2 w_o}}, \quad (2.27)$$

$$g_2(d) = F_{edge} - \sum_{i=1}^n d_i c_i. \quad (2.28)$$

定义罚函数

$$G(d, r) = \sum_{i=1}^n E_i + rB(d). \quad (2.29)$$

其中

$$B(d) = -(\log_e(g_1(d)) + \log_e(g_2(d))), \quad (2.30)$$

r 是一个很小的正数。当 d 趋向于可行域的边界时, $B(d)$ 趋向于 $+\infty$, 则 $G(d, r)$ 趋向于 $+\infty$; 否则, 由于 r 取值很小, 则 $G(d, r)$ 的值近似与 P2.3 的目标函数, 因此可以通过求解 P2.4 来获得 P2.3 的近似解, P2.4 为

$$\begin{aligned} \text{P2.4: } \min_d G(d, r) \\ \text{s.t. c2.4: } d \in S. \end{aligned} \quad (2.31)$$

由于 $B(d)$ 的阻拦作用, 可以将 P2.4 的解控制在可行域的内部。P2.4 可以当作无约束问题来处理, 更容易求解。

根据式(2.29), 显然 r 的值越小, P2.4 的最优解越接近与 P2.3 的最优解。然而, 如果 r 的值太小, 将给 P2.4 的计算带来很大困难。因此, 本节采用序列无约束极小化方法, 取一个严格单调递减且趋于 0 的罚因子数列 $\{r_k\}$ 。罚因子数列 $\{r_k\}$ 满足

$$r_{k+1} = \beta r_k \quad (2.32)$$

其中, β 为缩小系数, $\beta \in (0, 1)$, k 为迭代次数。P2.4 可以转换为

$$\begin{aligned} \text{P2.5: } \min_d G(d, r_k) \\ \text{s.t. c2.4: } d \in S. \end{aligned} \quad (2.33)$$

根据目标函数 $G(d, r_k)$ 在其负梯度方向 $-\nabla G(d, r_k)$ 具有函数值下降最快的特

性，可以将 n 维无约束优化问题转化为一系列按照负梯度方向的一维搜索问题，即按照式(2.34)进行迭代搜索：

$$d_i^{(j+1)} = d_i^j - \alpha_i^j \nabla G(d_i^j, r_k). \quad (2.34)$$

其迭代的终止条件为 $\|\nabla G(d_i^j, r_k)\| \leq \varepsilon$ ，其中 α_i^j 是步长， ε 为允许的误差，为了平衡收敛速度与误差，本节中将误差设置为随移动设备数量动态变化的值，定义为

$$\varepsilon = \eta \cdot n. \quad (2.35)$$

其中 η 为误差选择系数，是一个正常数。具体求解过程如算法 2.1 所示。

算法 2.1: MECOS(Minimize Energy Consumption of Offloading Strategy, 能耗最小化卸载策略)算法

输入：设备参数 o_i, c_i, κ ，系统参数 $B, T, \omega_0, g_i, F_{edge}$ ，迭代步长 α ，缩小系数 β ，误差选择系数 η ，罚因子 r_0 。

输出：移动设备卸载数据 d ，移动设备消耗能耗 E_i 之和。

- 1 初始化 $k=0$ ，移动设备卸载数据 d^0 ，根据式(2.35)计算误差 ε 。
 - 2 **repeat**
 - 3 初始化 $j=0$;
 - 4 **while** $\|\nabla G(d_i^j, r_k)\| > \varepsilon$
 - 5 **for** $i=1$ 到 n
 - 6 根据式(2.34)计算 $d_i^{(j+1)}$;
 - 7 **end for**
 - 8 $j = j + 1$;
 - 9 **end**
 - 10 $r_{k+1} = \beta r_k$;
 - 11 $k = k + 1$;
 - 12 $d^k = d^j$;
 - 13 根据式(2.30)计算 $B(d^k)$;
 - 14 **until** $\|r_k B(d_k)\| < \varepsilon$
 - 15 根据(2.23)计算移动设备消耗能耗 E_i 之和。
-

2.3.5 时间复杂度分析

对于 MECOS 算法包括三层循环(2-14 行)，第一层循环是最外围的内罚函数求解过程，迭代次数为 k ，第二层循环是移动设备卸载数据的选择(4-9)行，其迭代次数为 j ，最后一层循环是 n 个移动设备重新搜索卸载数据的迭代次数为 n (5-7)行，综上所述 MECOS 的时间复杂度为 $O(kjn)$ 。

2.4 MECOS 的仿真验证与分析

在本节中对所提出的 MECOS 算法的性能进行了评估与验证，并与一些基准算法进行了比较。仿真实验结果验证了算法的有效性。

2.4.1 MECOS 的仿真实验设计

为了评估 MECOS 算法的性能，对一个由多个移动设备和一个移动边缘服务器组成的边缘计算系统进行了仿真，该仿真是采用蒙特卡洛方法随机进行实现的。除非另有额外说明，否则仿真参数如表 2.1 所示，其中参数设置参考文献[52, 72, 73]中的数据。

表 2.1 仿真参数设置

参数	取值
B	5MHz
ω_0	10^{-9} W
T	1s
κ	10^{-27}
F_{edge}	10G cycle
o_i	300~500Kbit
c_i	200~400 cycle/bit
g_i	10^{-6}

仿真程序在 Matlab 平台下编写，通过选取系统参数生成不同场景下的边缘用户代价，边缘云利用率等数据。

2.4.2 MECOS 的仿真结果分析

本节首先考虑了 MECOS 算法中，系统参数取不同值时 MECOS 算法的表

现。

图 2.2 展示了 MECOS 算法下边缘云移动设备数目对边缘云利用率的影响。仿真实验在移动设备数量 $n=5,10,15,\dots,50$ 的系统参数下进行。可以观察到,在移动设备数量较少时,边缘云和信道的资源充足,移动设备分配更多的时隙卸载更多的数据,但由于移动设备数量较少此时边缘云计算资源利用率不高。随着移动设备数目逐渐增多,移动设备对有限的计算资源之间出现了竞争,卸载数据增长速率开始逐渐降低,边缘云利用率的增长速率也开始降低。由于边缘云的计算资源是有限的,随着移动设备数目达到 25 时,边缘云计算资源被全部利用。算法通过调度边缘云计算资源与信道资源分配给不同移动设备来使总能耗最低。随着设备数量继续增多,边缘云使用率始终保持在 100%的水平,说明边缘云计算资源被充分利用了。

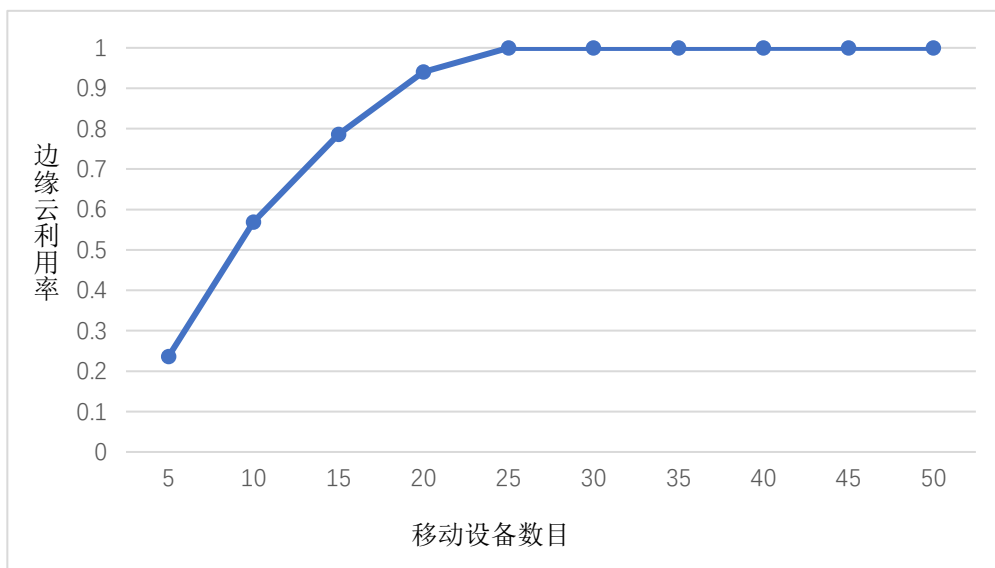


图 2.2 移动设备数目对边缘云利用率的影响

算法运行时间以及算法运行结果是评价算法性能的重要标准。表 2.2 展示了当误差选择系数 η 为不同值时算法迭代次数以及最终结果,此时设置移动设备数量为 50。平均运行时间是通过在 AMD Ryzen R7 3700x 3.6GHz 处理器和 16GB RAM 的计算机上运行 MECOS 算法获得的。

表 2.2 η 对算法运行时间及算法结果的影响

误差选择系数 η	迭代次数	运行时间 (s)	完成任务的总 能耗 (J)
------------------	------	-------------	------------------

10^{-2}	4	0.06	30.6136
10^{-3}	16	0.19	28.4482
10^{-4}	44	0.52	27.3482
10^{-5}	206	2.46	27.3234
10^{-6}	742	8.96	27.3224

从表 2.2 中可以发现取误差选择系数 η 为 10^{-4} 最合适。当 η 分别取 10^{-2} 和 10^{-3} 时, 虽然运行时间少, 但 $\eta=10^{-4}$ 相比与 $\eta=10^{-2}$ 和 $\eta=10^{-3}$ 消耗的总能耗分别降低了 10.66%, 3.86%, 并且 $\eta=10^{-4}$ 的运行时间也是可以接受的。 $\eta=10^{-6}$ 和 $\eta=10^{-5}$ 相比与 $\eta=10^{-4}$ 总能耗分别降低了 0.094%, 0.091%, 几乎没有差别, 但运行时间相差很大, 综上所述取 η 为 10^{-4} 最合适。

为了进一步判断 MECOS 的扩展性和收敛性, 表 2.3 展示了不同移动设备数目下算法的平均运行时间。

表 2.3 不同移动设备数目下算法的平均运行时间

移动设备数目	10	20	30	40	50	100
平均运行时间(s)	0.147	0.244	0.386	0.452	0.526	0.951

从表 2.3 中可以发现, 随着移动设备数目的增加 MECOS 的平均运行时间呈线性增加, 具有良好的扩展性和收敛性。

为了更直观的评估 MECOS 的性能, 本文模拟了以下基准卸载策略来与 MECOS 算法进行比较。

(1) 本地计算策略(Local Computing Only, LCO): 在 LCO 策略中, 所有移动设备产生的计算任务都在设备本地执行而不进行计算卸载。

(2) 资源平均分配策略(Resource Equal Allocation, REA): 在 REA 策略中, 用于计算卸载的信道资源以及边缘云计算资源平均分配给各个移动设备。

(3) 按任务量优先级分配策略(Task-based Priority Allocation, TPA): 在 TPA 策略中, 计算任务更大的移动设备具有较高的卸载优先级, 边缘云资源与信道资源以此为权重来进行分配。

图 2.3 展示了在四种策略下不同移动设备数目下平均每个移动设备的能耗。移动设备数量设置为 $n=10, 20, \dots, 100$ 。

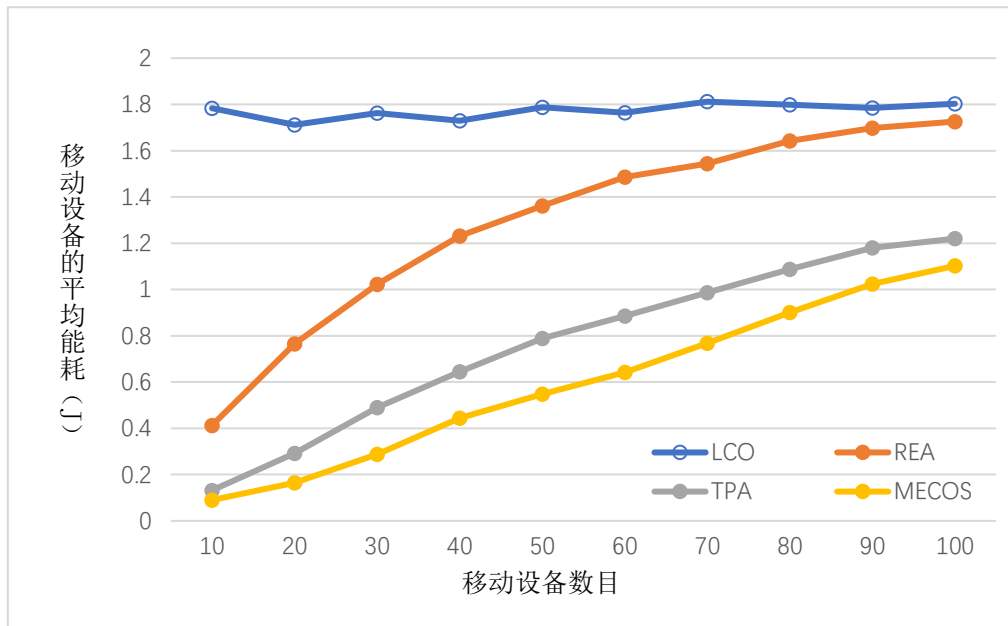


图 2.3 不同策略下移动设备数目对平均能耗的影响

由图 2.3 中可知，由于 LCO 策略下每个移动设备都在本地执行计算任务，没有占用信道资源与边缘云计算资源，因此能耗消耗一直最大，而且平均能耗与移动设备数目无关，一直维持在一个稳定的水平。MECOS 算法相比与 LCO 策略能耗降低了 38.82%~94.95%，随着移动设备数目的增多，能耗降低的比例在逐渐减少，主要原因是随着移动设备数目的增多，信道资源与计算资源都是有限的，单个移动设备所能使用的资源逐渐减少。在移动设备数量相对较少时(10-30)时，边缘云计算资源比较充足，计算资源调度对不同策略影响效果较小，REA，TPA 和 MECOS 算法主要差距在信道资源调度方面，MECOS 算法相比与 REA 策略能耗降低了 71.87%~78.20%，相比与 TPA 策略能耗降低了 31.81%~41.27%。在移动设备数量相对较多(40-100)时，MECOS 算法相比与 REA 策略能耗降低了 36.14%~63.93%，相比与 TPA 策略能耗降低了 9.83%~24.04%。其主要原因是 MECOS 算法充分利用了信道资源与边缘云计算资源，而 REA 策略以及 TPA 策略对于信道资源与边缘云计算资源利用都有不足，但随着设备数量增加，边缘云计算资源和信道资源是有限的，绝大部分计算任务要在移动设备本地进行计算，边缘云计算资源和信道资源带来的减少的能耗占总能耗的比重不断降低。但 MECOS 策略一直是最优的。

图 2.4 展示了在移动设备数目为 30 时，不同策略下边缘云计算资源对移动

设备的平均能耗的影响。

由图 2.4 可以发现,当边缘云计算资源有限时,完成任务的总能耗都非常高,四种策略的节能效果并不明显,但是 MECOS 的节能效果始终是最优秀的。随着边缘云计算资源的增加,完成任务的总能耗开始迅速降低,节能效果越来越明显,但能耗降低的速率开始减慢,主要由于信道资源是有限的,卸载数据不能无限制的增加,形成了制约。

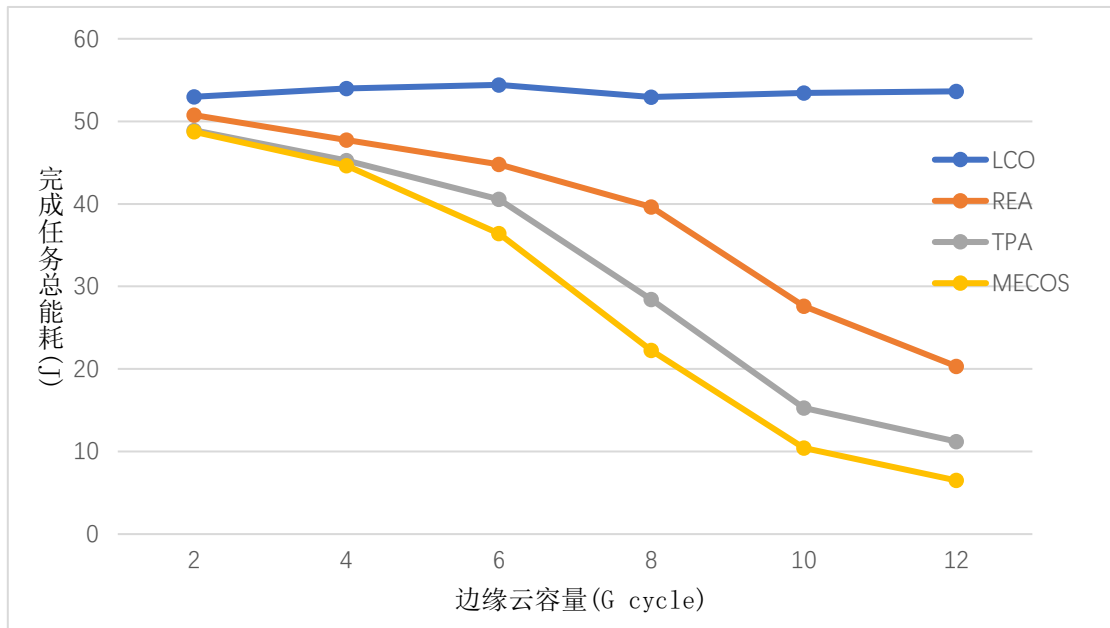


图 2.4 不同策略下边缘云计算资源对完成任务总能耗的影响

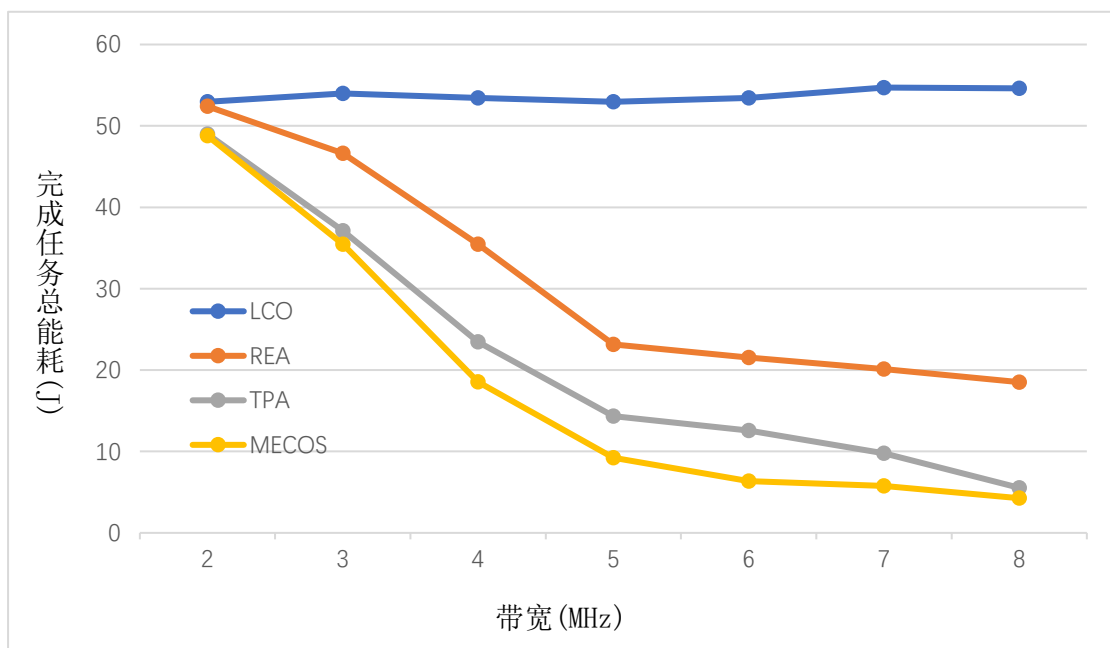


图 2.5 不同策略下带宽资源对完成任务总能耗的影响

图 2.5 展示了移动设备数量为 30 时，带宽的变化对于完成任务总能耗的影响。当带宽较低时，在规定时间内卸载相同数据时需要更高的发射功率，传输能耗会增加，移动设备更倾向于卸载较少的数据，带宽成为节能效果的瓶颈。而随着带宽增大，相同时隙资源内卸载的数据更多，传输能耗相应的降低，节能效果显著增加。但当带宽增加到一定程度后，边缘云的计算资源成为瓶颈，节能效果增速减慢。然而，无论带宽大小变化，MECOS 的节能效果始终是最好的。

2.5 本章小结

本章针对移动设备忽视信道与边缘云计算资源而导致卸载性能低下乃至任务失败问题，提出了一种有限资源下的能源高效型计算卸载决策方法。综合考虑了移动设备的计算频率，传输功率，卸载数据，边缘云的计算能力以及信道资源等变量，改进了卸载任务的能源模型。分别建立了最佳卸载数据量和本地设备最优计算频率的模型、最佳卸载数据量和移动设备发射功率的模型以及信道资源与最佳卸载数据量的模型，将问题简化为单变量优化问题，并基于内罚函数法进行求解，提高了卸载策略的实际可行性以及资源配置的效率。最后进行仿真实验，将仿真结果与其它基准算法进行比较。在不同移动设备数目下(10-100)，MECOS 算法相比与 LCO 策略移动设备的能耗降低了 38.82%~94.95%。在移动设备数量相对较少时(10-30)时，MECOS 算法相比与 REA 策略移动设备的能耗降低了 71.87%~78.20%，相比与 TPA 策略移动设备的能耗降低了 31.81%~41.27%。在移动设备数量相对较多(40-100)时，MECOS 策略相比与 REA 策略移动设备的能耗降低 36.14%~63.93%，相比与 TPA 策略移动设备的能耗降低了 24.04%~9.83%。

第三章 基于 Stackelberg 博弈的资源定价与卸载决策

3.1 引言

边缘云的部署和维护是有成本的,用户在使用边缘云的计算资源时,除了能耗与延迟等自身开销外,往往还需要对占用的边缘云资源进行付费,目前大多数关于计算卸载的研究聚焦于降低用户的开销,却较少有研究聚焦于边缘云的收益问题。因此需要考虑边缘云定价策略对用户卸载策略的影响。

Stackelberg 博弈是 MEC 系统中使用最广泛的定价博弈模型,其特点是博弈参与者的层次结构清晰。在 Stackelberg 博弈模型中,边缘云占据主导地位,可以调整定价以实现自身利益最大化,而用户处于跟随者位置,可以跟随边缘云的定价调整卸载策略以确保自身利益,这种博弈模型更加符合实际。

目前的 Stackelberg 博弈大都是对边缘云计算资源平均分配假设的研究,但在实际中,不同移动设备的延迟敏感度不同必然会导致计算资源需求的差异,因此边缘云有选择地将不同计算能力的计算资源出售给用户更为现实。

不同于以往的研究忽略了边缘云定价策略和非平均定价的影响,本章提出了一种将边缘云卸载决策与 Stackelberg 博弈定价有机结合的新方法。分别建立了边缘云定价和购买最优计算资源块的决策模型以及最佳卸载数据量和购买计算资源块数量之间的关系模型,实现了用户的最优卸载决策,确立了边缘云定价的上下界。最后,提出了一种基于动态规划的边缘云定价卸载算法,实现了边缘云效用最优定价和各用户自身效用的最大化。

3.2 系统模型

3.2.1 任务时延模型

本文考虑一个多用户边缘计算系统,该系统包含一个边缘云服务器和 n 个用户,由集合 $N = \{1, 2, \dots, n\}$ 表示。为了模拟系统的异构性,使用户设备的 CPU 频率、任务数据量和任务的计算复杂度各不相同。设 f_i (Hz) 是表征用户设备的计算能力的用户设备 i 的 CPU 频率。设任务数据量为 o_i (bit), d_i (bit) 表示从用户卸

载到边缘云的数据量, 计算每一 bit 数据所需的 CPU 周期数为 c_i (CPU cycle/bit), 其表征了任务的计算复杂性。

用户设备的本地执行模型与数据传输模型与第二章是一致的, 在本节中, 边缘云的计算能力是有限的, 大小为 F_{edge} (HZ), 边缘云将计算资源分成 M 个计算资源块出售, 计算资源块计算速率为 f_c ($F_{edge} = M * f_c$), 每个用户可以根据自己的需要购买计算资源块, 因此用户 i 的边缘云执行时延为

$$t_i^e = \frac{d_i c_i}{k_i f_c}. \quad (3.1)$$

其中, k_i ($k_i \geq 0$) 是用户 i 根据需要购买的计算资源块的数量。

总而言之, 用户 i 完成任务所需的时间延迟为

$$t_i = \max(t_i^l, t_i^e + t_i^o). \quad (3.2)$$

其中, t_i^l 和 t_i^o 分别为第二章介绍的用户设备的执行时延和传输时延, 完成任务所需要的延迟为用户设备本地执行时延和边缘云执行时延加上传输时延的较大者。

3.2.2 博弈模型

边缘云需要一定的维护和部署成本, 所以用户需要为自己占用的边缘云计算资源付费。边缘云通过出售其计算资源块获得收益, 其效用最大化问题如 P3.1 所示:

$$\begin{aligned} P3.1: \max U_{edge} &= \sum_{i=1}^n \mu_i k_i \\ s.t. \quad &\sum_{i=1}^n k_i f_c \leq F_{edge}. \end{aligned} \quad (3.3)$$

其中 μ_i 是边缘云对单位计算资源块的定价。

用户支付占用边缘云资源的费用, 通过将部分计算任务卸载到边缘云来降低延迟成本, 因此用户的成本分为两部分, 第一部分是如式(3.2)所示的延迟成本, 第二部分是支付给边缘云的费用。为了统一时延成本和支付成本的维度, 使用支付成本相对于时延的加权因子 λ_i ($\lambda_i \geq 0$)。因此, 用户 i 完成计算任务的成本为

$$U_i = t_i + \mu_i k_i \lambda_i. \quad (3.4)$$

引理 3.1: 当用户卸载的数据量小于或等于

$$x_i = o_i c_i \left/ \left(c_i + \frac{f_i}{r_i} + \frac{c_i f_i}{k_i f_c} \right) \right., \quad (3.5)$$

完成这项任务所需要的时间为 t_i^l 。当用户卸载的数据量大于 x_i ，完成任务所需的时间为 $(t_i^e + t_i^o)$ 。

证明：当用户购买的计算资源块个数 $k_i=0$ 时，用户不能卸载数据，否则任务永远不会完成，此时 $x_i=0$ 满足引理 3.1。

当 $k_i > 0$ ，我们可以很容易地发现 t_i^l 随着用户卸载数据而减小， $(t_i^e + t_i^o)$ 随着卸载数据的增加而增加。当 $t_i^e + t_i^o = t_i^l$ 时，我们可以得到此时的卸载数据量 $d_i = x_i$ 。引理 3.1 得证。

根据引理 3.1，用户 i 完成计算任务所需的成本可以转换为

$$U_i = \begin{cases} \frac{(o_i - d_i)c_i}{f_i} + \mu_i k_i \lambda_i, & 0 \leq d_i \leq x_i \\ \frac{d_i c_i}{k_i f_c} + \frac{d_i}{r_i} + \mu_i k_i \lambda_i, & x_i < d_i < o_i \end{cases}. \quad (3.6)$$

因此，用户 i 的成本最小化问题如 P3.2 所示：

$$\begin{aligned} P3.2: \min U_i \\ s.t. \quad & 0 \leq d_i \leq o_i, \\ s.t. \quad & 0 \leq k_i \leq \frac{C_{edge}}{f_c}. \end{aligned} \quad (3.7)$$

其中， C_{edge} 表示此时边缘云的剩余计算能力。

3.3 基于 Stackelberg 博弈的资源定价与卸载决策设计

本节采用反向归纳法来求解最大化边缘云的收益，最小化每个用户成本的资源定价与卸载决策^[74]。首先，在边缘云给定价格的情况下解决 P3.2，每个用户自主确定自己的最优卸载数据量和购买的最佳计算资源块数目；第二，将用户的卸载策略代入到 P3.1 中，边缘云以增加自身效用为目标，调整到最优定价 μ 。

3.3.1 用户成本的优化

假定边缘云已经确定了边缘云计算资源块的价格，用户将会从自身利益最大

化出发确立最优卸载数据量和购买的最佳计算资源块数目。在定理 3.1 中，确立了用户购买的最佳计算资源块数目与最优卸载数据量的关系模型。

定理 3.1: 当用户购买的最佳计算资源块确定时，用户要卸载的最佳数据量为 $d_i^* = x_i$ 。

证明: 假设存在 $d_i^* \neq x_i$ ，使得 $U_i(d_i^*) < U_i(x_i)$ 。当 $d_i^* < x_i$ 时，根据引理 3.1，我们可以发现 U_i 在 $d_i \leq x_i$ 时随着 d_i 的增加单调递减，则一定存在 $U_i(d_i^*) > U_i(x_i)$ 。当 $d_i^* > x_i$ 时，根据引理 3.1，我们还可以发现 U_i 在 $d_i > x_i$ 时随着 d_i 的增加单调递增，必然存在 $U_i(d_i^*) > U_i(x_i)$ 。因此假设是错误的，定理 3.1 得证。

根据定理 3.1，P3.2 可以转换为

$$\begin{aligned} P3.3: \min U_i &= \frac{(o_i - x_i)c_i}{f_i} + \mu_i k_i \lambda \\ s.t. \quad 0 &\leq k_i \leq \frac{C_{edge}}{f_c}. \end{aligned} \quad (3.8)$$

求解博弈问题时需要获得其纳什均衡解。当博弈达到纳什均衡时，用户不能通过单方面调整卸载策略来增加效用^[75]。根据定理 3.1 的结果，进一步确定用户购买的最优计算资源块数量，获得博弈的纳什均衡解。

定理 3.2: 对于边缘云给定的价格，用户之间的非合作博弈存在纳什均衡。纳什均衡解为

$$k_i^* = \begin{cases} 0 & , k_i^s < 0 \\ k_i^s & , 0 \leq k_i^s \leq \frac{C_{edge}}{f_c} \\ \frac{C_{edge}}{f_c} & , k_i^s > \frac{C_{edge}}{f_c} \end{cases} \quad (3.9)$$

其中

$$k_i^s = \frac{c_i r_i \sqrt{\lambda_i c_i f_c o_i \mu_i} - \lambda_i c_i f_i r_i \mu_i}{(\lambda_i c_i f_c r_i + \lambda_i f_i f_c) \mu_i}. \quad (3.10)$$

证明: P3.3 的可行域是非空凸子集，目标函数在可行域是凸的，P3.3 是凸优化问题。在凸优化问题中，局部最优解就是全局最优解。所有用户都取全局最优解，即博弈模型的纳什均衡解。

为了确定用户对边缘云定价的最优响应即所购买的计算资源块的数量，P3.3 的拉格朗日函数构造如下：

$$L_i(p_i, \eta_i) = \frac{(o_i - x_i)c_i}{f_i} + \mu_i k_i \lambda_i + \eta_i (k_i - \frac{C_{edge}}{f_c}), \quad (3.11)$$

其中拉格朗日乘数 $\eta_i \geq 0$ ，然后应用 KKT(Karush-Kuhn-Tucker)条件^[76]，得到包含拉格朗日乘数的购买最优计算资源块数量的表达式

$$\frac{\partial L_i}{\partial k_i} = \lambda_i \mu_i - \frac{c_i^3 f_i^2 o_i}{f_c (\frac{c_i f_i^2}{f_c k_i} + \frac{f_i^2}{r_i} + c_i f_i)^2 k_i^2} + \eta_i, \quad (3.12)$$

$$k_i^* = \frac{c_i r_i \sqrt{\lambda_i c_i f_c o_i \mu_i + c_i f_c o_i \eta_i} - \lambda_i c_i f_i r_i \mu_i - c_i r_i f_i \eta_i}{(\lambda_i c_i f_c r_i + \lambda_i f_i f_c) \mu_i + c_i r_i f_c \eta_i + f_i f_c \eta_i}. \quad (3.13)$$

根据 KKT 条件的互补松弛，我们可以得到

$$\eta_i (k_i - \frac{C_{edge}}{f_c}) = 0. \quad (3.14)$$

当 $\eta_i = 0$ 时，最优解位于可行域内，即 $0 < k_i^* < (C_{edge}/f_c)$ ，将 $\eta_i = 0$ 代入式(3.13)可得

$$k_i^* = k_i^s = \frac{c_i r_i \sqrt{\lambda_i c_i f_c o_i \mu_i} - \lambda_i c_i f_i r_i \mu_i}{(\lambda_i c_i f_c r_i + \lambda_i f_i f_c) \mu_i}. \quad (3.15)$$

当 $\eta_i > 0$ 时，最优解位于可行域的边界，即 0 或 (C_{edge}/f_c) 。由于二阶偏导数 $(\partial^2 U_i / \partial k_i^2) > 0$ ，因此用户 i 购买的最优计算资源块数量为

$$k_i^* = \begin{cases} 0 & , k_i^s < 0 \\ k_i^s & , 0 \leq k_i^s \leq \frac{C_{edge}}{f_c} \\ \frac{C_{edge}}{f_c} & , k_i^s > \frac{C_{edge}}{f_c} \end{cases}. \quad (3.16)$$

显然，该策略使得用户的非合作卸载博弈达到纳什均衡点。定理 3.2 得证。

此外，根据式(3.16)可以观察到，当边缘云的定价 μ_i 降低（增加）时，用户购买的计算资源块的数量增加（减少），相应的卸载的数据量增加（减少）。当定价 μ_i 低于某个阈值时，用户将购买所有剩余的计算资源块。而当定价 μ_i 高于某个

阈值时，用户将不会购买任何计算资源块。

3.3.2 边缘云效用优化

边缘云通过出售其计算资源块获得收益，根据定理 3.2 可知当价格确定时，可以确定用户购买的计算资源块，因此边缘云通过对定价的调整来提高自身收益。在定理 3.3 中给定了边缘云定价的搜索范围。

定理 3.3: 为了最大化边缘云的效用，边缘云对用户 i 的计算资源块定价有一个上下界，即

$$\frac{c_i f_c o_i}{\lambda_i [f_i + C_{edge} (1 + \frac{f_i}{c_i r_i})]^2} \leq \mu_i \leq \frac{c_i f_c o_i}{\lambda_i f_i^2}. \quad (3.17)$$

证明: U_i 对 k_i 的一阶偏导数为

$$\frac{\partial U_i}{\partial k_i} = \lambda_i \mu_i - \frac{c_i^3 f_i^2 o_i}{f_c (\frac{c_i f_i}{f_c k_i} + \frac{f_i}{r_i} + c_i f_i)^2 k_i^2}, \quad (3.18)$$

二阶偏导数为

$$\frac{\partial^2 U_i}{\partial k_i^2} = \frac{2c_i^3 f_c^2 o_i r_i^2 (c_i r_i + f_i)}{((c_i f_c r_i + f_i f_c) k_i + c_i f_i r_i)^3}. \quad (3.19)$$

从物理意义上看，用户设备的所有参数都是非负的。二阶偏导数 $(\partial^2 U_i / \partial k_i^2) > 0$ 。因此一阶偏导数一定是单调递增的。由于 $k_i = 0$ 是 P3.3 的一种可行解，对于边缘云中的给定价格，用户必须采用一种成本低于完全本地计算的成本的策略。所以边缘云给出的价格必须是当偏导数 $(\partial U_i / \partial k_i) = 0$ 时，用户购买的计算资源块 $k_i \geq 0$ ，因此可得

$$\mu_i \leq \frac{c_i f_c o_i}{\lambda_i f_i^2}. \quad (3.20)$$

作为博弈中的主导参与者，边缘云会根据每个用户的卸载决策动态调整其定价，以最大化其效用。根据式(3.16)可知，当 $0 \leq k_i^s \leq (C_{edge} / f_c)$ 时，边缘云给予用户 i 的定价 μ_i 越低，则用户购买的计算资源块越多，相应卸载的数据也会更多。由于用户购买的计算资源块 $k_i^s = (C_{edge} / f_c)$ 及其对应的定价是解决 P3.1 的可行方案，

如果此时边缘云继续降低定价，用户 i 购买的计算资源块数量不能继续增加，定价降低必然会导致边缘云的效用下降。因此，边缘云出于以自身效用最大化为目标，对每个用户出售的计算资源块定价有一个下限。让 $k_i^s \leq (C_{edge}/f_c)$ ，可得

$$\mu_i \geq \frac{c_i f_c o_i}{\lambda_i [f_i + C_{edge} (1 + \frac{f_i}{c_i r_i})]^2}. \quad (3.21)$$

定理 3.3 得证。

根据定理 3.3，边缘云通过调整定价来控制每个用户购买的计算资源块的数量，从而将 P3.1 简化为 P3.4，

$$\begin{aligned} P3.4: \max U_{edge} &= \sum_{i=1}^n \mu_i k_i^s \\ s.t. & \frac{c_i f_c o_i}{\lambda_i [f_i + C_{edge} (1 + \frac{f_i}{c_i r_i})]^2} \leq \mu_i \leq \frac{c_i f_c o_i}{\lambda_i f_i^2}. \end{aligned} \quad (3.22)$$

从中可以看出 c_i ， o_i 越大， λ_i ， f_i 越小，边缘云对单位计算资源块定价的上下界越高。这表明，边缘云更有可能给那些任务计算复杂度高、任务总数据量大的用户提供较高的定价；而对于那些自身计算能力高、支付权重高的用户，给出的定价可能相对较低。

3.4 算法设计

3.4.1 算法分析

根据定理 3.3，我们可以得到边缘云最优定价 μ_i^* 与用户购买最优计算资源块数量 k_i^* 之间的关系，即

$$k_i^* = \frac{c_i r_i \sqrt{\lambda_i c_i f_c o_i \mu_i^*} - \lambda_i c_i f_i r_i \mu_i^*}{(\lambda_i c_i f_c r_i + \lambda_i f_i f_c) \mu_i^*}. \quad (3.23)$$

根据 $(\partial k_i^* / \partial \mu_i^*) < 0$ 可知 k_i^* 是关于 μ_i^* 的双射函数。因此 P3.4 中边缘云确定最优定价的问题就可以转化为用户确定购买最优计算资源块数量的问题。将 k_i^* 的所有可能值表示为集合 K_i ，可得

$$k_i^* \in K_i = \left\{ 0, 1, 2, \dots, \frac{C_{edge}}{f_c} \right\}. \quad (3.24)$$

对于用户 i , k_i^* 只能是 K_i 中的一个元素, 即每个取值是互斥的, 因此 K_i 可以看作是一组物品。边缘云中的计算资源块总数可视为背包的总容量。用户 i 购买的计算资源块 k_i^* 的数量是物品的体积。当确定用户 i 购买的计算资源块 k_i^* 时, 边缘云的定价 μ_i^* 是唯一的, 相应的 $k_i^* \mu_i^*$ 是此时物品的价值。P3.4 等同于解决哪些物品可以装入背包中, 从而使这些物品的体积之和不超过背包的容量, 并且这些物品价值的总和是最大的。因此, P3.4 是一个可用动态规划概念求解的分组背包问题, 求解的关键是推导出描述最优解的状态转移方程。

设 $E[i][k]$ 表示边缘云通过向前 i 个用户出售 k 块计算资源块可以获得的最大收益。根据式(3.21)可知, 当 k_i 确定时, 此时边缘云的最优定价应该为 $(c_i f_c o_i / \lambda_i [f_i + k_i f_c (1 + f_i / c_i r_i)]^2)$ 。边缘云所获得的收益为

$$E_i(k_i^o) = \frac{c_i f_c o_i k_i^o}{\lambda_i [f_i + k_i^o f_c (1 + \frac{f_i}{c_i r_i})]^2}, \quad (3.25)$$

此时边缘云可以分配给其他 $(i-1)$ 个用户的剩余计算资源块是 $(k - k_i^o)$ 。在此基础上, 我们可以得到

$$E[i][k] = E[i-1][k - k_i^o] + E_i(k_i^o). \quad (3.26)$$

所以 P3.4 的状态转移方程为

$$E[i][k] = E[i-1][k - k_i^o] + E_i(k_i^o), \quad k_i^o \in K_i = \left\{ 0, 1, 2, \dots, \frac{C_{edge}}{f_c} \right\}. \quad (3.27)$$

然后通过回溯方法得到每个用户购买的计算资源块的数量, 从而分别通过定理 3.1 和式(3.21)得到用户卸载的数据量和边缘云定价。基于动态规划的定价卸载算法的详细步骤在算法 3.1 中进行了阐述。

算法 3.1: DPPO(Dynamic Programming Pricing Offloading, 动态规划定价卸载)算法

输入: 系统参数 $F_{edge}, B, \omega_0, f_c, \lambda_i, g_i$, 设备参数 o_i, c_i, f_i, p_i 。

输出: 设备的卸载数据量 d^* , 购买的计算资源块数量 k^* 以及边缘云定价 μ^* 。

1 for $i=1$ 到 n

```

2      for  $k=0$  到  $(C_{edge}/f_c)$ 
3           $E[i][k] = E[i-1][k]$ ;
4          for  $k_i^o=1$  到  $(C_{edge}/f_c)$ 
5              if  $k \geq k_i^o$ 
6                  根据式(3.25)计算  $E_i$ ;
7                   $E[i][k] = E[i-1][k - k_i^o] + E_i(k_i^o)$ ;
8              end if
9          end for
10     end for
11 end for
12 回溯得到每个用户购买的计算资源块数量  $k^*$ ;
13 根据定理 3.3 计算  $k^*$  对应的唯一定价  $\mu^*$ ;
14 根据式(3.21)计算用户要卸载的数据量  $d^*$ 。

```

3.4.2 时间复杂度分析

对于 DPPO 算法, 其主体包含三层循环(1-11 行), 第一层循环的次数为 n , 第二层循环和第三层循环以 1 为步长从 0 遍历到 (C_{edge}/f_c) , 每次平均循环次数为 (C_{edge}/f_c) , 所以这部分的时间复杂度为 $O(n(C_{edge}/f_c)^2)$ 。经过以上步骤后, 状态空间由所有可能的值 $E[i][k](i=1, \dots, n; k=0, \dots, (C_{edge}/f_c))$ 组成, 并且使用回溯来确定用户购买的计算资源块的数量部分(12 行)需要遍历状态空间, 其复杂度为 $O(n(C_{edge}/f_c))$ 。剩余部分(13-14 行)是通过遍历一次不同用户购买的计算资源块数量获得的, 时间复杂度为 $O(n)$ 。总而言之, DPPO 算法的时间复杂度为

$$O(n(C_{edge}/f_c)^2 + n(C_{edge}/f_c) + n) = O(n(C_{edge}/f_c)^2). \quad (3.28)$$

3.5 DPPO 的仿真验证与分析

在本节中通过仿真对所提出的 DPPO 算法的性能进行了评估。首先给出了仿真实验设计, 随后的仿真结果表明了该方法的实际有效性。

3.5.1 DPPO 的仿真实验设计

DPPO 算法采用 Stackelberg 博弈论和动态规划来解决边缘云计算资源分配

和定价问题。根据不同用户对计算资源的不同需求，给出了最大化边缘云效用的边缘云最优定价以及在此定价下每个用户最大化自身效用的最优卸载策略。

为了评估 DPPO 算法的性能，对一个由多个用户和一个移动边缘服务器组成的边缘计算系统进行了仿真，该仿真是采用蒙特卡洛方法随机进行实现的。除非有额外说明，否则仿真参数如表 3.1 所示，其中参数设置参考文献[52, 72, 73]中的数据。

表 3.1 仿真参数设置

参数	取值
B	1MHz
ω_0	10^{-9} W
N	60
f_i	0.1~0.5 GHz
f_c	0.01GHz
F_{edge}	10GHz
o_i	300~500 Kbit
c_i	200~400 cycle/bit
g_i	10^{-6}
p_i	0.1W
λ_i	0.5~1.5

仿真程序在 Java 平台下编写，通过选取系统参数生成不同场景下的边缘云效用、用户效用等数据，然后将得到的数据用于 Matlab 平台下的绘图。

3.5.2 DPPO 的性能验证

仿真的第一部分是 DPPO 算法相对于用户数目的性能。为了更直观的评估所提出的 DPPO 算法，我们将其与以下定价策略进行了比较：

统一定价策略(Uniform Pricing, UP): 在 UP 策略中，基于公平原则，边缘云对每个用户以相同的价格对其计算资源块进行统一定价。

贪婪定价策略(Greedy Pricing, GP): 在 GP 策略中，边缘云以最优价格出售其计算资源块，而不考虑边缘云计算资源是有限的限制。

平均定价策略(Average Pricing, AP): 在 AP 策略中，边缘云将其计算资源块平均分配并出售给用户。

图 3.1 比较了在用户相对较少的情况下, 不同定价策略下边缘云的效用。从图 3.1 可以看出, 首先, 每种定价策略下的边缘云收入都随着用户数量的增加而增加。其次, 所提出的 DPPO 策略的边缘云收入总是最高的。UP 策略没有根据用户对价格的不同接受程度设定不同的价格, 不能最大程度地提高边缘云的效益。GP 策略忽略了边缘云资源是有限的限制, 没有从全局因素考虑, 当用户数超过一定数量时, 边缘云的收益会低于 DPPO 策略。AP 策略对计算资源进行平均分配, 忽视了不同用户对计算资源的不同需求, 没有充分利用边缘云计算资源以最大程度地提高其效益。与 UP、AP 和 GP 定价策略相比, DPPO 策略下边缘云获得的收益分别提高了 36.12%、13.05%和 5.98%。

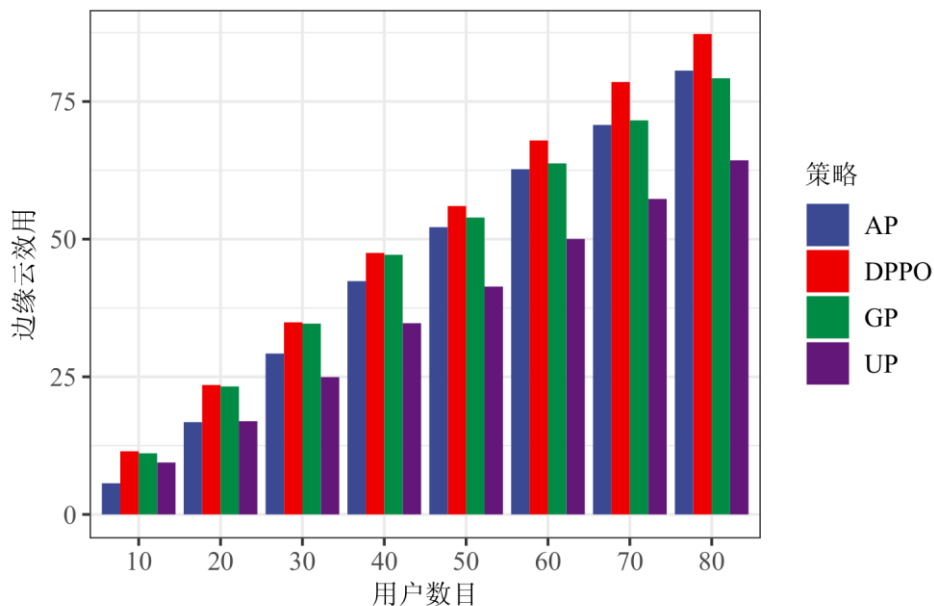


图 3.1 当用户相对较少时, 采用不同定价策略的边缘云效用的比较

图 3.2 比较了用户相对较多的情况下, 不同定价策略下边缘云的效用。DPPO 策略下的边缘云的效用相对于其他策略有了比较显著的提升, 因为 DPPO 策略可以充分利用用户之间的竞争来提高其效用。除此之外, 由于边缘云计算资源是有限的, 边缘云效用的增长速度随着用户数目的增加在逐渐放缓。与 UP、AP 和 GP 定价策略相比, DPPO 策略下边缘云的收益分别提高了 22.53%、36.08%和 25.29%。

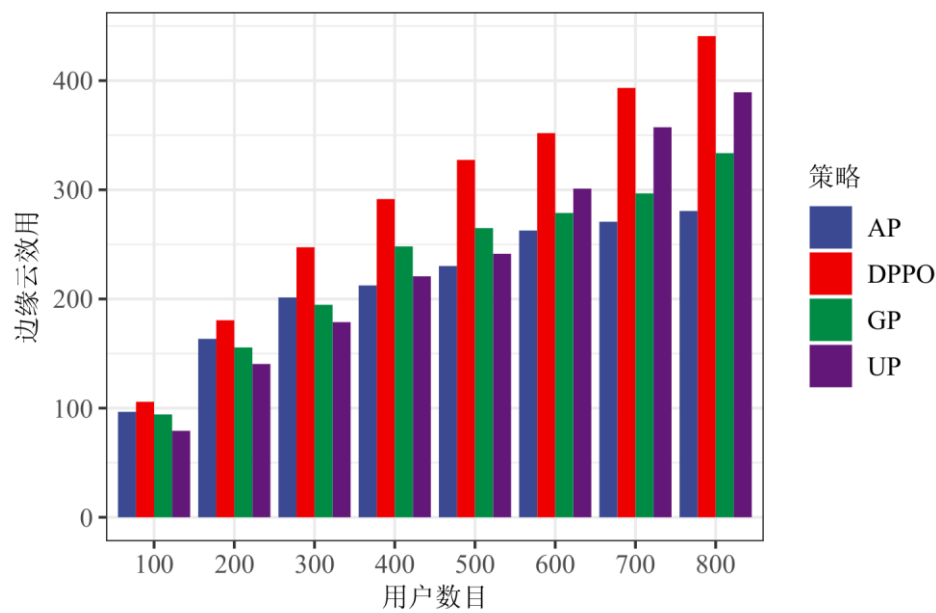


图 3.2 当用户相对较多时，采用不同定价策略的边缘云效用的比较

为了进一步评估所提出的 DPPO 算法的性能，我们比较了以下卸载策略：

本地计算卸载策略(Local Computing Offloading, LCO)：在 LCO 策略中，所有用户的计算任务都在本地执行，而不需要计算卸载。

最佳资源卸载策略(Optimal Resource Offloading, ORO)：在 ORO 策略中，用户将选择购买最佳数量的计算资源块，但用户卸载的数据量是随机的。

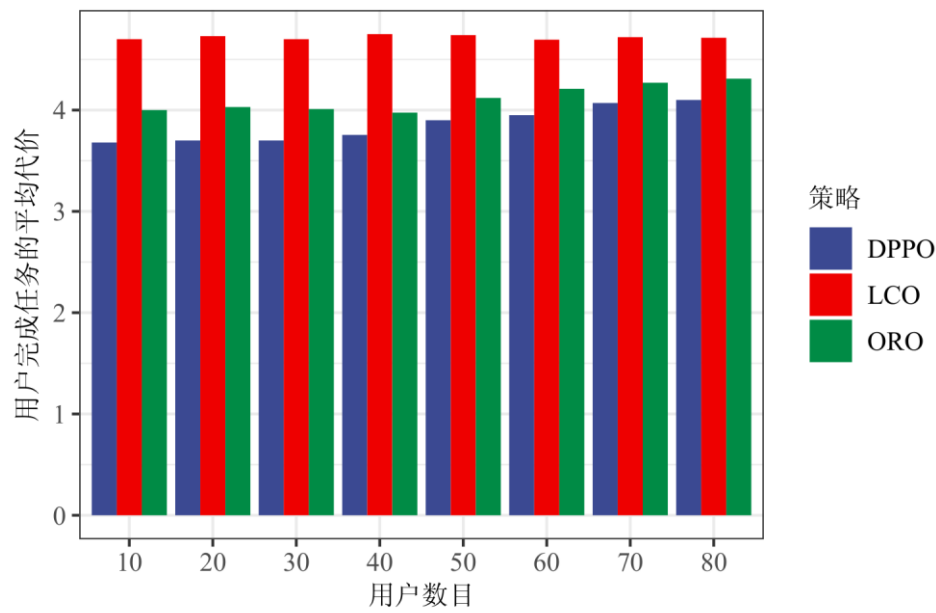


图 3.3 当用户相对较少时，不同卸载策略下的用户成本比较

图 3.3 和图 3.4 分别展示出了在用户相对较少和用户相对较多的情况下使用不同卸载策略下的用户卸载成本的比较。当用户数量相对较少时，DPPO 策略下的用户成本明显降低，因为此时用户数量相对较少，用户之间的竞争并不激烈，用户可以用更低的价格购买足够的计算资源块来降低成本。然而，随着用户数量的增加，边缘云中的计算资源是有限的，用户之间的竞争变得更加激烈，用户需要以更高的价格购买计算资源块，用户的成本开始逐渐增加。除此以外，相比其它策略，DPPO 策略下，用户的成本始终是最低的。在用户数量较少的情况下，与 LCO 和 ORO 卸载策略相比，DPPO 策略分别降低了 18.26%和 6.28%的任务完成成本；在用户数量较多的情况下，DPPO 策略比 LCO 和 ORO 卸载策略分别降低了 4.9%和 1.97%的任务完成成本。

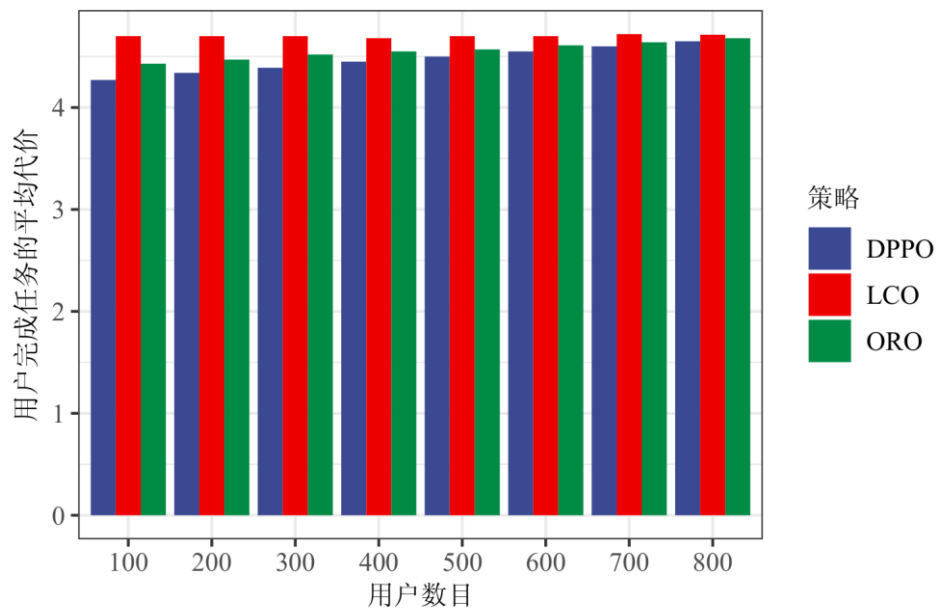


图 3.4 当用户相对较多时，不同卸载策略下的用户成本比较

仿真的第二部分评估了系统参数的影响。选择信道带宽 B 和边缘云计算能力 F_{edge} 对 DPPO 算法进行评估。在这一部分中，用户数量设置为 60。

图 3.5 显示了带宽对边缘云效用的影响。实验是在 $B = 0.2, 0.4, \dots, 1.2 \text{ MHz}$ 的系统参数下进行的。从图 3.5 中可以看出，所有策略都随着带宽的增加而增加，但边缘云效用的增加速度随着带宽的增加而逐渐放缓。这是因为信道带宽的增加提高了传输速度，传输的代价会相应的减少，用户会倾向于卸载更多的数据，购

买更多的计算资源块，从而提高了边缘云的效用，但是边缘云的计算资源是有限的，所以增长的速度会逐渐放缓。UP 策略下的边缘云效用最小且增长最慢，这是因为 UP 策略是对边缘云计算资源的统一定价，没有考虑不同用户对定价的敏感性，因此在带宽不是影响边缘云效用的决定性因素时，用户购买的计算资源块更少，卸载的数据更少。

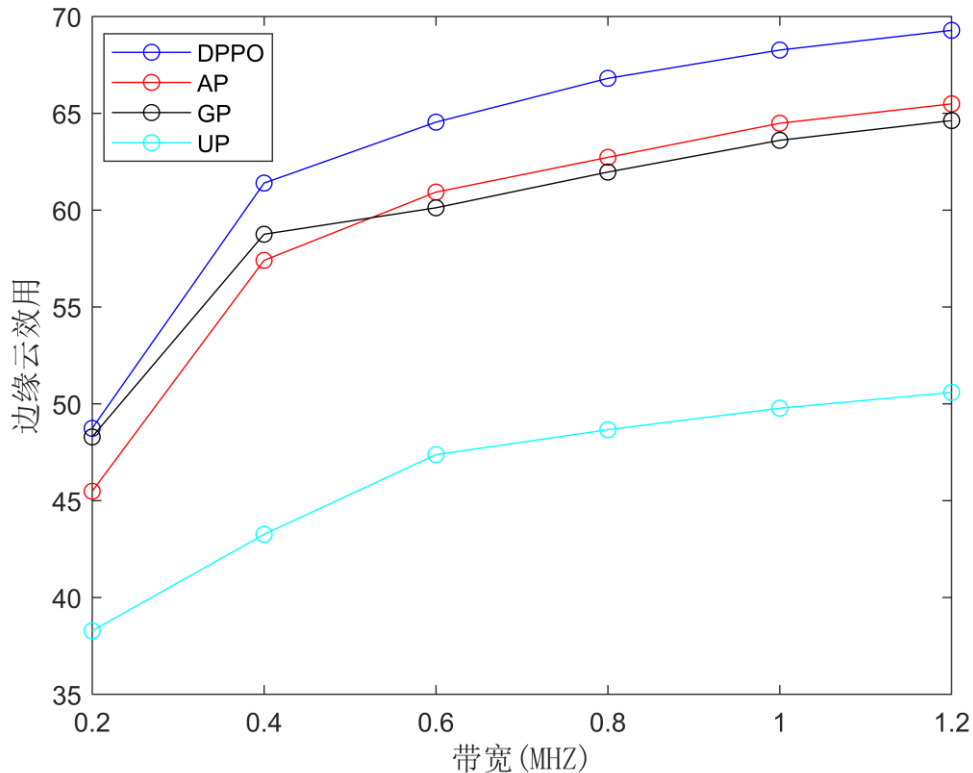


图 3.5 带宽对边缘云效用的影响

图 3.6 显示了边缘云计算资源大小对边缘云效用的影响。实验是在 $F_{edge} = 2, 4, \dots, 12GHz$ 的系统参数下进行的。从图 3.6 可以看出，所有策略都会使得边缘云效用随着计算资源的增加而增加，但边缘云效用的增加速度随着计算资源的增加而逐渐放缓。这是因为随着计算资源的增加，边缘云可以出售更多的计算资源，从而提高其效用，但是用户对计算资源的需求并不是无限增加的，并且带宽资源是有限的形成了制约，所以边缘云效用的增长速度会逐渐放缓。

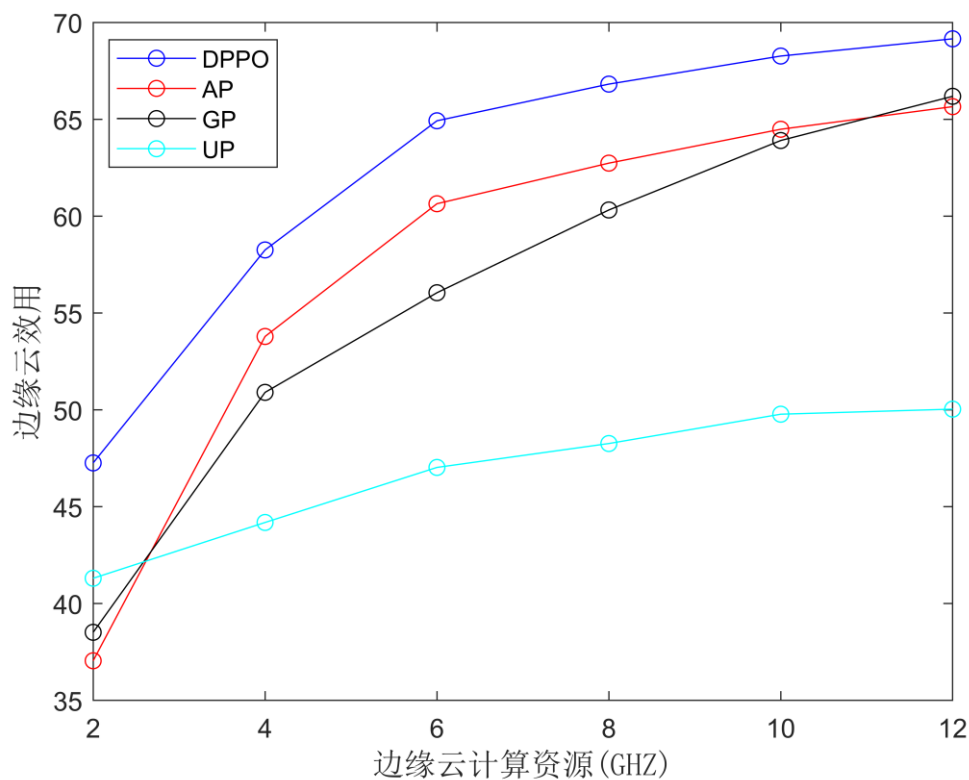


图 3.6 边缘云计算资源对边缘云效用的影响

综上所述，单一的资源丰富并不是提高边缘云效用的有效途径，在为边缘计算系统部署资源时需要协调资源的部署。此外，无论带宽和边缘云计算资源的取值如何，DPPO 策略下的边缘云效用都是最高的。

在第三部分的仿真中，DPPO 算法下程序的平均运行时间和边缘云效用相对于不同 f_c 值的结果如表 3.2 所示。平均运行时间是通过在 Intel Core i5-7200 2.5GHz 处理器和 8GB RAM 的计算机上运行 DPPO 算法获得的。在本部分中，用户数量设置为 60。可以看出，随着 f_c 值的增加，算法的平均运行时间几乎呈指数下降，但边缘云效用也随之降低。当 f_c 选择合适的值(例如 0.01 GHZ、0.02 GHZ)时，DPPO 算法可以在可接受的时间内获得接近最优的卸载决策。这表明该算法具有良好的收敛性。

表 3.2 f_c 对算法运行时间和边缘云效用的影响

f_c (GHZ)	0.002	0.005	0.01	0.02	0.05	0.1	0.2
平均运行时间(s)	7.765	1.408	0.283	0.069	0.014	0.003	0.002
边缘云效用	70.1	69.0	68.2	67.4	66.6	65.9	61.6

表 3.3 显示了 DPPO 算法相对不同用户数目下的平均运行时间。此时, f_c 设置为 0.01 GHz。可以看出, DPPO 算法的平均时间随着用户数的增加而线性增加, 说明该算法具有良好的可扩展性。

表 3.3 用户数目对算法运行时间的影响

用户数目	10	20	30	40	50	60	70	80
平均运行时间(ms)	74	128	170	199	244	283	318	354

3.6 本章小结

在本章中, 我们针对目前 Stackelberg 博弈都是均匀分配边缘云计算资源, 忽略不同用户对计算资源需求差异的问题, 提出了一种将边缘云卸载决策与 Stackelberg 博弈定价有机结合的新方法。首先, 采用 Stackelberg 博弈理论建立了用户最佳卸载数据量和购买最佳计算资源块数量的模型, 将用户的多变量卸载决策问题转换为单变量优化问题, 使用户的卸载决策问题得到简化, 并证明了纳什均衡的存在性。其次, 运用 KKT 条件实现了用户购买最优计算资源块的卸载决策, 建立了边缘云定价和购买最优计算资源块的决策模型, 确立了边缘云定价的上下界。最后, 提出了一种基于动态规划的边缘云定价卸载算法, 实现了边缘云效用最优定价和各用户自身效用的最大化。仿真结果表明, 本文提出的方法不但实现了边缘云效用和用户效用的均衡, 而且具有良好的收敛性和可扩展性。在用户数量较少的情况下, 与 LCO 和 ORO 卸载策略相比, 所提出的 DPPO 策略分别降低了 18.26%和 6.28%的任务完成成本, 与 UP、AP 和 GP 定价策略相比, DPPO 策略下边缘云获得的收益分别提高了 36.12%、13.05%和 5.98%。在用户数量较大的情况下, DPPO 策略比 LCO 和 ORO 卸载策略分别降低了 4.9%和 1.97%的任务完成成本, 与 UP、AP 和 GP 定价策略相比, DPPO 策略下边缘云的收益分别提高了 22.53%、36.08%和 25.29%。

第四章 基于边云协同的多目标卸载决策优化

4.1 引言

目前,移动边缘计算研究主要集中与在一个准静态场景中多用户对各自卸载任务比例进行划分决定边缘云完成的任务大小及用户各自完成的任务大小从而降低任务完成的时延与能耗。但是,边缘云的计算能力通常是有限的,随着用户数目的增加,时延与能耗的降低将陷入瓶颈。为了减少单个边缘云的有限资源对移动边缘计算的移动性和计算性能的限制,技术人员提出在宏小区(macro cell)中的每个小区(small cell)的基站内均部署边缘服务器。除此之外,传统中心云虽然相较于边缘云服务器距离用户相对较远,但在人群密集的场景下,由于其充足的计算能力,将任务卸载到中心云能获得更低的时延及能耗。密集化的边缘服务器的部署及边云协同的调度,比如多边缘云和中心云的存在和任务划分使得计算卸载的选择更具多样性。解决多边缘网络的计算卸载决策问题是充分挖掘移动边缘计算的潜能过程中不可避免的挑战。

针对以上问题,本章依据排队论的思想,以能耗和时延性能最小化为目标,对移动设备的服务请求卸载速率、边缘云的服务请求到达速率以及中心云的服务请求到达速率进行了协同建模与优化。采用二分查找迭代方法对边缘云集群和中心云的服务请求到达速率进行决策,并利用蚁群算法实现了移动设备卸载速率的寻优和服务请求的合理分配。

4.2 系统模型

4.2.1 MEC 场景描述

如图 4.1 所示,我们所考虑的场景是由一个遥远的中心云, k 个边缘云和 n 个移动设备组成的,其中移动设备由集合 $N=\{1,2,...,n\}$ 表示,边缘云由集合 $K=\{1,2,...,k\}$ 表示。每个移动设备执行一个应用程序并生成一系列服务请求,这些服务请求可以由移动设备处理,也可以通过无线接入点将任务卸载到边缘云计算或通过边缘云卸载任务到中心云计算。云计算控制器需要对移动设备的服务请求

的卸载位置进行决策, 比如 k 个边缘云处理的服务请求的分配, 以及是否需要卸载到中心云计算。在所提出的体系结构中, 假设被卸载的任务是独立的, 并且可以被划分为子任务进行并行执行^[77]。根据移动设备, 边缘云, 以及中心云的计算资源不同, 我们将移动设备处的处理模型视为 $M/M/1$ 模型, 边缘云中的处理模型视为 $M/M/c$ 模型, 中心云中的处理模型视为 $M/M/\infty$ 模型^[60,77,78]。

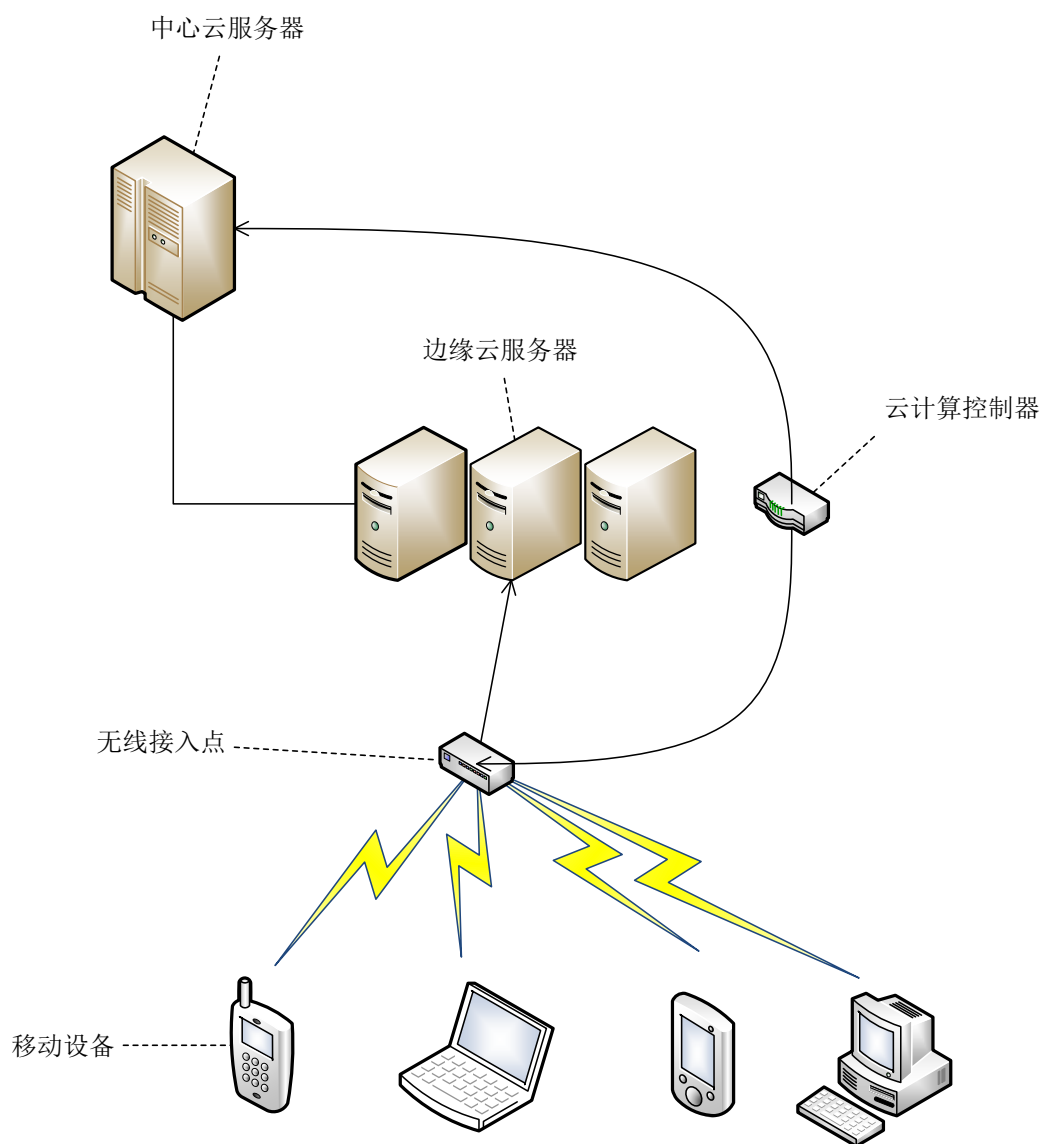


图 4.1 多边缘服务器的边云协同场景

卸载可能会出现三种场景。第一个场景: 卸载到边缘云上, 这种选择在边缘云上负载比较轻的情景下是一种最佳选择, 当随着边缘云负载增加时, 卸载到边缘云上会有更长的排队时间。第二个场景: 卸载到中心云上, 中心云相对于用户距离更远, 需要更长的通信时间, 当边缘云与用户的负载都比较重时, 排队时间

超过了通信时间，卸载到中心云上是一个更好的选择。第三个场景：由移动设备自己处理，当任务比较少时，且边缘云负载较重，卸载到边缘云上会有更长的排队时间，卸载到中心云上需要更长的通信时间，所以将任务由移动设备自己处理是一个更好的选择。

4.2.2 任务执行模型

我们假设移动设备 i ， $i \in N$ 产生的请求遵循泊松过程，平均到达速率为 λ_i 。假设这些请求是计算密集型的，并且相互独立的。移动设备 i 选择将每个服务请求以 p_i^c 的概率卸载到边缘云和中心云中进行计算。因此卸载到云的服务请求遵循平均到达速率为 $p_i^c \lambda_i$ 的泊松过程，移动设备本地处理的服务请求也遵循平均到达速率为 $(1-p_i^c) \lambda_i$ 的泊松过程。用户可以通过优化 p_i^c 来降低完成任务的代价，当 p_i^c 值变大时，传输到边缘云和中心云的请求会更多，相应的边缘云的平均执行时间会增加，本地执行的平均时间会减少，从功耗角度而言，本地处理任务的功耗会随着本地处理服务请求的降低而降低，而用于发送服务请求的射频组件中的功耗随着传输服务请求的增加而增加。

由于移动设备相比于边缘云以及中心云的资源最少，我们将其视为 M/M/1 模型，同时只能处理一个请求，多余的请求需要排队。M/M/1 排队系统的请求数量可以用生灭过程建模，其暂态图如图 4.2 所示^[77]。该系统的稳定条件为

$$\frac{\lambda}{\mu} < 1. \quad (4.1)$$

其中 λ 代表任务请求到达速率， μ 代表任务请求处理速率。

我们用 μ_i^l 表示移动设备 i 中的服务请求的平均处理速度，则移动设备 i 处理服务请求的平均执行时间为

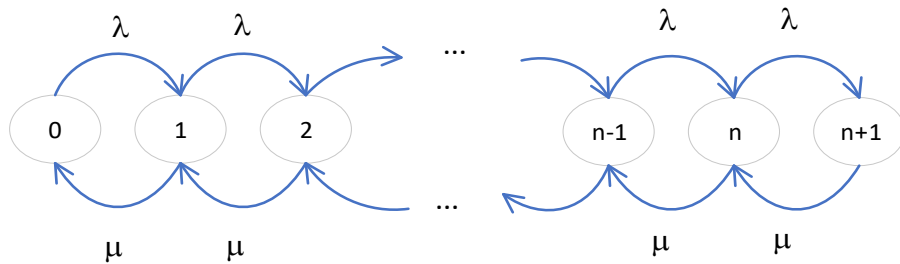


图 4.2 M/M/1 排队系统的暂态图

$$t_i^l = \frac{1}{\mu_i^l - (1 - p_i^c)\lambda_i}. \quad (4.2)$$

我们用 μ_i^r 表示移动设备 i 中发送服务请求的平均速度, 则移动设备 i 传输服务请求的平均传输时间为

$$t_i^r = \frac{1}{\mu_i^r - p_i^c \lambda_i}. \quad (4.3)$$

其中 μ_i^r 与从移动设备到接入点的无线信道容量成正比^[80]。 $p_i^c \lambda_i$ 为移动设备的卸载请求产生的速率。

移动设备 i 的能量消耗包括两部分: (1)移动设备用于本地服务请求处理的能量消耗; (2)向边缘云中发送数据的能量消耗。移动设备 i 的平均执行能耗为

$$e_i^l = \frac{(1 - p_i^c)\lambda_i}{\mu_i^l} \cdot e_i^{l, \max}. \quad (4.4)$$

其中 $e_i^{l, \max}$ 是 CPU 一直处于工作状态时的能量能耗。与平均执行能耗类似, 移动设备 i 的平均传输能耗为

$$e_i^r = \frac{p_i^c \lambda_i}{\mu_i^r} \cdot e_i^{r, \max}. \quad (4.5)$$

其中 $e_i^{r, \max}$ 是射频组件一直处于工作状态时的能量能耗。

边缘云相比于移动设备具有更加丰富的计算资源, 但相比于中心云并不足以承载太多的服务请求。我们认为边缘云为一个 $M/M/c$ 队列模型, 每个服务器的服务速率用 μ_e 来表示。由于每个边缘云的资源有限, 它们只能同时服务于 c 个任务。其他任务应该在队列中等待稍后处理。 $M/M/c$ 排队系统中的请求数量可以用生灭过程建模, 其暂态图如图 4.3 所示。该系统的稳定条件为

$$\rho = \frac{\lambda}{c\mu} < 1. \quad (4.6)$$

其中 ρ 表示排队强度。

来自不同移动设备的服务请求被汇集在一起, 根据泊松过程的性质, 总速率为

$$\lambda_{total} = \sum_{i=1}^n \lambda_i p_i^c. \quad (4.7)$$

由于边缘云的计算资源相对于中心云的资源是有限的,如果边缘云的工作负载太重,排队时间会变得很长,从而降低移动设备的应用程序处理速度,导致无法处理接受到的所有移动设备的服务请求,因此,将任务卸载到中心云是一个常见的选择。我们用 ϕ_c 表示由云处理的服务请求中由中心云处理的比例,则中心云处理的服务请求的平均到达速率为

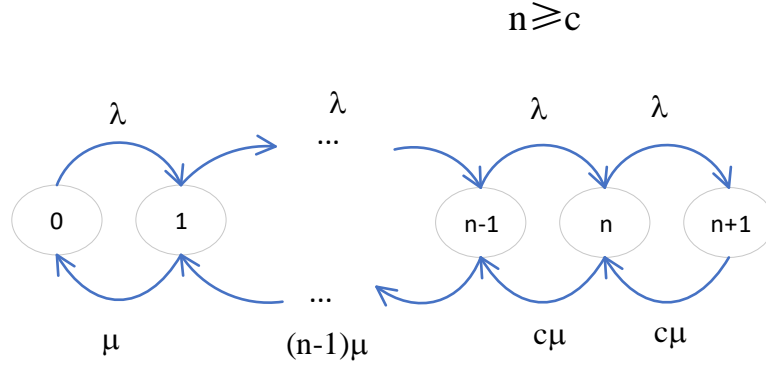


图 4.3 M/M/c 排队系统的暂态图

$$\lambda_c = \phi_c \cdot \lambda_{total}. \quad (4.8)$$

与之相对应的边缘云处理的服务请求的平均到达速率为

$$\lambda_e = (1 - \phi_c) \cdot \lambda_{total}. \quad (4.9)$$

我们假设分配给边缘云 j , $j \in K$ 的服务请求的平均到达速率为 λ_j^e , 基于边缘云 j 的 M/M/c 队列模型和 Erlang 公式^[81], 我们可知边缘云的任务排队强度为

$$\rho_j^e = \frac{\lambda_j^e}{c\mu_e}. \quad (4.10)$$

因此, 边缘云 j 的平均队长 (包含请求排队队长和正在处理的请求数) 为

$$L_j^s = \frac{C(c, \rho_j^e) \lambda_j^e}{(c\mu_e - \lambda_j^e)} + \frac{\lambda_j^e}{\mu_e}. \quad (4.11)$$

根据 Little 定理, 边缘云每个服务请求在边缘云 j 的平均等待时间 (包含等待时间和执行时间) 表示为^[78]

$$t_j^e = \frac{C(c, \rho_j^e)}{c\mu_e - \lambda_j^e} + \frac{1}{\mu_e}. \quad (4.12)$$

其中

$$C(c, \rho_j^e) = \frac{\left(\frac{(c\rho_j^e)^c}{c!}\right)\left(\frac{1}{1-\rho_j^e}\right)}{\sum_{k=0}^{c-1} \frac{(c\rho_j^e)^k}{k!} + \left(\frac{(c\rho_j^e)^c}{c!}\right)\left(\frac{1}{1-\rho_j^e}\right)}. \quad (4.13)$$

因此, 边缘云的总响应时间可以通过将所有的边缘云的响应时间相加来表示, 即

$$t_e = \sum_{j=1}^k p_j t_j^e = \sum_{j=1}^k \frac{\lambda_j^e}{\lambda_e} t_j^e. \quad (4.14)$$

其中 p_j 为卸载到边缘云 j 上的概率。在上述等式中, 系统的总响应时间表示为变量 $\lambda_j^e, j=1 \dots k$ 的函数, 满足限制条件

$$\lambda_1^e + \lambda_2^e + \dots + \lambda_k^e = \lambda_e. \quad (4.15)$$

如果边缘云服务器由于计算资源的限制而无法处理所有请求, 则会通过边缘云将过载的请求传输到中心云。我们假设将这些请求传输到中心云会导致固定的时间延迟 t_f 。由于中心云有足够的计算资源来处理这些请求, 因此中心云中请求的排队时间可以忽略不计。中心云的排队模型为 $M/M/\infty$, 服务率为 μ_c , 通常比边缘云服务器的服务率 μ_e 快, 因此每个服务请求在中心云的处理时间为

$$t_c = t_f + \frac{1}{\mu_c}. \quad (4.16)$$

我们忽略移动设备接受计算结果的时间和能量消耗, 因为通常而言计算结果的大小通常比输入的数据大小小的多, 因此根据式(4.4), (4.5)我们可以得到移动设备 i 完成一个服务请求的平均能耗为

$$E_i = (1 - p_i^c) \cdot e_i^l + p_i^c \cdot e_i^r. \quad (4.17)$$

根据式(4.2), (4.3), (4.14), (4.16)我们可以得到移动设备 i 完成一个请求的平均时延为

$$T_i = (1 - p_i^c) \cdot t_i^l + p_i^c \cdot t_i^r + p_i^c (1 - \phi_c) \cdot t_e + p_i^c \phi_c \cdot t_c. \quad (4.18)$$

相应地, 系统中所有移动设备的平均能耗和平均延时分别为

$$E = \frac{1}{n} \sum_{i=1}^n E_i, \quad (4.19)$$

$$T = \frac{1}{n} \sum_{i=1}^n T_i. \quad (4.20)$$

基于以上分析，将执行任务的延迟与能耗的加权和作为待优化的目标函数。从而目标函数可以表示为

$$\mathbf{Cost} = \alpha \cdot T + \beta \cdot E. \quad (4.21)$$

其中， α 、 β 分别为执行延迟以及能耗的权重，并且限制 $\alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$ 。通过调整权重一方面可以消除延迟、能耗两者量纲的影响，另一方面能够使模型适应不同要求的任务场景，比如若移动设备的电量不足而移动设备执行的服务请求对时延没有要求，则可以设置 $\alpha = 0, \beta = 1$ ；若移动设备电量充足而且对移动设备执行的服务请求的时延要求很高，则可以设置 $\alpha = 1, \beta = 0$ 。

对于决策变量——移动设备的卸载速率和服务请求卸载到边缘云的到达速率以及服务请求卸载到中心云的到达速率，存在如下约束条件：

$$\mu_i^l > (1 - p_i^c) \lambda_i, \quad (4.22)$$

$$\mu_i^r > p_i^c \lambda_i, \quad (4.23)$$

$$c \mu_e > \lambda_j^e, \quad (4.24)$$

$$\sum_{i=1}^n p_i^c \lambda_i = \lambda_c + \sum_{j=1}^k \lambda_j^e. \quad (4.25)$$

基于此，可以将服务请求开销最小化问题用 P4.1 进行描述，

P4.1: $\min \mathbf{Cost}$

$$\begin{aligned} s.t. \text{ c4.1: } & \mu_i^l > (1 - p_i^c) \lambda_i, i \in N, \\ \text{ c4.2: } & \mu_i^r > p_i^c \lambda_i, i \in N, \\ \text{ c4.3: } & c \mu_e > \lambda_j^e, j \in K, \\ \text{ c4.4: } & \sum_{i=1}^n p_i^c \lambda_i = \lambda_c + \sum_{j=1}^k \lambda_j^e. \end{aligned} \quad (4.26)$$

其中，c4.1 限制移动设备本地的请求到达速率不应超过移动设备的处理速率。c4.2 限制移动设备的卸载速率不应超过移动设备发送服务请求的处理速率。c4.3 限制卸载到边缘云上的请求速率不应超过边缘云的处理速率。c4.4 限制移动设备

的卸载速率为服务请求卸载到边缘云的到达速率以及服务请求卸载到中心云的到达速率之和。

4.3 基于蚁群算法的多目标卸载决策设计

我们可以将这个移动设备和云计算控制器组成的交互系统描述为一个两阶段的优化问题。第一个阶段，每个移动设备 i 确定其卸载请求产生的速率 $p_i^c \lambda_i$ 。第二个阶段，云计算控制器将服务请求分配到各个边缘云服务器，并决定是否需要将请求进一步卸载到中心云服务器，确定边缘云和中心云的请求到达速率。

4.3.1 服务请求卸载位置的优化

本节在已知移动设备卸载决策的前提下，对边缘云和中心云的资源分配进行优化，决策移动设备卸载任务的卸载位置，确定边缘云和中心云的请求到达速率。

定理 4.1: 当移动设备确定其卸载请求产生的速率 $p_i^c \lambda_i$ 后，卸载到边缘云的请求到达速率存在一个最优值 λ_e^* ，高于该值 λ_e^* 时，服务请求将进一步卸载到中心云去处理。

证明: 易知，当移动设备确定其卸载请求产生的速率 $p_i^c \lambda_i$ 后， E 的大小不受 λ_e 影响，只有 T 的大小受 λ_e 影响，对于任意移动设备 i ，只有边缘云的执行时间 $p_i^c(1-\phi_c) \cdot t_e$ 以及中心云的执行时间即 $p_i^c \phi_c \cdot t_c$ 受 λ_e 影响，其中 $t_c = t_f + 1/\mu_c$ 是一个固定值， $\phi_c = (\lambda_{total} - \lambda_e)/\lambda_{total}$ ， $(1-\phi_c) = \lambda_e/\lambda_{total}$ 。对 T 求关于 λ_e 的偏导数可得

$$\frac{\partial T}{\partial \lambda_e} = \frac{1}{n} \sum_{i=1}^n \left(\frac{p_i^c \cdot t_e}{\lambda_{total}} + p_i^c \frac{\lambda_e}{\lambda_{total}} \cdot \frac{\partial t_e}{\partial \lambda_e} - \frac{p_i^c t_c}{\lambda_{total}} \right). \quad (4.27)$$

其中 $(\partial t_e / \partial \lambda_e)$ 无法直接求解，求单个边缘云 j 执行时间 t_j^e 关于服务请求到达率 λ_j^e 的偏导数

$$\frac{\partial t_j^e}{\partial \lambda_j^e} = \frac{\frac{\partial C(c, \rho_j^e)}{\partial \lambda_j^e} \cdot (c\mu_e - \lambda_j^e) + C(c, \rho_j^e)}{(c\mu_e - \lambda_j^e)^2}. \quad (4.28)$$

其中，

$$\begin{aligned}
 C(c, \rho_j^e) &= \frac{\left(\frac{(c\rho_j^e)^c}{c!}\right)\left(\frac{1}{1-\rho_j^e}\right)}{\sum_{k=0}^{c-1} \frac{(c\rho_j^e)^k}{k!} + \left(\frac{(c\rho_j^e)^c}{c!}\right)\left(\frac{1}{1-\rho_j^e}\right)} \\
 &= \frac{\left(\frac{1}{c!}\right)}{\sum_{k=1}^c \frac{(1-\rho_j^e)}{k!(c\rho_j^e)^k} + \left(\frac{1}{c!}\right)} \\
 &= \frac{\left(\frac{1}{c!}\right)}{\sum_{k=1}^c \frac{(1-\frac{\lambda_j^e}{c\mu})}{k! \left(\frac{\lambda_j^e}{\mu}\right)^k} + \left(\frac{1}{c!}\right)}.
 \end{aligned} \tag{4.29}$$

设

$$f(\lambda_j^e) = \sum_{k=1}^c \frac{(1-\frac{\lambda_j^e}{c\mu})}{k! \left(\frac{\lambda_j^e}{\mu}\right)^k}, \tag{4.30}$$

可知

$$\frac{\partial C(c, \rho_j^e)}{\partial \lambda_j^e} = \frac{-f'(\lambda_j^e) \cdot \frac{1}{c!}}{(f(\lambda_j^e) + \frac{1}{c!})^2}. \tag{4.31}$$

其中

$$\begin{aligned}
 f'(\lambda_j^e) &= \sum_{k=1}^c \frac{\frac{-1}{c\mu} \cdot \left(\frac{\lambda_j^e}{\mu}\right)^k - k \cdot \left(\frac{\lambda_j^e}{\mu}\right)^{k-1} \cdot \frac{1}{\mu} \cdot (1-\frac{\lambda_j^e}{c\mu})}{k! \left(\frac{\lambda_j^e}{\mu}\right)^{2k}} \\
 &= \sum_{k=1}^c \left(-\frac{1}{k! c\mu \left(\frac{\lambda_j^e}{\mu}\right)^k} - \frac{1-\frac{\lambda_j^e}{c\mu}}{(k-1)! \lambda_j^e \left(\frac{\lambda_j^e}{\mu}\right)^k} \right) \\
 &= \sum_{k=1}^c \left(\frac{(\lambda_j^e - c\mu)k - \lambda_j^e}{k! c\mu \lambda_j^e \left(\frac{\lambda_j^e}{\mu}\right)^k} \right).
 \end{aligned} \tag{4.32}$$

由 (4.3) 可知， $(\lambda_j^e - c\mu) < 0$ ，又因为 $\lambda_j^e > 0$ ，可知 $f'(\lambda_j^e) < 0$ ，则

$(\partial C(c, \rho_j^e) / \partial \lambda_j^e) > 0$ ，最终可得 t_j^e 是关于 λ_j^e 的单调增函数，单个边缘云关于服务请求到达速率是单调增的，显然 t_e 关于边缘云的服务请求到达速率 λ_e 也是单调增的，即 $(\partial t_e / \partial \lambda_e) > 0$ 。

根据式(4.27)可知，当 $\lambda_e = 0$ 时， $t_e = 0$ ， $(\partial T / \partial \lambda_e) < 0$ ，完成一个请求的平均执行时间 T 随着边缘云服务请求到达速率 λ_e 的增加而减少，当随着 λ_e 增加，使得 $t_e \geq t_c$ 时， $(\partial T / \partial \lambda_e) \geq 0$ ， T 在之后是关于 λ_e 的单调增函数。因此当用户确定远程处理的总服务请求比例后，卸载到边缘云的服务请求到达率必然存在一个最优值 λ_e^* 使得完成任务的代价最小，定理 4.1 得证。

由定理 4.1 可知，边缘云的服务请求到达速率存在最优值 λ_e^* ，我们可以很容易发现当 $t_e > t_c$ 时，此时 T 是关于 λ_e 的单调增函数，此时的 λ_e 超过边缘云的服务请求最优到达速率，因此我们可以计算出低于 λ_e 下的 k 个边缘云的服务请求到达速率，并将值缓存在云计算控制器上避免每次对服务请求的分配进行重复计算。具体计算过程如算法 4.1 所示。

算法 4.1: ISC(Iteration Search Cache, 遍历搜索缓存)算法

输入：单位服务请求到达速率 $\Delta \lambda_e$ ，系统参数 c, μ_e, μ_c, t_f, K 。

输出：不同 λ_e 下各边缘服务器的最优服务请求到达速率以及边缘云处理时延。

- 1 初始化边缘云上服务请求到达速率 $\lambda_e = \Delta \lambda_e$ 。
- 2 定义变量 $index, \min$ 。
- 3 根据式(4.16)计算 t_c 。
- 4 **while** $\min < t_c$
- 5 设置所有 $\lambda_j^e = 0, j \in K$ ，总服务请求到达速率与单位服务请求到达速率的比值 $a = (\lambda_e / \Delta \lambda_e)$ ；
- 6 **for** $i = 1$ 到 a
- 7 $index = 1$, \min 为 $\lambda_{index}^e = E[a][index] + \Delta \lambda_e$ 时的 t_e 的值；
- 8 **for** $j = 2$ 到 k
- 9 计算 $\lambda_j^e = \lambda_j^e + \Delta \lambda_e$ 时的 t_e ；
- 10 **if** $t_e < \min$
- 11 $\min = t_e, index = j$ ；
- 12 **end if**
- 13 **end for**
- 14 $E[a][index] = E[a][index] + \Delta \lambda_e$ ；

```

15      end for
16      根据式(4.14)计算  $T[a]$ ;
17       $\lambda_e = \lambda_e + \Delta\lambda_e$ ;
18  end

```

由定理 4.1 可知当移动设备确定其卸载请求产生的速率 $p_i^e \lambda_i$ 后, 卸载到边缘云的请求到达速率必然存在一个最优值 λ_e^* 使得完成任务的代价最小, 本节采取二分查找的方法寻找到近似最优的 λ_e , 再根据通过算法 4.1 确定的云计算控制器上缓存的数据, 以 $O(1)$ 的时间复杂度获得 k 个边缘云的请求到达速率。具体计算过程如算法 4.2 所述。

算法 4.2 BLI(Binary Lookup Iteration, 二分查找迭代)算法

输入: 远程服务请求到达率 λ_{total} , 单位服务请求到达速率 $\Delta\lambda_e$, 计算精度 ε , 迭代步长 σ , 系统参数 c, μ_e, μ_c, t_f, k 。

输出: 多个边缘服务器的最优服务到达率总和 λ_e 。

```

1  在  $[0, \text{Math.min}(\lambda_{total}, kc\mu_e - \varepsilon)]$  范围中随机取初值  $\lambda_e$ 。
2  初始化  $l = 0, r = \text{Math.min}(\lambda_{total}, Kc\mu_e - \varepsilon)$ 。
3   $t_e = T[\lambda_e / \Delta\lambda_e]$ 。
4  计算  $y(\lambda_e) = (1 - \phi_c) \cdot t_e + \phi_c \cdot t_c$ 。
5  while  $|l - r| \geq \varepsilon$ 
6      if  $y(\lambda_e + \sigma) > y(\lambda_e)$ 
7           $r = \lambda_e$ ;
8      else
9           $l = \lambda_e + \sigma$ ;
10     在  $[l, r]$  范围中随机选取  $\lambda_e$  的值;
11 end

```

根据算法 4.2, 我们可以在确立移动设备卸载速率的前提下直接确立相应的多个边缘云的到达速率以及中心云的到达速率, 将 P4.1 转换为一个单变量优化问题, 进一步简化问题, 降低求解难度。

4.3.2 移动设备卸载速率的优化

针对一般性多约束问题, 采用蚁群算法能够快速求得全局最优解。蚁群算法是一种仿生算法, 它吸收了蚂蚁觅食的行为特征, 其基本原理是: 蚂蚁在觅食时

会在走过的路径上留下自己分泌的信息素，蚁群之间利用这种信息素来传递信息，蚂蚁在选择路径时会受到信息素浓度的影响，即某条路径上的信息素浓度越高，蚂蚁选择该路径的概率就越大，从而进一步增强该路径上的信息素浓度。蚁群利用信息素的正反馈机制建立起一套增强型学习系统，经过多轮强化能够找到觅食的最短路径^[82]。依据蚁群算法的思想，对建立求解 P4.1 的数学模型的过程进行说明。

不失一般性，设整个蚁群中蚂蚁的数量为 m ，在每次迭代中，蚂蚁随机选择一组移动设备的任务卸载速率，然后利用 BLI 算法计算出边缘云和中心云的服务请求到达速率，从而根据式(4.21)进一步计算出完成任务的平均代价作为蚁群算法的代价函数。因此，每个蚂蚁的路径的信息素的量为

$$\Delta\tau_i = \frac{A}{COST_i}. \quad (4.33)$$

其中 A 是正的常量值。第 i 个蚂蚁路径的成本 $COST_i$ 根据式(4.21)通过计算完成任务的平均能耗和平均时延来计算。

蚂蚁按照一定的概率进行状态转移，设 $p_i(t)$ 为 t 时刻第 i 只蚂蚁转移的概率，由于蚁群是通过随机搜索来优化移动设备卸载速率的取值，因此状态转移概率仅受信息素浓度影响，表示为

$$p_i(t) = \frac{\tau^*(t) - \tau_i(t)}{\tau^*(t)}. \quad (4.34)$$

其中 $\tau^*(t)$ 为蚁群中 t 时刻最高的信息素浓度，状态转移概率的大小体现了 t 时刻每只蚂蚁的状态与当前时刻的最优状态之间的差距，如果状态转移概率小于状态转移概率阈值 p_0 ，表明该蚂蚁 t 时刻的状态处于最优解的附近，则 $t+1$ 时刻的状态只需要在 t 时刻的状态附近进行局部搜索即可；如果状态转移概率大于状态转移概率阈值，表明该蚂蚁 t 时刻的状态距离最优解较远，则 $t+1$ 时刻的状态需要在搜索范围内随机分配，在进行随机分配时需要注意对越界值的处理。 $t+1$ 时刻的状态之所以不按照状态转移概率的值进行逼近，是因为 t 时刻的最优解并不一定是全局最优解，随机分配可以发现相对于 t 时刻的最优解更优的状态，从而使搜索不易陷入局部最优解。

在每一次状态转移中, 蚂蚁都会根据这些信息素浓度来更新他们下一次状态转移的信息素浓度, 如下所示:

$$\tau_i(t+1) = (1-\rho) \cdot \tau_i(t) + \Delta\tau_i(t). \quad (4.35)$$

其中 ρ 是信息素的蒸发系数。此参数可防止算法偏向一条路径, 并有助于算法利用更多路径。

通过以上分析, 可以看出蚁群算法具有一定的随机性, 主要体现在初始时刻蚁群状态的选择上。但是通过参数的设定, 可以使求解的过程充分利用到信息素的有效信息, 尽快收敛到最优值; 同时使蚁群的状态空间尽量覆盖到全局, 避免陷入局部最优解^[83]。其求解过程如算法 4.3 所示。

算法 4.3: ACMODM(Ant Colony Multi-objective Offloading Decision Making, 蚁群多目标卸载决策)算法

输入: 蚂蚁数量 m , 转移概率阈值 p_0 , 蒸发系数 ρ , 系数 A , 蚂蚁搜索次数 k , 系统参数 $c, \mu_e, \mu_c, t_f, k, \alpha, \beta$ 。

输出: 完成服务请求的平均代价 $Cost$ 。

- 1 调用算法 4.1 缓存不同 λ_e 下各个边缘服务器的最优服务到达率。
- 2 初始化迭代次数 $t=0$, 初始化移动设备卸载请求的服务到达率 $\lambda_{total}(0)$ 。
- 3 **while** $t < k$
- 4 **for** $i=1$ 到 m
- 5 调用算法 4.2 确立 λ_e, λ_c ;
- 6 根据算法 4.1 的结果确立各个边缘服务器的最优服务到达率;
- 7 根据式(4.33)计算蚂蚁状态的信息素浓度 τ_i ;
- 8 根据式(4.34)计算蚂蚁状态的转移概率 $p_i(t)$;
- 9 **if** $p_i(t) < p_0$
- 10 各移动设备的卸载请求到达率进行局部搜索, 确定 $\lambda_{total}(i)$;
- 11 **else**
- 12 各移动设备的卸载请求到达率进行全局搜索 $\lambda_{total}(i)$;
- 13 **end if**
- 14 调用 5~7 行, 计算 $\Delta\tau_i$;
- 15 **if** $\tau_i > \Delta\tau_i$
- 16 $\lambda_{total}(i) = \lambda_{total}(i-1)$;
- 17 **end if**

18 **end for**

19 **end**

20 根据式(4.21)计算完成服务请求的平均代价 $Cost$ 。

4.3.3 时间复杂度分析

对于 ACMODM 算法其主体包括两层循环（3-19 行），第一层循环是迭代次数 k （第 3 行），第二层循环是蚂蚁个数 m （第 4 行），其中调用算法 4.2 的迭代次数为 $\log(\lambda_{total})$ （第 5 行，第 14 行），移动设备卸载请求到达率重新搜索的迭代次数为 n （9~13 行），综上所述 ACMODM 的时间复杂度为 $O(km(n + \log(\lambda_{total})))$ 。

4.4 ACMODM 的仿真设计与分析

4.4.1 ACMODM 的仿真实验设计

本章主要考虑的是多用户设备与多边缘服务器的场景，给出大量仿真结果检验所提出的 ACMODM 算法的性能。该仿真是采用蒙特卡洛方法随机进行实现的。除非有额外说明，否则仿真参数如表 4.1 所示，其中参数设置参考文献[64, 77]中的数据。

表 4.1 仿真参数设置

参数	取值
λ_i	1~1.5 request/s
μ_i^l	1.6 request/s
μ_i^r	2 request/s
$e_i^{l,max}$	4~6 J
$e_i^{r,max}$	1~2 J
c	4
μ_e	2.5~3.5 request/s
μ_c	10 request/s
t_f	1 s

仿真程序在 Java 平台下编写，通过选取系统参数生成不同场景以及不同参数下的完成服务请求的平均代价、边缘云负载率等数据，然后将得到的数据用于 Matlab 平台下的绘图。

4.4.2 ACMODM 的仿真结果分析

不失一般性, 本节实验中首先设置延迟权重 α , 能耗权重 β 为 $\{1/2, 1/2\}$ 。考虑的是有 120 个移动设备, 以及 5 个边缘服务器的多用户设备多边缘服务器的场景, 其中每个边缘服务器有 4 个工作台在工作。

图 4.4 展示了完成服务请求的平均代价, 展示了移动设备完成服务请求的平均代价随着迭代次数的增加逐渐降低并最终达成稳态。相比于初始状态, 经过迭代后完成任务的平均代价降低了 28.61%, 效果明显。

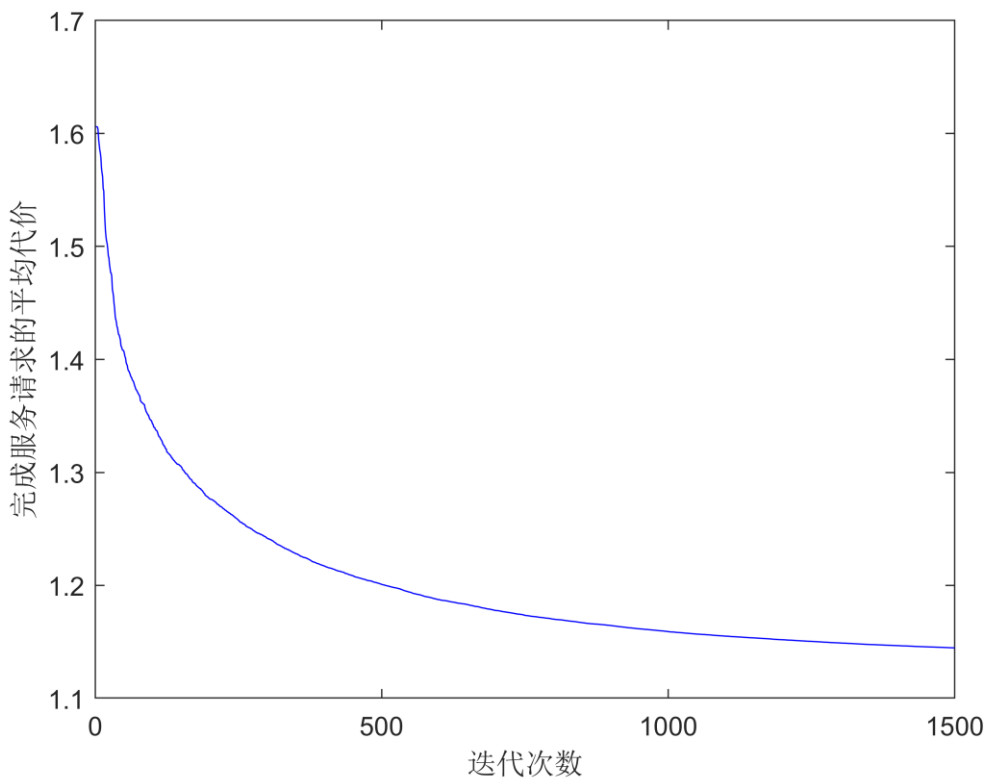


图 4.4 迭代次数对完成服务请求的平均代价的影响

算法运行时间以及算法效果是衡量算法最重要的两个指标, ACMODM 算法的平均运行时间和完成请求的平均代价下降的百分比相对于不同迭代次数的结果如表 4.2 所示。平均运行时间是通过在 Intel Core i5-7200 2.5GHz 处理器和 8GB RAM 的计算机上运行 ACMODM 算法获得的。可以看出随着迭代次数的增加, 算法平均运行时间呈线性增加, 完成请求的平均代价下降的百分比的增长速率逐渐减小趋于 0, 因为随着迭代次数的增加, 算法愈发逼近最优解, 优化的空间逐

渐减少。综上所述,可以综合考虑运行时间以及算法效果,选择迭代次数在 500~1500 之间。

表 4.2 迭代次数对算法效果的影响

迭代次数	50	100	200	500	1000	1500	2000
平均运行时间(s)	0.049	0.082	0.168	0.375	0.623	0.776	0.989
平均代价下降率	12.23%	16.14%	20.36%	25.10%	27.71%	28.61%	28.77%

表 4.3 展示了 5 个不同计算能力的边缘服务器的负载状态。此时五个边缘服务器的处理能力分别为 2.5 request/s, 2.75 request/s, 3.0 request/s, 3.25 request/s, 3.5 request/s。

表 4.3 μ_e 对边缘云负载的影响

μ_e (request/s)	2.5	2.75	3	3.25	3.5
边缘云处理请求速率(request/s)	7.89	8.97	10.04	11.09	12.13
边缘云负载率	78.90%	81.54%	83.66%	85.30%	86.64%

易知边缘云的处理能力越强,边缘云的负载率相应的会越高,但是负载率的增长速率会逐渐降低,因为负载率越高的相应的排队时间会越长,导致负载率的增加不能随着处理能力的增加而线性增加。

表 4.4 展示了中心云不同处理能力下,边缘云的平均负载以及移动设备卸载概率的变化。易知,随着中心云处理能力的增加,边缘云的平均负载率随之降低,移动设备卸载概率随之提升。主要原因是,卸载到中心云的代价随着中心云能力的增加会降低,用户更倾向于在任务较多时卸载到中心云去处理。但边缘云的负载率下降速率逐渐降低,这是因为卸载到中心云的任务代价有固定的通信时延,随着中心云的计算能力提高,通信时延对完成任务的代价的影响会逐渐提高。

表 4.4 μ_c 对边缘云平均负载以及卸载概率的影响

μ_c (request/s)	5	10	15	20	25
边缘云平均负载率	85.03%	83.53%	83.00%	82.68%	82.50%
p_c	52.31%	55.75%	56.94%	57.82%	58.34%

为了更直观的评估所提出的 ACMODM 算法,我们比较了以下基准算法:

本地计算策略(Local Computing, LC): 在 LC 策略中,服务请求均由移动设备本地完成。

边缘云负载均衡策略(Edge Load Balance, ELB): 在 ELB 策略中, 服务请求卸载到边缘云上去处理时, 选择边缘云的依据将依靠边缘云的负载率, 使得边缘云达到负载均衡。

用户随机卸载策略(User Random Offloading, URO): 在 URO 策略中, 移动设备随机卸载服务请求, 卸载的服务请求将根据所提出的算法决定卸载到边缘云或者中心云。

完全随机卸载策略(Completely Random Offloading, CRO): 在 URO 策略中, 移动设备随机卸载服务请求, 卸载的服务请求随机决定卸载到边缘云或中心云。

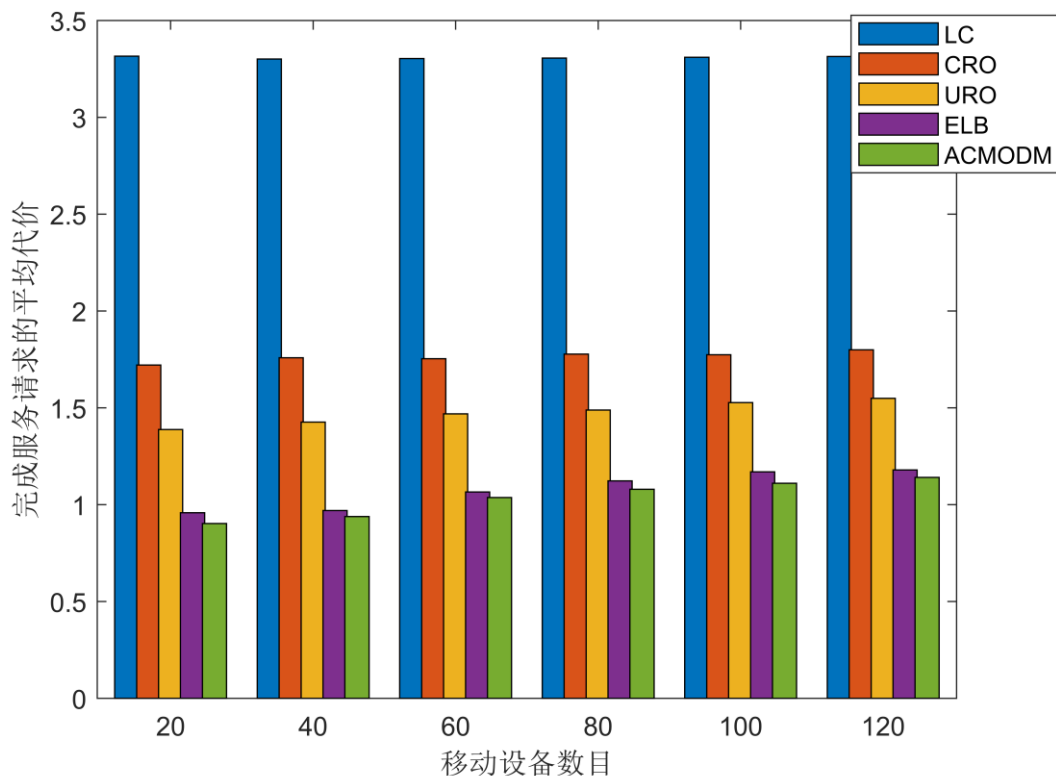


图 4.5 不同策略下完成服务请求的平均代价随移动设备数目变化的对比

由图 4.5 可知, ACMODM 策略在不同移动设备数目下完成服务请求的平均代价总是最低的。LC 策略对服务请求不进行计算卸载, 所以随着移动设备数目的增加不会对完成服务请求的平均代价有任何改变; CRO 策略不考虑边缘云的拥塞状态, 也不考虑移动设备的计算能力以及正在处理的服务请求, 完全随机, 相比于提出的 ACMODM 策略缺少了蚂蚁寻优的移动设备卸载决策的优化以及

BLI 算法的边缘云与中心云服务请求的分配,完成服务请求的平均代价一直第二高。除此之外,CRO 策略随着用户数目的增加完成服务请求的平均代价略微有提升,主要原因是随着用户的增加随机卸载到边缘云的请求更多,边缘云的负载会更高,边缘云的平均执行时间提高了;URO 策略相比于提出的 ACMODM 策略缺少了蚂蚁寻优的移动设备卸载决策的优化,但相比与 CRO 策略增加了 BLI 算法的边缘云与中心云请求的分配,URO 策略会根据边缘云的负载状况决定是否选择将请求卸载到中心云去处理,所以完成任务的平均代价会比 CRO 更低;ELB 策略相比于提出的 ACMODM 策略少了 ISC 算法中对根据边缘服务器计算能力以及负载状况不同进行服务请求分配的部分,采取的是均衡分配,边缘云服务器计算能力相差不大,均衡分配也考虑到了边缘服务器的负载状况,所以 ELB 策略也能取得较好的结果,但完成服务请求的平均代价还是高于 ACMODM 策略。ACMODM 策略相比与 LC, CRO, URO, ELB 策略,完成任务的平均代价分别降低了 65.55%~72.80%, 36.31%~47.67%, 26.29%~34.78%, 3.51%~5.56%。由于 ACMODM 策略已经充分利用了所有计算资源,其它策略并没有完全利用所以计算资源,随着移动设备数目的增加,服务请求的数目更多,其它策略会利用更多的计算资源,因此完成服务请求的平均代价下降的百分比随着移动设备数目的增加在减少,但 ACMODM 策略下完成服务请求的平均代价一直是最低的。

在移动边缘计算系统中,边缘服务器数目是否充足对完成服务请求的代价的影响也值得关注,可以根据服务请求的数目来部署合适的边缘服务器数量。由图 4.6 可知,不同边缘服务器数目下,所提出的 ACMODM 策略下完成服务请求的平均代价总是最低的。ACMODM 策略完成服务请求的平均代价相比与 LC, CRO, URO, ELB 策略分别降低了 64.35%~70.05%, 34.07%~40.83%, 27.67%~33.33%, 2.53%~3.56%。随着边缘云服务器数目的增加,ACMODM 策略可以充分的利用所增加的边缘服务器的计算能力,使得其相比与其它基准算法随着边缘服务器数目的增加平均代价下降的百分比更高。另外值得注意的是,ACMODM 下完成服务请求的平均代价与 ELB 策略下完成的服务请求的平均代价近似时,ELB 所需要的边缘云服务器数目将比 ACMODM 策略下所需要的边缘云服务器数目增加两台。

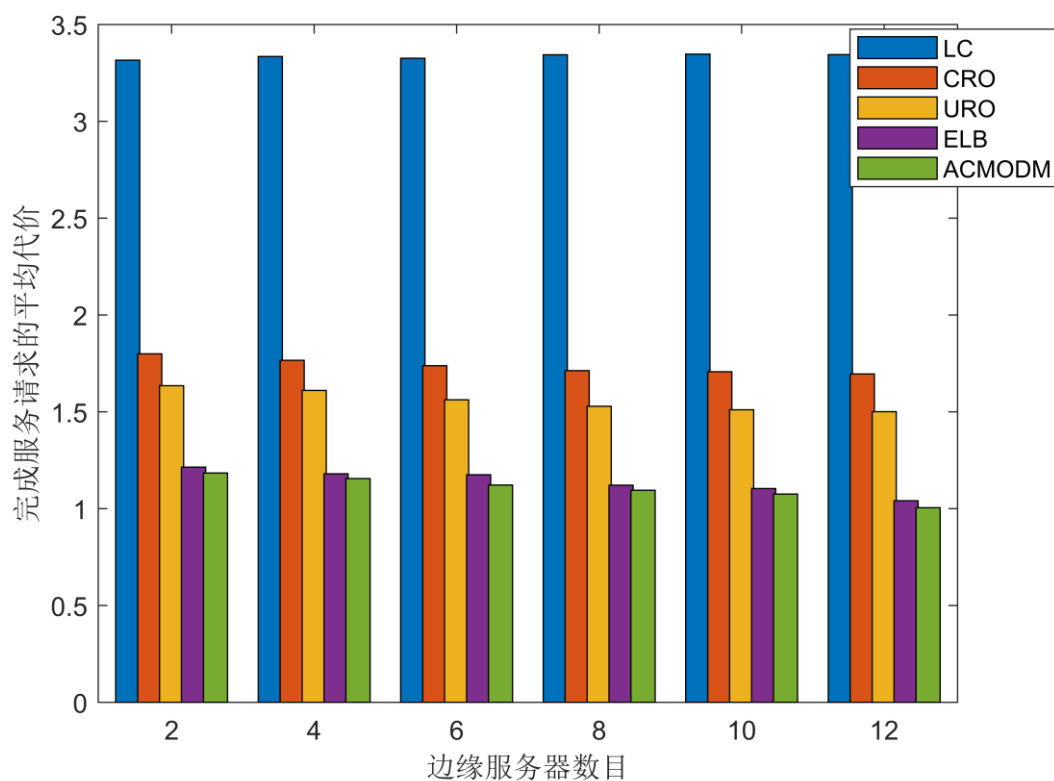


图 4.6 不同策略下完成服务请求的平均代价随边缘服务器数目变化的对比

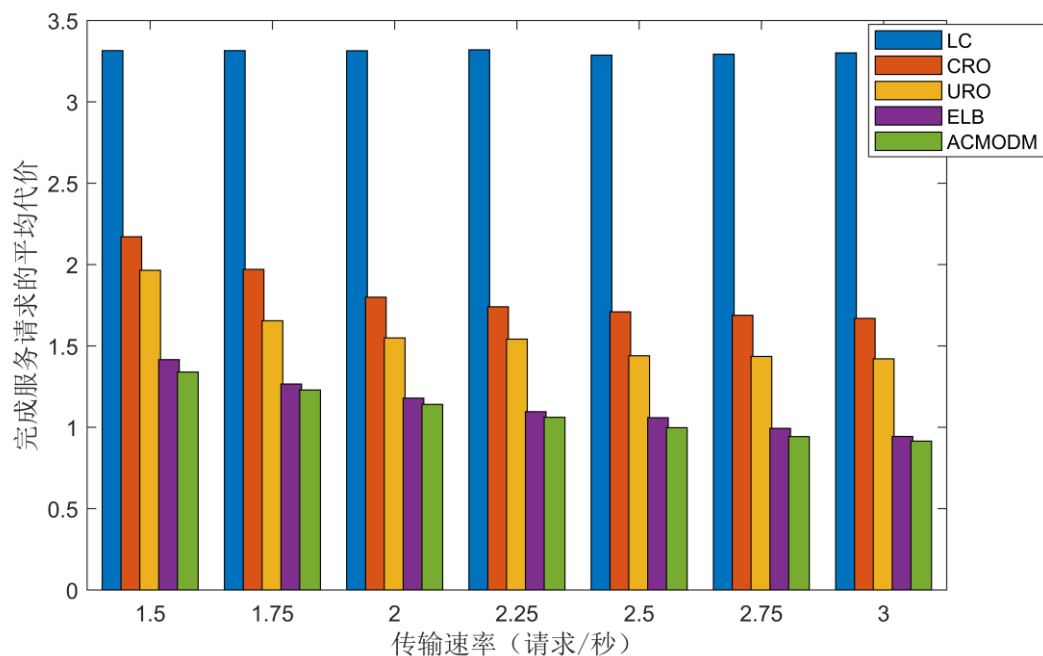


图 4.7 不同策略下完成服务请求的平均代价随传输速率变化的对比

信道资源对不同策略结果的影响也值得关注。传输速率与信道资源是成正比的，因此在图 4.7 中根据不同策略下完成服务请求的平均代价随传输速率变化的

对比来评估信道资源的影响。

由图 4.7 可知,随着传输速率的增加完成服务请求的平均代价逐渐降低,但是降低的速率逐渐减少,主要原因是由于边缘云的计算资源以及到中心云的通信时延是固定的,随着传输速率的增加,它们成为降低能耗的制约因素。CRO 策略与 URO 策略对于是否需要卸载到云上均是采用随机卸载策略,因此当信道资源不足即传输速率较低时,完成服务请求的代价偏高,但随着信道资源开始增加,完成服务请求的代价显著下降,最终在信道资源充足之后趋于稳定。LC 策略由移动设备本地完成不受信道资源的影响,完成服务请求的平均代价保持稳定。ACMODM 策略完成服务请求的平均代价相比与 LC, CRO, URO, ELB 策略分别降低了 59.81%~72.42%, 38.07%~45.53%, 32.14%~35.91%, 2.53%~6.15%。

4.5 本章小结

本章针对边云协同中多边缘服务器的计算能力未得到充分利用问题,提出了一种基于蚁群算法的多边缘卸载决策优化策略。依据排队论思想并以能耗和时延性能最小化为目标,对移动设备的服务请求卸载速率、边缘云和中心云服务请求到达速率进行了协同建模与优化。采用二分查找迭代方法对边缘云集群和中心云的服务请求到达速率进行决策,并利用蚁群算法实现了移动设备卸载速率的寻优和服务请求的合理分配。最后进行仿真实验,将仿真结果与其它基准策略进行比较。在不同移动设备数目(20-120)下,ACMODM 策略下完成服务请求的平均代价相比与 LC, CRO, URO, ELB 策略分别降低了 65.55%~72.80%, 36.31%~47.67%, 26.29%~34.78%, 3.51%~5.56%;在不同边缘服务器数目(2-12)下,ACMODM 策略完成服务请求的平均代价相比与 LC, CRO, URO, ELB 策略分别降低了 64.35%~70.05%, 34.07%~40.83%, 27.67%~33.33%, 2.53%~3.56%;在不同传输速率(1.5-3 请求/秒)下,ACMODM 策略完成服务请求的平均代价相比与 LC, CRO, URO, ELB 策略分别降低了 59.81%~72.42%, 38.07%~45.53%, 32.14%~35.91%, 2.53%~6.15%。

第五章 总结与展望

5.1 全文总结

本文首先介绍了移动边缘计算的研究背景及其研究意义,对移动边缘计算与计算卸载技术及其国内外研究现状进行了综述,分析了计算卸载决策对充分发挥移动边缘计算优势的重要性,并且简要介绍了本文研究的重点及难点。

针对移动设备忽视信道与边缘云计算资源而导致卸载性能低下乃至任务失败问题,提出了一种有限资源下的能源高效型计算卸载决策方法。综合考虑了移动设备的计算频率,传输功率,卸载数据,边缘云的计算能力以及信道资源等变量,改进了卸载任务的能源模型。分别建立了最佳卸载数据量和本地设备最优计算频率的模型、最佳卸载数据量和移动设备发射功率的模型以及信道资源与最佳卸载数据量的模型,将问题简化为单变量优化问题,并基于内罚函数法进行求解,提高了卸载策略的实际可行性以及资源配置的效率。仿真结果表明:在不同移动设备数目下(10-100),所提出的 MECOS 算法相比与 LCO 策略移动设备的能耗降低了 38.82%~94.95%。在移动设备数量相对较少时(10-30)时,MECOS 算法相比与 REA 策略移动设备的能耗降低了 71.87%~78.20%,相比与 TPA 策略移动设备的能耗降低了 31.81%~41.27%。在移动设备数量相对较多(40-100)时,MECOS 策略相比与 REA 策略移动设备的能耗降低 36.14%~63.93%,相比与 TPA 策略移动设备的能耗降低了 9.83%~24.04%。

针对现有定价策略平均分配边缘云计算资源而忽略用户需求差异问题,提出了一种边缘云卸载决策和 Stackelberg 博弈定价有机融合的一种新方法。分别建立了边缘云定价和购买最优计算资源块的决策模型以及最佳卸载数据量和购买计算资源块数量之间的关系模型,实现了用户的最优卸载决策,确立了边缘云定价的上下界。最后,提出了一种基于动态规划的边缘云定价卸载算法,实现了边缘云效用最优定价和各用户自身效用的最大化。仿真结果表明:在用户数量较少的情况下,与 LCO 和 ORO 卸载策略相比,所提出的 DPPO 策略分别降低了 18.26% 和 6.28%的任务完成成本,与 UP、AP 和 GP 定价策略相比,DPPO 策略下边缘云获得的收益分别提高了 36.12%、13.05%和 5.98%。在用户数量较大的情况下,

DPPO 策略比 LCO 和 ORO 卸载策略分别降低了 4.9%和 1.97%的任务完成成本,与 UP、AP 和 GP 定价策略相比,DPPO 策略下边缘云的收益分别提高了 22.53%、36.08%和 25.29%。

针对边云协同中多边缘服务器的计算能力未得到充分利用问题,提出了一种基于蚁群算法的多边缘卸载决策优化策略。依据排队论思想并以能耗和时延性能最小化为目标,对移动设备的服务请求卸载速率、边缘云和中心云服务请求到达速率进行了协同建模与优化。采用二分查找迭代方法对边缘云集群和中心云的服务请求到达速率进行决策,并利用蚁群算法实现了移动设备卸载速率的寻优和服务请求的合理分配。仿真结果表明:在不同移动设备数目(20-120)下,ACMODM 策略下完成服务请求的平均代价相比与 LC, CRO, URO, ELB 策略分别降低了 65.55%~72.80%, 36.31%~47.67%, 26.29%~34.78%, 3.51%~5.56%。

5.2 未来的展望

本文对 MEC 系统中的多用户、多边缘服务器以及云边协同的计算卸载决策做了一定的研究,包括本地计算频率、传输功率控制、信道资源分配、任务划分、边缘云计算资源块定价等内容,但由于作者的能力水平有限,本文仍存在一些可以改进之处,值得进一步深入地研究:

1. 在多用户单边缘服务器场景的计算卸载决策方面,本文提出的 MECOS 算法与 DPPO 算法均是考虑设备位置相对固定后的准静态情况。在未来的研究中还需关注用户设备位置移动的一般情况,这涉及到应对小区切换的设备移动性管理,云边协同的服务迁移方案以及更深层次的内容缓存策略的制定,尤其是如何以较低的系统开销保证设备移动时服务的连续性,值得深入的探索和研究。

2. 在云边协同的计算卸载决策方面,本文主要以能耗、延迟为目标,研究了任务划分和边缘云集群的资源分配问题。实际中,一些服务的子任务粒度大,或者子任务之间存在关联关系,这种情况制约着任务划分和执行顺序的决策。因此,首先需要建立标准模型对这种情况进行描述,在此基础上进行资源联合分配,这将是具有挑战性的工作。

3. 在研究通信时延方面,本文主要以信道带宽以及传输功率的调整来决定

通信时延，到中心云的通信时延采取了常见的固定时延。实际中，决定通信时延的还有各个边缘服务器到移动设备之间的距离，到中心云的通信时延需要考虑信道的拥塞程度。这将是一个动态变化的过程，需要我们深入的探索和研究。

参考文献

- [1] Atzori L, Iera A, Morabito G. The internet of things: A survey[J]. Computer networks, 2010, 54(15): 2787-2805.
- [2] Chen D, Zhang N, Qin Z, et al. S2M: A lightweight acoustic fingerprints-based wireless device authentication protocol[J]. IEEE Internet of Things Journal, 2016, 4(1): 88-100.
- [3] Sun H, Zhou F, Hu R Q. Joint offloading and computation energy efficiency maximization in a mobile edge computing system[J]. IEEE Transactions on Vehicular Technology, 2019, 68(3): 3052-3056.
- [4] Li A, Yang X, Kandula S, et al. CloudCmp: comparing public cloud providers[C]//Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. 2010: 1-14.
- [5] Andrews J G, Buzzi S, Choi W, et al. What Will 5G Be?[J]. IEEE Journal on Selected Areas in Communications, 2014, 32(5): 1065-1082.
- [6] Satyanarayanan M. The Emergence of Edge Computing[J]. Computer, 2017, 50(1): 30-39.
- [7] Shi W, Cao J, Zhang Q, et al. Edge Computing: Vision and Challenges[J]. Internet of Things Journal, IEEE, 2016, 3(5): 637-646.
- [8] Yang G, Hou L, He X, et al. Offloading time optimization via Markov decision process in mobile-edge computing[J]. IEEE Internet of Things Journal, 2020, 8(4): 2483-2493.
- [9] Li S, Zhang N, Lin S, et al. Joint admission control and resource allocation in edge computing for internet of things[J]. IEEE Network, 2018, 32(1): 72-79.
- [10] ETSI Mobile Edge Computing publishes foundation specifications[EB/OL]. <http://www.etsi.org/index.php/news-events/news/1078-2016-04-etsi-mobile-edge-computing-publishesfoundation-specifications>.
- [11] Senyo P K, Addae E, Boateng R. Cloud computing research: A review of research themes, frameworks, methods and future research directions[J]. International Journal of Information Management, 2018, 38(1): 128-139.
- [12] Hanan E. Internet of Things (IoT), mobile cloud, cloudlet, mobile IoT, IoT cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions - ScienceDirect[J]. Journal of Network and Computer Applications, 2019, 128: 105-140.
- [13] Delp E J. The future of mobile computing[C]. Systems, Signals and Image Processing (IWSSIP), 2013 20th International Conference on. IEEE, 2013: 121-121.
- [14] Albert, Greenberg, James, et al. The cost of a cloud: Research problems in data center networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 39(1): 68-73.
- [15] Cuervo E, Balasubramanian A, Cho D K, et al. MAUI: Making smartphones last longer with code offload[C]. International Conference on Mobile Systems. DBLP, 2010: 49-62.
- [16] Satyanarayanan M, Bahl P, Cáceres R, et al. The Case for VM-Based Cloudlets in Mobile Computing[J]. Pervasive Computing IEEE, 2009, 8(4): 14-23.
- [17] Flavio B, Rodolfo M, Jiang Z, et al. Fog Computing and its Role in the Internet of Things[C]. Proceedings of the MCC workshop on Mobile Cloud Computing. ACM, 2012: 13-16.
- [18] Connected vehicles, the internet of things, and fog computing[EB/OL]. <https://www.sigmobile.org/mobicom/2011/vanet2011/program.html>.
- [19] IBM and Nokia Siemens Networks Announce World's First Mobile Edge Computing Platform[EB/OL]. <https://www-03.ibm.com/press/us/en/pressrelease/40490.wss>.
- [20] Khan A U R, Othman M, Madani S A, et al. A Survey of Mobile Cloud Computing Application

- Models[J]. IEEE Communications Surveys & Tutorials, 2014, 16(1): 393-413.
- [21] Chang C Y, Alexandris K, Nikaein N, et al. MEC architectural implications for LTE/LTE-A networks[C]. Workshop on Mobility in the Evolving Internet Architecture. ACM, 2016: 13-18.
- [22] Chen J, Zheng X, Rong C. Survey on Software-Defined Networking[C]. Springer, 2015: 115-124.
- [23] Mijumbi R, Serrat J, Gorricho J L, et al. Network Function Virtualization: State-of-the-Art and Research Challenges[J]. IEEE Communications Surveys & Tutorials, 2017, 18(1): 236-262.
- [24] Armbrust M. Above the clouds: A berkeley view of cloud computing[J]. Science, 2009, 53: 07-013.
- [25] Bhattacharya A, De P. A Survey of Adaptation Techniques in Computation Offloading[J]. Journal of Network & Computer Applications, 2016, 78(JAN.): 97-115.
- [26] Liu Q, Huang S, Opadere J, et al. An Edge Network Orchestrator for Mobile Augmented Reality[C]. IEEE INFOCOM 2018 - IEEE Conference on Computer Communications. IEEE, 2018: 756-764.
- [27] Wang L, Jiao L, He T, et al. Service Entity Placement for Social Virtual Reality Applications in Edge Computing[C]. IEEE INFOCOM 2018 - IEEE Conference on Computer Communications. IEEE, 2018: 468-476.
- [28] Shanhe Y, Zijiang H, Qingyang Z, et al. LAVEA: Latency-Aware Video Analytics on Edge Computing Platform[C]. 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2017: 2573-2574.
- [29] Zhang Q, Zhang Q, Shi W, et al. Firework: Data Processing and Sharing for Hybrid Cloud-Edge Analytics[J]. IEEE Transactions on Parallel and Distributed Systems, 2018, 29(9): 2004-2017.
- [30] Cicirelli F, Guerrieri A, Spezzano G, et al. An edge-based platform for dynamic Smart City applications[J]. Future Generation Computer Systems, 2017, 76(NOV.): 106-118.
- [31] 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). IEEE, 2016: 1451-1455.
- [32] 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.
- [33] You C, Huang K, Chae H, et al. Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading[J]. IEEE Transactions on Wireless Communications, 2017, 16(3): 1397-1411.
- [34] Guo F, Ma L, Zhang H, et al. Joint load management and resource allocation in the energy harvesting powered small cell networks with mobile edge computing[C]. IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2018: 299-304.
- [35] Chen X, Jiao L, Li W, et al. Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing[J]. IEEE/ACM Transactions on Networking, 2016, 24(5): 2795-2808.
- [36] Tang L, Chen X. An Efficient Social-Aware Computation Offloading Algorithm in Cloudlet System[C]. 2016 IEEE Global Communications Conference (GLOBECOM). IEEE, 2016: 1-6.
- [37] Chen, Xu. Decentralized Computation Offloading Game for Mobile Cloud Computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2015, 26(4): 974-983.
- [38] Zheng J, Cai Y, Wu Y, et al. Dynamic Computation Offloading for Mobile Cloud Computing: A Stochastic Game-Theoretic Approach[J]. IEEE Transactions on Mobile Computing, 2019, 18(4): 771-786.
- [39] Zhang J, Hu X, Ning Z, et al. Energy-Latency Tradeoff for Energy-Aware Offloading in Mobile

- Edge Computing Networks[J]. IEEE Internet of Things Journal, 2018, 5(4): 2633-2645.
- [40] Wang Y, Sheng M, Wang X, et al. Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling[J]. IEEE Transactions on Communications, 2016, 64(10): 4268-4282.
- [41] Klas G. Edge Computing and the Role of Cellular Networks[J]. Computer, 2017, 50(10): 40-49.
- [42] Flores H, Hui P, Tarkoma S, et al. Mobile code offloading: from concept to practice and beyond[J]. IEEE Communications Magazine, 2015, 53(3): 80-88.
- [43] Joseph A D, Delespinasse A F, Tauber J A, et al. Rover: A Toolkit for Mobile Information Access[J]. ACM SIGOPS Operating Systems Review, 1995, 29(5): 156-171.
- [44] Kosta S, Aucinas A, Hui P, et al. ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading[C]. Proceedings IEEE Infocom. IEEE, 2012: 945-953.
- [45] 詹文翰. 移动边缘网络计算卸载调度与资源管理策略优化研究[D]. 电子科技大学, 2020.
- [46] 深度 | 边缘计算方兴未艾, 全产业链协同发展 [EB/OL]. <https://mp.weixin.qq.com/s/QM3vKB2oWuSt7mJK66LNpA>.
- [47] 盘点|2019十大边缘计算项目[EB/OL]. <https://xueqiu.com/4719507903/138666597>.
- [48] Liu C F, Bennis M, Debbah M, et al. Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing[J]. IEEE Transactions on Communications, 2019, 67(6): 4132-4150.
- [49] Ding Z, Ng D W K, Schober R, et al. Delay minimization for NOMA-MEC offloading[J]. IEEE Signal Processing Letters, 2018, 25(12): 1875-1879.
- [50] Li C, Xia J, Liu F, et al. Dynamic offloading for multiuser multi-CAP MEC networks: A deep reinforcement learning approach[J]. IEEE Transactions on Vehicular Technology, 2021, 70(3): 2922-2927.
- [51] Elgendy I A, Zhang W Z, He H, et al. Joint computation offloading and task caching for multi-user and multi-task MEC systems: reinforcement learning-based algorithms[J]. Wireless Networks, 2021, 27(3): 2023-2038.
- [52] Liu Z, Fu J, Zhang Y. Computation offloading and pricing in mobile edge computing based on Stackelberg game[J]. Wireless Networks, 2021, 27(7): 4795-4806.
- [53] Moura J, Hutchison D. Game theory for multi-access edge computing: Survey, use cases, and future trends[J]. IEEE Communications Surveys & Tutorials, 2018, 21(1): 260-288.
- [54] Xiong Z, Feng S, Wang W, et al. Cloud/fog computing resource management and pricing for blockchain networks[J]. IEEE Internet of Things Journal, 2018, 6(3): 4585-4600.
- [55] Hazra A, Adhikari M, Amgoth T, et al. Stackelberg game for service deployment of IoT-enabled applications in 6G-aware fog networks[J]. IEEE Internet of Things Journal, 2020, 8(7): 5185-5193.
- [56] Liu Y, Xu C, Zhan Y, et al. Incentive mechanism for computation offloading using edge computing: A Stackelberg game approach[J]. Computer Networks, 2017, 129: 399-409.
- [57] Jie Y, Tang X, Choo K K R, et al. Online task scheduling for edge computing based on repeated Stackelberg game[J]. Journal of Parallel and Distributed Computing, 2018, 122: 159-172.
- [58] Li M, Wu Q, Zhu J, et al. A computing offloading game for mobile devices and edge cloud servers[J]. Wireless Communications and Mobile Computing, 2018, 2018.
- [59] Sun W, Zhang H, Wang L, et al. Profit maximization task offloading mechanism with d2d collaboration in mec networks[C]//2019 11th International Conference on Wireless Communications and Signal Processing (WCSP). IEEE, 2019: 1-6.
- [60] Wang Y, Lin X, Pedram M. A nested two stage game-based optimization framework in mobile cloud computing system[C]//2013 IEEE Seventh International Symposium on Service-Oriented

- System Engineering. IEEE, 2013: 494-502.
- [61] Li X, Zhang C, Gu B, et al. Optimal pricing and service selection in the mobile cloud architectures[J]. IEEE Access, 2019, 7: 43564-43572.
- [62] Chin T L, Chen Y S, Lyu K Y. Queuing model based edge placement for work offloading in mobile cloud networks[J]. IEEE Access, 2020, 8: 47295-47303.
- [63] Du J, Zhao L, Feng J, et al. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee[J]. IEEE Transactions on Communications, 2017, 66(4): 1594-1608.
- [64] Liu L, Chang Z, Guo X, et al. Multiobjective optimization for computation offloading in fog computing[J]. IEEE Internet of Things Journal, 2017, 5(1): 283-294.
- [65] Ren J, Yu G, He Y, et al. Collaborative cloud and edge computing for latency minimization[J]. IEEE Transactions on Vehicular Technology, 2019, 68(5): 5031-5044.
- [66] Guo M, Li L, Guan Q. Energy-efficient and delay-guaranteed workload allocation in IoT-edge-cloud computing systems[J]. IEEE Access, 2019, 7: 78685-78697.
- [67] 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.
- [68] Shannon C E. A mathematical theory of communication[J]. Bell System Technical Journal, 1949, 27(4): 379-423.
- [69] Liu Y, Yuen C, Hassan N U, et al. Electricity cost minimization for a microgrid with distributed energy resource under different information availability[J]. IEEE Transactions on Industrial Electronics, 2014, 62(4): 2571-2583.
- [70] Wang X, Yuen C, Chen X, et al. Cost-aware demand scheduling for delay tolerant applications[J]. Journal of Network and Computer Applications, 2015, 53: 173-182.
- [71] You C, Huang K, Chae H, et al. Energy-efficient resource allocation for mobile-edge computation offloading[J]. IEEE Transactions on Wireless Communications, 2016, 16(3): 1397-1411.
- [72] Chen X, Jiao L, Li W, et al. Efficient multi-user computation offloading for mobile-edge cloud computing[J]. IEEE/ACM transactions on networking, 2015, 24(5): 2795-2808.
- [73] Mao Y, Zhang J, Song S H, et al. Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems[J]. IEEE Transactions on Wireless Communications, 2017, 16(9): 5994-6009.
- [74] 薛建彬, 关向瑞, 王璐, 等. 基于 Stackelberg 博弈的资源动态定价策略[J]. 华中科技大学学报: 自然科学版, 2020, 48(4): 121-126.
- [75] MacKenzie A B, DaSilva L A. Game theory for wireless engineers[J]. Synthesis Lectures on Communications, 2006, 1(1): 1-86.
- [76] Boyd S, Boyd S P, Vandenberghe L. Convex optimization[M]. Cambridge university press, 2004.
- [77] Rashidi S, Sharifian S. A hybrid heuristic queue based algorithm for task assignment in mobile cloud[J]. Future Generation Computer Systems, 2017, 68: 331-345.
- [78] Jia M, Cao J, Liang W. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks[J]. IEEE Transactions on Cloud Computing, 2015, 5(4): 725-737.
- [79] Bolch G, Greiner S, De Meer H, et al. Queueing networks and Markov chains: modeling and performance evaluation with computer science applications[M]. John Wiley & Sons, 2006.
- [80] Tse D, Viswanath P. Fundamentals of wireless communication[M]. Cambridge university press, 2005.
- [81] Ngo B, Lee H. Analysis of a pre-emptive priority M/M/c model with two types of customers and

- restriction[J]. Electronics Letters, 1990, 26(15): 1190-1192.
- [82] 段海滨, 王道波, 朱家强, 等. 蚁群算法理论及应用研究的进展[J]. 控制与决策, 2004, 19(12): 1321-1326,1340.
- [83] 卓金武, 魏永生, 秦健. MATLAB 在数学建模中的应用[J]. 北京航空航天大学出版社, 北京, 2011.

作者在攻读硕士学位期间公开发表的论文及科研成果

- 【1】. Computation offloading and pricing in mobile edge computing based on Stackelberg game[J]. //Wireless Networks, volume 27, pages 4795–4806 (2021), doi:10.1007/s11276-021-02767-z(第一作者, JCR: Q2, IF: 2.602)
- 【2】. Resource pricing and offloading decisions in mobile edge computing based on the Stackelberg game[J] //The Journal of Supercomputing (2022) , doi:10.1007/s11227-021-04246-w(第一作者, JCR: Q2, IF: 2.474)
- 【3】. 移动边缘计算(MEC)系统仿真软件 V1.0, 授权号: 2021R11L1935047, 完成日: 2021 年 6 月(除导师外第一作者)

致 谢

光阴似箭，三年的研究生学习生活转眼间就迎来了尾声，而新生开学好像还是昨天的事，第一次踏入上海大学的场景犹历历在目。回忆起这三年，感慨不已，欣慰之余而又庆幸无比。值得欣慰的是，三年来我确切地感受到了自己的进步。三年的时间我从刚入学对科研的懵懂无知变成了如今与课题组成员一起取得了一定的科研成果，这段宝贵的经历使我学到了许多终身受用的东西。值得庆幸的是我周围有很多良师益友，他们在我学习的道路上给了我很多的指导、陪伴与鼓励，使我能够顺利地完成学业，在此谨向他们表示衷心的感谢！

首先感谢我的导师付敬奇教授三年以来对我的指导与栽培。我知道，千言万语并不能确切地表达我对老师的感激之情。付老师在科研中，付老师给我树立了一个很好的榜样，刚入学时，付老师就教导我，不要停留在本科生阶段，要将自己定位为一个科研工作者，把自己培养成研究型创新型人才，要学会独立思考问题，发现问题。付老师用严谨的治学态度和深厚的专业知识在科研的道路上指引我前行，让我学会发现问题、思考问题、解决问题，培养我的科研能力与抗压能力。除此之外，付老师更是我的人生导师，在闲暇时间付老师会向我们分享历史、时事、人文等各方知识，拓宽了我的视野，面对我的一些困惑，也会站在更高的角度引导我去解决，再次衷心感谢付老师三年以来对我的指导与栽培。

感谢李昂师姐、张越师兄，在我对课题的迷茫导致科研无法得到进展，是师兄师姐站在一个过来人的角度指导我如何更好的进行科研工作，解决问题。

感谢王益杰师弟以及沈子逸师弟，他们与我共同讨论课题，我们互相帮助互相成长，很多科研的灵感是在我们讨论中取得的。

感谢同门张添宽和何丹婷，三年来我们一同从懵懂无知成长为如今的样子，是你们在课题上给予帮助与建议，在生活上给予安慰和鼓励，祝你们在接下来的生活中工作顺利。

感谢研究生期间的室友江天赐、朱凌昀以及侯震宇，三年的朝夕相处，让我收获了在上海大学的美好回忆。

感谢所有帮助过我的老师、同学和朋友，感谢所有关心过和帮助过我的人，没有他们对我的关心，就没有我今天的任何成绩。

感谢上海大学，它给我的人生添上了精彩的一笔，给了我一个更新更高的起点，让我更加从容自信地面对未来的挑战！

感谢我的父母及家人在我读研期间在物质和精神上给予的帮助和支持。你们的支持给予了我不断前进，永不言弃的动力。

最后感谢审评本文的专家们，你们宝贵的意见是我不断进步的动力。