



本科毕业设计（论文）

**多源多中继边缘计算设备**

**能耗延迟权衡研究**

学 院 计算机学院

专 业 计算机科学与技术

（大数据技术方向）

年级班别 2016级（4）班

学 号 3116004617

学生姓名 姚棉阳

指导教师 武继刚

2020年 5 月

**多源多中继边缘计算设备能耗延迟权衡研究**

**姚棉阳**

**计算机学院**

**摘 要**

随着科学技术的发展，移动边缘计算的提出加快了物联网设备的发展，使得越来越多的计算密集型应用可以在智能移动设备上运行。现有研究主要考虑智能移动设备和边缘服务器之间的协作，却没有考虑到中继设备的巨大潜能。基于此，本文提出一个多移动设备、多中继设备和一个边缘服务器的架构，移动设备上的任务可以由不同移动设备本身、中继设备或边缘服务器执行。基于该架构，本文研究能耗优化问题以及时延能耗权衡最小化问题。对于能耗优化问题，我们将其分为总能耗最小化问题以及最小化最大能耗对问题，并提出TEOA算法和MPMA算法分别解决；对于时延能耗权衡问题，我们将其归类为一个多维背包问题，证明其属于NP-Hard问题，并提出EDTA算法进行解决。数值实验结果表明了我们提出的TEOA算法、MPMA算法、EDTA算法的有效性。在最小化总能耗方面，本文提出的TEOA算法比随机算法提升了23.65%；在最小化最大能耗对方面，在取得相同能耗效果的情况下，MPMA算法的运行时间比TEOA算法减少了50%；在时延能耗权衡方面，EDTA算法在性能方面均比TEOA算法、MPMA算法优异。

**关键词：**边缘计算、协作、能耗优化、权衡

**Abstract**

With the development of science and technology, the proposal of mobile edge computing has accelerated the development of Internet of things devices, enabling more and more computationally intensive applications to be run on mobile phones. Existing researches focus on the collaboration between mobile devices and edge server. But they fail to take into account the great potential of relay devices. Based on this, this paper proposes an architecture with mobile device, relay equipment and an edge server, the task on mobile device can be executed by itself, relay equipment or edge server. Based on this architecture, this paper studies the energy consumption optimization problem and delay energy tradeoff minimization problem. Then, we address the energy consumption optimization problem and the delay energy consumption tradeoff problem. For energy consumption optimization problem, we divide it into total energy consumption minimization problem and maximum energy consumption pair minimization problem, and propose TEOA algorithm and MPMA algorithm to solve these problems, respectively. For the delay-energy consumption tradeoff problem, we classify it as a multi-dimensional knapsack problem and prove that it is np-hard. To solve this problem, we propose an EDTA algorithm to address it. Numerical experiment shows the effectiveness of TEOA algorithm, MPMA algorithm and EDTA algorithm. In total energy consumption minimization problem, the proposed TEOA algorithm outperforms 23.65% on average better than random algorithm. In the case of achieving the same performance of energy consumption, the running time of MPMA algorithm is half shorter than that of TEOA algorithm. Besides, in terms of delay energy tradeoff, EDTA algorithm is superior to TEOA, MPMA algorithm.

**Key words:** edge computing, collaboration, energy consumption optimization, tradeoff

**目 录**

[1 绪论 1](#_Toc38453923)

[1.1 研究背景 1](#_Toc38453924)

[1.2 相关工作 2](#_Toc38453925)

[1.3 研究内容和主要贡献 2](#_Toc38453926)

[1.4 论文架构 3](#_Toc38453927)

[2 系统模型与问题定义 4](#_Toc38453928)

[2.1 网络模型 4](#_Toc38453929)

[2.2 任务模型 5](#_Toc38453930)

[2.3 通信模型 5](#_Toc38453931)

[2.4 计算模型 6](#_Toc38453932)

[2.4.1 本地计算模式 6](#_Toc38453933)

[2.4.2 直接边缘计算模式 7](#_Toc38453934)

[2.4.3 协作计算模式 7](#_Toc38453935)

[2.4.4 联合协作模式 8](#_Toc38453936)

[2.5 问题定义 9](#_Toc38453937)

[2.5.1 总能耗最优化问题 9](#_Toc38453938)

[2.5.2 最小化最大能耗对问题 10](#_Toc38453939)

[2.5.3 能耗与时延的权衡优化问题 11](#_Toc38453940)

[3 算法 13](#_Toc38453941)

[3.1 总能耗最优化问题 13](#_Toc38453942)

[3.1.1 不同计算模式的最优解 13](#_Toc38453943)

[3.1.2 迁移策略算法 14](#_Toc38453944)

[3.1.3 时间复杂度分析 17](#_Toc38453945)

[3.2 最小化最大能耗对问题 17](#_Toc38453946)

[3.2.1 迁移策略算法 17](#_Toc38453947)

[3.2.2 复杂度分析 21](#_Toc38453948)

[3.3 能耗与时延的权衡优化问题 21](#_Toc38453949)

[3.3.1 最优CPU频率决策 21](#_Toc38453950)

[3.3.2 迁移策略算法 23](#_Toc38453951)

[3.3.3 时间复杂度分析 26](#_Toc38453952)

[4 实验结果与分析 27](#_Toc38453953)

[4.1 参数设置 27](#_Toc38453954)

[4.2 总能耗最优化问题 27](#_Toc38453955)

[4.3 最小化最大能耗对问题 29](#_Toc38453956)

[4.4 能耗与时延的权衡优化问题 31](#_Toc38453957)

[结论 34](#_Toc38453958)

[参考文献 35](#_Toc38453959)

[致 谢 38](#_Toc38453960)

# 绪论

## 研究背景

随着移动智能设备越来越受欢迎，用户倾向于在移动设备上完成诸如虚拟现实、在线游戏等应用。移动智能设备的体积小，这也表示其电池容量小，计算能力有限。然而，移动智能设备需要巨大的计算资源、充分的存储资源、足够的电量来支撑此类应用的运行。这对于移动设备的性能而言，是非常具有挑战性的[1,2]。所以，如何在智能移动设备中实现高效的节能的任务调度具有非常重要的意义[3]。

随着物联网的快速发展，互联网中数据的传输量呈指数增长[4]。虽然移动智能设备可以将计算密集型任务迁移到云服务器进行处理，但数量过多的任务迁移会导致网络拥塞进而导致传输延迟，从而带来低劣的用户服务体验[1,4,5]。

在上述背景下，移动边缘计算被提出。移动边缘计算技术的使用可以把更多的时间用于任务执行方面，从而满足用户良好的服务体验[6,7]。为减少通信方面的时延，它会将服务器部署在靠近移动智能设备附近的基站上。此外，当大部分任务只被迁移到边缘服务器，而不需要通过网络发送到云服务器时，网络的拥塞概率也能大大降低。

智能移动设备间的协作通信模式也可以提升移动设备的性能。这是因为一方面，协作通信可以提高网络通道利用率，减少设备之间的传输时间以及传输能耗；在另一方面，考虑到中继设备拥有的计算性能，协作通信也可以进一步节省能耗。

在移动边缘计算领域下，处理任务的时延（任务的传输时延和任务的执行时延）以及对应的能耗都是衡量移动边缘计算的重要权衡因素。基于此，我们提出了一个具有多个移动设备、多个中继设备和一个位于基站的边缘服务器架构。该架构具有如下特点：

1、移动设备上存在一个不可分解的任务，该任务可由移动设备本身执行，或者迁移到中继设备或位于基站上的边缘服务器并由其执行；

2、中继设备拥有着比移动设备更强的计算能力，它可以帮助移动设备执行任务，或与移动设备进行协作通信，帮助移动设备以解码转发的形式将任务发送到边缘服务器上；

3、边缘服务器位于基站上，其拥有多个中央处理器，可以同时执行多个任务。

多种计算方式并存导致了问题的复杂性；此外，移动设备与中继设备之间的共同协作也使得问题具有很高的复杂性。考虑到上述问题的复杂性，如何在移动边缘协作计算架构下权衡任务的完成时延以及能耗是一个挑战。

## 相关工作

随着科学技术的发展，计算密集型应用的不断发展对移动智能设备的性能提出了很大的挑战。移动边缘计算的提出，不仅可以优化移动智能设备的能耗，也可以更好的满足用户的服务质量需求。现已有大量的关于移动边缘计算领域的研究。

在移动边缘计算能耗方面，文献[1]提出了多个移动设备和边缘服务器结构下的节能计算迁移策略方案。在保证服务性能的前提下，文献[6]研究并设计一个任务迁移算法，从而解决在移动设备-边缘服务器-云服务器架构下移动设备能耗优化问题。文献[3]则解决了移动设备终端之间的节能高效协作。此外，文献[8]研究了多个移动设备-边缘服务器-云服务器架构下的能耗优化问题。然而，上述的文献均只考虑了移动设备和边缘服务器、云服务器之间的协作计算，未能考虑到中继设备蕴含丰富的计算能力和通信容量，从而进一步优化系统能耗。

在移动边缘计算时延方面，文献[4]考虑了三层架构下的任务处理时延优化问题。在移动边缘计算权衡方面，文献[7]研究了单移动设备和边缘服务器模型下的计算时延和设备能耗权衡最小化问题；文献[5]则是研究了多个移动设备和边缘服务器模型下的总计算能量效率最大化问题；文献[9]提出了一种在线动态任务分配调度方法，解决具有能量收集能力的移动边缘计算系统的能量消耗与执行延迟之间的权衡问题。然而，这些文献也未能考虑到中继设备的协作能力。此外，文献[10]考虑了中继设备的潜能，但其模型并不适用于现实情况中多移动设备、多中继设备并存的情形。

## 研究内容和主要贡献

在移动边缘协作计算领域下，本文主要研究多移动设备、多中继设备和一个边缘服务器的架构下的能耗优化以及时延与能耗的权衡。针对能耗优化方面，本文将其分为两个子问题进行研究，分别为总能耗最小化问题、最大能耗对最小化问题，为使任务完成时延满足用户服务质量需求，本文加入了时间约束；针对时延与能耗的权衡，本文同时考虑时延以及能耗两方面的约束。为解决这些问题，我们需要面对以下挑战：

1、与单个中继设备的系统不同，当多个中继设备共同存在时，低效的中继设备选择不利于系统能耗的优化；

2、迁移策略的选择对于系统的能耗优化起着关键性的作用，多种计算方式并存不仅会增加迁移算法的时间复杂性，也会提升系统的节能效益；

3、迁移算法的设计必须是高效的，也就是迁移算法必须在多项式时间内结束。

针对这些问题，我们一一提出了相应的算法进行解决。此外，为度量本文提出架构的性能，我们与文献[11]的模型进行对比。同时，为对比不同算法在最小化最大能耗对的性能，本文同时与模拟退火进行了对比。本文的主要贡献点如下：

1、我们根据本地计算、迁移后计算设计出四种计算模式，并讨论在不同计算模式下的能耗最优解；此外，本文中提出的架构与文献[11]的模型进行了对比，从实验结果可知，中继设备的加入不仅提高了架构中的通信效能，也使得架构下的能耗进一步优化；

2、针对能耗优化问题，我们考虑了时延约束条件，在保证用户的服务体验下优化能耗，对于能耗优化方面，我们将其分为两个子问题：总能耗最小化问题和最大能耗对最小化问题。针对第一个子问题，我们在假设边缘服务器具有无限资源的情况下，将其转变为一个二分图带权匹配问题，并运用KM算法[12,13]进行解决。对于第二个子问题，我们证明其为线性整形规划问题，并提出了一个基于迭代的算法进行求解；针对时延与能耗的权衡问题，我们同时考虑了时延和能耗的约束，将其归纳为一个多维背包问题，因此该问题属于NP-hard；

3、在最小化总能耗方面，本文提出的架构与文献[11]的模型对比节约了45.27%的能耗，同时在性能方面比随机算法节约了23.65%的能耗；在最小化最大能耗对方面，虽然TEOA算法和MPMA算法取得了相同的能耗效果，但MPMA算法花费更少的时间，可见MPMA算法在一定程度上的优异性；在时延能耗权衡方面，我们所提出的EDTA算法在不同环境下均比TEOA算法和MPMA算法优异。

## 论文架构

本文一共4章，具体结构如下：

在第1章中，我们主要介绍了本文的研究背景、移动边缘计算领域的相关工作、本文主要工作、面临的挑战以及本文的贡献；

接着，我们在第2章中介绍了基础概念，对移动边缘协作计算的网络模型、任务模型、通信模型、计算模型进行形式化定义，最后提出了本文的优化目标；

在第3章中，我们针对三个优化目标，推导出对应的算法；

然后，为测试我们提出的算法的有效性，我们在第4章进行了数值实验；

最后，我们总结了本文的结论。

# 系统模型与问题定义

## 网络模型

如图2.1所示，系统中包括了多个移动设备、多个中继设备以及一个位于基站的边缘服务器，其中机动车、智能移动设备上含有一个待执行的任务，因此我们将其归为移动设备层；而中继设备层方面，设备本身没有待执行的任务，因此可以充当中继设备，用于传输数据或帮助移动设备执行完成任务；边缘服务器与基站连接，其可以接受移动设备层传输过来的任务，然后执行并将结果返回给移动设备。从图2.1中可以看出，移动设备上的任务可以由移动设备本身执行，也可发送到中继设备或边缘服务器，并由其执行完成，再返回相应结果。我们定义移动设备集为，中继设备集为，其中，分别为移动设备，中继设备的数量。最后，我们定义为位于基站的边缘服务器。



图2.1

系统模型

移动设备均配备了一条天线，以支持任务的传输。每一个移动设备均含有一个待执行的任务，对于移动设备，，它含有一个不可分割的任务，这意味着该任务无法分成两部分，并由两个设备完成。架构下的中继设备，，可以接受移动设备的任务，并帮助移动设备完成该任务，也可以作为中继器，将任务以解码转发（Decode and Forward, DF）的方式发送到位于基站的边缘服务器，并由边缘服务器完成。位于基站的边缘服务器拥有多条天线，这也意味着它可以同时与多个设备进行通信、接收数据。与文献[14,15]相似，边缘服务器拥有*C*个中央处理器（Center Process Unit, CPU），这表示边缘服务器可以同时执行多个任务。不失一般性，本文中假设边缘服务器上的CPU拥有相同的性能，例如：CPU周期，能量消耗等。

## 任务模型

移动设备均含有一个任务，其执行完成方式本文将会在2.4节中进行介绍。此外，为了保证用户体验，任务的执行完成时间有最大完成时间约束：，因此，我们有：

,

（2.1）

其中和分别为完成任务的传输时间和执行时间。

在2.4节中，我们将会介绍四种计算模式，其中包括本地计算模式，协作计算模式，直接边缘计算模式，联合协作模式。因此，我们将引入四个参数来表示对应的计算策略。

定义为本地计算模式的对应策略。特别地，表示任务将由移动设备本身计算完成，也就是本地计算模式，而则表示其他情况。与的定义相似，我们设定、和分别为移动设备协作计算模式、直接边缘计算模式和联合协作模式的对应策略。与符号相似，表示移动设备将任务发送到边缘服务器并由边缘服务器完成，而则表示其他情况；表示移动设备将任务发送到中继设备，并由中继设备执行完成，而则表示其他情况；表示移动设备与中继设备进行协作，以DF方式将任务发送到边缘服务器并由边缘服务器处理完成，而则表示其他情况。注意任务只能在一个设备上执行，因此也只能采用一种计算模式。因此，我们有：

（2.2）

## 通信模型

在这一部分，我们将会介绍通信模型。在本模型下，我们采用正交频分多址（orthogonal frequency division multiplexing, OFDMA）技术。因此，我们可以忽略不同设备间的相互干扰[16]。此外，为了有效利用网络中的带宽资源，我们将总带宽*B*分成*M*个子信道，每个移动设备的信道带宽可以表示为[17,18]。定义移动设备和中继设备之间的信道增益为，其中为两个设备间的欧几里得距离，和分别表示对应的瑞利衰减系数和一个被设为4的常量[19]。根据香农公式可知，当移动设备与中继设备进行通信传输时，对应的通道容量可以表示为：

,

（2.3）

其中，为移动设备的传输功率，设为27dBm，表示中继设备的加性高斯白噪声（additive white Gaussian noise, AWGN）功率[20]。

定义移动设备与边缘服务器之间的通信容量为，与公式（2.3）类似，我们有：

,

（2.4）

其中和分别表示移动设备和边缘服务器之间的信道增益和边缘服务器的AWGN功率。与公式（2.3）、公式（2.4）相似，中继设备和边缘服务器之间的通道容量可以表示为：

,

（2.5）

其中和分别为中继设备和边缘服务器之间的信道增益和中继设备的传输功率，设为27dBm。

接下来，我们要介绍以DF方式进行的协作通信。以DF方式进行协作通信包含两个阶段，其中第一个阶段为移动设备将数据同时传输到中继设备和位于基站的边缘服务器，持续时间假设为，第二个阶段为中继设备将数据发送到边缘服务器，持续时间假设为。当移动设备与中继设备协作以DF方式发送任务，对应的通道容量设置为。因此，以DF的协作通信传输的数据量可以表示为[21]。所以，基于DF方式的传输速率可以表示为：

,

（2.6）

其中是一个系数。在我们的模型中，由于的取值超出我们本次的研究范围，因此我们将设置为一个常量。

## 计算模型

### 本地计算模式

如果任务由移动设备自身执行，则称为本地计算模式。值得注意的是，该模式下，任务不需要发送到其他设备，只由移动设备本身完成，因此我们不需要考虑此模式下的传输时间以及传输能耗。因此在该计算模式下，我们有。定义移动设备的CPU频率为，其不可以超过移动设备最大的CPU频率，即。因此，如果任务由本地计算模式完成，则完成任务的时间表示为：

,

（2.7）

其中为任务的工作量大小，表示完成任务所需要的CPU周期频率。该模式下，完成任务的能耗为：

,

（2.8）

其中为一个取决于移动设备架构的系数[22]。

### 直接边缘计算模式

在这个模式下，任务将会由移动设备直接迁移到位于基站上的边缘服务器，接着再由边缘服务器执行完成。定义移动设备将任务直接迁移到边缘服务器的时间为。因此，我们有：

,

（2.9）

其中为任务的数据量大小。因此，传输部分的能耗可以表示为：

.

（2.10）

与公式（2.7）相似，边缘服务器执行完成任务所需要的时间可以表示为：

,

（2.11）

其中表示边缘服务器的CPU频率，并且其不能超过最大的CPU频率，即。因此，根据文献[23]可知，对应的能量消耗可以表示为：

,

（2.12）

其中，和都是一个正常量，可以通过离线功率拟合得到，此外，的取值范围为[2.5, 3][24]。因此，如果任务以直接边缘计算模式完成，完成该任务的总时延可以表示为：

.

（2.13）

相应地，完成该任务的总能耗可以表示为：

.

（2.14）

### 协作计算模式

在这个模式下，移动设备会将任务迁移到中继设备，并由该中继设备完成。因此，在这个模式下，能耗以及时延均包括以下两个阶段：任务传输阶段和任务执行阶段。定义和分别为传输阶段的时延和对应的能耗，与公式（2.9）和（2.10）相似，我们有：

,

（2.15）

和

.

（2.16）

接着，让和分别表示任务由中继设备执行时的时延和对应的能耗。因此，在这个模式下的第二个阶段，我们有：

,

（2.17）

其中表示中继设备的CPU频率，且服从，公式中表示中继设备最大的CPU频率。此外，对应的能耗可以表达为：

,

（2.18）

其中表示为一个取决于中继设备架构的系数，设为。因此，当任务通过这种方式与中继设备协作完成任务，总的执行时延可以表示为：

.

（2.19）

对应的能耗可以表示为：

.

（2.20）

### 联合协作模式

在这个模式下，中继设备将会协助移动设备，以DF方式将任务迁移到位于基站的边缘服务器上，并由边缘服务器完成任务。如果任务在中继设备的协作下完成了此任务，那么传输阶段的时延可以表示为：

.

（2.21）

因此，对应的传输阶段的能耗可以表达为：

.

（2.22）

与公式（2.11）相似，在任务的执行阶段，完成任务的时延可以表示为：

.

（2.23）

与公式（2.12）类似，任务执行阶段的能耗可以表达为：

.

（2.24）

因此，当移动设备与中继设备协作，通过联合协作模式完成任务时，总时延可以表示为：

.

（2.25）

对应地，该模式下的总能耗表示为：

（2.26）

## 问题定义

在这一部分中，我们将会介绍文中将要解决的问题。首先，我们将介绍总能耗最优化问题；接着，我们将阐明最大移动-中继设备能耗对最小化问题；最后，我们阐述关于能耗与时延的权衡问题。

对于一个移动设备完成任务的能耗，可以表示为：

.

（2.27）

对应的时间延迟为：

.

（2.28）

### 总能耗最优化问题

因此，总能耗最优化（TEO）问题可以表示为：

其中，条件C1表示了对每一个任务的时间约束；条件C2、C3和C4分别表示对移动设备、中继设备和边缘服务器上CPU的频率约束；条件C5和C6确保了每一个任务只能由一种计算模式完成；条件C7则表明了一个中继设备最多与一个移动设备配对；条件C8则是边缘服务器资源的限制，表明边缘服务器最多同时执行任务数量。

### 最小化最大能耗对问题

最小化最大能耗对（MPM）问题可以表示为：

其中条件C1表示了对不同移动设备上的任务的时间约束；条件C2、C3和C4则分别表示对移动设备、中继设备和边缘服务器上CPU的频率约束；条件C5和C6确保了每一个任务只能由一种计算模式完成；条件C7限制了一个中继设备最多与一个移动设备配对；条件C8则是对边缘服务器的CPU资源条件限制，条件C9中*R*则表示了最大的能耗对。

**定理**：MPM问题为一个整形线性规划问题，一般属于NP-hard。

**证明**：当最优CPU频率解决时，该问题可以转变为一个线性规划问题。此外，由于该问题的解均属于整数，因此该问题可转变为整形线性规划问题。通常解决整形线性规划问题是NP-hard的[19]。因此MPM问题属于NP-hard。

### 能耗与时延的权衡优化问题

定义和分别为执行任务的时延和能耗的权重，其中，。因此，对于移动设备的时延与能耗的权衡值可以表示为：

.

（2.29）

因此，能耗与时延权衡（EDT）问题可以表示为：

其中条件C1为不同任务的时间约束；移动设备、中继设备和边缘服务器的CPU频率约束条件分别为条件C2、C3和C4；条件C5和C6确保了计算模式的唯一性；C7限制条件则是占用中继设备唯一性的约束；条件C8则是边缘服务器的资源的条件限制，条件C10为能耗方面的约束。

我们首先给出多维背包问题的定义，然后再来证明EDT问题属于NP-hard。

**定义**：多维背包问题。定义*X*个不同类型的物品为。对于每一个物品，其价值为，并且有不同的资源条件需要满足。多维背包问题是在给定约束条件的情况下最大化背包的总价值，那么其可以表达为：

,

（2.30）

其中第一个限制条件是一个0-1约束，第二个限制条件是对第*k*维资源的约束。接下来我们来证明EDT问题属于多维背包问题。

**定理**：EDT问题属于多维背包问题，属于NP-hard。

**证明**：通过对条件C6的转换，我们得到：和。因此，限制条件C6可以表述为：和。此外，由限制条件C6可知，。因此，我们可以认为问题一共有*M*项因子相加。那么问题可以表述为：

其中为一个引入的正常量，并且服从。因此问题转化为，其中条件C5是对变量，，，的二元约束，条件C6.1、C6.2、C7、C8则是在不同维度的容量约束。因此可以得知问题是一个多维背包问题，属于NP-hard[25]。所以，问题是NP-hard。

# 算法

## 总能耗最优化问题

对于，它可以转化为以下几个子问题：1）不同计算模型下的最优解；2）最优调度问题。对于第一个子问题，我们将在3.1.1节进行讨论；而对于第二个子问题，则在3.1.2节进行讨论。

### 不同计算模式的最优解

首先是关于本地计算模式的最优解，根据公式（2.7）和（2.8），可得到移动设备上CPU频率与该模式下的能耗之间关系的函数为。可知，该函数为一个增函数。又因为限制条件以及，其中。因此，问题中本地计算模式的最优解为：

（3.1）

接着，我们讨论直接边缘计算模式的最优解。假如移动设备采用这种计算模式完成任务。那么，该模式下的总能耗与边缘服务器上CPU的频率之间的函数关系为。可知，关于的一次求导函数为，其中是一个取值范围在2.5到3之间的整数，而和均是一个正常量[24]，因此该函数是一个单调增函数。又因为限制条件和。因此，问题中直接边缘计算模式的最优解为：

,

（3.2）

其中。

接着，我们讨论协作计算模式下的最优解。假如移动设备上的任务将任务迁移到中继设备上，根据公式（2.18）和公式（2.20），中继设备的CPU频率和该模式下的总能耗之间的函数关系可以表示为，其中约束条件为和。显而易见，该函数是一个单调递增函数，因此，此模式下最优的CPU频率为：

,

（3.3）

其中，。

最后，我们再来讨论下当移动设备以联合协作模式和中继设备完成任务时的总能耗。根据公式（2.22）、（2.24）和（2.26），我们建立关于总能耗和边缘服务器上的CPU频率之间的函数关系：，其中与之间没有相应的函数关系，可将视为一个常数。其中，关于的一次求导公式为：。由于是一个取值范围在2.5到3之间的整数，而和均是一个正常量[24]，因此易知该函数是一个单调递增函数。此外，在这个模式下需要满足如下约束条件：和。因此，该模式下边缘服务器的最优CPU频率为：

,

（3.4）

其中。

### 迁移策略算法

在这一小节中，我们将设计一个迁移策略。首先，我们将问题转化为以下形式：

其中。由于本地计算模式和直接边缘计算模式不会与中继设备进行协作，而另外两个模式会与中继设备进行协作，因此，我们将迁移策略分为下面两个子问题进行求解：

s.t.

其中问题为本地计算模式与直接边缘计算模式之间的决策的选择，不同移动设备间的决策不会相互干扰，因此可通过贪心策略实现；在忽略条件下，我们将问题转化为一个二分图最大带权匹配问题，该问题可通过KM算法[12,13]获得最优匹配结果，接着再通过多次迭代满足限制条件C8。的算法过程如下：我们先计算本地计算模式以及直接边缘计算模式的最优CPU频率，然后算出该模式完成任务的时延以及能耗，再进行贪心选择，该算法的整个流程描述在表3.1中。

表3.1 的解法

|  |
| --- |
| Input: ,  Output: , , , , |
| 1. **for** *i* from 1 to *M* **do** 2. compute and by （3.1）and（3.2）, respectively 3. compute , , , and by （2.7）,（2.8）,（2.13）and（2.14）, respectively 4. set 5. **if** **then** 6. , , , , 7. **end if** 8. **if** **and** **then** 9. , , , , 10. **end if** 11. **end for** |

TEO问题的算法过程思想如下：第一步，完成算法1确定默认的任务计算模式；接着，我们计算出协作计算模式和联合协作模式的最优CPU频率，然后算出协作计算模式和联合协作模式下的执行时延以及能耗，再由权重处理形成矩阵，通过KM算法[12,13]进行二分图匹配；最后，算法对匹配结果进行迭代，从而满足边缘服务器上CPU的限制条件。算法的详细过程在表3.2中。

表3.2 TEOA算法

|  |
| --- |
| Input: *,*  Output: *, , ,* , |
| **step1**：ensure the default mode   1. solve   **step2**: form a matrix and generate the match policy   1. **for** *i* from 1 to *M* **do** 2. **for** *j* form 1 to *N* **do** 3. compute and by （3.3）and（3.4） 4. compute , , , 6. **if** **and** **do** 7. , , , , 8. **end if** 9. **if** , **and** ( **or** ) **do** 10. , , , , 11. **end if** 12. **end for** 13. **end for** 14. using KM to get the match policy and update the default mode   **step 3**: iterate to satisfy condition   1. for each mobile device ensure its energy consumption 2. **while** *C8* is not met**do** 4. **if** **and** **then** 5. , , , 6. **else if** **and** **then** 7. , , , 8. **else if** **and** **then** 9. , , , 10. **end if** 12. **end while** |

### 时间复杂度分析

接下来我们对TEOA算法进行时间复杂度分析。假设移动设备的数量为，中继设备的数量为，那么TEOA算法的时间复杂度为。

在第一步中，我们会确定默认计算方式，这是一层循环，因此第一步的时间复杂度为。第二步，我们会形成一个矩阵，该矩阵为两层循环完成，因此，形成该矩阵的时间复杂度为；接下来则是KM算法的匹配过程，由于KM算法最多有轮需要匹配，在每一次匹配中，KM算法均会调整顶点标记，并找出初始相等图上的最大匹配，而每一次调整节点的时间复杂度为，因此这一步骤中KM算法的时间复杂度为。第三步，迭代以满足限制条件，因为移动设备的数量为，因此最大的迭代次数不超过，所以第三步的时间复杂度为。

综上所述，TEOA算法的时间复杂度为。

## 最小化最大能耗对问题

对于问题，我们将其转化为以下两个子问题：1）不同模式下最优CPU的选择；2）迁移策略的选择。由于MPM问题与TEO问题一致，都是时延条件满足下的能耗优化问题，因此，该算法中不同计算模式的最优CPU频率与TEOA算法中一致。我们在3.2.1节中，我们将讨论迁移策略的思路，并在3.2.2节中讨论迁移策略的时间复杂度。

### 迁移策略算法

对于每一个移动设备，我们可以将其能耗转变为：

（3.5）

为方便解决问题，迁移策略的选择选择可以拆分成以下两个子问题：

对于问题，其形式与一致，因此我们用解决的算法解决该问题。同理，问题也能达到一个最优解。对于问题，我们先忽视限制条件*C8*，然后通过多次迭代来达到一个近似解。首先，我们给每一个移动设备建立一个队列，队列中存放着多个元组。当移动设备和中继设备进行匹配，那么会有两种计算方式，分别是协作计算模式和联合协作模式。因此，我们可以算出两种模式下的能耗，选择其中能耗较小的一个，形成元组，其中元组存储的信息如下：（计算模式，中继设备，对应能耗）。如果，那么该元组将会存放到移动设备的队列中。因此，我们可以看出，当列表为空时，默认计算策略将是此移动设备最优的计算策略。第二步，算法将会给每个移动设备生成上述的队列，并给移动设备随机匹配一个中继设备。第三步，算法会找到最大的能耗对。接着，算法会不断寻找一个更好的移动-中继设备能耗对来替换原先的能耗对。第三步将会不断的重复执行，直到该算法无法找到一个更好的能耗对进行替换。该算法将会迭代多次，从而满足限制条件。最小化最大能耗对算法（MPMA）的过程展示在表3.3中。

表3.3 MPMA算法

|  |
| --- |
| Input: ,  Output: *, , ,* , |
| **step1**：ensure the default computing mode   1. address problem   **step2**: Generate queues   1. **for** *i* from 1 to *M* **do** 2. **for** *j* from 1 to *N* **do** 3. compute and by （3.3）and （3.4）, respectively 4. compute , , , by （2.19）,（2.20）,（2.25）,（2.26）, respectively 6. **if** **do** 7. , , , , 8. **end if** 9. **if** **and** **do** 10. , , , , 11. **end if** 12. **if** **then** 13. store the information of this mobile-relay pair to the queue of mobile device 14. **end if** 15. **end for** 16. **end for** 17. assign mobile-relay pair randomly and update   **step 3**: iterative process    2. set all the relay devices as“unused” 3. **for** the element in queue of in a non-decreasing order **do** 4. execute algorithm check-relay-availability (, ) 5. **if** relay device can be used **then** 6. assign relay device to mobile device 7. turn to run step 3 again 8. **end** **if** 9. **end for** 10. **if** mobile device cannot match a new relay device **then** 11. end the process of iteration 12. **end if**   **step 4**: iterate to satisfy condition   1. **while** *C8* is not met **do** 3. **if** **and** **then** 4. , , , 5. **else** **if** **and** **then** 6. , , , 7. **else** **if** **and** **then** 8. , , , 9. **end** **if** 11. **end while** |

在算法3的执行过程中，会需要调用算法4来检查中继设备的可用性。当中继设备没有与移动设备匹配时，或者该中继设备与此移动设备匹配，或者该中继设备原先匹配的移动设备可以找到另一个移动设备进行匹配，那么该中继设备是可用的，其中算法4的过程在表3.4中。

表3.4 check-relay-availability 算法

|  |
| --- |
| **Input:** relay device , energy consumption pair *E*  **Output:** the availability of relay device |
| 1. **if** relay device is not assigned **then** 2. relay device can be used 3. **end if** 4. set relay device “used” 5. /\*assume the relay device *j* matches mobile device *a*\*/ 6. execute algorithm find-new-relay(, *E*) 7. **if** mobile device can find a new match **then** 8. relay device can be used 9. **end if** 10. relay device cannot be used |

而在算法4的执行过程中，存在调用算法5的过程，该过程则是为与中继设备原先匹配的移动设备寻找另一个中继设备进行匹配。算法5在调用过程中，算法会检测中继设备的可用性，因此会在算法5的运行过程中会不断地调用算法4。算法5的具体过程在表3.5中。

表3.5 find-new-relay 算法

|  |
| --- |
| **Input**: mobile device , energy consumption pair *E*  **Output**: the result of new match of mobile device |
| 1. **for** all the “unused” relay device in queue of in a non-decreasing order **do** 2. **if** **then** 3. execute algorithm check-relay-availability (, *E*) 4. **if** relay device can be used **then** 5. assign relay device to mobile device 6. mobile device can find a new relay device 7. **end if** 8. **end if** 9. **end for** 10. mobile device cannot find a new relay device |

### 复杂度分析

现在我们来分析下MPMA算法的时间复杂度。假设移动设备的数量为，中继设备的数量为，那么MPMA算法的时间复杂度为。证明过程如下：

第一步，为一个贪心的迭代过程。该过程为一个一层循环嵌套，因此第一步的时间复杂度为。接着，第二步中，队列元素的形成以及添加，这是在一个双层循环语句下完成的，因此其时间复杂度为。对于每一个移动设备，最多有个中继设备可以与其形成能耗对，因此最多有个能耗对可以迭代。此外，每一次寻找能耗对至多有*N*次。因此，迭代寻找能耗对的过程的时间复杂度为。最后，迭代满足约束条件，由于移动设备的数量为，因此最大会迭代次。因此，该过程的时间复杂度为。

综上所述，MPMA算法的时间复杂度为。

## 能耗与时延的权衡优化问题

接下来让我们分析下EDT问题。同样的，我们将该问题分为两部分进行解决：1）不同模式下，最优CPU频率的选择；2）迁移策略的选择。

### 不同模式最优决策

我们先来考虑不同模式下的CPU频率的不同。首先是本地计算模式的最优CPU频率考虑，根据公式（2.7）和（2.8），该模式下的权衡值与CPU频率之间的函数关系为：。因此，我们有，可知先减小后增长。所以，。考虑到，和，因此当时，该模式可以被选取为任务的计算模式。所以，问题中本地模式最优CPU频率为：

.

（3.6）

接着，我们来讨论直接边缘计算模式的最优CPU频率，根据公式（2.11）—（2.14），该计算模式下边缘服务器的CPU频率与该模式下的能耗存在以下函数关系：，其中，与的取值无关，因此我们可将其视为一个常数。所以，我们有，其中是个取值范围在2.5到3之间的整数，而和均是一个正常量[24]。因此，我们可知先减小后变大。所以，。考虑到EDT问题的限制条件，因此当时，该模式有解。所以，问题中直接边缘计算模式最优CPU频率为：

,

（3.7）

其中，，。

接着，我们来讨论协同计算模式下的最优解。基于公式（2.17）—（2.20），可知该模式下权衡值与中继设备CPU频率的函数关系为：，其中，的值与无关，可将其视为一个常数。因此，我们有。可知，随着的逐渐增大，先减小后增大。因此。考虑到EDT问题的限制条件，因此时，任务可由该模式完成。所以，问题中协作计算模式最优CPU频率为：

,

（3.8）

其中，。

最后，我们来考虑联合协作模式的权衡值最优情况。根据公式（2.23）—（2.26），我们可知该模式下权衡值与边缘服务器CPU频率的关系为：。由于很难确定该公式的最小值，因此，我们对其进行求导，可得：。因此，我们有。考虑到EDT问题的限制条件，因此我们有：，。所以，任务采用联合协作模式最优CPU频率为：

,

（3.9）

其中，。

### 迁移策略算法

在这一小节中，我们将设计一个迁移模式，决定不同的任务应该采用哪一种计算模式，以及移动设备与中继设备的协作执行任务。

由EDT问题中限制条件C5，C6，我们有：

.

（3.10）

假如，那么。那么我们有，且当或。因此，我们有

.

（3.11）

为方便表达式的书写，设定，，，。因此，问题可以等价转化为：

（3.12）

对（3.12）做进一步变形，那么我们有：

其中，，，。为解决问题，我们将其分为以下两个子问题进行解决：

,

,

其中，问题中的为问题的输出。对于问题，由于不同移动设备互不干扰，因此我们采用一个贪心算法进行解决。在限制条件满足的情况下，我们选取和中较小的一项进行解决。对于问题，我们将其转变为一个带权二分图匹配问题。因此我们可以用匈牙利算法或KM算法[12,13]完成。最后对匹配结果进行迭代，以满足边缘服务器CPU的限制条件。

关于问题的解法，我们在表3.6中进行了描述。由于不同移动设备间在本地计算模式以及直接边缘计算模式中的选择不会产生相互冲突。因此，该贪心策略是最优的。

表3.6 关于的贪心算法

|  |
| --- |
| Input: ,  Output: , , , , |
| 1. **for** *i* from 1 to *M* **do** 2. compute and by （3.6）and（3.7）, respectively 3. compute , , , and  by （2.7）,（2.8）,（2.13）and（2.14）, respectively 4. set 5. **if** **then** 6. , , , , 7. **end if** 8. **if** **and** **then** 9. , , , , 10. **end if** 11. **end for** |

问题的解法在表3.7展示，其中问题的解法，我们在表3.7的第2步进行阐述。最后会在第3步中通过迭代满足边缘服务器CPU的限制条件。

表3.7 EDTA算法

|  |
| --- |
| Input: ,  Output: , , , , |
| **step1**：ensure the default mode   1. address problem   **step2**: matching process   1. **for** *i* from 1 to *M* **do** 2. **for** *j* from 1 to *N* **do** 3. execute and by （3.8）and（3.9） 4. execute , , , 6. **if** **and** **do** 7. , , , , 8. **end if** 9. **if** , **and** ( or ) **do** 10. , , , , 11. **end if** 12. **end for** 13. **end for** 14. using KM algorithm to gain the matching result   **step 3**: iterative to satisfy condition   1. for all ensure its energy consumption 2. **while** *C8* is not met **do** 4. **if** **and** **then** 5. , , , 6. **else** **if** **and** **then** 7. , , , 8. **else** **if** **and** **then** 9. , , , 10. **end for** 12. **end while** |

### 时间复杂度分析

接下来我们分析EDTA算法的时间复杂度。

假设移动设备的数量为，中继设备的数量为。在第1步中，是通过一个贪心策略求解问题，这个过程通过一次循环得到，所以第1步的时间复杂度为。第二步中，首先会通过两个嵌套循环得到一个矩阵，因此该步骤的时间复杂度为；接下来是一个匈牙利算法或KM算法[12,13]的策略，该步骤的时间复杂度为，因此第2步的时间复杂度为。第三步中，会通过迭代从而满足边缘服务器CPU的限制条件。一共有个移动设备，那么该步骤至多进行次，最终满足约束条件，因此该步骤的时间复杂度为。

综上所述，EDTA算法的时间复杂度为。

# 实验结果与分析

这一章中，我们将提供一些仿真结果来说明所提算法的性能。在4.1节中，我们将阐述本次数值实验的参数设置，而在4.2节到4.4节中，我们将对不同的子问题进行实验模拟。

## 参数设置

实验参数除非稍后另外指定，否则将我们将使用描述的默认参数。首先假设架构下存在20个移动设备，15个中继设备；此外，所有的移动设备、中继设备随机的地理位置随机分布在一个100m x 100m的区域内[26,27]。移动设备上均有一个任务，其数据量大小遵循正态分布，并且它的大小为[0.2, 0.5] mb。为方便起见，每一个任务的时延均设置为0.1s。对于任务的工作量大小，我们将从以下两种情况进行分析：

Case 1：工作量大小为数据量大小的11倍；

Case 2：工作量大小和数据量大小之间没有明显的关系，任务的工作量大小设置为[0.1, 1] mb。

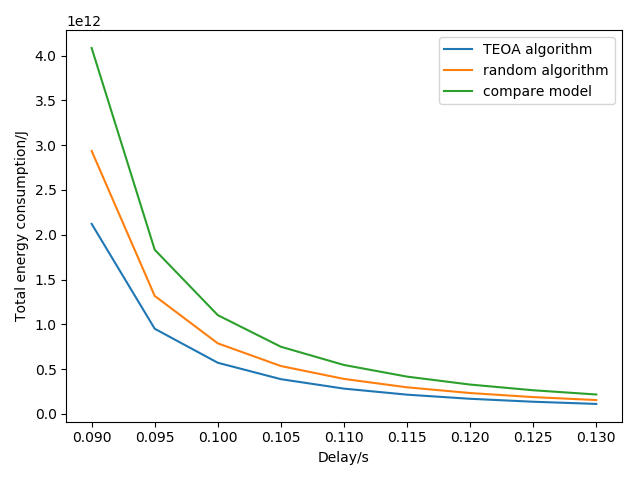
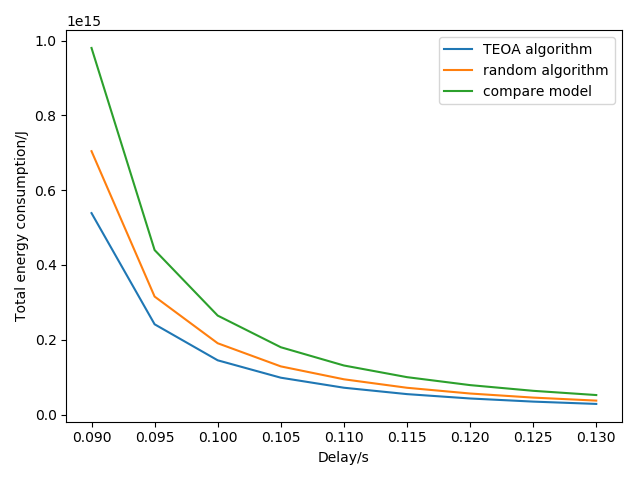
在任务传输方面，移动设备的带宽W设置为50Mb[28]；移动设备的传输能耗，中继设备的传输能耗设置为27dBm[29]；中继设备附近的AWGN设置为-70dBm，边缘服务器所在的基站附近的AWGN设置为-50dBm[21]；此外，瑞利衰减信道系数遵从复杂高斯随机变量分布CN(0, 1)[30]。在设备性能方面，移动设备的最大CPU频率设置为，中继设备的最大CPU频率设置为，边缘服务器的最大CPU频率设置为[31]；此外，，[32]。

此外，为表示该架构提出的有效性，我们与文献[11]所提出的模型进行对比。为方便起见，下面将文献[11]所提出的模型统称为对比模型。

## 总能耗最优化问题

在这一小节中，我们将研究TEOA算法的性能。由于没有现成的算法进行对比，因此我们与随机算法进行对比。此外，我们也将本文所提出的架构与对比模型进行对比。

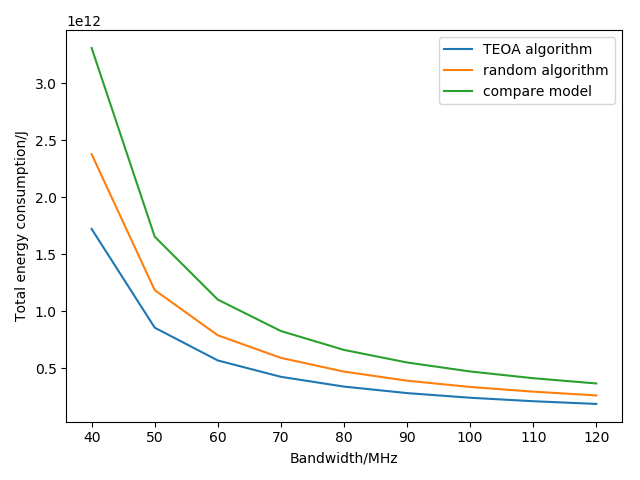
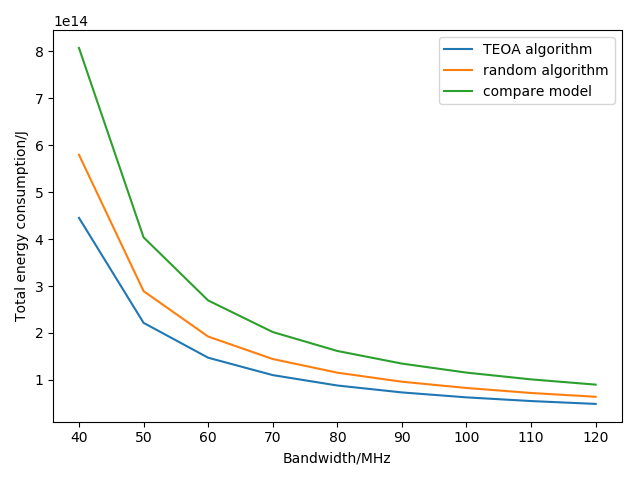
首先，我们来比对下任务的时延与设备总能耗之间的关系。图4.1中描绘了这种关系，其中图4.1(a)是Case 1下进行的，图4.1(b)描绘的是Case 2下进行的。由图4.1(a)、图4.1(b)可知，随着移动设备上任务的完成时延的提升，移动设备的总能耗降低。由公式（2.7）、（2.11）、（2.17）可知，当传输时延不变时，随着移动设备上完成任务的时延的提升，设备也有更多的时间用于完成该任务，那么设备就可以以一个低功率CPU频率完成任务。由公式（2.8）、（2.12）、（2.18）可知，在其他因素不变的情况下，完成任务的能耗与CPU频率成正比。也就是说，随着完成任务的CPU频率的降低，完成同一个任务所需要的总能耗降低，因而解释了图中曲线下降的原因。由图4.1可知，我们提出的TEOA算法与对比模型相比，在总能耗降低方面性能效果更佳。此外，在Case 1时，TEOA算法的性能比随机算法平均提升了23.65%，也比对比模型相比提升了45.27%；在Case 2时，TEOA算法的性能比随机算法平均提升了27.61%，也比对比模型相比平均提升了48.25%。



(a) Case 1; (b) Case 2；

图4.1 时延与总能耗之间的关系

接着，图4.2描绘了设备的带宽方面的变化与设备的总能耗之间的关系。由图4.2可知，随着带宽的增加，设备完成任务的总能耗也在逐渐的降低。根据公式（2.2）、（2.3）、（2.4）可知，移动设备的带宽与通道容量之间成正比关系，那么当移动设备的带宽增加时，对应的通道容量也在逐渐增加。接着，再根据公式（2.9）、（2.15）、（2.21）可知，随着通道容量的增加，传输时间相应降低。那么在传输功率等其他因素不变的情况下，传输方面的能耗也逐渐降低。此外，传输时间的降低，用于设备完成任务的时间也增加，根据之前的分析，随着设备用于计算时间方面的增加，完成任务所产生的能耗也在逐渐降低。综上分析，可知，随着带宽的增加，设备完成任务的总能耗会逐渐降低。图4.2(a)，图4.2(b)解释了这一点。此外，由总能耗这一因素进行分析，可知TEOA算法在性能方面比随机算法好，在系统结构方面，本文提出的架构也比对比模型优异。在Case 1时，TEOA算法比随机算法提升了23.71%；同种情况下，本文的架构所产生的总能耗比对比模型平均降低了45.50%；在Case 2时，TEOA算法平均比随机算法提升了27.97%；同种情况下，本文的架构所产生的总能耗比对比模型平均降低了48.52%。



(a) Case 1; (b) Case 2;

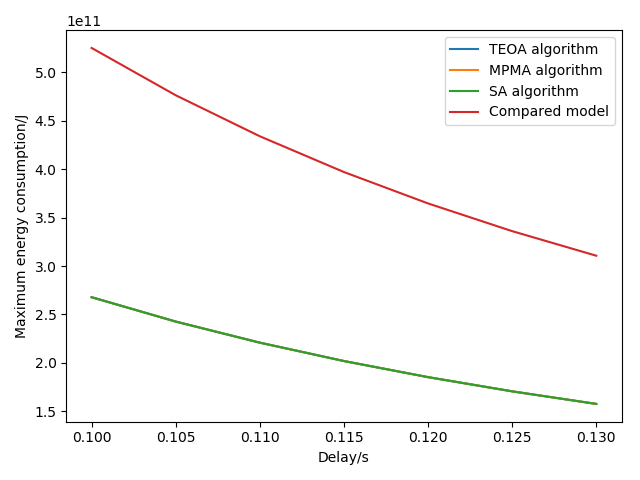
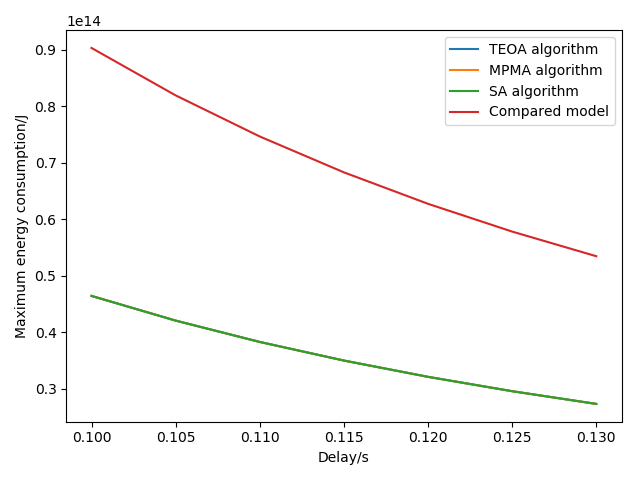
图4.2

总能耗在不同带宽下的对比

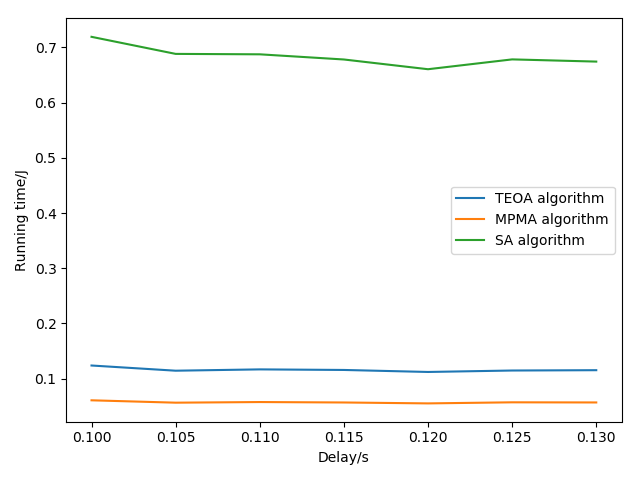
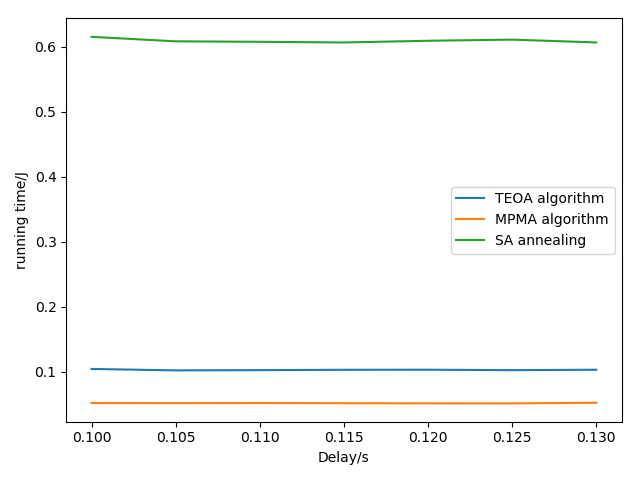
## 最小化最大能耗对问题

在这一节中，我们将研究MPM算法的性能。此外，我们与对比模型进行对比，与TEOA算法、模拟退火（simulated annealing, SA）算法进行对比。

首先，我们来比对随着时延的增长，最大能耗对的变化情况。由图4.3(a)、图4.3(b)可知，随着时延的增长，最大的能耗对在逐渐降低。这是因为当完成任务的时延增长时，在传输速率不变的情况下，传输能耗、传输时间也不会发生变化。因此，可用于任务计算的时间增加了。根据公式（2.7）、（2.11）、（2.17）等可知，随着任务计算时间的增长，设备可以以一个更低的CPU频率完成任务；接着，由公式（2.8）、（2.12）、（2.18）等可知，在其他因素不变的情况下，随着CPU频率的降低，完成任务的能耗也随之降低。因此，完成该任务的总能耗降低。所以，当所有任务的完成能耗均降低时，最大的能耗对也因此降低，这与图4.3(a)、图4.3(b)的曲线趋势一致。从图4.3(a)、图4.3(b)中可知，SA算法取得了最好的性能，接着是MPMA算法、TEOA算法。但由图4.3(c)、图4.3(d)可知，SA算法花费了最长的时间取得了最佳的性能，而虽然三个算法的性能接近，但MPMA算法在运行时间方面却拥有着比其他两个算法更低的运行时间。在最大能耗对最小化情况下，在Case 1时，对，MPMA、TEOA、SA算法所产生的最大能耗对比对比模型减少48.79%；但MPMA算法平均花费了0.0519s，而TEOA算法和SA算法分别花费了0.1031s和0.6093s；在Case 2时，三个算法均比对比模型减少了49.12%，而MPMA算法平均花费了0.0572s，但TEOA算法和SA算法分别花费了0.1160s和0.6838s。因此可以看出，虽然三个算法在性能方面无异，但在算法运行时间方面，MPMA算法比其他两个算法更高效。



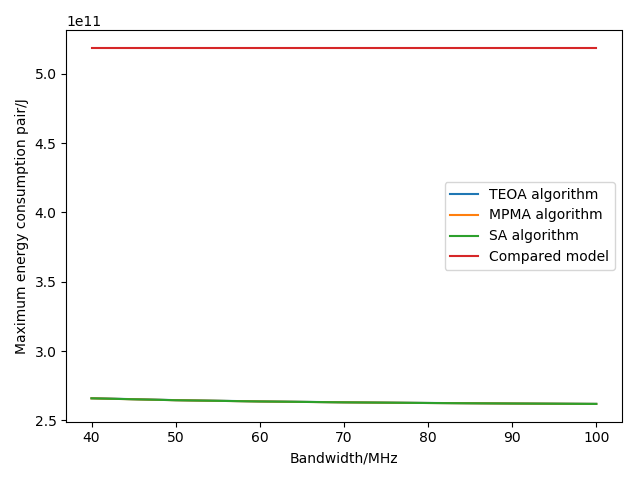
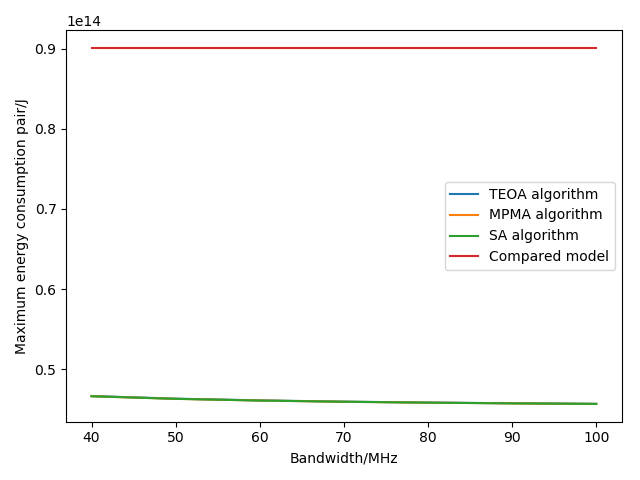
(a) Case 1下不同时延下最大能耗对情况； (b) Case 2下不同时延下最大能耗对情况；



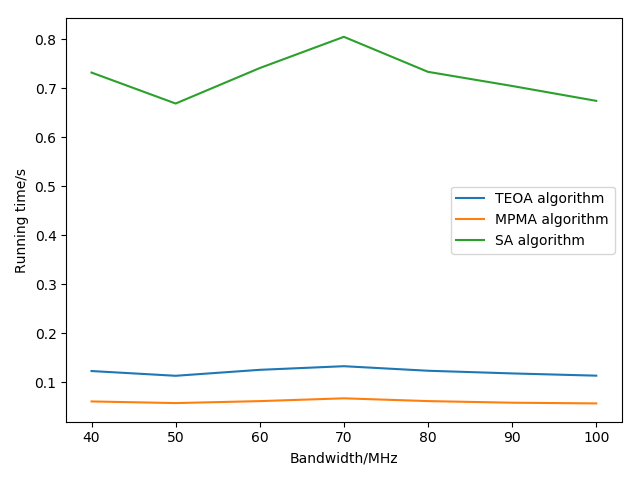
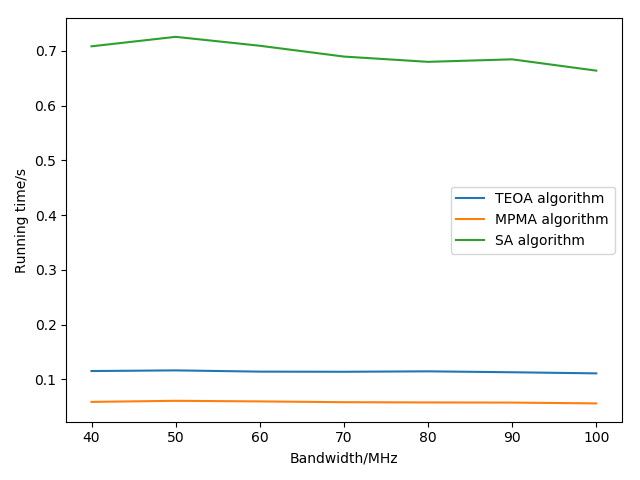
(c) Case 1下不同算法运行时间； (d) Case 2下不同算法运行时间；

图4.3 不同时延条件下不同算法运行结果

接着，我们来比较下不同的带宽条件对最大能耗对的影响。由图4.4(a)、图4.4(b)可知，随着带宽的增加，最大的能耗对在降低。根据公式（2.2）、（2.3）、（2.4）可知，带宽与通道容量成正比。随着带宽的增大，通道容量也在逐渐的增加。那么传输方面的能耗、时延则会相应的减小。因此，用于计算的时间则对应的增加，而由前面的分析可知，计算时间的增长有利于计算方面的能耗降低。因此，当所有的能耗对都降低时，最大的能耗对也相应降低。由图4.4(a)、图4.4(b)可知，TEOA算法、MPMA算法、SA算法获得了相似的性能，但是由图4.4(c)、图4.4(d)可知，在取得相似性能的情况下，在Case 1时，MPMA算法平均运行了0.0587s，而TEOA算法、SA算法分别运行了0.1142s、0.6942s。而在Case 2时，MPMA算法、TEOA算法和SA算法分别运行了0.0607s、0.1215s和0.7232s。可见，虽然算法效果一致，但在运行时间方面，MPMA算法表现性能更加突出。



(a) Case 1下不同带宽与最大能耗对的关系； (b) Case2下不同带宽与最大能耗对的关系；



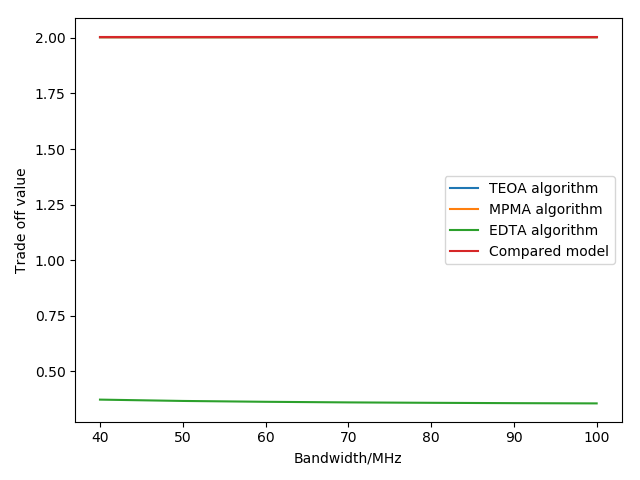
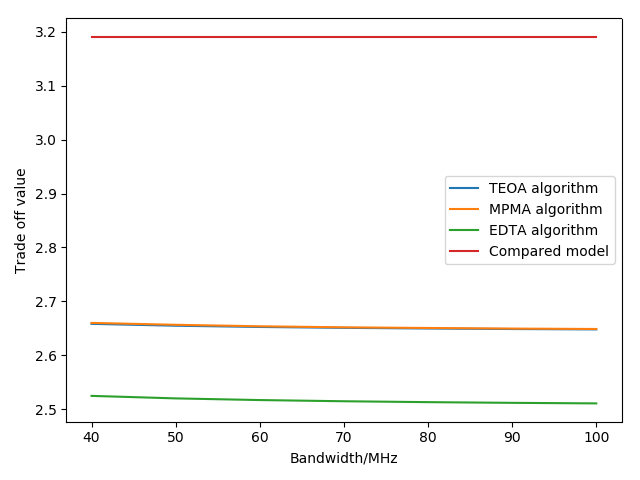
(c) Case 1情况下不同算法运行时间； (d) Case 2情况下不同算法运行时间；

图4.4 不同带宽与最大能耗对之间的关系

## 能耗与时延的权衡优化问题

在这一节中，我们将研究EDTA算法的性能，同时与TEOA算法和MPMA算法进行对比。此外，我们将对比模型作为基准线，对本文所提出的三个算法进行比较。

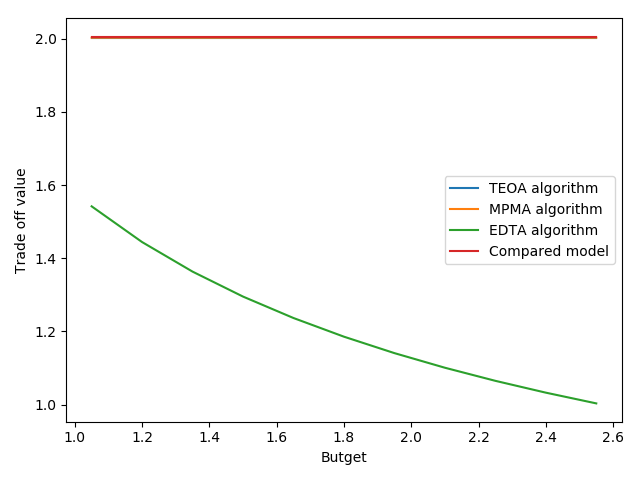
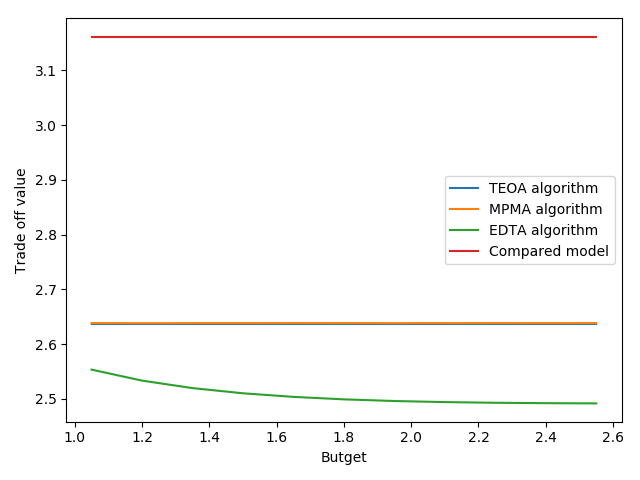
首先，我们来比对不同带宽情况下，权衡值的变化情况。在计算能耗不设置约束的条件下。由图4.5可知，随着带宽的增加，架构下能耗与时延之间的权衡值在下降。根据公式（2.2）、（2.3）、（2.4）可知，随着带宽的增加，通道容量在逐渐增加。因此，在其他条件不变的情况下，能耗会不断降低，进而导致权衡值的减小。而MPMA算法、TEOA算法是能耗方面的优化算法，因而在总延迟不变的情况下，它们会朝着能耗最小化的方向进行优化，所以权衡值可能有下降的趋势但变化不大。而EDTA算法则是朝着权衡值优化的方向进行优化，因此，其曲线的下降幅度会比MPMA算法和TEOA算法更加明显。其中，图4.5(a)是在Case 1下进行的，与对比模型相比，TEOA、MPMA、EDTA算法分别提升了16.90%、16.86%、21.16%的性能。在Case 2下，我们的实验结果展示在图4.5(b)。与对比模型相比，三个算法平均分别提升了0.12%、0.12%、81.85%的性能。



(a) Case 1； (b) Case 2；

图4.5 不同带宽情况下权衡值变化情况

接下来，我们比对不同的能耗约束下的权衡值。我们以任务在本地完成时的最小能耗作为预算。在忽略预算的情况下，由公式（3.6）—（3.9）可知，最优CPU频率可以被确定。而在考虑预算时，CPU频率的选择区间变小，因而无法达到最优的CPU频率的选择，因此，随着预算的增加，CPU频率的选择区间不断的变大，至此可选择的CPU频率越来越靠近最优的CPU频率，所以完成任务的时延能耗权衡值也在不断的下降，最后，总的权衡值也在不断的下降，直到不同设备可以用最佳的CPU频率进行工作，曲线开始趋于稳定。与对比模型相比，在Case 1时，由图4.6(a)可知，EDTA算法、TEOA算法和MPMA算法分别提升了20.67%、16.59%和16.54%；在Case 2时，由图4.6(b)可知，在性能方面，EDTA算法平均提升了39.20%，但TEOA算法、MPMA算法基本不提升。

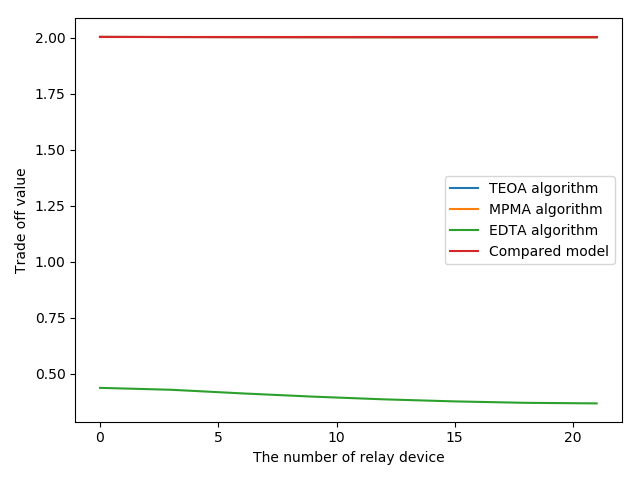
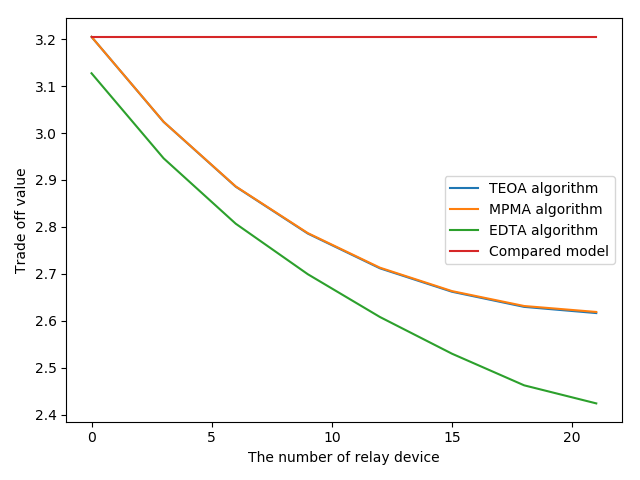


(a) Case 1； (b) Case 2；

图4.6

不同能耗预算下权衡值变化情况

最后，我们比对不同的中继设备数量对于权衡值的影响，随着中继设备数量的增加，越来越多的移动设备可以选择与中继设备进行协作从而降低该设备的权衡值。当架构中中继设备数量不足时，中继设备的数量是制对时延能耗的权衡优化关键。而随着中继设备数量的增多，该因素制约性逐渐降低，图4.7(a)和图4.7(b)正是描绘了这一点。由图4.7(a)可知，与对比模型相比，EDTA算法提升了15.74%，而TEOA算法、MPMA算法则分别提升了12.17%、12.13%；而在Case 2时，EDTA算法提升了80.24%，而其他两个算法在性能方面与对比模型类似。



(a) Case 1； (b) Case 2；

图4.7 不同中继设备数量对权衡值的影响

# 结论

本文提出了一种新的基于移动边缘计算的多源、多中继协同计算迁移结构，移动设备上含有一个待执行的任务；此外，为执行完成该任务，我们提出了多种任务计算方式。基于此，本文研究了该架构下的能耗优化问题以及时延能耗权衡优化问题。对于能耗优化问题，本文将其分为最小化总能耗和最小化最大能耗对两个子问题，其中第一个子问题，我们转化为最大二分图匹配问题，第二个子问题归类为整形线性规划，因此其属于NP-hard。基于此，本文分别提出了TEOA算法和MPMA算法进行解决。对于时延能耗权衡优化问题，我们将其归纳为一个多维背包问题，因此该问题属于NP-Hard。通过对目标方程式的转化，以及对条件的松弛，我们将其转变为一个二分图匹配问题。因此，我们提出一个基于KM的算法来解决该问题。此外，我们进行了大量的数值实验，大量实验结果表明本文所提出算法的有效性、鲁棒性。

# 参考文献

1. 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.
2. Kuang Z, Shi Y, Guo S, et al. Multi-User Offloading Game Strategy in OFDMA Mobile Cloud Computing System[J]. IEEE Transactions on Vehicular Technology, 2019, 68(12): 12190-12201.
3. Chen X , Pu L , Gao L , et al. Exploiting Massive D2D Collaboration for Energy-Efficient Mobile Edge Computing[J]. IEEE Wireless Communications, 2017, 24(4):64-71.
4. Li G, Wang J, Wu J, et al. Data processing delay optimization in mobile edge computing[J]. Wireless Communications and Mobile Computing, 2018, 2018:1-9.
5. 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.
6. Tao X, Ota K, Dong M, et al. Performance guaranteed computation offloading for mobile-edge cloud computing[J]. IEEE Wireless Communications Letters, 2017, 6(6): 774-777.
7. Mao Y, Zhang J, Letaief K B. Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems[C]//2017 IEEE wireless communications and networking conference (WCNC). IEEE, 2017: 1-6.
8. 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.
9. Zhang G, Zhang W, Cao Y, et al. Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices[J]. IEEE Transactions on Industrial Informatics, 2018, 14(10): 4642-4655.
10. Wang F, Xu J, Wang X, et al. Joint offloading and computing optimization in wireless powered mobile-edge computing systems[J]. IEEE Transactions on Wireless Communications, 2017, 17(3): 1784-1797.
11. Mao S, Leng S, Yang K, et al. Fair energy-efficient scheduling in wireless powered full-duplex mobile-edge computing systems[C]//GLOBECOM 2017-2017 IEEE Global Communications Conference. IEEE, 2017: 1-6.
12. Kuhn H W . The Hungarian method for the assignment problem[J]. 2005, 52(1):7-21.
13. Munkres J. Algorithms for the assignment and transportation problems[J]. Journal of the society for industrial and applied mathematics, 1957, 5(1): 32-38.
14. Ouyang T, Li R, Chen X, et al. Adaptive User-managed Service Placement for Mobile Edge Computing: An Online Learning Approach[C]//IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, 2019: 1468-1476.
15. Sun Y, Zhou S, Xu J. EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks[J]. IEEE Journal on Selected Areas in Communications, 2017, 35(11): 2637-2646.
16. Yu Y, Zhang J, Letaief K B. Joint subcarrier and CPU time allocation for mobile edge computing[C]//2016 IEEE Global Communications Conference (GLOBECOM). IEEE, 2016: 1-6.
17. Tareq M M K, Semiari O, Salehi M A, et al. Ultra reliable, low latency vehicle-to-infrastructure wireless communications with edge computing[C]//2018 IEEE Global Communications Conference (GLOBECOM). IEEE, 2018: 1-7.
18. Kuang Z, Shi Y, Guo S, et al. Multi-User Offloading Game Strategy in OFDMA Mobile Cloud Computing System[J]. IEEE Transactions on Vehicular Technology, 2019, 68(12): 12190-12201.
19. Chen L, Huang L, Xu H, et al. Optimal channel assignment schemes in underlay CRNs with multi-PU and multi-SU transmission pairs[C]//International Conference on Wireless Algorithms, Systems, and Applications. Springer, Cham, 2015: 29-39.
20. Xing H, Liu L, Xu J, et al. Joint task assignment and resource allocation for D2D-enabled mobile-edge computing[J]. IEEE Transactions On Communications, 2019, 67(6): 4193-4207.
21. Cao X, Wang F, Xu J, et al. Joint computation and communication cooperation for energy-efficient mobile edge computing[J]. IEEE Internet of Things Journal, 2018, 6(3): 4188-4200.
22. Guo S, Xiao B, Yang Y, et al. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing[C]//IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. IEEE, 2016: 1-9.
23. Chen X, Shi Q, Cai Y, et al. Joint cooperative computation and interactive communication for relay-assisted mobile edge computing[C]//2018 IEEE 88th Vehicular Technology Conference (VTC-Fall). IEEE, 2018: 1-5.
24. Rao L, Liu X, Ilic M D, et al. Distributed coordination of internet data centers under multiregional electricity markets[J]. Proceedings of the IEEE, 2011, 100(1): 269-282.
25. Fréville A. The multidimensional 0–1 knapsack problem: An overview[J]. European Journal of Operational Research, 2004, 155(1): 1-21.
26. Yao M, Chen L, Liu T, et al. Energy Efficient Cooperative Edge Computing with Multi-Source Multi-Relay Devices[C]//2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). IEEE, 2019: 865-870.
27. Bozorgchenani A, Tarchi D, Corazza G E. Centralized and distributed architectures for energy and delay efficient fog network-based edge computing services[J]. IEEE Transactions on Green Communications and Networking, 2018, 3(1): 250-263.
28. Huang M, Liu W, Wang T, et al. A Cloud-MEC Collaborative Task Offloading Scheme with Service Orchestration[J]. IEEE Internet of Things Journal, 2019: 1-1.
29. Rodrigues T G, Suto K, Nishiyama H, et al. Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration[J]. IEEE Transactions on Computers, 2018, 67(9): 1287-1300.
30. Yang T, Hu Y, Gursoy M C, et al. Deep reinforcement learning based resource allocation in low latency edge computing networks[C]//2018 15th International Symposium on Wireless Communication Systems (ISWCS). IEEE, 2018: 1-5.
31. Chen M H, Dong M, Liang B. Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints[J]. IEEE Transactions on Mobile Computing, 2018, 17(12): 2868-2881.
32. Park C B, Park B S, Uhm H J, et al. IEEE 802.15. 4 based service configuration mechanism for smartphone[J]. IEEE Transactions on Consumer Electronics, 2010, 56(3): 2004-2010.

# 致 谢

盛年不重来，一日难再晨。大学四年时光很快就过去了，作为一名广工学子，我很庆幸这四年在学校的成长、收获。

首先，我要感谢我的指导老师武继刚老师。在大二时，武老师指导我的大学生创新创业训练项目，最后顺利结项；在大四时，武老师指导我的毕业设计论文，老师的谆谆教诲、辛勤栽培，让我受益匪浅。老师给予的活跃性的指导，使我在学术道路上可以顺利前进。

其次，我要感谢陈龙副教授。他一直以扎实的理论基础、不厌其烦的教导感染我、鞭策我。不仅在我学术道路上，在日常生活中，师兄也一直如北极星般指引我正常的方向。师兄对待生活的努力态度也深深感染我，让我在成长的道路上一直以只争朝夕、不负韶华的态度积极进取。

感谢的话说不完，我要感谢我的家人，感谢你们让我在求学、求知的道路上没有后顾之忧。此外，我还要感谢我的班主任杨易扬老师，我的支部书记杨博书记，我的辅导员刁嘉程老师、杨博老师，我的同学们，感谢你们在我求学道路上的陪伴、鼓励。

此外，我非常感谢本论文涉及领域的学者们。正是因为你们在本领域的辛勤劳作，我才能找到自己的研究方向，从而在学术道路上继续前行下去。

最后，衷心感谢评审论文的老师们，感谢你们的辛勤劳动。本文如有错误，敬请斧正。