边缘计算与算力网络

文献调研报告

姓　名：张武轩

学　号：23031212148

日　期：2023.12.07

目　录

目　录 I
一、 DRL for Online Computation Offloading in Wireless Powered MEC Networks 1
（一） 研究背景 1
（二） 主要内容 1
（1） 研究内容一 无线供电 MEC 场景建模 2
（2） 研究内容二 在线算法设计 3
（三） 研究结论 4
（四） 存在问题/改进建议/启发 5
二、 Fairness Aware Task Offloading and Resource Allocation in Cooperative MEC 6
（一） 研究背景 6
（二） 主要内容 6
（1） 研究内容一 公平感知的任务卸载场景建模 7
（2） 研究内容二 分层算法 8
（三） 研究结论 9
（四） 存在问题/改进建议/启发 10
三、 Energy-Efficient Multi-Access Mobile Edge Computing With Secrecy Provisioning 11
（一） 研究背景 11
（二） 研究内容 11
（1） 具有保密配置的 MEC 场景建模 11
（2） 单 WD 场景的问题描述 13
（3） 问题的层结构 14
（4） 解决 TECM 问题提出的分层算法 15
（三） 研究结论 17
（四） 存在问题/改进建议/启发 17
四、 Cost-Efficient Server Configuration and Placement for Mobile Edge Computing 18
（一） 研究背景 18
（二） 研究内容 18
（三） 研究结论 18

（1） MEC 环境下 ES 配置和部署场景建模 18
（2） 基于排队论的问题建模 19
（3） 解决问题采用的算法 20
（四） 存在问题/改进建议/启发 20
五、 Mechanisms for Resource Allocation and Pricing in MEC Systems 21
（一） 研究背景 21
（二） 研究内容 21
（1） 基于拍卖理论的社会福利和用户效益建模 21
（2） 边缘（云-边）资源分配和定价问题（ERAP） 22
（3） 基于贪心的资源分配和定价策略（G-ERAP） 23
（4） 基于线性规划（LP）的逼近策略（APX-ERAP） 23
（三） 研究结论 24
（四） 存在问题/改进建议/启发 24
六、 Energy-Minimized Scheduling of IRT Tasks in a CPU-GPU CC Platform 25
（一） 研究背景 25
（二） 研究内容 25
（1） 间歇性实时任务下异构云平台场景建模 25
（2） 异构云平台的能耗建模 27
（3） 基于动态规划的算法设计 28
（三） 研究结论 29
（四） 存在问题/改进建议/启发 29
七、 EdgeDR An Online Mechanism Design for Demand Response in Edge Clouds 30
（一） 研究背景 30
（二） 研究内容 30
（1） 紧急需求响应(EDR)场景建模 30
（2） 社会成本建模 31
（3） 问题描述 32
（4） 问题分解 33
（5） 在线算法设计 34

（三） 研究结论 35
（四） 存在问题/改进建议/启发 35
参考文献： 36
课程意见及建议反馈 37

一、DRL for Online Computation Offloading in Wireless Powered MEC Networks

（一）研究背景

由于外形尺寸小和严格的生产成本限制，现代物联网(IoT)设备的电池寿命和计算能力往往受到限制。由于无线电力传输(WPT)技术的最新进展，无线设备(WD)的电池可以通过空中持续充电，而无需更换电池。同时，最近发展的移动边缘计算（MEC）技术可以有效增强设备计算能力。借助 MEC，无线设备可以将计算密集型任务卸载到附近的边缘服务器，以减少计算延迟和能耗。

（二）主要内容

无线供电的移动边缘计算（MEC）最近已成为增强低功耗网络（例如无线传感器网络和物联网（IoT））的数据处理能力的有前途的范例。文章考虑采用二进制卸载策略的无线供电 MEC 网络，无线设备（WD）的每个计算任务要么在本地执行，要么完全卸载到 MEC 服务器。文章给出一种在线算法，该算法可以最佳地适应任务卸载决策和无线资源分配以适应时变的无线信道条件。有别于传统算法，该算法需要在通道相干时间内快速解决硬组合优化问题。文章提出了一种基于深度强化学习的在线卸载（DROO）框架，该框架将深度神经网络实现为可扩展的解决方案，从经验中学习二进制卸载决策。它消除了解决组合优化问题的需要，从而大大降低了计算复杂度，特别是在大型网络中。为了进一步降低复杂性，文章提出了一种自适应过程，可以动态自动调整 DROO 算法的参数。数值结果表明，与现有优化方法相比，该算法可以实现接近最优的性能，同时将计算时间显着减少一个数量级以上。例如，在 30 个用户的网络中，DROO 的 CPU 执行延迟小于 0.1 秒，即使在快速衰落的环境中，也能真正实现实时、最佳的卸载。

（1）研究内容一 无线供电 MEC 场景建模

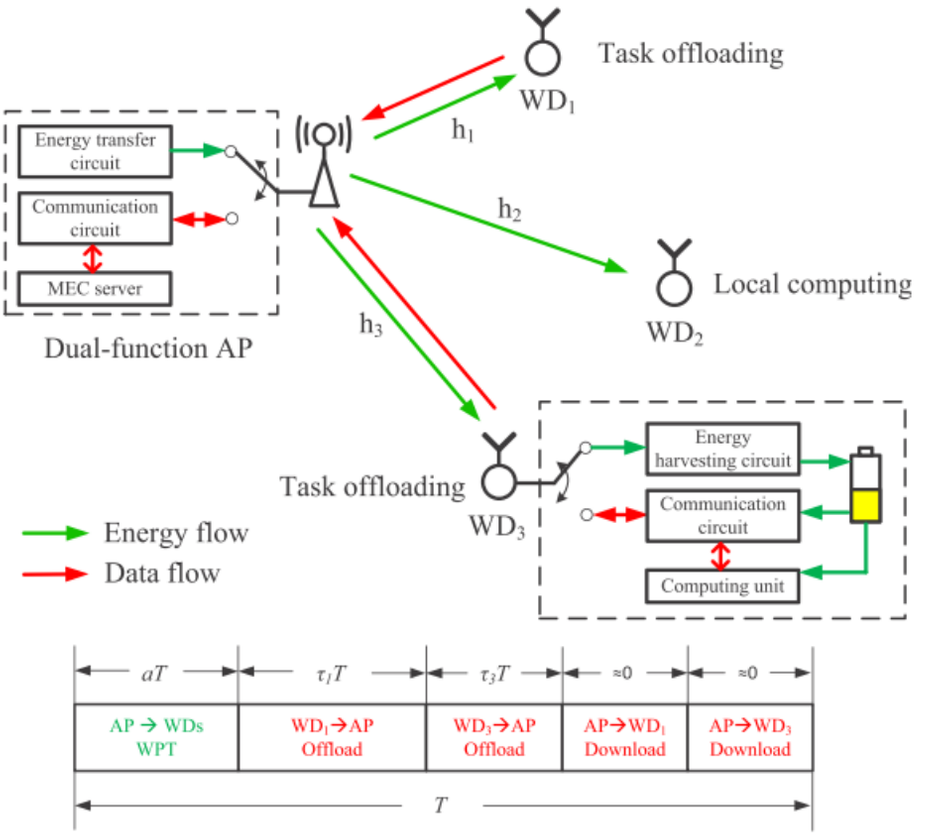

图 1.1 无线供电 MEC 场景描述图

基于无线充电技术（WPT），可接入点（AP）具有充电、提供计算资源两项基本功能。而无线设备（WD）的位置频繁改变引起信道增益 h 改变，在快衰竭环境下，要求算法具有较高的执行速度。

模型把系统时间划分为等长的连续时间帧T，其中T≤相干时间（信道保持恒定的最大时间差范围）。模型中的任务不可分割，采用二进制卸载策略。如果假设计算资源 AP≫WD，上行数据大小≫下载处理结果大小，则有：边缘计算时间≈0，下载处理结果时间≈0。

（1.1）卸载到边缘执行的最大计算速率

为了最大化计算速率，卸载时应耗尽其收集的能量$E_i = \mu P h_i T a$，即耗尽收集电量的最大卸载速率，则发射功率 $P_i = \frac{E_i}{\tau_i T}$，则有信噪比 $\frac{P_i h_i}{N_0} = \frac{E_i h_i}{\tau_i T N_0} = \frac{\mu P {h_i}^2 a}{\tau_i N_0}$，则有最大传输速

率$v = B\log_2\left(1+\frac{\mu P {h_i}^2 a}{\tau_i N_0}\right)$，最大卸载速率为$\frac{vT\tau_i}{T}$，即为

$$r_{O,i}(a,\tau_i) = B\log_2\left(1+\frac{\mu P {h_i}^2}{N_0}\times\frac{a}{\tau_i}\right)\tau_i$$

（1.2）本地最大计算速率

本地计算速率$r_{L,i} = \frac{f_i t_i}{T}$，而本地计算资源$f$受收集电量$E_i = \mu P h_i T a$的影响，即

$$k_i {f_i}^3 t_i \le E_i$$

其中k为计算能效系数。

当f取最大值时，须有t=T，不等式等号成立。解得最大值为$r_{L,i}(a) = (\mu P)^{\frac{1}{3}}\left(\frac{h_i}{k_i}\right)^{\frac{1}{3}} a^{\frac{1}{3}}$。

（1.3）问题建模

定义加权计算速率$Q(h,a,\tau,x) = \sum_{i=1}^{N} w_i\left((1-x_i)r_{L,i}(a) + x_i r_{O,i}(a,\tau_i)\right)$，问题建模如下

$$\begin{aligned} Q^*(\mathbf{h}) = \underset{\mathbf{x},\boldsymbol{\tau},a}{\text{maximize}} \quad & Q(\mathbf{h},\mathbf{x},\boldsymbol{\tau},a) \\ \text{subject to} \quad & \sum_{i=1}^{N}\tau_i + a \le 1, \\ & a \ge 0,\ \tau_i \ge 0,\ \forall i \in \mathcal{N}, \\ & x_i \in \{0,1\}. \end{aligned}$$

当给定卸载决策x后，子问题建模如下

$$\begin{aligned} Q^*(\mathbf{h},\mathbf{x}) = \underset{\boldsymbol{\tau},a}{\text{maximize}} \quad & Q(\mathbf{h},\mathbf{x},\boldsymbol{\tau},a) \\ \text{subject to} \quad & \sum_{i=1}^{N}\tau_i + a \le 1, \\ & a \ge 0,\ \tau_i \ge 0,\ \forall i \in \mathcal{N}. \end{aligned}$$

（2）研究内容二 在线算法设计

算法分为内外两层：外层 DRL 根据当前信道增益 h 求解卸载决策x，内层凸问题在得到x后通过一维二分搜索求解(a,τ)，如图 1.2、1.3 所示。

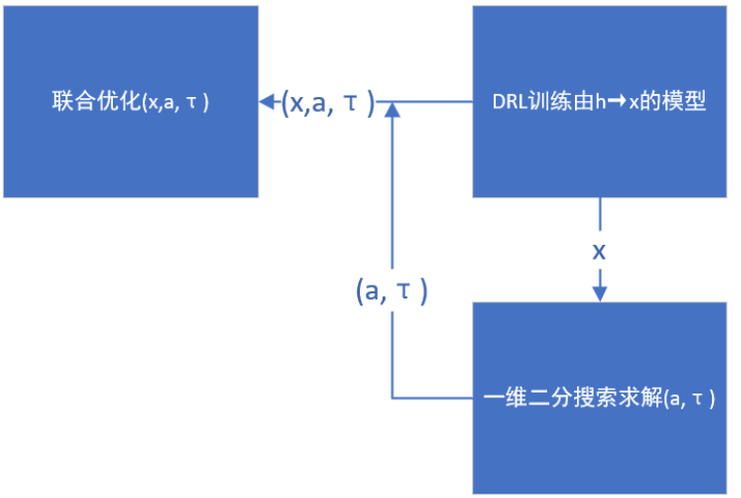

图 1.2 算法层级结构图

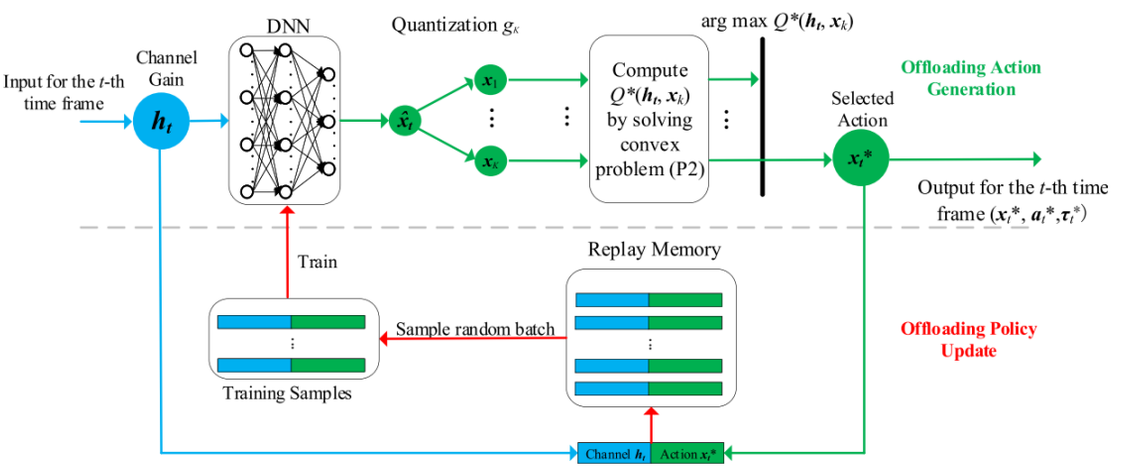

图 1.3 算法结构图

（2.1）外层深度强化学习算法（DRL）

h_t表示第t个时间帧的信道增益信息，由 DNN 网络得到松弛卸载决策$\hat{x}_t$，再由文章提出保序量化方法生成 K 个卸载决策x^1到x^k，其中文章提到的保序量化方法相比于 KNN 有更好的收敛性能。

（三）研究结论

文章提出了一种基于深度强化学习的在线卸载算法 DROO，以最大化具有二进制计算卸载的无线供电 MEC 网络中的加权和计算速率。该算法从过去的卸载经验中学习，通过强化学习改进 DNN 生成的卸载动作。设计了保序量化和自适应参数设置方法以实现算法的快速收敛。与传统的优化方法相比，所提出的 DROO 算法完全消除了解决硬混合整数规划问题的需要。仿真结果表明，DROO 实现了与现有基准测试方法类似的近乎最佳性能，但将

CPU 执行延迟降低了一个数量级以上，使得实时系统优化真正适用于衰落环境中的无线供电 MEC 网络。

（四）存在问题/改进建议/启发

文章对边缘计算中新颖的场景进行了数学建模，量化分析了 WPT 场景下的任务卸载问题。

给出了采用深度强化学习来解决复杂组合优化问题的思路，为我们在边缘计算环境中优化资源分配、提高系统性能提供了新的思考路径。深度强化学习的引入不仅能够处理实时变化的环境，还具备自适应性，使得系统能够更好地适应不同的工作负载和场景需求。

二、Fairness Aware Task Offloading and Resource Allocation in Cooperative MEC

（一）研究背景

随着 5G 的蓬勃发展，来自各个领域的大量计算密集型应用（例如增强现实和自动驾驶）预计将能够与实时响应无缝协作。然而，由于处理速度和存储有限，移动/物联网设备很难处理这些计算任务。为了应对这一挑战，传统的方法是将计算密集型任务卸载到配备丰富计算资源和大存储容量的远程云服务器上。然而，云服务器通常距离终端设备较远，在空间上远离终端设备，这导致不可接受的延迟和大量回程使用。为了解决这一矛盾，无线城域网 (WMAN) 中的移动边缘计算 (MEC) 被视为一项有前途的技术，使移动/物联网设备能够将计算密集型任务卸载到 MEC 服务器，它们配备有计算资源并且通常安装在基站（BS）中。

尽管 MEC 服务器具有较高的处理速度和较大的存储容量，但由于设备尺寸的限制，无法满足所有要求。通常，移动云计算（MCC）被视为与 MEC 协作的补充方式，这意味着当 MEC 服务器或终端设备上无法完成的任务将被卸载到云服务器上。然而，这种策略会带来两个缺点。一方面，将任务卸载到云服务器可能会导致较长的延迟，无法满足某些延迟敏感任务的时间要求。另一方面，由于不同服务器上的任务负载不均衡，一些边缘服务器可能会闲置，从而导致资源浪费。由于 BS 之间的通信成本相对较低，MEC 服务器之间的合理合作可以被认为是满足更多延迟敏感任务的有前途的方法，而这些任务是云服务器无法完成的，[9]。另外，边缘服务器之间的互助可以解决部分边缘服务器资源不足的问题，避免不同边缘服务器上的任务负载不均衡。也就是说，通过协作，MEC 层的服务器成为具有更强计算和通信能力的超级服务器，服务于时延敏感的任务。

（二）主要内容

目前，移动边缘计算（MEC）成为解决延迟敏感任务与资源有限的移动/物联网设备之间矛盾的新兴范例。然而，考虑到其有限的存储和计算能力，单个 MEC 服务器通常无法满足繁重的计算任务。因此，MEC 服务器的协作提供了解决这一问题的有效方法。文章研究了协作 MEC 服务器场景下的联合任务卸载和资源分配问题。首先从用户体验的角度定义物联网设备之间的资源公平性。然后，通过考虑系统效率和公平性来制定联合优化问题，该问题被证明是 NP 困难的。为了解决这个问题，文章提出了一种两层算法：上层算法受进化策略的启发，能够全局搜索优越的卸载方案；而底层算法考虑到所有任务之间的公平性，能够生成充分利用服务器资源的资源分配方案。

（1）研究内容一 公平感知的任务卸载场景建模

如图 2.1 所示，文章描述的场景中每个设备有不同的服务类型需求，每个边缘服务器部署了不同类型的服务，任务不能被本地执行，任务被提交到最近的边缘服务器，边缘服务器之间可以相互转发任务。

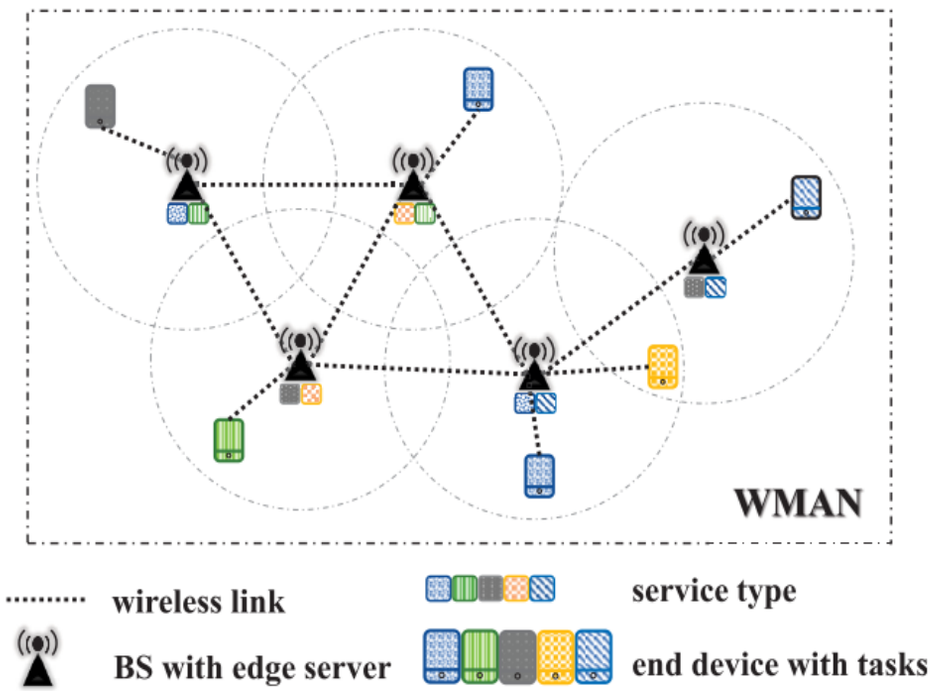

图 2.1 场景描述图

（1.1）任务的实际执行时间

如图 2.2 所示，任务的实际执行时间由任务的上传传播时间、上传传输时间、转发传播时间、转发传输时间、计算执行时间组成。从图中可以简单看出，任务的实际执行时间可以表示为传播时间总和与上传传输时间和转发传输时间较大值与计算时间之和，公式表示为：

$$d_i = \max\left\{\alpha_{i,j}\cdot\frac{DS_i}{r^{\uparrow}_{i,upl}},\quad \beta^i_{j,k}\cdot\frac{DS_i}{\min_{e_{j,k}\in\mathbb{E}^i_{upl,\mathrm{comp}}} r^{\leftrightarrow}_{j,k}}\right\} + \alpha_{i,j}\cdot l_{i,upl} + \beta^i_{j,k}\cdot\sum_{e_{j,k}\in\mathbb{E}^i_{upl,\mathrm{comp}}} l_{j,k} + \gamma_{i,k}\cdot\frac{C_i}{f_{\mathrm{comp},i}}$$

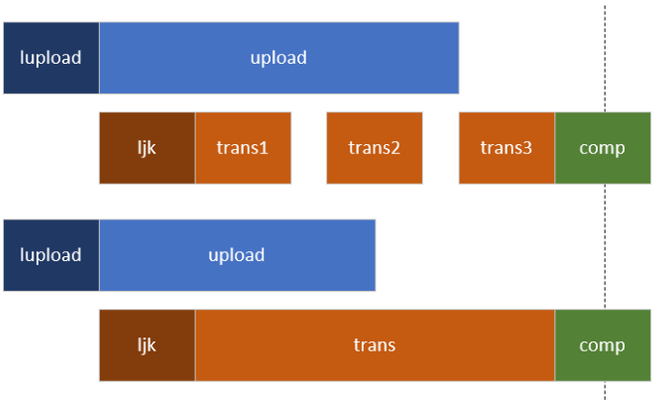

图 2.2 任务实际执行时间示意图

考虑到任务的公平性，定义辅助变量$G_i = \frac{d_i}{D_i}$，G_{max}为所有任务中执行最差（比例最大）的比例，问题建模如下：

$$
\begin{aligned}
\min_{\Gamma,\mathbb{F}} \quad & G_{\max} \\
\text{s.t.} \quad & \sum_{j=1}^{m} \alpha_{i,j} = 1 \quad T_i \in \mathbb{T}, M_j \in \mathbb{M} \\
& \beta_{j,k}^{i} \in \{0,1\} \quad T_i \in \mathbb{T}, M_j, M_k \in \mathbb{M} \\
& \sum_{k=1}^{m} \gamma_{i,k} = 1 \quad T_i \in \mathbb{T}, M_k \in \mathbb{M} \\
& \sum_{i=1}^{n} f_{\text{comp},i} \le F_{\text{comp}} \quad T_i \in \mathbb{T}, M_{\text{comp}}^{i} \in \mathbb{M} \\
& d_i \le D_i \quad T_i \in \mathbb{T} \\
& S_i \in \mathbb{S}_{\text{comp}} \quad T_i \in \mathbb{T}, M_{\text{comp}}^{i} \in \mathbb{M}.
\end{aligned}
$$

（2）研究内容二 分层算法

如图 2.3 所示，文章的算法设计分为两层，外层元启发式算法更新二元组(β,γ)，β表示任务的转发决策，γ表示任务的计算执行决策；内层求解资源分配策略f。

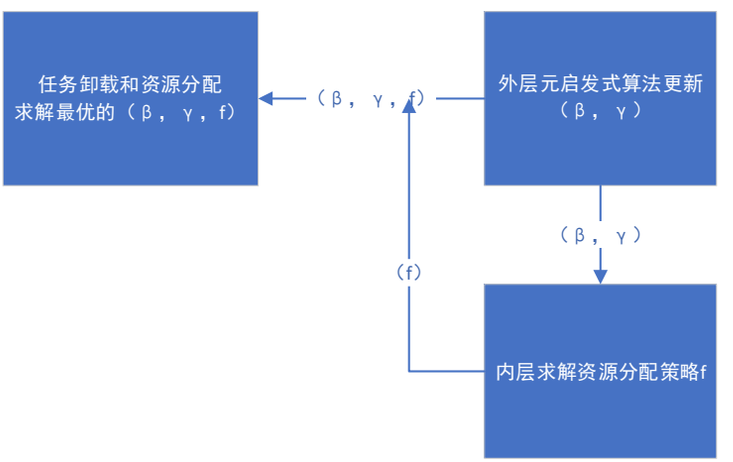

图 2.3 算法层级结构图

（2.1）内层求解资源分配策略f

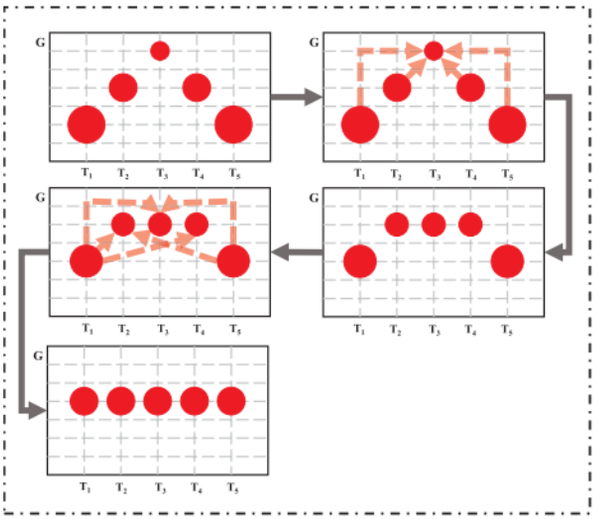

图 2.4 条件 1 示意图

如图 2.4，可以证明，当一个服务器上的所有任务的比值G_i都相同时， G_{max}最小，即

$$\frac{d_i}{D_i} = \theta_i + \frac{\eta_i}{f_{comp,i}} = g_{comp}$$

$$f_{comp,i} = \frac{\eta_i}{g_{comp} - \theta_i}$$

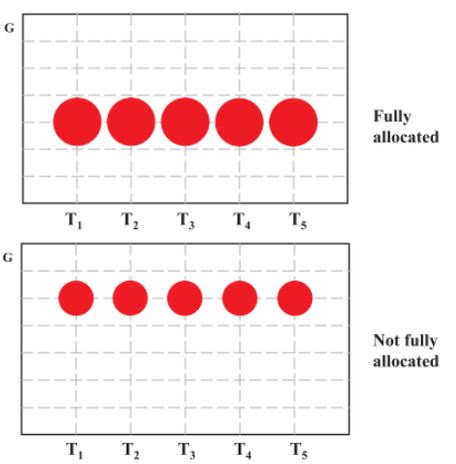

图 2.5 条件 2 示意图

如图 2.5，另外可以证明，当资源被完全利用时G_{max}最小

$$\sum f_{comp,i} = F_{comp}$$

（三）研究结论

文章研究了配备 MEC 服务器的 WMAN 的公平感知任务卸载和资源分配问题。文章

全面地表述了该问题，并表明所表述的问题是 NP 困难的。为了有效地解决该问题，文章提出了一种两级算法 APGTO+FGRA，其中 APGTO 可以搜索优越的卸载方案，而 FGRA 可以找到公平地充分利用服务器资源的资源分配方案。大量的仿真结果表明，APGTO+FGRA 能够给出相对公平、高效的任务卸载和资源分配方案，该方案考虑了服务质量最差的任务。

（四）存在问题/改进建议/启发

大部分文献往往研究整体的最优而忽略的个体的情况，例如使得整体的平均执行时间最短，但某个任务的执行情况可能很差。与其他任务卸载的论文相比，该文章设计的卸载策略考虑到了每个任务的公平性而不是追求全局最优。文章把优化目标新颖地设置为最小化最差任务的任务实际执行时间与截止时间的比例，巧妙地完成了对个体情况的照顾。

但在建模过程中，本文在出现了两个问题：1.建模中多余地考虑了α；2.考虑了相比较而言可以忽略的传播时间。

三、Energy-Efficient Multi-Access Mobile Edge Computing With Secrecy Provisioning

（一）研究背景

由于过去几十年异构无线接入网络（RAN）的广泛部署，多接入移动边缘计算的新兴范例，它允许移动终端通过多接入同时将计算工作负载卸载到多个不同的边缘计算服务器。RAN 为在未来无线系统中实现计算密集型移动互联网服务提供了一种有前景的方案。

MEC 的安全问题近年来也引起了广泛关注。特别是，由于无线电传输的广播性质，多接入 MEC 中的关键安全问题之一是从无线设备卸载的任务可能会被恶意节点窃听。例如，恶意节点可能故意从正在卸载任务的无线设备收集无线电信号，并以暴力方式解码任务数据。因此，积极卸载计算工作负载但没有保密规定可能会导致大量计算工作负载被窃听。如何利用多路访问计算卸载的特性来提高无线设备的卸载性能（例如，完成任务时的能耗），同时为卸载传输提供有保证的保密性，一直是一个关键问题。完成任务的保证延迟。这样的解决方案很大程度上取决于对计算卸载、保密配置和卸载传输的正确管理。

（二）研究内容

文章研究了具有保密配置的节能多路访问移动边缘计算。具体来说，首先研究一个无线设备（WD）的多路访问卸载受到恶意节点窃听的情况。通过表征 WD 在其卸载传输中基于保密的吞吐量，制定了 WD 多路访问计算卸载、保密配置和卸载传输持续时间的联合优化，目标是最大限度地减少 WD 的总能耗，同时提供有保证的保密中断。卸载并保证完成 WD 工作负载的总体延迟。尽管该联合优化问题具有非凸性，但利用其分层结构并提出了一种有效的算法来解决它。在单 WD 场景研究的基础上，进一步研究了多个 WD 的场景，其中一组 WD 顺序执行多路访问计算卸载，同时受到恶意节点的窃听。考虑到不同 WD 之间的耦合效应，我们提出了一种基于交换启发式的算法（使用提出的单 WD 算法作为子程序）来查找 WD 的顺序以执行多访问计算卸载，其目标是最大限度地减少所有 WD 的总能耗。

（1）具有保密配置的 MEC 场景建模

如图 3.1 所示，文章 MEC 场景中假定存在一个窃听者（Eavesdropper），无线设备在向边缘服务器卸载任务时，会因为窃听者占用部分信道而受到干扰。

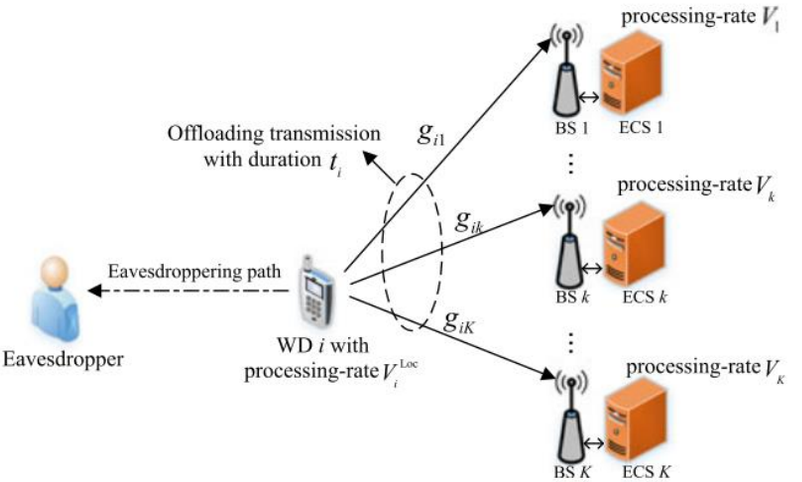

图 3.1 具有保密配置的 MEC 场景描述图

香农定理解释了在有限的带宽和特定信道条件下，可靠数据传输的极限。基于此得出安全吞吐量（secure throughput）可以表示为：

$$C_{ik}^{\mathrm{sec}}=\left(W_k\log_2\left(1+\frac{p_{ik}g_{ik}}{n_k}\right)-W_k\log_2\left(1+\frac{p_{ik}g_{ik\mathrm{E}}}{n_{k\mathrm{E}}}\right)\right)^{+}.$$

其中，p_{ik}表示发射功率，g_{ik}表示信道增益，n_k表示噪声功率，设定ε_i来表示中断概率（secrecy outage probability）。由于窃听者可能故意隐藏其位置，因此无法准确获得g_{ikE}的值，假设 g_{ikE}服从指数分布，均值等于 a_{iE}，可以表示为

$$\theta_{ik}=-\alpha_{i\mathrm{E}}\ln\left(1-\left(1-e^{-\frac{\hat{g}_{ik}}{\alpha_{i\mathrm{E}}}}\right)(1-\epsilon_i)\right).$$

因此，卸载速率x_{ik}可以表示为

$$x_{ik}=W_k\log_2\left(\frac{p_{ik}\hat{g}_{ik}+n_{k\mathrm{E}}}{p_{ik}\theta_{ik}+n_{k\mathrm{E}}}\right),$$

由此可以得到发射功率p_{ik}的表达式，该式包含变量s_i、 t_i和ε_i。

$$p_{ik}=\frac{n_{k\mathrm{E}}2^{\frac{s_{ik}}{t_iW_k(1-\epsilon_i)}}-n_{k\mathrm{E}}}{\hat{g}_{ik}-\theta_{ik}2^{\frac{s_{ik}}{t_iW_k(1-\epsilon_i)}}},$$

（2）单 WD 场景的问题描述

如图 3.2 所示，任务的总时延L_i为本地执行时间和边缘执行时间的最大值，表示为

$$L_i = \max\left\{ t_i + \max_{k\in\mathcal{K}}\{\frac{s_{ik}}{V_k}\}, \frac{S_i^{\mathrm{req}} - \sum_{k\in\mathcal{K}} s_{ik}}{V_i^{\mathrm{Loc}}} \right\},$$

显然，一个约束条件为总时延应当小于截止时间。

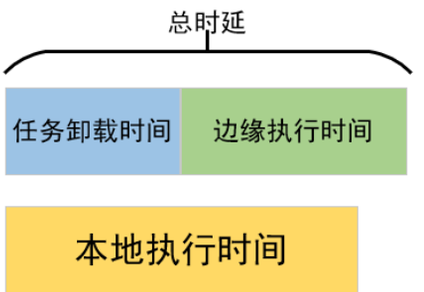

图 3.2 任务执行总时延示意图

文章以最小化无线设备的能耗为优化目标，联合优化计算卸载决策s_i、资源分配决策t_i和保密配置ε_i。如图 3.3 所示，无线设备的总能耗由本地执行能耗和卸载能耗组成。

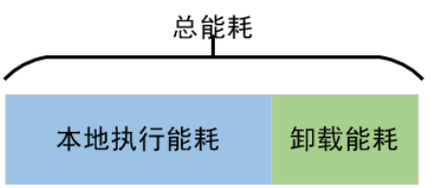

图 3.3 无线设备总能耗示意图

优化问题（TECM）建模如下

$$(\mathrm{TECM}): E_i^{\min} = \min \frac{S_i^{\mathrm{req}} - \sum_{k\in\mathcal{K}} s_{ik}}{V_i^{\mathrm{Loc}}} \rho_i^{\mathrm{Loc}} + t_i \sum_{k\in\mathcal{K}} p_{ik}$$

$$\text{subject to: } 0 \le s_{ik} \le S_i^{\mathrm{req}}, \forall k, \tag{7}$$

$$\sum_{k\in\mathcal{K}} s_{ik} \le S_i^{\mathrm{req}}, \tag{8}$$

$$0 \le \epsilon_i \le \epsilon_i^{\max}, \tag{9}$$

$$L_i \le T_i^{\max}, \tag{10}$$

$$\text{constraints } (4), (5), \text{ and } (6),$$

$$\text{variables: } \mathbf{s}_i, t_i \text{ and } \epsilon_i.$$

（3）问题的层结构

对于给定的二元组(t_i, ε_i)，子问题（TECM-Sub）是一个严格凸问题。于是有了 TECM 问题的以下分解，如图 3.4 所示。

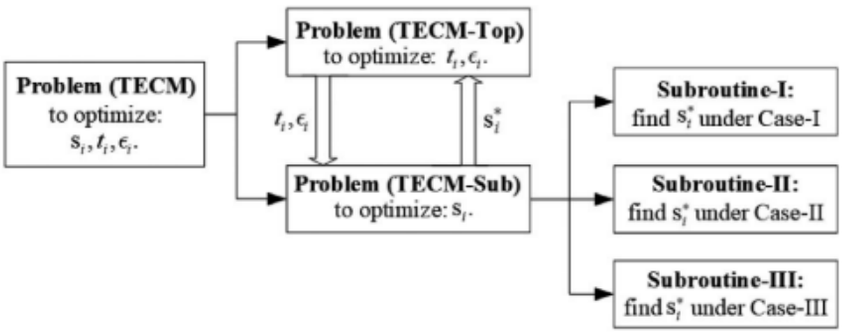

图 3.4 问题层级结构图

对于给定的(t_i,ε_i)元组，问题 (TECM) 变成优化变量s_i的子问题（TECM-Sub），如下所示：

$$
\begin{aligned}
(\text{TECM-Sub}): \quad E^{\min}_{i,(\epsilon_i,t_i)} &= \min \frac{S_i^{\text{req}} - \sum_{k\in\mathcal{K}} s_{ik}}{V_i^{\text{Loc}}} \rho_i^{\text{Loc}} \\
&\quad + t_i \sum_{k\in\mathcal{K}} \frac{n_{k\text{E}} 2^{\frac{s_{ik}}{t_i W_k (1-\epsilon_i)}} - n_{k\text{E}}}{\hat{g}_{ik} - \theta_{ik} 2^{\frac{s_{ik}}{t_i W_k (1-\epsilon_i)}}} \\
\text{subject to:} \quad & 0 \le s_{ik} \le s_{ik}^{\text{upp}}, \forall k, \qquad (11) \\
& S_i^{\text{req}} - T_i^{\max} V_i^{\text{Loc}} \le \sum_{k\in\mathcal{K}} s_{ik} \le S_i^{\text{req}}, \qquad (12) \\
\text{variables:} \quad & \mathbf{s}_i.
\end{aligned}
$$

约束(11) s_{ik}小于等于上界s_{ik}^{upp}，这个上界是问题总计算量、服务器在截止时间内的最大计算量和最大传输量中的最小值。在边缘服务器的计算量应当大于等于总计算量减去本地最大计算量的剩余计算量。

子问题（TECM-Sub）属于凸优化问题，于是能够使用 KKT 来解决问题，按以下方式构建拉格朗日函数

$$
\begin{aligned}
L(\mathbf{s}_i, \lambda, \mu) = & \sum_{k\in\mathcal{K}} t_i n_{k\text{E}} \frac{2^{\frac{s_{ik}}{t_i W_k (1-\epsilon_i)}} - 1}{\hat{g}_{ik} - \theta_{ik} 2^{\frac{s_{ik}}{t_i W_k (1-\epsilon_i)}}} \\
& + \frac{S_i^{\text{req}} - \sum_{k\in\mathcal{K}} s_{ik}}{V_i^{\text{Loc}}} \rho_i^{\text{Loc}} + \lambda \left(\sum_{k\in\mathcal{K}} s_{ik} - S_i^{\text{req}} \right) \\
& + \mu \left(S_i^{\text{req}} - \sum_{k\in\mathcal{K}} s_{ik} - T_i^{\max} V_i^{\text{Loc}} \right).
\end{aligned}
\qquad (16)
$$

在极值处有下式成立

$$\frac{n_{k\mathrm{E}} \ln 2(\hat{g}_{ik}-\theta_{ik})z_{ik}}{W_k(1-\epsilon_i)(\hat{g}_{ik}-\theta_{ik}z_{ik})^2}=\Gamma_i(\lambda,\mu),$$

其中，定义中间变量z_{ik}，和中间函数 Γ_i

$$z_{ik}=2^{\frac{s_{ik}}{t_i W_k(1-\epsilon_i)}},$$

$$\Gamma_i(\lambda,\mu)=\frac{\rho_i^{\mathrm{Loc}}}{V_i^{\mathrm{Loc}}}+\mu-\lambda,$$

对于一组给定的一组(λ,μ)，总有唯一的根z_{ik}，于是最优解s_i可以表示为

$$s_{ik}^{\mathrm{root}}=(1-\epsilon_i)t_i W_k \log_2(z_{ik}^{\mathrm{root}}).$$

另外可以证明，对于给定的一组(t_i,ε_i)，s_i的值随 λ 减少，随 μ 增加，在最优状态下，λ 和 μ 不能同时为正

对于给定的一组(λ,μ)，最优的s_i存在三种情况，分别是：任务全部卸载到边缘执行，μ=0；截止时间内本地和边缘一起执行，即本地执行达到最大量，剩余未完成的部分由边缘执行，λ=0；除上述外的情况，位于两个约束内部，λ 和 μ 同时=0。

如图 3.4 所示，对于以上三种情况，分别对应三个子程序。

（4）解决 TECM 问题提出的分层算法

最底层的三个子程序分别对应上述的三种情况：对于情况 1，文中采用二分查找来找到使得任务全部卸载到边缘情况下的 λ 值；对于情况 2，采用与情况 1 相同的方法找到 μ 值；对于情况 3，设置 μ 和 λ=0，并使s_i处于约束范围内。

如图 3.5 所示，子问题算法（TECM-sub）执行流程为：

1.如果s_i达到上界也无法在截至时间内完成除本地执行外的剩余部分，说明问题不可行，跳至 5；

2.如果s_i达到上界也无法在截止时间内完全卸载到边缘执行完，则仅比较情况 2、3 的结果，跳至 4；

3.比较三种情况给出的结果；

4.选择能耗最低的s_i；

5.算法结束。

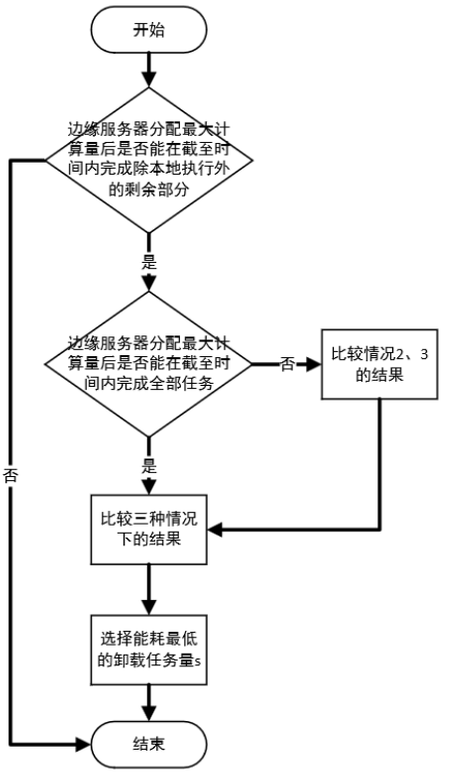

图 3.5 子问题算法流程图

t_i, ε_i 都落在各自给定的区间内，与其他参数无关。所以可以采用二维线性搜索方法来解决顶层问题（TECM-Top）。顶层问题算法执行流程如图 3.6 所示。

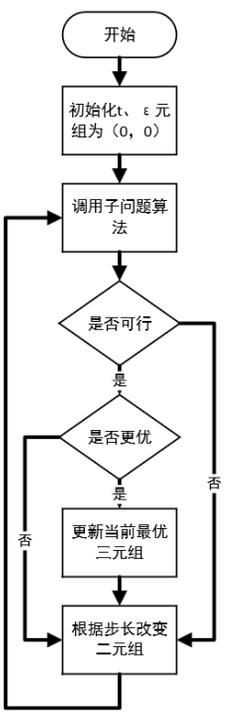

图 3.6 顶层问题算法流程图

（三）研究结论

文章研究了具有保密配置的节能多路 MEC。针对单 WD 场景，文章制定了 WD 的多路计算卸载、保密配置和卸载传输持续时间的联合优化，目标是最小化 WD 完成工作负载的总能耗，同时提供保证保密规定以及保证完成所需工作负载的总体延迟。尽管该联合优化问题具有非凸性，但利用其分层结构并提出了一种有效的算法来解决它。以对单 WD 场景的研究为基础，进一步研究了受恶意节点窃听的多 WD 场景。考虑到不同 WD 之间的耦合效应，提出了一种有效的算法来查找执行多路访问计算卸载时 WD 的排序，其目标是最小化所有 WD 的总能耗。已经提供了大量的数值结果来验证我们提出的算法的有效性和效率。结果表明，文章所提出的算法在节省能耗方面具有优势，同时在完成任务时提供有保证的保密性和有保证的延迟。

（四）存在问题/改进建议/启发

该文章考虑了移动边缘计算中的安全问题，以通俗的 MEC 环境的数学模型为基础，通过引入一个保密配置参数，以极小的成本，巧妙完成了对具有保密配置的 MEC 环境的建模。

四、Cost-Efficient Server Configuration and Placement for Mobile Edge Computing

（一）研究背景

构建 MEC 平台的第一步是部署边缘服务器（ES），这就带来了 ES 的选址问题。然而，该问题与传统的设施选址问题不同，因为 ES 的放置方案与 ES 的资源配置方案耦合在一起，两者都会影响运营支出（OPEX）和系统性能。因此，在构建 MEC 系统之前需要解决 ES 配置和放置的联合优化问题。ES 的高密度部署和过多的计算资源配置将直接降低移动网络运营商（MNO）的投资回报率（ROI），而部署密度和资源配置不足会降低服务质量（QoS）和体验质量（QoE）。一般来说，将计算资源集中在几个大型边缘节点可以帮助 MNO 节省运营成本，但这是以 QoS 和 QoE 潜在下降为代价的。不仅需要保证 MEC 平台的 QoS 和 QoE，还需要控制系统的 OPEX，这对这一问题的解决提出了重大挑战。

（二）研究内容

边缘服务器（ES）的计算资源配置和选址是构建移动边缘计算（MEC）平台的两个关键步骤。文章研究了 MEC 环境中 ES 配置和放置的联合优化问题。首先，将每个 ES 视为 M/G/m 排队模型，并建立数学模型来表征 MEC 环境，以便可以分析计算系统的性能和运营支出（OPEX）。然后，文章设计了一种两阶段方法，并开发了一系列基于二分算法和遗传算法（GA）的算法来获得 ES 的最优配置方案和次优放置方案（包括部署数量），目标是最大限度地降低运营成本，同时将系统性能保持在预定水平。最后，文章基于上海电信提供的真实基站数据集进行实验，以证明所提算法的有效性。这项工作是 MEC 环境中 ES 配置和放置联合优化问题的首次研究，其主要目标是提高成本效率

（三）研究结论

文章提到了 MEC 中服务器配置和放置优化的重要性。回顾了现有的相关研究并总结了这些研究的缺陷。本文从 MNO 的角度，研究了 MEC 环境中 ES 配置和放置的联合优化问题，其主要目标是提高成本效率。考虑到放置方案对配置方案的依赖性，文章设计了两阶段方法并开发了一系列算法来获得 ES 的最佳部署数量和位置，以及每个 ES 的最佳处理器数量和速度。文章提出的算法算法只需要在系统部署前使用一次，当环境发生重大变化时可以重复使用。实验基于上海电信提供的真实基站数据集进行，结果证明我们提出的算法是有效的。本文的研究成果为 MEC 平台的建立提供了理论和实践见解。

（1）MEC 环境下 ES 配置和部署场景建模

如图 4.1 所示，场景中存在多个通讯基站，在基站附近可以部署协同工作的边缘服务器，

协同工作的基站和边缘服务器之间有线连接，基站之间通过城域网（MAN）有线连接，无线设备可以通过与基站建立无线连接，再通过基站间的有限转发或直接访问协同工作的边缘服务器上的服务。

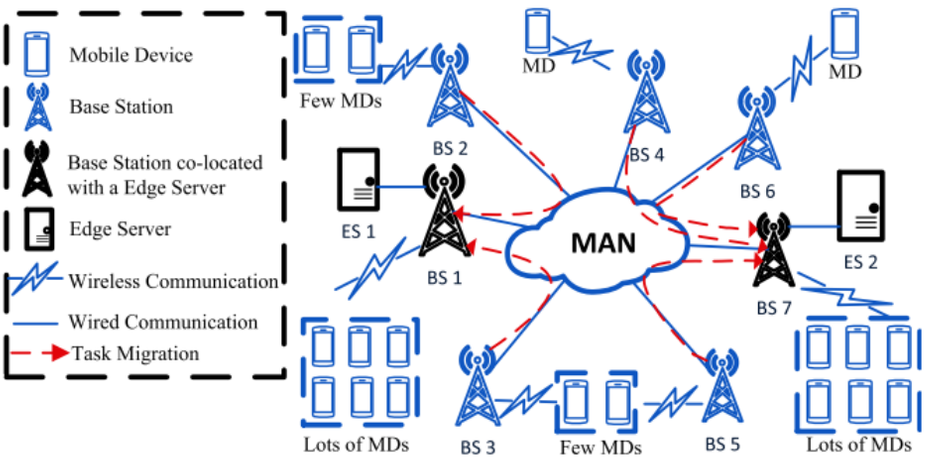

图 4.1 场景描述图

文章假定基站(BS)b_j的任务到达率λ_i为定值，任务到达率可以通过对时间序列进行预测。每个基站可以被定义为其位置和任务到达率组成的二元组，边缘服务器除了要确定位置信息和计算得到的任务到达率（协同工作基站接收的任务请求与转发到协同基站的任务请求），还要包含服务器的配置信息（包括 CPU 核心数和 CPU 频率），基站b_j、边缘服务器s_i、边缘服务器的任务到达率的定义如下：

$$b_j \triangleq (l_j, \lambda_j),\ 1 \le j \le n,\ l_j \in L,$$

$$s_i \triangleq (\hat{l}_i, m_i, f_i, \hat{\lambda}_i),\ 1 \le i \le k,\ \hat{l}_i = l_{h(i)} \in L,$$

$$\hat{\lambda}_i = \lambda_{h(i)} + \sum\nolimits_{l_v \in ne(i)} \lambda_v,$$

$$\begin{cases} l_v \in L - \hat{L}, \\ d(l_v, \hat{l}_i) < d(l_v, \hat{l}_w),\ \forall \hat{l}_w \in \hat{L}\ and\ \hat{l}_w \ne \hat{l}_i. \end{cases}$$

（2）基于排队论的问题建模

根据排队论，可以将系统的平均响应时间表示如下：

$$\overline{T} = \sum_{i=1}^{k} \frac{\hat{\lambda}_i}{\lambda} (\overline{t_i} + \overline{w_i}),$$

其中s_i的平均任务执行时间可以表示为：

$$\overline{t_i} = \frac{\lambda_{h(i)}}{\hat{\lambda}_i} \left(\frac{\overline{r_i}}{f_i} + \frac{\overline{d_i}}{\overline{\dot{c}_i}} \right) + \frac{\sum_{l_v \in ne(i)} \lambda_v}{\hat{\lambda}_i} \left(\frac{\overline{r_i}}{f_i} + \frac{\overline{d_i}}{\overline{\dot{c}_i}} + \frac{\overline{d_i}}{\overline{\ddot{c}_i}} \right),$$

s_i的服务强度（利用率）可以表示为

$$\rho_i = (\hat{\lambda}_i \overline{t_i}) / \breve{m}_i$$

MEC 平台的运营成本（OPEX）由站点租金和能源消耗成本组成。其中站点租金可以表示为：

$$C_s = \kappa \sum_{j=1}^{n} u_j c_j.$$

其中，c_j为边缘服务器的租金，由边缘服务器放置位置的房屋平均租金决定，为了使模型计算的站点租金更加准确，文章的作者爬取了恋家网站上海地区的房屋租赁信息，然后利用百度地图提供的坐标拾取系统来获取这些出租房屋的位置。。

基于 DVFS 技术，能源消耗成本可以表示为：

$$P_i = m_i(\rho_i \xi {f_i}^{\alpha} + P^*),$$

因此 OPEX 可以表示为：

$$C = C_s + C_p = \kappa \sum_{j=1}^{n} u_j c_j + \kappa \upsilon C_e \sum_{i=1}^{k} (m_i(\rho_i \xi {f_i}^{\alpha} + P^*)).$$

（3）解决问题采用的算法

将考虑成本的边缘服务器(ES)资源分配和部署问题表述为组合优化问题后，文章设计了基于二分法和遗传算法(GA)的算法，使系统在达到预定标准的前提下 OPEX 最小化。

（四）存在问题/改进建议/启发

文章对 MEC 环境中平台性能（平均响应时间）和运营成本（OPEX）建模，据文章作者所述，本问是对 MEC 环境中 ES 配置和放置联合优化问题的首次研究，对问题的部分数学建模具有一定开创性，包括作者根据排队论对模型中的各项关注的指标进行数学建模。

五、Mechanisms for Resource Allocation and Pricing in MEC Systems

（一）研究背景

近年来，诸多研究集中于在 MEC 系统中设计有效的资源分配算法，但服务货币化，即为移动用户和边缘提供商制定激励方案，仍然是 MEC 系统开发中的重大挑战。美国国家科学基金会（NSF）关于边缘计算重大挑战的报告将激励和货币化确定为边缘计算系统开发的五个重大挑战之一。 MEC 服务器的分散分布、资源需求的异构性以及用户之间为获取高质量服务而进行的竞争使得 MEC 系统中的资源分配和定价成为一个具有挑战性的问题。许多为 MCC 系统开发的拍卖机制并不直接适用于 MEC 设置。在 MEC 系统中，资源分布在多个级别上，用户通常请求多种类型的资源捆绑，并且他们对不同网络级别（即云和边缘）提供的服务有不同的评估。

（二）研究内容

文章解决了移动边缘计算（MEC）系统中的资源分配和货币化挑战，其中用户具有异构需求并争夺高质量服务。文章将边缘资源分配问题 (ERAP) 表述为混合整数线性规划 (MILP)，并证明 ERAP 是 NP 困难的。为了有效地解决问题，本文提出了两种资源分配机制。首先，开发了一种基于拍卖的机制，并证明所提出的机制是个体理性的，并且产生无嫉妒的分配。文章还提出了一种基于 LP 的近似机制，它不能保证无嫉妒，但它提供的解决方案保证与最优解决方案在给定距离内。通过对各种大小的 ERAP 实例进行广泛的实验分析来评估所提出机制的性能。使用商业求解器求解 MILP 模型得到的最优解作为基准来评估解的质量。分析表明，所提出的机制在合理的时间内为问题的相当大的实例获得了接近最优的解决方案。

（1）基于拍卖理论的社会福利和用户效益建模

文章将社会福利定义为中标用户的投标总和，可以表示为：

$$V = \sum_{i=1}^{n} \sum_{l=1}^{2} \alpha_l \cdot b_i \cdot x_{il}$$

其中，b_i表示用户 i 的投标价格，α_l表示用户资源分配在 l 层时的价值系数，l=1 表示边缘层，l=1 表示云层，若$\alpha_1 > \alpha_2$，则表示用户更倾向于在边缘获得服务。

用户估值可以表示为：

$$v_i(a_i) = \begin{cases} \alpha_1 b_i & \text{if } r_i \preceq a_i^1 \\ \alpha_2 b_i & \text{if } r_i \preceq a_i^2 \text{ and } r_i \not\preceq a_i^1 \\ 0 & \text{otherwise} \end{cases},$$

其中，a_i^l表示用户 i 通过分配机制在 l 层分配得到的资源数量。

因此用户效益即可表示为：

$$u_i = v_i - p_i$$

其中p_i表示用户的实际支出，用户支出的实际费用将根据用户分配得到的资源计算，表示为：

$$p_i = \sum_{j=1}^{m} \sum_{k=1}^{d} r_{ij} \cdot q_{jk} \cdot \pi_i \cdot w_k.$$

平台中不同的种类的资源被最小可划分单元表示，平台将配置不同种类的资源打包成虚拟机实例，如图 5.1 所示。

TABLE 1
Types of VM Instances Used in the Experiments

Type	vCPU	Memory (GB)	SSD Storage (GB)
medium	1	3.75 (1 unit)	1×4 (1 unit)
Large	2	7.5 (2 units)	1×32 (8 units)
Xlarge	4	15 (4 units)	2×40 (20 units)
2xlarge	8	30 (8 units)	2×80 (40 units)

图 5.1 不同种类的虚拟机实例资源分配图

（2）边缘（云-边）资源分配和定价问题（ERAP）

优化目标为最大化社会福利（用户估值总和最高/把资源分配给最看重该资源的用户），问题被建模为：

$$\text{maximize} \quad V = \sum_{i=1}^{n} \sum_{l=1}^{2} \alpha_l \cdot b_i \cdot x_{il}$$

$$\text{subject to:} \sum_{i=1}^{n} \sum_{j=1}^{m} r_{ij} \cdot q_{jk} \cdot x_{il} \leq C_{kl} \quad \forall k, \forall l$$

$$\sum_{l=1}^{2} x_{il} \leq 1 \quad \forall i$$

输出决策为由用户i的二值分配策略和用户i的基本价格组成的二元组。

（3）基于贪心的资源分配和定价策略（G-ERAP）

首先根据用户的出家密度B_i（单位资源的平均出价）对它们进行优先级排序，其中B_i可以表示为：

$$B_i = \frac{b_i}{\sum_{j=1}^{m}\sum_{k=1}^{d} r_{ij}\cdot q_{jk}\cdot w_k}.$$

由此确定边缘层的基本价格π_i，表示为：

$$\pi_i = \alpha_2 B^* + \frac{(\alpha_1-\alpha_2)}{2}(B_u + B_{u+1}).$$

云层的基本价格为：

$$\pi_i = \alpha_2 B^*,$$

$$B^* = \begin{cases} B_{u'+1} & \text{if } u' < n \\ B_{u'} - \epsilon & \text{otherwise} \end{cases},$$

（4）基于线性规划（LP）的逼近策略（APX-ERAP）

文章设计了一种基于 LP 的近似机制 APX-ERAP，用于求解 ERAP。APX-ERAP 是 *d* 维背包问题 MDKP 的近似算法的扩展。

ERAP 可以看作是由两个背包（两层资源）组成的加权多维多重背包问题。每个背包有 *d* 个维度（*d* 种资源），每个维度 *k* 的容量有限。有 *n* 个物品（*n* 个用户）要分配给背包。目标是最大化总利润。原始算法首先解决仅考虑一个 *d* 维背包的 MDKP 的 LP 松弛。LP 松弛解包含一组完全分配的项目 I 和一组至多 *d* 个分数分配的项目 F。

然后，算法考虑背包的两个可行解决方案：将 I 中的所有项目分配给背包或将 F 中最值钱的物品放入背包。ERAP-MILP 的 LP 松弛解包含一组边缘级别完全分配的用户 I1、一组云级别完全分配的用户 I2、一组至多 d 个边缘级别部分分配的项目 F1，以及云级别的一组最多 d 个分数分配的项目 F2。然后，根据这些集合，算法确定边缘级别和云级别最有价值的分配。

一次分配的例子如图 5.2 所示。

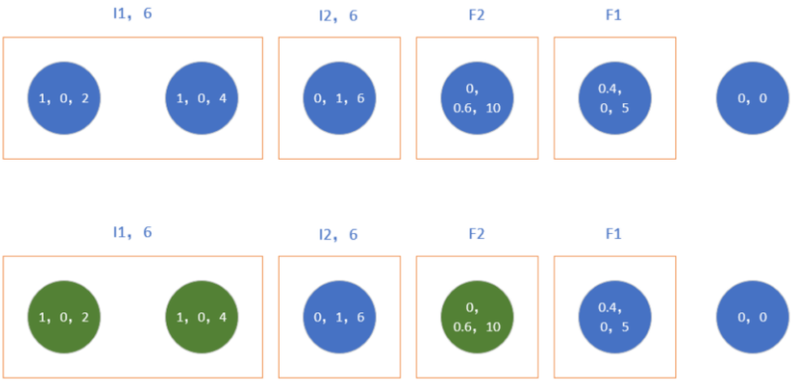

图 5.2 APX-ERAP 机制举例

（三）研究结论

文章提出了边缘计算系统中的两种资源分配和定价机制，其中用户具有异构请求并争夺高质量服务。通过进行广泛的实验分析证明了所提出机制的特性并评估了其效率。对于小型实例，将所提出的机制获得的解决方案与 CPLEX 求解器获得的最优解决方案在执行时间、服务用户百分比、社会福利和收入方面进行了比较。对于大型实例，将两种提出的机制的性能与用于分析小型实例的相同指标进行了比较。实验结果表明，通过所提出的机制获得的资源分配产生了接近最优的解决方案。除了解决方案的质量之外，较短的执行时间使得所提出的机制有望在边缘计算系统中部署。

（四）存在问题/改进建议/启发

文章针对 MEC 中(云-边两层)的资源分配问题，考虑了异构服务器和多种资源类型，对社会福利和用户效用进行建模。文章以社会福利和用户效用的双重目标，提出了一种具有前瞻性的计算资源动态配置方法。这种方法在资源分配和定价机制上的创新性为解决边缘计算中的资源管理问题提供了新的思路。

六、Energy-Minimized Scheduling of IRT Tasks in a CPU-GPU CC Platform

（一）研究背景

从用户或消费者的角度来看，他们希望通过云计算获得高质量的服务。服务延迟是服务质量的一个基本方面。用户提交的计算任务通常需要足够快的处理速度，以满足用户的期望，并留下计算资源来处理其他任务。通常有两种方法可以提高系统的服务质量。一方面，可以在云计算系统中部署更大的计算能力。目前，多核 CPU+多 GPU 的异构计算模式已广泛应用于云计算服务器中，多核 CPU 可同时运行多种任务，GPU 可有效加速 AI 应用的执行。另一方面，如果计算资源非常有限，则需要良好的调度算法来决定计算任务分配和资源分配，以保证系统的性能。

（二）研究内容

由于云计算服务的灵活性、可用性和可扩展性，越来越多的用户通过云计算技术寻求解决方案。云计算平台往往由大量的基础设施组成，其能耗是一个大问题。文章研究如何在处理一些间歇性实时任务时最小化云计算平台的能耗。与之前将用户提交的任务抽象为单个计算作业并使用 CPU 进行处理的工作不同，本工作提出使用 CPU 和 GPU 来处理不规则间隔的间歇性实时任务，并且其发布的计算作业必须在规定的时间限制内完成。能耗最小化问题被表述为一个整数非线性规划问题，需要决定任务分配计划和具体的资源分配计划。为了有效地解决这个问题，文章定义了一个代表给定资源量的给定任务集的最优解决方案的状态，以及代表状态值的值函数。通过这种方式，推导了状态转移方程并开发了动态规划方法来解决该问题。该方法还可以扩展到处理到达时间是动态且不可预测的任务。实验表明，该算法能够有效降低能耗，同时与其他贪心方法相比，其计算时间也相当短。

（1）间歇性实时任务下异构云平台场景建模

如图 6.1 所示，移动设备通过接入点连接到边缘服务器，从设备卸载的任务由边缘服务器或云服务器处理。由于移动设备始终处于运动状态，因此它们可能会时不时地与互联网断开连接。因此，它们的卸载任务并不是连续存在的，而是以不规则的间隔出现。与之前的研究假设计算任务连续地留在系统上不同，本研究通过假设它们是不规则间隔发生的间歇性实时任务来考虑这种现象。

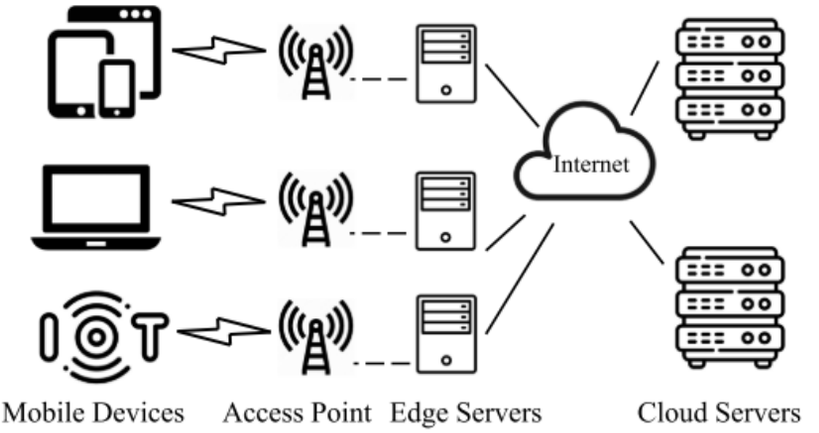

图 6.1 场景说明图

如图 6.2 所示，云-边-端平台将服务器中的 CPU 计算资源划分为相同的虚拟 CPU（vCPU），而 GPU 的最小划分则对应实际的物理 GPU，GPU 资源在虚拟机之间相互隔离，即虚拟机之间无法共享使用同一个物理 GPU。

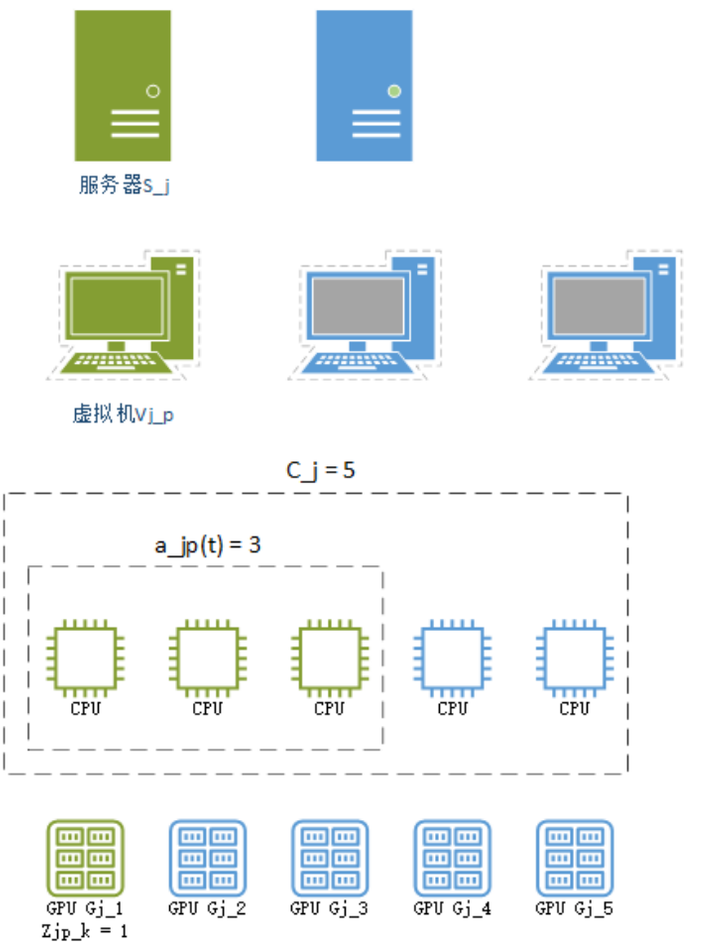

图 6.2 资源分配示意图

如图 6.3 所示，由于移动设备始终处于运动状态，偶尔发生断连，因此卸载任务并不是

连续存在，而是以不规则的间隔出现，文章将每个间歇性实时任务（IRT）划分为多个任务片段，并将每个片段看作独立的任务，根据任务片段的到来时间重新映射。

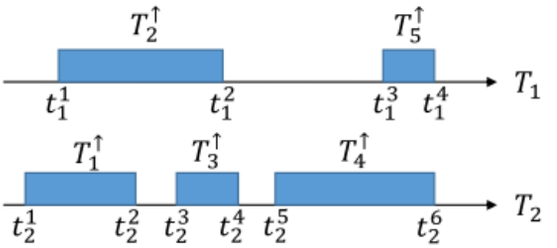

Fig. 2. Tasks are re-denoted based on the time of their occurrence.

图 6.3 IRT 任务示意图

（2）异构云平台的能耗建模

基于 DVFS 技术，异构云平台的功率由静态功率和动态功率组成，可以表示为

$$P_{S_j}(t) = P_{S_j}^s(t) + \sum_{p=1}^{C_j} P_{j,p}^d(t),$$

总功耗为时隙内功率对时间的积分，可以表示为：

$$E_\Delta = \int_0^\Delta \sum_{j=1}^{m} P_{S_j}(t)\mathrm{d}t.$$

其中，虚拟机的动态功率取决于虚拟机中的计算资源的使用情况，可以表示为：

$$P_{j,p}^d(t) = \alpha_j \cdot a_{j,p}(t) + P_{j,k} \cdot z_{j,p,k}(t),$$

本文的优化目标是：在满足任务响应时间等约束下，总能耗最小，表示为：

$$\min \ E_\Delta.$$

$$x_{i,j,p}(t) \in \{0,1\}, \sum_{j=1}^{m}\sum_{p=1}^{C_j} x_{i,j,p}(t) = 1,\ i = 1,\ ..,\ n.$$

$$z_{j,p,k}(t) \in \{0,1\},\ \sum_{p=1}^{C_j} z_{j,p,k}(t) = 1,\ j = 1 \ldots,\ m,\ k = 1,\ ..,\ m_j.$$

其中，为了满足实时性约束，需要下式成立：

$$\sum_{i=1}^{n} \frac{e_i}{\tau_i} \le n(2^{1/n} - 1).$$

若该式成立，说明时隙内的所有作业都可以满足速率单调调度(RMS-一种静态优先级调

度算法)的响应时间要求，即：

$$\sum_{i=1}^{\xi_{j,p}(t)} \frac{e_i(t)}{\tau_i} \leq \xi_{j,p}(t)(2^{1/\xi_{j,p}(t)} - 1).$$

当任务T_i分配到虚拟机$V_{j,k}$上时，可以计算得到执行时间$e_i(t)$为：

$$e_i(t) = \begin{cases} \frac{l_i}{\lambda_{i,j} a_{j,p}(t)}, & x_{i,j,p}(t)=1,\ \sum_{k=1}^{m_j} y_{i,j,p,k}(t)=0 \\ \frac{c_i}{\lambda_{i,j} a_{j,p}(t)} + \frac{g_i}{\chi_{i,j,k}}, & x_{i,j,p}(t)=1,\ y_{i,j,p,k}(t)=1, \end{cases} \tag{11}$$

（3）基于动态规划的算法设计

首先定义状态三元组为（执行任务集，平台 CPU 计算资源，GPU），表示为：

$$\mathcal{S}(\mathcal{T}_i^{\uparrow}, C_x, \mathcal{G})$$

定义价值函数$V(S)$为：

$$V(\mathcal{S}) = \begin{cases} \frac{1}{E_\Delta}, & \text{if all constraints are met} \\ 0, & \text{if not all constraints are met,} \end{cases}$$

在是否开启新的虚拟机的决策上，根据价值函数给出决策，决策的结果将会影响此时系统的状态，因此可以得到状态转移方程：

$$\mathcal{S}(\mathcal{T}_{i+1}^{\uparrow}, C_x, \mathcal{G}) =$$
$$\begin{cases} \mathcal{S}^1(\mathcal{T}_{i+1}^{\uparrow}, C_x, \mathcal{G}), & V(\mathcal{S}^1(\mathcal{T}_{i+1}^{\uparrow}, C_x, \mathcal{G})) > V(\mathcal{S}^2(\mathcal{T}_{i+1}^{\uparrow}, C_x, \mathcal{G})) \\ \mathcal{S}^2(\mathcal{T}_{i+1}^{\uparrow}, C_x, \mathcal{G}), & \text{else.} \end{cases}$$

算法的执行过程如图 6.3 所示。

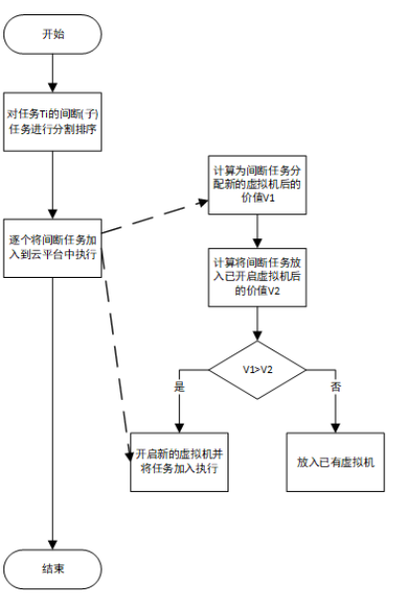

图 6.3 基于动态规划的算法流程图

（三）研究结论

云计算是一种新的计算范式，通过互联网向用户按需提供计算服务。但其能源消耗问题也日益突出。在虚拟化技术的基础上，合理分配虚拟计算单元不仅可以有效提高资源利用率，还可以降低能耗。本文研究如何利用多台服务器的云计算系统中的计算资源来调度间歇性实时任务的调度问题。文章将计算资源抽象为许多相同的虚拟 CPU，并将调度问题转化为具有整数变量和非线性约束的优化问题。通过推导状态转移方程，开发了一种动态规划算法来找到提前已知到达信息的任务的时间表。对于任务到达时间不可预测的场景，文章将算法扩展为在运行时使用。大量的仿真和实验结果表明，运行时动态规划算法具有很高的成功率，并且只需要很少的计算就可以为不可预测的间歇性实时任务找到有效的调度。所提出的方法有一些局限性。一方面，它无法应用于调度可以分解为许多小部分且每个部分可以在单个虚拟机中运行的 DAG 应用程序。另一方面，它没有考虑虚拟机的启动和关闭时间。

（四）存在问题/改进建议/启发

这项研究在考虑移动设备的运动状态以及断连可能性的背景下，对间歇性实时任务（IRT）进行了创新性的分析：将每个间歇性实时任务划分为多个任务片段，并将每个片段看作独立的任务单元。

七、EdgeDR An Online Mechanism Design for Demand Response in Edge Clouds

（一）研究背景

人们付出了很多努力来设计机制来解决电力需求响应的分裂激励问题；然而，现有的解决方案有局限性并且不适合微云。首先，采购缺乏灵活性，适应不断变化的市场条件受到限制。基于拍卖的机制促使租户对固定承诺出价，而基于奖励的机制则设定奖励率并不经意地接受租户的报价。其次，他们大多假设租户减少工作量以减少功耗，而不是自己管理租户的工作量或安排资源分配。我们所知的唯一工作负载感知机制探索了批处理作业的时间灵活性，这与云工作负载类型和工作负载分配的空间灵活性不匹配。第三，它们主要依靠昂贵、不环保的柴油发电机来补偿电网所需的能源减少。随着紧急需求响应（EDR）变得更加频繁，当前来自电网的奖励可能不足以覆盖发电成本。运营商实际上是通过缩小发电机容量来削减电力基础设施投资，这可能会损害 EDR 能力。因此，寻求其他方法来补偿能量减少是很有趣的

（二）研究内容

计算前沿正在从集中式大型数据中心转向网络边缘的分布式云。由于其分布式特性，云计算更加灵活的工作负载管理，因此非常适合处理电力需求响应，以帮助电网保持稳定性。然而，它们还需要计算需求响应以避免过载并保持可靠性。文章提出了一种新颖的在线市场机制 EdgeDR，以实现边缘需求响应计划的成本效率。在较高的层面上，观察到云运营商可以动态地打开/关闭整个云，以补偿电网所需的能源减少或为边缘服务提供足够的计算资源。文章制定了一个长期的社会成本最小化问题，并将其分解为一系列单轮采购拍卖。在每个拍卖实例中，作者建议让 cloudlet 租户以其二维服务质量退化容忍度的成本函数进行投标，并让 cloudlet 运营商选择服务质量、管理工作负载并安排 cloudlet 激活状态。此外，文章还为运营商提出了动态支付机制，以在更实际的场景中平衡短期利润和长期利益之间的权衡。通过严格的分析，证明所提出的投标政策是合理和真实的；所提出的工作负载管理算法在每次拍卖中都具有接近最佳的性能；所提出的整体在线算法达到了可证明的竞争比。最后通过广泛的跟踪驱动模拟进一步确认了所提出的机制的性能。

（1）紧急需求响应(EDR)场景建模

如图 7.1 所示，服务提供商要将服务部署到微云，用户通过 AP 连接微云使用服务，所以每个服务可以被定义为一组(i,j)，i代表 AP，j代表服务提供商。微云运营商根据服务提供商对服务的 QoS 部署微云。

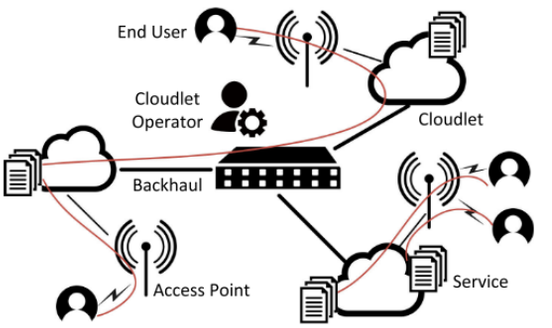

Fig. 1. An example of the cloudlet network.

图 7.1 场景描述图

如图 7.2 所示，当紧急需求响应发生后，需求响应探测器向微云运行商发送需求响应信号；微云运营商向服务提供商征求投标；随后服务提供商提交投标；微云运营商根据服务提供商提交的投标决定边缘系统中的微云部署状态；并通知服务提供商中标信息和支付价格；最后确定本地发电机提供的发电量。

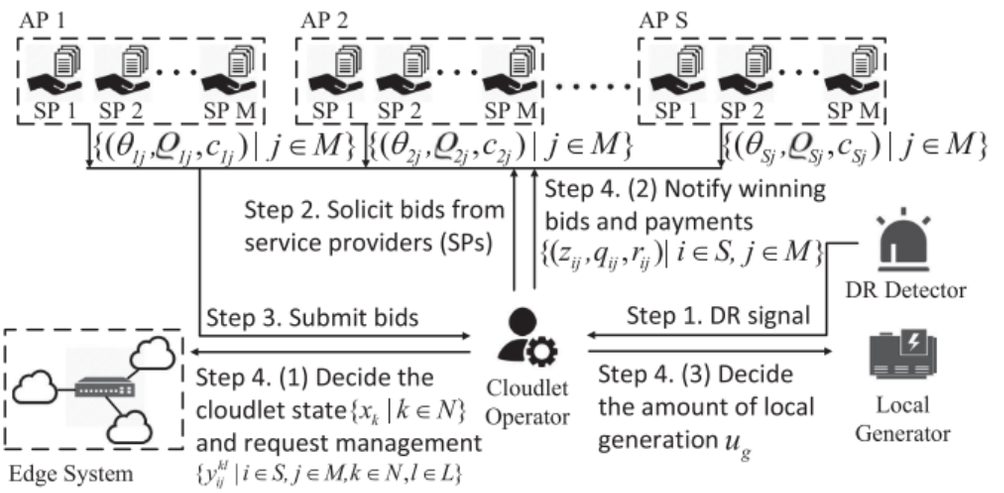

图 7.2 紧急需求响应过程示意图

（2）社会成本建模

微云运营商的总成本由四部分组成，分别是本地发电成本、微云开关机造成的成本、微云运行的成本和由于服务损耗支付给服务提供商的成本，表示为：

$$pu_g(t)$$

$$\sum_k \alpha_k[x_k(t)-x_k(t-1)]^+$$

$$\varsigma\sum_k x_k(t).$$

$$r_{ij}(t)$$

服务提供商的总收益为服务提供的预期效益和微云服务商由于服务损耗的赔偿费用之和减去由于服务损耗造成的收益损失。服务提供商的总成本即为总收益取负，表示为：

$$\sum_i (f_{ij}(z_{ij}(t)) + h_{ij}(q_{ij}(t)) - r_{ij}(t) - \sum_k a_{ij}^k(t) y_{ij}^k(t))$$

其中，由于延迟损耗和吞吐率损耗导致的损失函数表示为：

$$f_{ij}(z_{ij}(t)) = \begin{cases} c_{ij}(t) z_{ij}(t), & z_{ij}(t) \leqslant \theta_{ij}(t) \\ +\infty, & z_{ij}(t) > \theta_{ij}(t) \end{cases},$$

$$h_{ij}(q_{ij}(t)) = \begin{cases} c_{ij}(t) q_{ij}(t), & q_{ij}(t) \leqslant \varrho_{ij}(t) \\ +\infty, & q_{ij}(t) > \varrho_{ij}(t) \end{cases}.$$

文章将总的社会福利定义为微云运营商的服务提供商总收益之和，即前述两者之和取负，表示为：

$$\sum_t \Bigg(\sum_{i,j,k,l} a_{ij}^{kl} y_{ij}^{kl}(t) - \sum_{i,j} f_{ij}(z_{ij}(t)) - \sum_{i,j} h_{ij}(q_{ij}(t)) - p u_g(t) - \varsigma \sum_k x_k(t) - \sum_k \alpha_k [x_k(t) - x_k(t-1)]^+ \Bigg)$$

（3）问题描述

在服务提供商提出的延时和吞吐量损耗约束内，最大化社会福利的问题描述表示如下，其中对于每条约束可以解释为：(a)微云 k 的工作负载<=总资源量；(b)微云消耗电量<=存储电量+充电量；(c)理论转发延迟损耗<=分配转发延迟损耗；(d)理论吞吐量损耗<=分配吞吐量损耗；(e\g)服务智能部署到一个微云；(f)微云 k 开启与否（二值变量）；(h)分配转发延迟损耗<=服务提供商可容忍上限；(i)分配吞吐量损耗<=服务提供商可容忍上限；(j)充电量>=0。

$$\max \quad \sum_t \Bigg(\sum_{i,j,k,l} a_{ij}^{kl} y_{ij}^{kl}(t) - \sum_{i,j} f_{ij}(z_{ij}(t)) - \sum_{i,j} h_{ij}(q_{ij}(t)) - p u_g(t) - \varsigma \sum_k x_k(t) - \sum_k \alpha_k [x_k(t) - x_k(t-1)]^+ \Bigg) \tag{1}$$

$$\text{s.t.} \quad \sum_{i,j,l} \lambda_{ij}^l(t) y_{ij}^{kl}(t) \leqslant R_k x_k(t), \forall k \in \mathcal{N}, \forall t \in \mathcal{T} \tag{1a}$$

$$\sum_k e_k(t) \leqslant P_{EDR}(t) + u_g(t), \forall t \in \mathcal{T} \tag{1b}$$

$$\sum_k d_{ij}^k(t) \sum_l y_{ij}^{kl}(t) \leqslant z_{ij}(t), \forall i \in \mathcal{M}, \forall j \in \mathcal{M}, \forall t \in \mathcal{T} \tag{1c}$$

$$\sum_l \omega_{ij}^l(t) \sum_k y_{ij}^{kl}(t) \leqslant q_{ij}(t), \forall i \in \mathcal{M}, \forall j \in \mathcal{M} \tag{1d}$$

$$\sum_{k,l} y_{ij}^{kl}(t) \leqslant 1, \forall i \in \mathcal{S}, \forall j \in \mathcal{M}, \forall t \in \mathcal{T} \tag{1e}$$

$$x_k(t) \in \{0,1\}, \forall k \in \mathcal{N}, \forall t \in \mathcal{T} \tag{1f}$$

$$y_{ij}^{kl}(t) \in \{0,1\}, \forall i \in \mathcal{S}, \forall j \in \mathcal{M}, \forall k \in \mathcal{N}, \forall l \in \mathcal{L}, \forall t \in \mathcal{T} \tag{1g}$$

$$0 \leqslant z_{ij}(t) \leqslant \theta_{ij}(t), \forall i \in \mathcal{S}, \forall j \in \mathcal{M}, \forall t \in \mathcal{T} \tag{1h}$$

$$0 \leqslant q_{ij}(t) \leqslant \varrho_{ij}(t), \forall i \in \mathcal{S}, \forall j \in \mathcal{M}, \forall t \in \mathcal{T} \tag{1i}$$

$$u_g(t) \geqslant 0, \forall t \in \mathcal{T}. \tag{1j}$$

（4）问题分解

可以观察到，切换能耗成本与时间耦合，需要知道两个时间的微云部署(启动)状态，即：

$$\sum_k \alpha_k [x_k(t) - x_k(t-1)]^+$$

去除掉与时间耦合的部分，问题可以被分解为C_{SC}^t和W_{-SC}^t。假设给定微云部署状态$x_k(t)$，问题转化为：

$$\max \quad \sum_{i,j,k,l} a_{ij}^{kl} y_{ij}^{kl} - \sum_{i,j} f_{ij}(z_{ij}) - \sum_{i,j} h_{ij}(q_{ij}) - g(u)$$

将二元变量松弛，得到对偶松弛问题：

$$\min \quad \sum_{k} R_k x_k \mu_k + \sum_{i,j} \rho_{ij} + \sum_{i,j} f_{ij}^{*}(\xi_{ij}) + \sum_{i,j} h_{ij}^{*}(\varpi_{ij}) + g^{*}(\varphi) \tag{3}$$

$$\text{s.t.} \quad \rho_{ij} \geqslant a_{ij}^{kl} - \left(\lambda_{ij}^{l}\mu_k + d_{ij}^{k}\xi_{ij} + \omega_{ij}^{l}\varpi_{ij} + \beta_k \lambda_{ij}^{l}\varphi\right), \quad \forall i \in \mathcal{S}, j \in \mathcal{M}, k \in \mathcal{N}, l \in \mathcal{L} \tag{3a}$$

$$\mu_k, \varphi, \xi_{ij}, \varpi_{ij}, \rho_{ij} \geqslant 0 \tag{3b}$$

其中，对偶变量解释为：μ_k微云k单位资源价格，ε_{ij}单位延迟损耗惩罚，ω_{ij}单位吞吐量损耗惩罚，φ单位电量成本。因此，将服务负载λ_{ij}部署到具有资源配置l的微云k的总成本可表示为：

$$\lambda_{ij}^{l}\mu_k + d_{ij}^{k}\xi_{ij} + \bar{\omega}_{ij}^{l}\varpi_{ij} + \beta_k \lambda_{ij}^{l}\bar{\varphi}$$

（5）在线算法设计

如图 7.3 所示，在线算法包含三个子算法。

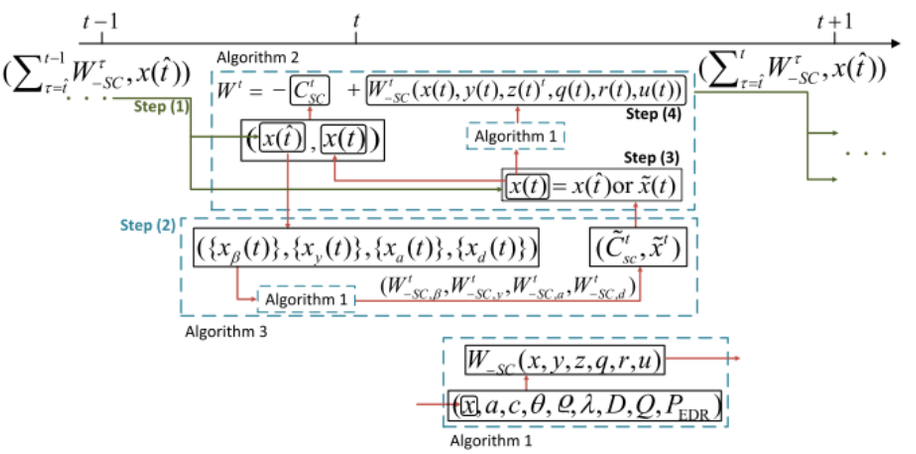

图 7.3 在线算法框架图

算法 1-单轮拍卖算法执行过程为：为服务(i,j)选取(k,l)使得每次部署产生的成本最小，由此确定的一组(i,j,k,l)为对服务(i,j)的最优分分配，从而确定最优的y；根据约束，确定最优的u；因为有了最优的(k,l)，所以可以使得$z = d^k$，$q = \omega^l$；因此通过对偶的方法，即可确定一组最优的(y,z,q,u)。

算法 2 比较在四种微云的开关状态下的W_{-SC}^{t}收益，与微云状态不变时C_{SC}^{t}的收益，根据收益情况确定微云的当前状态。

算法 3 根据社会收益计算公式计算社会收益。

（三）研究结论

文章研究了分布式云的电力应急需求响应和计算需求响应。在所提出的机制中，作者设计了一系列采购拍卖来解决运营商与租户的分割激励问题。文章专注于使运营商能够直接在云中分配租户的工作负载并调度计算资源，以便在获取租户的投标方面获得更大的灵活性并适应不断变化的市场条件。文章还第一个提出让运营商打开/关闭整个云，以提高电网的稳定性和/或确保边缘的可靠性，同时动态地在需求响应收益和所产生的切换成本之间取得平衡。文章提出了一种在线算法，该算法在每次拍卖中都采用多项式时间近似算法，与现有方法相比，具有可证明的性能保证和经过验证的实际优越性。作者还调查了一个更实际的案例，如果效用低于其预期，投标人可能会退出 EdgeDR。文章为运营商提供动态的支付机制，降低运营商的成本，同时保证个体的合理性、真实性以及 EdgeDR 的长期利益。

（四）存在问题/改进建议/启发

文章对在紧急需求响应(EDR)场景下对微云运营商和服务提供商的成本(长期社会福利\成本)进行建模，计的用于解决需求响应的在线算法，强调了对实时性的关注，使得该算法在真实场景中或类似的时延敏感的场景中具有泛用性和实际应用的潜力

参考文献：

[1] Huang Liang, et al. "Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks". IEEE Transactions on Mobile Computing, vol. 19, 2020, pp. 2581-2593

[2] Zhou Jiayun, Zhang Xinglin. "Fairness-Aware Task Offloading and Resource Allocation in Cooperative Mobile-Edge Computing". IEEE Internet of Things Journal, vol. 9, 2022, pp. 3812-3824.

[3] Qian Li Ping, et al. "Energy-Efficient Multi-Access Mobile Edge Computing With Secrecy Provisioning". IEEE Transactions on Mobile Computing, vol. 22, 2023, pp. 237-252.

[4] He Zhenli, et al. "Cost-Efficient Server Configuration and Placement for Mobile Edge Computing". (TPDS) IEEE Transactions on Parallel and Distributed Systems, vol. 33, 2022, pp. 2198-2212.

[5] Bahreini Tayebeh, et al. "Mechanisms for Resource Allocation and Pricing in Mobile Edge Computing Systems". (TPDS) IEEE Transactions on Parallel and Distributed Systems, vol. 33, 2022, pp. 667-682.

[6] Hu Biao, et al. "Energy-Minimized Scheduling of Intermittent Real-Time Tasks in a CPU-GPU Cloud Computing Platform". (TPDS) IEEE Transactions on Parallel and Distributed Systems, vol. 34, 2023, pp. 2391-2402.

[7] Chen Shutong, et al. "EdgeDR: An Online Mechanism Design for Demand Response in Edge Clouds". (TPDS) IEEE Transactions on Parallel and Distributed Systems, vol. 33, 2022, pp. 343-358.

课程意见及建议反馈

若有关于本课程的任何意见或建议，可附在此处。谢谢大家对本课程的支持！