VEC 中基于 DRL 的服务多分类缓存与计算卸载方案

IEEE Publication Technology Department

I. System model

A. System Overview

考虑一个三层的 VEC 网络模型, 物理层包括配备有 MEC 服务器的路边单元 RSU, 任务车辆以及云服务器.

系统中包含 L 个任务车辆, N 个 RSU, 一个云服务中心, 任务车辆表示为 $\mathcal{V} = \{1, 2, \ldots, l, \ldots, L\}$, 配备 MEC 服务器的 RSU 为 $\mathcal{R} = \{1, 2, \ldots, n, \ldots, N\}$, $\mathcal{AP} = \{\mathcal{C} \cup \mathcal{R}\} = \{1, 2, \ldots, i, \ldots, I\}$, 车辆和路边单元 RSU 具备有限的计算资源和缓存空间, 云服务中心有无限大的计算资源和缓存空间, 系统中的时间划分为 $\mathcal{T} = \{1, 2, \ldots, t, \ldots, T\}$, 每个时隙都有相同的时值 Δt, 每个车辆在每个时隙开始时刻生成一个任务.

假设互联网存在 K 类与计算型的任务相对应的缓存执行文件, 当任务所需的一类或多类执行文件在节点 (也就是车辆或路边单元 RSU) 上进行了预缓存时, 该任务才能在车辆本地或者卸载到路边单元上进行处理, 文件类型共有 K 种, 由 $\mathcal{F} = \{1, 2, \ldots, k, \ldots, K\}$ 表示, 假设每种文件类型大小固定为 θ, 且云中存储了所有类型的执行文件.

任务车辆在时刻 t 开始生成的任务请求可以表示为 $T_l(t) = \{d_l(t), \boldsymbol{f}_l(t), D_l(t)\}$, 其中,$d_l(t)$ 是任务输入数据, $\boldsymbol{f}_l(t) = \{f_1(t), f_2(t), \ldots, f_k(t), \ldots, f_K(t)\}$ 是任务请求所需资源类型的向量表示, $f_k(t) \in \{0, 1\}$ 是任务所需 k 类文件的二元变量, 即 $T_l(t)$ 需要依赖多个类型的文件才能进行处理, 例如当 $K = 3$, 也就是共有三类缓存执行文件, $\boldsymbol{f}_l(t) = \{1, 0, 1\}$ 表示任务 $T_l(t)$ 只需要第一类和第三类文件, $D_l(t)$ 是任务最大容忍时延

车辆 V_l 和路边单元 RSUR_n 的总存储资源分别为 S_l, S_n

由于任务车辆本身具备存储功能, 由车辆 V_l 生成的任务可以选择本地处理, 卸载到路边单元 R_n 或者云服务器中, 而路边单元具有有限的计算和缓存资源, 所以我们选择一个跳数的协同单元 $R_{n'}$ 作为对等卸载节点, 这样可以减轻路边单元计算负载.

B. Communication Model

为了描述实际的车辆移动环境, 考虑行驶速度在 $\mu_{\min}$ 到 $\mu_{\max}$ 之间的车辆, 并假设边缘节点知道属于其覆盖区域的所有车辆, 上行链路为多信道模型, 每个边缘节点的总体带宽均为 B, 注意不同边缘节点分配的频谱是相同的, 并且分配的信道都是正交的, 因此在同一边缘节点的覆盖区域内干扰可以忽略不计

在该研究中主要考虑的是服务缓存对应的卸载问题, 所以我们着重考虑任务车辆中用户的服务质量, 忽略车辆和路边单元的缓存时延, 假设在每个时隙每个车辆只能发送一个任务, 路边单元在每个时隙接收到的任务总数为 $q_n(t)$, 车辆 V_l 与路边单元之 R_n 间的信噪比为

$$\gamma_{l,n}(t) = \frac{p_{l,n}(t) g_{l,n}(t)}{\sum_{n=1}^{N} g_{l,n}(t) \sum_{i=1}^{q_n(t)} p_{l,n}(t) + \sigma^2} \tag{1}$$

基于香农定理, 车辆 V_l 与路边单元之 R_n 间的数据传输率为

$$r_{l,n}(t) = B_{l,n}(t) \log_2(1 + \gamma_{l,n}(t)) \tag{2}$$

其中 σ^2 是高斯白噪声方差, $g_{l,n}(t)$ 是平均信道增益, $p_{l,n}(t)$ 表示路边单元的平均信道传输功率, $B_{l,n}(t)$ 是分配给任务车辆的带宽, $B_{l,n}(t) = B/q_n(t)$

Manuscript created October, 2020; This work was developed by the IEEE Publication Technology Department. This work is distributed under the LaTeX Project Public License (LPPL) (http://www.latex-project.org/) version 1.3. A copy of the LPPL, version 1.3, is included in the base LaTeX documentation of all distributions of LaTeX released 2003/12/01 or later. The opinions expressed here are entirely that of the author. No warranty is expressed or implied. User assumes all risk.

C. Service Caching Mechanism

在每个时隙开始, 任务车辆生成新的任务 $T_l(t) = \{d_l(t), \boldsymbol{f}_l(t), D_l(t)\}$, 并且在该时隙完成任务的卸载和计算, 在每个时隙结束前, 车辆和路边单元要对每一类执行文件做出缓存更新决策并更新文件在节点的缓存状态, 协同车辆和路边单元的缓存容量限制为

$$\theta \sum_{k=1}^{K} \phi_{l,k}(t) \leq S_l \tag{3}$$

和

$$\theta \sum_{k=1}^{K} \phi_{n,k}(t) \leq S_n \tag{4}$$

其中 θ 是每个执行文件的大小, 为固定的常量, $\phi_{m,k} \in \{0,1\}$ 和 $\phi_{n,k} \in \{0,1\}$ 分别为第 k 类执行文件在车辆 V_l 和路边单元 R_n 中的缓存状态, $\phi_{l,k} = 1, \phi_{n,k} = 1$ 分别表示为第 k 类型文件在车辆 V_l 或 R_n 的缓存中, 否则不在缓存中

在每个时隙结束前, 车辆 V_l 和路边单元 R_n 要更新对每类缓存文件的决策, 用 $\psi_{l,k}(t) \in \{0,1,-1\}$ 和 $\psi_{n,k}(t) \in \{0,1,-1\}$ 分别表示表示对第 k 类文件的缓存更新策略, $\psi_{l,k}(t) = 0(\psi_{l,k}(t) = 0)$ 表示文件缓存状态保持不变, $\psi_{l,k}(t) = 1(\psi_{l,k}(t) = 0)$ 表示文件将被节点缓存, $\psi_{l,k}(t) = -1(\psi_{l,k}(t) = 0)$ 表示缓存文件将会被替换, 缓存更新决策的限制为

$$\psi_{i,k}(t) \geq -\phi_{i,k}(t), \forall i \in \mathcal{AP} \tag{5}$$

缓存状态的更新为

$$\phi_{i,k}(t+1) = \phi_{i,k}(t) + \psi_{i,k}(t) \tag{6}$$

更新完之后的缓存容量大小也要符合存储限制

$$\theta \sum_{k=1}^{K} (\phi_{i,k}(t) + \psi_{i,k}(t)) \leq S_i, \forall i \in \mathcal{AP} \tag{7}$$

D. Task Offloading Mechanism

在时隙开始时, 车辆生成任务请求, 该任务可以进行本地处理, 或者卸载到路边单元和协同路边单元或者是云服务器中, 且只能卸载到一个节点上, 任务车辆 V_l 的卸载决策为 $\boldsymbol{\omega}_l(t) = \{\omega_{local}(t), \omega_{l,n}(t), \omega_{l,n'}(t), \omega_{l,0}(t)\}$, $\omega(t) \in \{0,1\}$, 卸载决策限制为

$$\omega_{local}(t) + \omega_{l,n}(t) + \omega_{l,n'}(t) + \omega_{l,0}(t) = 1 \tag{8}$$

表 I
任务卸载决策

$\phi(t)$	卸载决策 $\omega(t)$			
	$local$	R_n	$R_{n'}$	Cloud
车辆缓存命中	1	0	0	0
车辆缓存未命中 且路边单元缓存命中	0	1	0	0
车辆缓存未命中 且路边单元缓存未命中 且协同路边单元缓存命中	0	0	1	0
车辆缓存未命中 且路边单元缓存未命中 且协同路边单元缓存未命中	0	0	0	1

其中 $\omega_{local}(t) = 1, \omega_{l,n}(t) = 1, \omega_{l,n'}(t) = 1, \omega_{l,0}(t) = 1$ 分别表示任务在 V_l 本地处理, 卸载到路边单元 R_n, 卸载到协同路边单元 $R_{n'}$ 和卸载到云服务器, 任务卸载时代理会先检查任务所需文件在节点的缓存情况, 当车辆缓存命中时, 任务进行本地处理, 若车辆缓存未命中且路边单元缓存命中则卸载到路边单元, 若协同车辆未命中且路边单元缓存未命中且路边单元临近一跳的协同路边单元缓存命中则卸载到协同路边单元, 否则卸载到云服务器, 具体的卸载案例见表 I

E. Computation Delay and Energy Consumption

当车辆处于路边单元的覆盖范围内时, 代理会根据缓存文件的缓存状态为任务选择卸载节点, 根据卸载决策可以计算出在不同的决策下相应的计算时延和传输时延.

任务进行本地处理的时延为

$$D_{local}(t) = \frac{\lambda d_l(t)}{f_V} \tag{9}$$

其中, f_V 是车辆 cpu 频率, λ 是计算 1bit 任务数据需要的 cpu 周期, 当任务选择卸载到路边单元 R_n 时, 时延为

$$D_{l,n}(t) = \frac{d_l(t)}{r_{l,n}(t)} + \frac{\lambda d_l(t)}{f_R} \tag{10}$$

其中, f_R 是路边单元的 cpu 频率, 当任务选择卸载到协同路边单元 $R_{n'}$ 时, 由于无线通信时延远大于路边单元的协同传输, 所以需要先将任务上传到路边单元再进行路边单元之间任务的传输, 卸载到协同路边单元的时延为

$$D_{l,n'}(t) = D_{l,n}(t) + \frac{d_l(t)}{r_{n,n'}} \tag{11}$$

其中, $r_{n,n'}$ 为路边单元之间的数据传输速率, 任务选择卸载到云服务器的时延为

$$D_{l,0}(t) = \frac{d_l(t)}{r_{Cloud}} \tag{12}$$

r_{Cloud} 为任务卸载到云服务器的数据传输速率, 由于云服务器有强大的计算资源, 这里忽略云服务器的计算时延, 并且计算任务的输出远小于数据的输入, 所以计算时延时忽略任务输出的返回时延, 考虑不同缓存和卸载决策下的传输延迟和计算延迟, 车辆 V_l 在时刻 t 生成任务卸载处理的总延迟为

$$\begin{aligned} D_{l,total}(t) = \omega_{local}(t)D_{local}(t) + \omega_{l,n}(t)D_{l,n}(t) \\ + \omega_{l,n'}(t)D_{l,n'}(t) + \omega_{l,0}(t)D_{l,0}(t) \end{aligned} \tag{13}$$

定义 t 时刻任务的平均处理延迟为

$$D_{ave}(t) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{q_n(t)}\sum_{i=1}^{q_n(t)} D_{l,total}(t) \tag{14}$$

F. Problem Formulation

随着交互式服务的兴起, 用户体验对用户的黏性至关重要, 黏性是这些服务成功的关键, 作为用户体验的重要方面, 任务处理时延逐渐成为衡量无线网络性能的重要指标, 设计一种联合计算卸载和服务缓存方案, 以最小化长期累计平均任务处理时间为目标

在时隙 t, 系统的决策包括时隙开始时任务车辆的卸载决策和边缘节点对某类执行文件的缓存决策, 以及时隙结束前边缘节点内某类型文件的缓存更新决策, 可以表示为 $\boldsymbol{Z}(t) = \{\boldsymbol{\omega}(t), \boldsymbol{\phi}(t), \boldsymbol{\psi}(t)\}$, $\boldsymbol{\omega}(t)$ 是车辆的卸载决策, $\boldsymbol{\omega}(t) = \{\boldsymbol{\omega}_1(t), \boldsymbol{\omega}_2(t), \ldots, \boldsymbol{\omega}_l(t), \ldots, \boldsymbol{\omega}_L(t)\}$, $\boldsymbol{\omega}_l(t) = \{\omega_{local}(t), \omega_{l,n}(t), \omega_{l,n'}(t), \omega_{l,0}(t)\}$, $\boldsymbol{\phi}_l(t)$, $\boldsymbol{\phi}_n(t)$ 分别是时隙开始时车辆和路边单元中每一类执行文件的缓存状态, $\boldsymbol{\phi}_l(t) = \{\boldsymbol{\phi}_1(t), \boldsymbol{\phi}_2(t), \ldots, \boldsymbol{\phi}_l(t), \ldots, \boldsymbol{\phi}_L(t)\}$, 每个 $\boldsymbol{\phi}_l(t)$ 又可以表示为 $\boldsymbol{\phi}_l(t) = \{\phi_{l,1}(t), \phi_{l,2}(t), \ldots, \phi_{l,k}(t), \ldots, \phi_{l,K}(t)\}$, $\boldsymbol{\psi}_l(t)$ 和 $\boldsymbol{\psi}_n(t)$ 分别是时隙结束前车辆和路边单元对每一类执行文件的缓存更新策略, $\boldsymbol{\psi}_l(t) = \{\boldsymbol{\psi}_1(t), \boldsymbol{\psi}_2(t), \ldots, \boldsymbol{\psi}_l(t), \ldots, \boldsymbol{\psi}_L(t)\}$, 每个 $\boldsymbol{\psi}_l(t)$ 又可以表示为 $\boldsymbol{\psi}_l(t) = \{\psi_{l,1}(t), \psi_{l,2}(t), \ldots, \psi_{l,k}(t), \ldots, \psi_{L,K}(t)\}$

具体地, 优化问题可以描述为:

$$\min_{\boldsymbol{Z}(t)} \quad \sum_{t=1}^{T}\alpha^{t-1}\frac{D_{ave}(t)}{D_{max}} \tag{15}$$

s.t.

$$\omega_{l,i}(t) \in \{0,1\}, \omega_{l,0}(t) \in \{0,1\}, \forall l \in \mathcal{V}, \forall t \in \mathcal{T}$$
$$\forall i \in \{\mathcal{AP}\} \tag{15a}$$
$$\omega_{local}(t) + \omega_{l,n}(t) + \omega_{l,n'}(t) + \omega_{l,0}(t) = 1,$$
$$\forall n, n' \in \mathcal{R}, \forall l, t \tag{15b}$$
$$\phi_{i,k}(t) \in \{0,1\}, \forall k \in \mathcal{F}, \forall i, t \tag{15c}$$
$$\psi_{i,k}(t) \in \{0,1\}, \forall i, k, t \tag{15d}$$
$$\psi_{i,k}(t) \geq -\phi_{i,k}(t), \forall i, k, t \tag{15e}$$
$$\phi_{i,k}(t+1) = \phi_{i,k}(t) + \psi_{i,k}(t), \forall i, k, t \tag{15f}$$
$$\theta\sum_{k=1}^{K}(\phi_{i,k}(t) + \psi_{i,k}(t)) \leq S_i, \forall i, k, t \tag{15g}$$
$$0 \leq \alpha \leq 1 \tag{15h}$$
$$D_{l,total}(t) \leq D_l(t), \forall l, t \tag{15i}$$

其中 $0 \geq \alpha \leq 1$ 是折扣因子, 用于反映未来收益或成本的时间价值, $D_{max} = \max\{D_{l,total}(t)\}$, 该问题是一个长期 MINLP 问题和 NP-hard 问题, 解决该优化问题的关键是在每个时隙对任务卸载和服务缓存做出合适的决策, 此外, 车辆参与状态的不断变化和短暂的交互也增加了边缘管理控制器的操作复杂度, 随着车辆数量和边缘节点的增加, 系统状态空间变大, 因此, 我们需要找到一种有效的方法来解决这些问题

II. DEEP REINFORCEMENT LEARNING FOR EDGE CACHING AND OFFLOADING

A. Problem Formulation Based on DRL

1) State space: 任意时刻开始时, 系统状态应该包括以下部分
   - $T_l(t)$: 车辆 V_l 在 t 时刻生成的任务请求, 可以用三元组表示为 $T_l(t) = \{d_l(t), \boldsymbol{f}_l(t), D_l(t)\}$
   - $q_n(t)$: 边缘节点 R_n 在 t 时刻接收到的任务数量
   - $\phi_{l,k}(t)$: t 时刻车辆 V_l 本地的执行文件缓存状态
   - $\phi_{n,k}(t)$: t 时刻边缘节点 R_n 的执行文件缓存状态
   - $B_{l,n}(t)$: t 时刻边缘节点 R_n 分配给车辆 V_l 的带宽

- $\gamma_{l,n}(t)$: t 时刻边缘节点无线通信的接收信噪比

t 时刻的系统状态定义为

$$\mathbf{S}(t) = \{\boldsymbol{T}(t), \boldsymbol{q}(t), \boldsymbol{\phi}_V(t), \boldsymbol{\phi}_R(t), \boldsymbol{\gamma}(t), \boldsymbol{B}(t)\} \tag{16}$$

2) Acation space: 系统代理在 t 开始要做出卸载决策, 时隙结束前要对节点的执行文件缓存做出缓存更新决策

- $\omega_l(t)$: t 时刻车辆 V_l 的更新决策
- $\psi_{l,k}(t)$: t 时刻车辆 V_l 对 k 类执行文件的缓存更新决策
- $\psi_{n,k}(t)$: t 时刻边缘节点 R_n 对 k 类执行文件的缓存更新决策

t 时刻的系统行动决策定义为

$$\mathbf{A}(t) = \{\boldsymbol{\omega}(t), \boldsymbol{\psi}_V(t), \boldsymbol{\psi}_R(t)\} \tag{17}$$

3) Reward: 强化学习主要用于使奖励值最大化, 本文优化的车辆处理任务的平均延迟, 换句话说, 奖励函数应与目标函数相对应, 奖励函数定义为

$$r(t) = R(\mathbf{S}(t), \mathbf{A}(t)) = -\frac{D_{ave}(t)}{D_{max}} \tag{18}$$