#### 3.1.3系统建模

|  |  |  |
| --- | --- | --- |
| Symbol | 单位 | 描述 |
|  |  | 用户终端集合 |
|  |  | 边缘服务器设备集合 |
|  |  | 云中心服务器 |
|  |  | 表示终端设备u在t时刻产生的任务流 |
|  | tasks/s | 任务到达率 |
|  | CPU cycles | 任务计算需求/计算量（CPU周期） |
|  | bits | 数据量 |
|  | 秒（s） | 任务截止时间 |
|  | Hz (CPU cycles/s) | 本地计算能力 |
|  | W | 传输功率 |
|  | ~~Hz~~ | ~~上行链路带宽~~ |
|  | Hz | 边缘服务器计算资源 |
|  | ~~Hz~~ | ~~总带宽资源~~ |
|  | CPU cycles | 待执行的任务队列长度 |
|  |  | 每个设备所能容纳的最大负载量，超过容量则不接收卸载任务 |
|  | Hz | 云中心计算资源 |
|  |  | 任务卸载百分比向量 |
|  |  | 本地执行比例、卸载到边缘e的比例，卸载到云的比例 |
|  | bool（向量） | 在t时刻设备（云/边缘）i上收集到的来自终端用户的任务，具体形式为一个向量列表，类似【0，1，1，1】，0表示没接收，1表示接收到 |
|  |  | 在t时刻设备i上的负载大小， |
|  | s | 计算时间 |
|  | W | 端边云设备计算功率 |
|  |  | 设备u与边缘服务器e和云服务器之间的信道增益 |
|  | W/Hz | 噪声功率谱密度 |
|  | bit/s | 用户终端到边缘服务器和云服务器的传输速率 |
|  | s | 用户终端与边缘服务器和云服务器通信时间 |
|  |  | 资源利用率 |
|  |  | 退火系数 |
|  |  |  |

##### 3.1.3.1系统模型：

云-边缘连续体系统包括跨多个层分布的不同资源，包括IoT设备、边缘节点和中央云。这些资源在计算能力、存储器和网络能力方面不同。在此环境中管理资源涉及将应用程序组件分配给可用资源，以在满足QoS要求的同时优化性能。边缘云由各种边缘服务器组成，云则表示云中心服务器。如果任务Ta不能在本地完成，则用户端将根据所选择的策略将任务卸载到边缘或云服务器。

考虑由U个用户端 ，E个边缘服务器和1个云服务器组成的6g端边缘云网络。任务是从资源有限的用户端生成的。

如果需求超过用户终端资源的容量，则可能无法满足任务的要求。因此，有必要协调边缘和云的计算资源以适应需求。

边缘服务器和云服务器的负载会随着任务大小的增加而增加，最大计算资源可以表示为，服务器上的任务计算量表示其负载，服务器在时刻的负载可以表示为，其中

在本文中，假设条件：

* 网络带宽和计算资源在短时窗口内稳定
* 任务分割后无依赖关系，每个计算任务都是细粒度的，并且可以按任意比例进行划分和卸载
* 边缘服务器和云端的资源分配为无先级的非抢占式调度

**因此任务分割比例如下：**



归一化约束：，

因此整体场景下的任务卸载决策为



##### 3.1.3.2 计算模型

在云边端三层的边缘计算场景下，任务卸载的计算建模需围绕**任务特性、节点资源、网络条件、优化目标**四大核心要素，构建从状态感知到决策输出的完整框架。以下是具体建模步骤与关键公式，结合典型场景展开说明：

**用户终端u在t时刻发起的任务特性**包含，其中表示计算量，单位为指令数/CPU周期数，表示数据量，表示实时性要求，任务允许的最大端到端延迟，其中。

我们假设，这是因为，在现实世界中，云服务器，边缘服务器、用户终端的算力差距较大。

（1）本地计算模型Local Comuputation Model

用户可以完成任务并同时收获任务。用户终端的计算速率由CPU周期的频率来量化，表示为表示用户的CPU的计算速率。可以使用本地计算时间在本地完成计算任



（2）计算本地将消耗能量，这是由经典模型计算的:



其中表示本地每秒的能耗，这可以通过使用测量方法来实现

###### （2）边缘计算模型Edge Computation Model

为了测量与计算单元u相关联的任务i的处理延迟，我们将等待处理的任务的积压定义为，具体描述如下：



由于UEs的能力瓶颈，所生成的任务将招致拥塞延迟。边缘服务器适用于在计算任务上进行协作。根据Little定律，任务的总处理延迟为等待延迟和服务时间，



本质：

其中，为边缘服务器CPU计算速率，由边缘服务器的性能决定。

同时，边缘服务器也将有能源成本



其中表示边缘服务器中计算功率。

###### （3）云计算模型Cloud Computation Model

从ue和边缘服务器出发，云服务器展示了先进的计算性能。因此，云服务器被认为能够并行处理无限数量的任务，而计算时间显示在



其中表示CPU的云计算速率。



##### 3.1.3.3 通信模型Communication Model（通信时间）

由于6g场景中的大规模ue接入，频分多址可能无法支持所有情况。同时，6g的高带宽和低延迟确保了每次传输的效率。因此，**我们采用时分多址 (TDMA) 来完成卸载任务的传输**。

根据香农公式、传输速率定义如下所示:





其中为中心云服务器和边缘节点e分配给设备u的带宽，分别表示设备u到边缘服务器的信道增益和设备u到云中心的信道增益，表示噪声功率谱密度，

从ue到边缘服务器的通信的时间成本为





（5）资源利用率提升

针对综合利用率进行定义：云中心、边缘服务器、终端设备利用率：~~~~ 

优化后利用率：~~~~  

综合利用率提升：

##### 3.1.3.4 问题描述（目标函数）

在6g场景中，需要及时执行大量任务，从而导致必须快速完成大量计算。因此，时间是最关键的因素，可以分为计算时间和传输时间。传输时间进一步分为两个过程，上行链路和下行链路。因此，传输时间被认为是加倍的多次。另外，在系统中还必须考虑由ue和边缘服务器引起的能量成本。因此，问题的公式





P3：综合利用率



目标函数：

在6g场景中，需要及时执行大量任务，从而导致必须快速完成大量计算。因此，时间是最关键的因素，可以分为计算时间和传输时间。传输时间进一步分为两个过程，上行链路和下行链路。因此，传输时间被认为是加倍的多次。另外，在系统中还必须考虑由ue和边缘服务器引起的能量成本。因此，问题公式集中在3个关键因素上

1) 总时间和2) 总能量（3）综合利用率。

总时间是卸载系统性能的关键指标，其中较低的时间转化为更好的性能。另一方面，总能量表示实现这种性能所需的成本。优化这3个目标至关重要（综合利用率、总时间）。6g网络中的资源分配问题通常集中在两个关键因素上: 1) 总时间和2) 总能量。总时间是卸载系统性能的关键指标，其中较低的时间转化为更好的性能。另一方面，总能量表示实现这种性能所需的成本。优化这两个目标以确保6g网络中的有效资源利用至关重要。

#### 3.1.4算法

在端边云上进行卸载的任务提出了重大挑战，因为它需要在边缘或云服务器之间进行选择以在任务上进行协作，同时优化时间和精力。该问题是NP困难的，并且随着节点 (诸如ue、ES和CS) 的数量增加，计算复杂度也呈指数增长。为了解决这个问题，与传统算法相比，MADRL算法MADDPG已被证明是有效的。在本节中，我们将描述如何使用MADDPG有效地解决6g网络中资源分配的优化问题。首先，我们将问题重新表述为分散的马尔可夫决策过程 (MDP)。然后，我们概述了MADDPG算法的详细逻辑，该算法将用于优化任务卸载。最后，对算法的复杂度进行了分析，验证了算法的可行性。

##### 3.1.4.1分散的部分可观测马尔可夫决策过程

RL是一种新兴的资源分配方法。具体来说，深度RL通常用于通过深度学习来更新策略的任务卸载，并且智能agent可以与环境交互以优化目标。但是，在端边云卸载场景中，从各个用户端生成的任务必须同时完成。因此，MADRL是更可取的选择。在多智能体环境中，智能体根据所有智能体的联合行动做出决策，从而导致全局最优决策而不是局部最优决策。

资源分配方案涉及时变参数和可以连续采取的一系列可能的动作。因此，优化资源分配的问题可以被重新定义为分散的部分可观察的MDP (POMDP)，由元组表示。这里，是指所涉及的多个agent，例如ue、边缘服务器和云服务器。集合包括包含所有agent的卸载状态。表示观察空间，其中包括agent的几个特征。表示采取动作后的环境奖励值。下面提供了分散的POMDP组件的详细说明，重点是任务卸载和资源分配。

* 智能体：

在端边云场景中，每个UE拥有做出关于任务分配和目标选择的决策的agent。MADDPG算法用于训练这些agent。在这种方法中，每个agent都有自己的观察和动作空间。尽管如此，agent必须共同努力才能获得丰厚的回报。他们可以从本地agent和其他agent获取策略更新。

* 状态:

状态空间具有所有主体的特征，如{各节点计算资源、用户与边缘基站的链路带宽状态、任务所需的计算资源、任务大小}



其中表示任务特性，具体为任务计算量，任务数据量，最大计算时延，表示用户终端u、边缘服务器，云中心服务器c的CPU计算频率，表示卸载比例，表示用户终端与边缘服务器和云中心通信速率

观测：

每个agent必须获得必要的信息，以做出明智的决定。这包括数据，例如**任务大小、CPU周期、计算能力、卸载任务的百分比和通信速率**。基于此信息，agent生成观察结果和迭代以促进其决策过程。定义是



* 动作：

UE应该通过与环境的交互从动作空间中选择动作来执行卸载决定，然后使用从该交互生成的数据来训练模型。动作空间定义为



在6g卸载场景中，给agent三个目标:

1) 边缘服务器（edge）；2) 云服务器（cloud）; 3) 本地（local）。

任务不会以二进制方式卸载。因此，卸载动作需要指定目标和要卸载的任务的百分比。

* 奖励函数设计

在问题表述的目标是最大程度地减少总时间、总能耗、同时提高综合利用率。为了适应分散的POMDP，我们将问题重新表述为R函数





## 算法设计

1. 算法总体框架

* 目标：在多智能体强化学习（MARL）中，通过LLM Agent的先验知识生成候选策略，控制RL算法探索方向，再由RL和环境交互中修正策略，提升训练效率和策略性能。
* 核心修改：
  1. Actor损失函数：引入LLM蒸馏损失。
  2. 动作生成机制：训练过程中融合LLM的动作建议。
  3. 退火策略：动态调整LLM指导权重。
* 适用场景：边缘计算资源调度、多机器人协同、交通控制等需要全局协调的连续动作空间任务。

2.算法步骤

**Step1.初始化**

（1）网络参数：

* 初始化每个智能体i的Actor网络和Critic网络
* 初始化对应的目标网络和，参数与主网络相同

（2）经验回放缓冲区：

* 创建共享缓冲区，容量为。

（3）LLM接口：

* 预训练LLM agent，设计文本到动作的解析器（例如，将“迁移30%任务到边缘，30%到云”映射为动作向量ai=[0.4,0.3,0.3]

（4）退火参数：

**Step2：数据收集**

1.环境交互：

* 每个智能体u观测局部状态
* **LLM动作生成**

1. 构建提示词模板，将每一个用户终端的部分观测状态信息编码为自然语言输入到LLM中，（如“节点A负载80%，节点B空闲”）。
2. LLM输出指令（如“将任务迁移至节点B”），解析为动作向量

2. 执行与存储

执行联合动作，获得奖励 和下一状态，将所有智能体的状态向量、动作向量、奖励向量、下一状态向量，构成经验元组存储至缓冲区

**Step3：网络更新**

**1.采样批次数据**：

* 从*经验缓冲区*中随机采样*B*个样本（如*B*=64）。

**2.计算目标Q值**：

* 对每个样本，计算目标Q值：



其中为Actor网络, 表示Critic网络，和为目标Critic网络和目标Actor网络。【这里是把（根据这是时刻动作a得到的下一时刻的局部观测状态输入给Actor网络得到的该状态下的动作）作为输入传入Critic网络获得Q‘值】

**3.更新Critic网络**：

* 最小化时序差分误差（TD error）：





* 反向传播更新参数：



4.**更新Actor网络**：

* **计算策略梯度损失**：



* **LLM与RL策略损失**：



示例：

LLM建议：“将30%任务迁移至节点B”，解析为aLLM=[0.7(本地),0.3(节点B)]

agent输出动作aθ=[0.6,0.4]，则MSE损失为(0.7−0.6)2+(0.3−0.4)2=0.02(0.7−0.6)2+(0.3−0.4)2=0.02。

* **总Actor损失**：





* 反向传播更新参数：



**5.更新目标网络**：

软更新目标网络参数（τ=0.01*τ*=0.01）：



**Step 4：动态退火与LLM依赖调整**

* **退火权重更新**：



* ：初始值（如1），k：线性衰减率0.0001。
* 控制衰减阶段的迭代次数
* ：中间阶段的恒定值（如0.1）。
* **~~LLM动作选择概率衰减~~**~~：~~

~~初始概率p0=0.8，指数衰减至p~~~~final~~~~=0.1~~

~~~~

|  |
| --- |
| LLM+RL算法 |
| 输入： 每个用户终端作为智能体，每个智能体u的环境观测信息：  初始化：（1）初始化网络参数：每个智能体u的Actor网络和Critic网络（2）初始化对应的目标网络和，参数与主网络相同；（3）经验回放缓冲区：创建共享缓冲区，容量为。（4）LLM：预训练LLM agent，设计文本到动作的解析器；（5）退火参数：； |
| 1. start for对于迭代次数  从 1 到  执行： 2. 对每个智能体的局部状态  进行文本描述，生成用于 LLM 智能体的提示词（prompt） 3. 获取 LLM 智能体的专家决策（expert decision） 4. 根据强化智能体的初始策略，采集数据，并将所有智能体联合数据（其中S 为所有智能体状态信息，A为所有智能体联合动作，R为所有智能体奖励向量）存储到缓冲区中 5. start for 从缓冲区抽取B个样本，对于每个转移样本 执行： 6. 更新Critic网络：计算Critic网络最小时许差分误差（TD-error）:      1. 反向传播参数更新 2. 计算Actor网络：策略梯度损失：      1. 总Actor损失梯度      1. 反向传播更新参数 2. 网络更新：通过如下更新规则调整策略参数      1. end for 2. end for |



