基于多智能体元强化学习的车联网协同服务缓存和计算卸载

宁兆龙[1,2]，张凯源[2]，王小洁[1]，郭磊[1]

（1. 重庆邮电大学通信与信息工程学院，重庆 400065；2. 大连理工大学软件学院，辽宁 大连 116620）

摘 要：为了降低求解优化问题的难度，提出一种双层的多路侧单元（RSU）协同缓存框架将问题进行解耦。外层采用多智能体元强化学习方法，在每个本地智能体进行决策学习的同时，采用长短期记忆网络作为元智能体来平衡本地决策并加速学习过程，从而得到最优的 RSU 缓存策略；内层采用拉格朗日乘子法求解最佳协同卸载策略，实现 RSU 间的任务分配。基于杭州真实交通数据的实验表明，所提方法具有理想的能效性能，并且能够在大规模任务流下保持网络稳健性。

关键词：车联网；边缘服务缓存；协同卸载；元强化学习

中图分类号：TN92

文献标识码：A

DOI: 10.11959/j.issn.1000−436x.2021104

Cooperative service caching and peer offloading in Internet of vehicles based on multi-agent meta-reinforcement learning

NING Zhaolong[1,2], ZHANG Kaiyuan[2], WANG Xiaojie[1], GUO Lei[1]

1. School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
2. School of Software, Dalian University of Technology, Dalian 116620, China

Abstract: In order to reduce computation complexity, a two-layer mutli-RSU (road side unit) service caching and peer offloading algorithm (MPO) was proposed to decouple the optimization problem. In the designed MPO, the outer layer utilized multi-agent meta-reinforcement learning, which established long short-term memory (LSTM) network as the meta-agent to balance decisions of local agents and accelerate learning progress. The inner layer utilized lagrange multiplier method to achieve optimal decision for RSU peer offloading. Experimental results based on real traffic data in Hangzhou demonstrate that the proposed method outperforms other methods and remains robust under large-scale workloads.

Keywords: Internet of vehicles, edge service caching, cooperative offloading, meta-reinforcement learning

1 引言

随着 5G 时代的到来和互联网设备的普及，万物互联的概念逐渐走进人们的生活，这推动了大量时延敏感型的移动应用，如增强现实、实时导航以及自动驾驶等[1-2]。虽然云技术逐渐成熟，但是随着移动设备的指数性增长，单纯依靠中央云服务器来控制广域网存在时延难以保证的瓶颈[3]，从而难以保证时延敏感型应用的服务质量。因此，移动边缘计算应运而生，成为目前解决上述问题的一种可靠方案。移动边缘计算将计算资源和存储资源以分布式的方式部署在距离用户层更近的边缘节点上，使这些边缘节点就近处理其覆盖区域内的相关业务，从而减轻回程链路的传输压力，并节约相应的服务响应时间。相对于中央云服务器的可扩展性，轻量化的边缘服务器存在资源容量受限和资源利用不

收稿日期：2020−09−17；**修回日期**：2021−03−24

通信作者：王小洁，xiaojie.kara.wang@ieee.org

基金项目：国家重点研发计划基金资助项目（No.2018YFE0206800）；国家自然科学基金资助项目（No.62025105, No.61971084, No.62001073）；重庆英才计划基金资助项目（No.CQYC2020058659）

Foundation Items: The National Key Research and Development Program of China (No.2018YFE0206800), The National Natural Science Foundation of China (No.62025105, No.61971084, No.62001073), Chongqing Talent Program (No.CQYC2020058659)

均等问题[4-5]。尤其是随着移动应用的多样性增强，其所需的资源也具有很强的异质性，这导致资源利用率低的问题日益凸显。

人工智能和机器学习技术的不断发展，以及其在多个领域的成功应用，使其正成为解决移动边缘计算瓶颈问题的关键技术[6-7]。和传统技术相比，人工智能技术对于环境的动态变化拥有更强大的感知能力。作为其重要分支，深度强化学习在资源分配方面已经得到一定的应用，文献[8-12]都表明基于强化学习的车联网资源分配解决方案具有较好的准确性和稳健性。随着用户需求的动态变化以及多方主体（设备节点、边缘节点和云服务器）的参与，车联网系统需要一种效率高、均衡性强的任务调度和资源分配方法。同时，由于边缘节点的资源有限，需要轻量化、分布式的机器学习技术与其进行适配，从而完成高效的学习过程。

车联网作为万物互联时代的重要一环，由于车辆的高移动特性和车辆应用需求的时变性，车辆应用的处理存在着更突出的难度[13]。为了更好地服务车辆用户和建设智慧城市，需要部署大量装配边缘服务器的路侧单元（RSU, road side unit）来更好地处理其覆盖区域内的车辆应用。因此，车辆、RSU 和云服务器构成了常见的三层车联网框架[14]。然而实际情况下城市中车流分布通常是不均匀的，这导致一些 RSU 没有足够的资源缓存车辆应用所需的服务，从而需要将计算任务卸载到云服务器；另一些 RSU 还有很多剩余的缓存空间没有得到利用，这就导致车联网系统的整体时延增加[15]。因此，为了充分利用车联网中的缓存和计算资源，需要网络运营商挖掘 RSU 之间的合作能力，从而提升车联网的服务效率。

车联网中的计算卸载和服务缓存得到学术界和工业界的广泛关注，因为通过制定相应策略可以更好地提升网络性能并减少能耗[16-17]。文献[16]提出一种多时间尺度的强化学习框架来进行缓存和计算资源的分配来最小化车辆应用的服务时延。文献[18]考虑了用户的移动性和网络连通性来进行内容缓存，从而能够缩短用户对内容的获取时间。文献[13]在能耗限制的情况下，通过计算卸载的方式满足了所有基站的能耗约束。很多研究关注车联网中的合作机制，文献[19]中，当车联网需要处理计算密集型任务时，多个边缘服务器会共同合作处理相关应用。文献[20]研究多接入车联网，将资源丰富的车辆与云服务器相结合，构建协同计算架构。也有很多相关研究利用车辆用户的属性，比如社会信任、位置区域等构建相应的车辆应用处理集群[5,14]。然而，这些研究大多集中在用户与服务器、用户与用户间的合作，缺少对于边缘节点之间合作的研究来提升车联网的整体服务性能。

本文考虑了车联网中 RSU 之间的合作来解决车辆应用处理过程中的服务缓存和任务调度问题。解决这一问题存在如下几个挑战：1) 车辆服务缓存和任务调度具有耦合性，车辆服务缓存决定任务调度的决策空间，任务调度的结果反映服务缓存的表现；2) 任务计算和传输的权衡，RSU 间合作会减少任务的计算时间，从而增加系统内的传输时间，如何在两者之间进行权衡得到最优解也是一个挑战；3) RSU 行为平衡，即 RSU 间合作能够降低系统时延，但求解过程中需避免陷入每个 RSU 的局部最优，而是求解全局最优策略。

本文主要的研究工作如下。

1) 本文构建了多边合作的车联网服务模型，它联合了任务缓存和边缘任务调度问题，在可用资源约束的情况下，最小化系统时延。本文将车联网服务问题建模成一个混合整数非线性规划问题，并证明求解该问题需要非多项式的计算复杂度。

2) 本文提出了一种双层的多 RSU 协同缓存框架求解上述问题，它采用多智能体元强化学习框架为 RSU 缓存车辆应用提供所需服务。每一个 RSU 作为一个本地智能体计算其对应状态下的缓存决策，云服务器作为元智能体，采用长短期记忆（LSTM, long short-term memory）结构的神经网络来平衡本地智能体的决策，并维护自己的状态信息来进行更快的策略学习。

3) 在缓存策略确定的情况下，本文提出一种自适应的 RSU 协同卸载算法，它采用拉格朗日乘子法来求解最佳协同卸载策略。本文通过二分迭代搜索的思想搜索最优拉格朗日乘子，从而调度系统中每一个 RSU 的计算任务，实现系统中所有 RSU 的工作量负载均衡。

4) 本文采用杭州交通流数据进行实验，结果表明本文提出的算法具有良好的效能和实用性。与其他 3 种基准算法相比，本文提出的算法能够获得更低的系统时延，并且能在大规模任务流下拥有相对稳定的表现。

2 系统模型

本文构建的多边车联网服务系统由 N 个 RSU 和一个提供服务的云服务器组成，如图 1 所示。

RSU 分布在城市中的不同区域，并配置边缘服务器为其相应区域内的车辆提供计算服务，不同的 RSU 之间通过局域网连接，且具有计算功能和服务缓存功能。车辆用户会和其邻近 RSU 通过无线通信的方式，将计算任务上传到对应的边缘服务器上，考虑它们之间的连接采用正交频分复用技术，因此多个车辆可以在不考虑干扰的情况下和同一个 RSU 通信。为了完成不同类别的车辆应用，系统需要从服务商处下载不同的服务，例如视频转码服务和障碍物识别服务等，设 $\mathcal{S}=\{1,2,\cdots,S\}$ 表示系统提供的服务集合，且缓存服务所需的存储空间为 p_s，F_n 和 C_n 分别表示 RSU n 拥有的计算能力和缓存能力，$\mathcal{M}=\{1,2,\cdots,M\}$ 和 $\mathcal{N}=\{1,2,\cdots,N\}$ 分别表示车辆用户和 RSU 的集合。假设 RSU n 接收车辆应用任务是一个泊松过程[20]，且任务接收速率为 π_n^t，处理一个任务所需的计算资源（CPU 周期数）服从期望为 h 的指数分布。本文主要变量及其含义如表 1 所示。

2.1 服务缓存模型

由于车辆资源有限，并且车辆应用对于处理时延具有严格要求，因此需要通过计算卸载的方式上传到 RSU 进行实时处理。此外，为了处理车辆任务，RSU 需要从中央云服务器上缓存任务所需的服务；否则，RSU 需要将任务上传到云服务器上进行处理。中央云服务器拥有充足的计算能力和缓存能力，因此，如果在云服务器上处理任务，时延主要由从 RSU 上传到云端的传输时延 T_{cloud} 造成。设 $A=\{\alpha_{ns}^t\in\{0,1\}\}_{n\in\mathcal{N},s\in\mathcal{S}}$ 表示服务的缓存策略，$a_n=\{\alpha_{ns}^t\in\{0,1\}\}_{s\in\mathcal{S}}$ 表示 RSU n 的缓存策略。由于 RSU 的缓存能力有限，因此对于每一个 RSU，不等式 $\sum_{s\in\mathcal{S}}\alpha_{ns}^t p_s\leqslant C_n$ 成立。同时，由于同一个服务可能缓存在多个 RSU 上，不同的 RSU 处理应用所需的服务可能缓存在其他 RSU 上，因此，系统需要根据服务缓存情况进行协同卸载，从而更好地利用系统中的空闲资源。

表 1 主要变量及其含义

变量	含义
α_{ns}^t	t 时刻由 RSU n 缓存计算任务 s
β_{ij}^t	t 时刻由 RSU i 卸载到 RSU j 的计算任务数量
π_n^t	t 时刻 RSU n 的任务接受速率
$T_i^{f,t}$	车联网系统 i 在 t 时刻的计算时延
$T^{g,t}$	车联网系统在 t 时刻的拥塞时延
r_t	t 时刻车联网系统的奖励函数

2.2 任务计算模型

在协同卸载过程中，本文假设计算任务在服务器间只能卸载一次，即如果计算任务从 RSU i 卸载到 RSU j，那么任务将在 RSU j 上的服务器上执行，而不会再卸载到其他 RSU。设 $B=\{\beta_{ij}^t\}_{i\in\mathcal{N},j\in\mathcal{N}}$ 表示

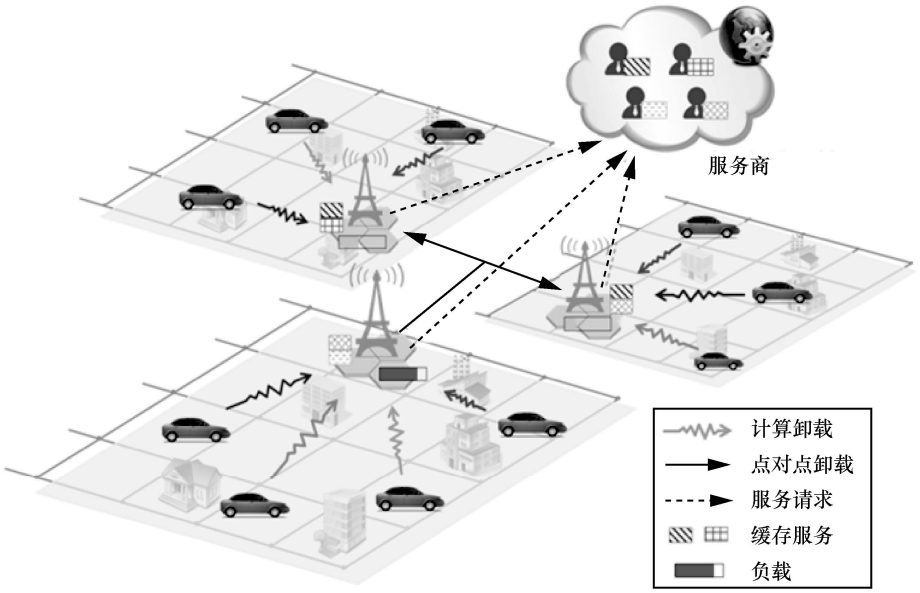

图 1 多边车联网服务系统模型

系统的协同卸载策略，其中 β_{ij}^t 表示 t 时刻由 RSU i 卸载到 RSU j 的计算任务数量，β_{ii}^t 表示 RSU i 自身处理的任务数量，则 RSU i 在 t 时刻处理的任务数量可以表示为 $\omega_i^t=\sum_{j=1}^{N}\beta_{ji}^t$。RSU 接收任务的过程是一个泊松过程，本文采用 $M/M/1$ 排队系统来为任务处理建模[21]，车联网系统的计算时延可以表示为

$$T_i^{f,t}=\frac{\omega_i^t}{\mu_i-\omega_i^t} \tag{1}$$

其中，$\mu_i=F_i/h$。为了满足任务队列处理的稳定性，$\omega_i^t\leqslant\mu_i$ 需要得到满足以确保每一个 RSU 的服务性能。

由于网络带宽有限，协同卸载会导致额外的拥塞时延。系统的拥塞时延由网络中的全部任务数量决定，系统中的总任务数量为 $\lambda^t=\sum_{i=1}^{N}\lambda_i^t$，其中 $\lambda_i^t=\sum_{j\in\mathcal{N}\setminus\{i\}}\beta_{ij}^t$ 表示 RSU i 卸载到其他 RSU 的任务数量。根据 $M/M/1$ 排队模型相关理论[18]，系统的拥塞时延为

$$T^{g,t}=\frac{\tau\lambda^t}{1-\tau\lambda^t},\tau\lambda^t<1 \tag{2}$$

其中，τ 表示在带宽充足情况下通过局域网传输一单位计算任务的时延。

2.3　问题描述

综上所述，系统时延主要由 3 个部分组成，分别是计算时延、拥塞时延和（从 RSU 上传到中央云服务器的）传输时延。系统时延 T_s 为

$$T_s=\sum_{s\in\mathcal{S}}\left[\sum_{i\in\mathcal{N}}(T_i^{f,t}+T^{g,t})+w_{os}T_{\text{cloud}}\right] \tag{3}$$

其中，$w_{os}\geqslant 0$ 表示系统中需要服务 s 且需要上传到云服务器上处理的任务数量。

本文联合考虑服务缓存策略和服务器间的协同卸载策略，目标是最小化车辆任务的处理时延，得到如下优化问题。

$$\begin{aligned}
&\text{P1}:\min_{A,B}\ \sum_{t\in\mathcal{T}}T_s\\
&\text{s.t.}\ \ \text{C1}:\sum_{s\in\mathcal{S}}\alpha_{ns}^t p_s\leqslant C_n,n\in\mathcal{N}\\
&\qquad\text{C2}:\sum_{j=1}^{N}\beta_{nj}^t\leqslant\pi_n^t,n\in\mathcal{N}\\
&\qquad\text{C3}:\sum_{j=1}^{N}\beta_{jn}^t\leqslant\mu_n,n\in\mathcal{N}\\
&\qquad\text{C4}:\alpha_{ns}^t\in\{0,1\},n\in\mathcal{N},s\in\mathcal{S}
\end{aligned}$$

其中，约束 C1 保证每个 RSU 缓存的服务不能超过其缓存能力；约束 C2 保证每个 RSU 协同卸载的任务数量不能超过其接受的车辆任务数量；约束 C3 保证每个 RSU 处理的任务数量不超过其计算能力。

定理 1　优化问题 P1 是一个混合整数非线性规划，求解其需要非多项式的计算复杂度。

证明　效用函数凸凹性

通过 2 种简化情况来分析优化问题 P1 的计算复杂度。

1) RSU 不进行协同卸载。当 RSU 不进行协同卸载时，不同服务所对应的计算任务只能由接收任务的 RSU 进行本地计算或者上传到云服务器上。因此，系统时延不仅由 RSU 的计算能力决定，也高度依赖于 RSU 的服务缓存能力。这时，优化问题 P1 可以转化为服务缓存问题和任务流输出问题，类似于文献[13]。文献[13]已经证明这个问题是一个混合整数非线性规划问题，并且拥有非多项式的计算复杂度。

2) 所有计算任务需要同一种服务。当所有计算任务需要同一种服务时，在计算资源充足的情况下，服务会被缓存在任一个缓存空间充足的 RSU。因此，该种情况可以被看作一个协同卸载问题，即在计算资源约束的情况下，进行计算任务的分配，类似于文献[18]。文献[18]已经证明求解这一问题拥有非多项式的计算复杂度。

通过上述分析，2 种简化情况都具有非多项式的计算复杂度。因此，求解本文的优化问题 P1 也具有非多项式的计算复杂度。证毕。

3　算法设计

由于求解优化问题 P1 具有非多项式的计算复杂度，本文提出一种双层的多 RSU 协同缓存算法（MPO, mutli-RSU service caching and peer offloading algorithm），外层采用多智能体元强化学习框架来为 RSU 缓存车辆应用所需的服务；内层在缓存策略确定的情况下，在缓存同一种服务的 RSU 间进行协同卸载，本文提出一种自动任务适应算法来求解系统的协同卸载策略。算法流程如图 2 所示。

3.1　基于多智能体学习的缓存分配策略

本文提出一种多智能体元策略的强化学习（MAMRL, multi-agent meta reinforcement learning）框架进行 RSU 的缓存分配，算法架构如图 3 所示。

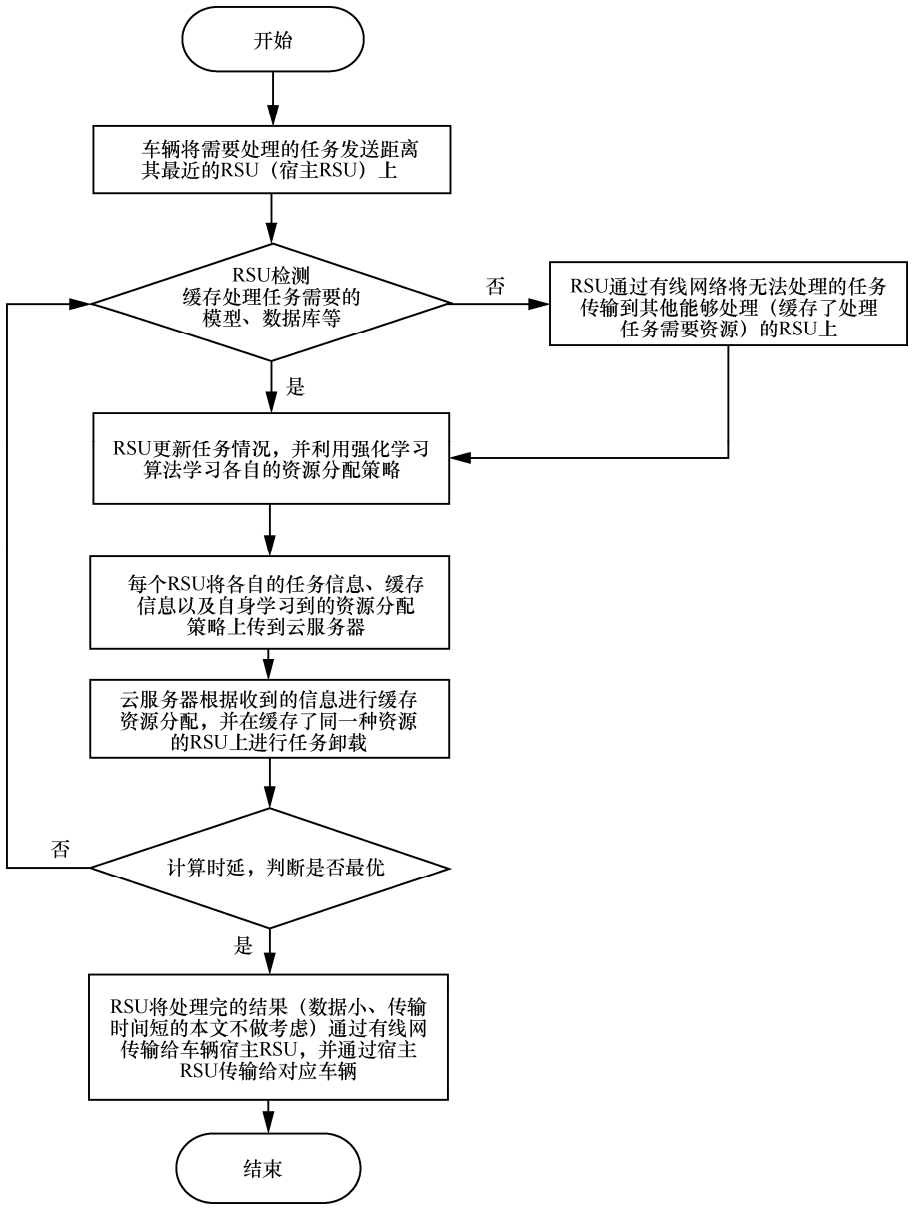

图 2　算法流程

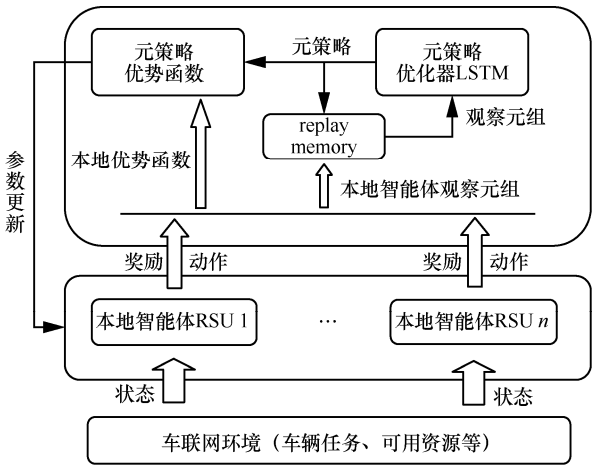

图 3　多智能体元策略的强化学习框架

和传统的强化学习相比，MAMRL 框架包含 2 种智能体：一种是本地智能体，它配置在每一个 RSU 上，根据任务量和 RSU 上的可用资源并利用强化学习算法进行自身缓存资源的分配；另一种是元智能体，它配置在云服务器上，根据每一个本地智能体学习到的信息和任务量的信息，利用 LSTM 进行全局缓存资源分配。MAMRL 减轻了因任务产生和资源需求带来的维度灾难，同时减少了 RSU 和云服务器之间的消息传递（本地智能体只需向元智能体上传其处理过的信息而不是全部信息），从而提供一个计算和通信复杂度更低的缓存分配方案。

定义状态空间为 $\prod=\{S_0,S_1,\cdots,S_N\}$，动作空间

为 $\mathcal{A}=\{A_0,A_1,\cdots,A_N\}$，本地智能体 i 在 t 时刻的状态空间为 $s_{ti}=\{\varepsilon_i,\varepsilon_i^s,C_i\}\in S_i$，其中，$\varepsilon_i$ 表示 RSU i 处理车辆任务所需的服务，ε_i^s 表示 RSU i 从云服务器上下载的任务量，C_i 表示 RSU i 的可用缓存资源和计算资源。

1) 本地智能体设计

每一个本地智能体在 t 时刻的动作空间为 $a_{ti}=\{\alpha_{is}\}\in A_i$，$s\in\mathcal{S}$，其中 $\alpha_{is}\in\{0,1\}$ 表示 RSU i 缓存服务 s 的决策，$\alpha_{is}=1$ 表示服务 s 缓存在 RSU i 上。奖励函数表示为

$$r_t(s_{ti},a_{ti})=\frac{\eta(\varepsilon_i-\varepsilon_i^s)}{\varepsilon_i} \tag{4}$$

其中，$\frac{\varepsilon_i-\varepsilon_i^s}{\varepsilon_i}$ 表示 t 时刻采取缓存策略后的缓存命中率，η 表示调整参数。缓存命中率的取值范围为 [0,1]，其值越大，说明制定的缓存策略越有效，因此会获得更高的奖励值。针对缓存问题，RSU i 的奖励最大化目标函数可以表示为

$$r_i(s_{ti},a_{ti})=\max_{a_{ti}}\mathrm{E}_{a_{ti}\sim s_{ti}}\left[\sum_{t'=t}^{\infty}\gamma^{t'-t}r_t(s_{ti},a_{ti})\right] \tag{5}$$

其中，$\gamma\in(0,1)$ 为折扣因子。在状态 s_{ti} 下采取动作定义为一个策略 π_{θ_i}，它是由参数 θ_i 决定的。策略 π_{θ_i} 决定了状态转移函数 $\Gamma:S_{ti}\times A_{ti}\mapsto S_{t'i}$，以及相应的奖励值函数 $r_i(s_{ti},a_{ti}):S_{ti}\times A_{ti}\mapsto R$。因此，在给定状态 s_{ti} 下，策略的状态值函数可以表示为

$$V^{\pi_{\theta_i}}(s_{ti})=\mathrm{E}_{a_{ti}\sim\pi_{\theta_i}(a_{ti}|s_{ti})}\left[\sum_{t'=t}^{\infty}\gamma^{t'-t}r_t(s_{ti},a_{ti})|_{a_{ti},s_{ti}}\right] \tag{6}$$

状态值函数可作为评论家的角色来评判每一个动作在该状态下的表现。因此，对于状态 s_{ti} 下的最佳策略 $\pi_{\theta_i}^*(a_{ti}\mid s_{ti})$，可以由该状态的值函数最大值确定，即最大的状态值函数可以获得其对应的最佳策略，最优值函数为

$$V^{\pi_{\theta_i}^*}(s_{ti})=\max_{a_t\in A}\mathrm{E}_{\pi_{\theta_i}^*}\left[r_{t'i}(s_{t'i},a_{t'i})+\sum_{t'=t}^{\infty}\gamma^{t'-t}V^{\pi_{\theta_i}}(s_{t'i})|_{a_{ti},s_{ti}}\right] \tag{7}$$

通过式(7)可以选择出每一个决策智能体 i 的最佳策略。采用时序差分（TD, temporal difference）法求解最优值函数和最优策略，采取策略 π_{θ_i} 的优势函数（TD 误差）定义为

$$\Lambda^{\pi_{\theta_i}}(s_{ti},a_{ti})=\left(r_{ti}(s_{ti},a_{ti})+\sum_{t'=t}^{\infty}\gamma^{t'-t}V^{\pi_{\theta_i}}(s_{t'i})\right)-V^{\pi_{\theta_i}}(s_{ti}) \tag{8}$$

其策略梯度为

$$\nabla_{\theta_i}\Lambda^{\pi_{\theta_i}}(s_{ti},a_{ti})=\mathrm{Ec}_{\pi_{\theta_i}}\left[\sum_{t'=t}^{\infty}\gamma^{t'-t}\nabla_{\theta_i}\log_{\pi_{\theta_i}}(s_{ti},a_{ti})\Lambda^{\pi_{\theta_i}}(s_{ti},a_{ti})\right] \tag{9}$$

通过式(9)可以将网络中每一个 RSU 的缓存决策离散化求解。在所有状态信息已知的情况下，本文可以采用集中式的解决策略，它的时间复杂度为 $O(S_iA_i\mid N\mid^2 T)$，这需要消耗大量的计算资源。同时，由于集中式的解决策略忽略了其他智能体的决策，导致强化学习过程中的探索和开发出现不平衡的现象，因此本文提出一种元策略的强化学习框架来解决上述困难。

2) 元策略强化学习模型

设本地智能体 i 的观察值为 O_i，它是由 $O_i:<r_i(s_{t'i},a_{t'i}),r_i(s_{ti},a_{ti}),a_{t'i},a_{ti},t,\Lambda^{\pi_{\theta_i}}(s_{ti},a_{ti})>$ 组成的数据元组，其中，$r_i(s_{t'i},a_{t'i})$ 表示下一状态的折扣奖励值，$r_i(s_{ti},a_{ti})$ 表示当前状态的折扣奖励值，$a_{t'i}$和a_{ti} 分别表示下一动作和当前动作，t 表示当前时刻，$\Lambda^{\pi_{\theta_i}}(s_{ti},a_{ti})$ 表示 TD 误差。这些都是由状态空间 S_i 决定的，并且由本地智能体处理得到的。设所有本地智能体的观察值的集合为 $\mathcal{O}=\{O_1,O_2,\cdots,O_N\}$，将观察值发送给元智能体，元智能体采用LSTM架构的神经网络处理收到的信息从而得到最优策略。对于策略 π_{θ_i}，它是由不断更新参数 θ_i 而得到的；因此，t 时刻的状态值函数和 TD 误差可以分别表示为 $V^{\pi_{\theta_i}}(s_{ti})\approx V^{\pi_{\theta_i}}(s_{ti};\theta_t)$ 和 $\Lambda^{\pi_{\theta_i}}(s_{ti},a_{ti})\approx\Lambda^{\pi_{\theta_i}}(s_{ti},a_{ti};\theta_t)$。根据式(8)和上述表示，可以将最优策略的 TD 误差写成

$$\Lambda^{\pi_{\theta_i}^*}(s_{ti},a_{t0},\cdots,a_{tN};\theta_t)=r_{ti}(s_{ti},a_{t0},\cdots,a_{tN};\theta_t)+\sum_{t'=t,s_{t'i}\in S_i}^{\infty}\gamma^{t'-t}\Gamma(s_{t'i}\mid s_{ti},a_{t0},\cdots,a_{tN})V^{\pi_{\theta_i}}(s_{t'i},\pi_{\theta_0}^*,\cdots,\pi_{\theta_N}^*)-V^{\pi_{\theta_i}}(s_{ti},\pi_{\theta_0}^*,\cdots,\pi_{\theta_N}^*)$$

其中，$(\pi_{\theta_0}^*,\cdots,\pi_{\theta_N}^*)$ 表示联合缓存分配策略，$\Gamma(s_{t'i}\mid s_{ti},a_{t0},\cdots,a_{tN})\mapsto[0,1]$ 表示状态转移概率。利用式(9)，最小化损失函数

$$L(\theta_i)=\min_{\pi_{\theta_i}}\frac{1}{N}\sum_{i\in\mathcal{N}}\frac{1}{2}\left[(r_{ti}(s_{ti},a_{ti};\theta_t)+\sum_{t'=tc}^{\infty}\gamma^{t'-t}V^{\pi_{\theta_i}^*}(s_{t'i};\theta_t))-V^{\pi_{\theta_i}^*}(s_{ti})^2\right] \tag{10}$$

为了进行低偏差的探索，算法采用熵正则化函

数 $\beta H(\pi_{\theta_i}(a_{ti}\mid s_{ti};\theta_t))$ 来处理随机的车辆任务需求，其中，β 表示正则化调整参数，$H(\pi_{\theta_i}(a_{ti}\mid s_{ti};\theta_t))$ 表示策略的熵值，较大的熵值说明智能体在搜寻策略时拥有更多的探索组合。结合熵正则化函数和式(10)，损失函数的策略梯度可以表示为

$$\nabla_{\theta_i}L(\theta_i)=\frac{1}{N}\sum_{i\in\mathcal{N}}\sum_{t'=t}^{\infty}\beta H(\pi_{\theta_i}(a_{ti}\mid s_{ti};\theta_t))+$$
$$\nabla_{\theta_i}\log\pi_{\theta_i}(a_{ti}\mid s_{ti})A^{\pi_{\theta_i}}(s_{ti},a_{ti}\mid\theta_t) \tag{11}$$

元智能体由 LSTM 结构的神经网络组成，设它的网络参数为 ϕ，且由 4 个门层来计算出下一个状态 $s_{t'i}$ 的最优决策 $\pi^*_{\theta_i}$ 和其对应的优势函数，4 个门层分别为遗忘门 $F_{t'}$、输入门 $I_{t'}$、单元状态层 $\widehat{E}_{t'}$ 和输出层 $Z_{t'}$。元智能体的具体实现为

$$M_{t'}(\boldsymbol{O}_{t'};\phi)=\mathrm{softmax}(\boldsymbol{H}_{t'}^{\mathrm{T}})$$

其中，

$$F_{t'}=\sigma(\phi_{FO}(\boldsymbol{O}_{it'})^{\mathrm{T}}+\phi_{FH}(\boldsymbol{H}_t)^{\mathrm{T}}+\boldsymbol{b}_F)$$
$$I_{t'}=\sigma(\phi_{IO}(\boldsymbol{O}_{it'})^{\mathrm{T}}+\phi_{IH}(\boldsymbol{H}_t)^{\mathrm{T}}+\boldsymbol{b}_I)$$
$$\widehat{E}_{t'}=\tanh(\phi_{EO}(\boldsymbol{O}_{it'})^{\mathrm{T}}+\phi_{EH}(\boldsymbol{H}_t)^{\mathrm{T}}+\boldsymbol{b}_E)$$
$$E_{t'}=\widehat{E}_{t'}\odot I_{t'}+F_{t'}\odot E_t$$
$$Z_{t'}=\sigma(\phi_{ZO}(\boldsymbol{O}_{it'})^{\mathrm{T}}+\phi_{ZH}(\boldsymbol{H}_t)^{\mathrm{T}}+\boldsymbol{b}_Z)$$
$$\boldsymbol{H}_{t'}=\tanh(E_{t'})\odot Z_{t'}$$

其中，遗忘门 $F_{t'}$ 负责确定哪些信息需要抛除；输入门 $I_{t'}$ 负责确定哪些信息需要更新；单元状态层 $\widehat{E}_{t'}$ 使用 tanh(•) 函数产生新的候选值向量，并通过公式 $E_{t'}=\widehat{E}_{t'}\odot I_{t'}+F_{t'}\odot E_t$ 更新单元状态层；输出层 $Z_{t'}$ 决定哪些信息输出，通过公式 $\boldsymbol{H}_{t'}=\tanh(E_{t'})\odot Z_{t'}$ 计算单元输出，并将最终的输出利用 softmax 函数输出最佳策略。因此，元智能体的损失函数是由本地智能体的分布所决定的，其损失函数的期望可以表示为

$$L(\phi)=\mathrm{E}_{L(\theta)}\left[\sum_{t=1}^{T}L(\theta_t)\right] \tag{12}$$

因此，MAMRL 框架将元智能体的学习参数传递给本地智能体，以便每个本地智能体更新自身的学习参数来计算出最优的缓存分配策略。其参数更新式为

$$\theta_{t'}=\theta_t+M_t(\nabla_{\theta_i}L(\theta_t);\phi) \tag{13}$$

MAMRL 框架可以理解成一个多参与人（N-player）的马尔可夫博弈模型。根据当前对多人马尔可夫博弈模型的研究[22]，MAMRL 模型至少存在一个纳什均衡点来保证最佳的缓存分配策略。因此，对于 MAMRL 模型求解的最优性，有命题 1 成立。

命题 1 对于 RSU i，其最佳的缓存分配策略 $\pi^*_{\theta_i}$ 是一个纳什均衡点，且其纳什均衡值为 $V^{\pi_{\theta_i}}(s_{t'i},\pi^*_{\theta_0},\cdots,\pi^*_{\theta_N})$。

证明 对于 RSU i 而言，$\pi^*_{\theta_i}$ 是综合考虑所有 RSU 动作后产生的纳什均衡的最优策略。因此，BS i 无法采取更优的策略来提升 $V^{\pi_{\theta_i}}(s_{ti})$，则对于式(10)，有如下不等式成立

$$V^{\pi^*_{\theta_i}}(s_{ti})\geqslant r_{ti}(s_{ti},a_{t0},\cdots,a_{tN};\theta_t)+$$
$$\sum_{t'=t,s_{t'i}\in S_i}^{\infty}\gamma^{t'-t}\Gamma(s_{t'i}\mid s_{ti},a_{t0},\cdots,a_{tN})V^{\pi_{\theta_i}}(s_{t'i},\pi^*_{\theta_0},\cdots,\pi^*_{\theta_N})$$

通过上述不等式可知，MAMRL 模型中的元智能体 $M_{t'}(\mathrm{O}_{t'};\phi)$ 能够在 RSU i 采取策略 $\pi^*_{\theta_i}$ 时达到纳什均衡，且 RSU i 的最优值为

$$V^{\pi^*_{\theta_i}}(s_{ti})=M_t(\nabla_{\theta_i}L(\theta_t);\phi)$$

因此，最优策略 $\pi^*_{\theta_i}$ 是缓存分配问题的一个纳什均衡点。证毕。

对于 MAMRL 模型的收敛性，有命题 2 成立。

命题 2 对于式(10)的梯度估计，可以建立估计值 $\widehat{\nabla}_{\theta_i}L(\theta_i)$ 和真实值 $\nabla_{\theta_i}L(\theta_i)$ 之间的关系为

$$P(\widehat{\nabla}_{\theta_i}L(\theta_i),\nabla_{\theta_i}L(\theta_i)>0)\propto(0.5)^N$$

证明 对于 RSU i 在时刻 t 采取动作 a_{ti} 的概率可以表示为

$$P(a_{ti})=\theta_i^{\,a_{ti}}(1-\theta_i)^{1-a_{ti}}=a_{ti}\log\theta_i+(1-a_{ti})\log(1-\theta_i)$$

考虑单个状态的情况下，策略梯度的估计量可表示为

$$\frac{\hat{\partial}}{\partial\theta_i}L(\theta_i)=r_{ti}(s_{ti},a_{t0},\cdots,a_{tN})\frac{\partial}{\partial\theta_i}\log P(a_{t0},\cdots,a_{tN})=$$
$$r_{ti}(s_{ti},a_{t0},\cdots,a_{tN})\frac{\partial}{\partial\theta_i}a_{ti}\log\theta_i+(1-a_{ti})\log(1-\theta_i)=$$
$$r_{ti}(s_{ti},a_{t0},\cdots,a_{tN})\left(\frac{a_{ti}}{\theta_i}-\frac{(1-a_{ti})}{(1-\theta_i)}\right)=$$
$$r_{ti}(s_{ti},a_{t0},\cdots,a_{tN})(2a_{ti}-1),\theta_i=0.5$$

因此，RSU i 的期望奖励可以表示为

$$\mathbb{E}(r_i)=\sum_{\forall i\in\mathcal{N}}r_{ti}(s_{ti},a_{t0},\cdots,a_{tN};\theta_t)(0.5)^N$$

令 $r_{ti}(s_{ti},a_{t0},\cdots,a_{tN};\theta_t)=1|r_{ti}(s_{ti},a_{t0},\cdots,a_{tN};\theta_t)$，可得 $\mathbb{E}(r_i)=(0.5)^N$。则梯度的期望可以表示为 $\mathbb{E}\left(\frac{\hat{\partial}}{\partial\theta_i}L(\theta_i)\right)=\frac{\partial}{\partial\theta_i}L(\theta_i)=(0.5)^N$，估计梯度的方差可以表示为

$$\mathbb{V}\left(\frac{\hat{\partial}}{\partial\theta_i}L(\theta_i)\right)=\mathbb{E}\left[\frac{\hat{\partial}}{\partial\theta_i}L^2(\theta_i)\right]-\mathbb{E}\left[\frac{\hat{\partial}}{\partial\theta_i}L(\theta_i)\right]^2=0.5^N-0.5^{2N}$$

因此，有

$$P(\widehat{\nabla}_{\theta_i}L(\theta_i),\nabla_{\theta_i}L(\theta_i)>0)=0.5^N\frac{\hat{\partial}}{\partial\theta_i}L(\theta_i)\propto(0.5)^N$$

上式说明，在求解过程中，梯度步长朝着正确的方向移动，且随着 RSU 数量的增加而呈指数级下降。证毕。

MAMRL 算法的伪代码如算法 1 和算法 2 所示，其中算法 1 为本地智能体训练过程，算法 2 为元智能体训练过程。

算法 1　本地智能体训练

输入　$s_{ti}=\{\varepsilon_i,\varepsilon_i^s,C_i\}$，$\forall s_{ti}\in S_i,\forall t\in\mathcal{T}$

输出　π_θ^*

初始化 $\theta_i,i\in\mathcal{N}$

1) for episode=1 to max_episode:
2)　replay_memory=[]
3)　　for each　$t\in\mathcal{T}$
4)　　　for step=1 to max_step:
5)　　　根据式(5)计算 $r_i(s_{ti},a_{ti})$
6)　　　根据式(6)计算 $V^{\pi_{\theta_i}}(s_{ti})$
7)　　　选取动作 $a_{ti}\sim\pi_{\theta_i}(a_{ti}|s_{ti})$ 并产生下一状态 $s_{t'i}$
8)　　　将 $<s_{ti},a_{ti},s_{t'i},r_{ti},\text{step},V^{\pi_{\theta_i}}(s_{ti})>$ 存入 replay_memory
9)　　　end for
10)　从 replay_memory 中取出元组，根据式(8)计算每个策略的优势函数并根据式(9)计算其策略梯度
11)　根据上述计算结果生成观察元组 $O_i:<r_i(s_{t'i},a_{t'i}),r_i(s_{ti},a_{ti}),a_{t'i},a_{ti},t,\Lambda^{\pi_{\theta_i}}(s_{ti},a_{ti})>$
12)　根据算法 2 计算元智能体的策略梯度
13)　参数更新：$\theta_{t'}=\theta_t+M_t(\nabla_{\theta_t}L(\theta_t);\phi)$
14)　end for
15) end for
16) 根据最优策略计算下一时刻的状态：$s_{t'i}=\arg\max_{\pi_{\theta_i}}(a_{ti}),i\in\mathcal{N}$

算法 2　元智能体训练

输入　$O_i:<r_i(s_{t'i},a_{t'i}),r_i(s_{ti},a_{ti}),a_{t'i},a_{ti},t,\Lambda^{\pi_{\theta_i}}\cdot(s_{ti},a_{ti})>$，$\forall t\in\mathcal{T}$

输出　$M_t(\nabla_{\theta_t}L(\theta_t);\phi)$

初始化参数 ϕ

1) for each　$t\in\mathcal{T}$
2)　for each　$i\in\mathcal{N}$
3)　根据式(10)和式(11)分别计算损失函数及其策略梯度
4)　根据元智能体实现方法计算 $M_t(\nabla_{\theta_t}L(\theta_t);\phi)$
5)　根据式(12)计算损失函数
6)　end for
7) end for
8)　return　$M_t(\nabla_{\theta_t}L(\theta_t);\phi)$

3.2　RSU 协同计算卸载算法

当所有 RSU 的缓存策略确定后，优化问题 P1 将转换为车辆任务在缓存同一服务的 RSU 之间进行协同计算卸载的子问题 P2

$$\text{P2}:\min_{\text{B}}\ T_s$$

$$\text{s.t. C1}:\sum_{j=1}^{N}\beta_{nj}^t\leqslant\pi_n^t,n\in\mathcal{N}$$

$$\text{C2}:\sum_{j=1}^{N}\beta_{jn}^t\leqslant\mu_n,n\in\mathcal{N}$$

$$\text{C3}:\beta_{ij}\geqslant 0,i,j\in\mathcal{N}$$

将式(1)~式(3)代入优化问题的目标函数中，可以得到

$$f(\beta)=\sum_{s\in\mathcal{S}}\left(\sum_{i\in\mathcal{N}}\frac{\omega_i^t}{\mu_i-\omega_i^t}+\frac{\tau\lambda^t}{1-\tau\lambda^t}\right)\tag{14}$$

变量 ω_i^t 和 λ^t 是 2 个独立变量，根据上文定义可知，它们都是由 RSU 协同计算卸载策略 β 决定的，其定义和关系见 2.2 节，因此可以通过求解式(14)来确定最优协同计算卸载策略。对于每一个缓存服务 s，它与其他的缓存服务之间是独立的。因此，在问题求解过程中，下文以服务 s 为例，对于子问题 P2，本文采用一种迭代的思路搜寻解空间中满足 KKT 条件的结果作为优化问题的解。在 RSU 协同计算卸载过程中，对于每一个 RSU 都有工作量负

载均衡等式 $\sum_{i=1}^{N} I_i = \sum_{i=1}^{N} O_i$ 成立，其中，I_i 表示 RSU i 的接收任务量，O_i 表示 RSU i 的输出任务量。根据定义，有 $\omega_i^t = \pi_i^t + I_i - O_i$ 和 $\lambda^t = \sum_{i=1}^{N} I_i$ 成立。将上述等式代入优化问题，可将优化问题 P2 转化为关于变量 I 和 O 的优化问题 P3

$$\text{P3}: \min_{I,O} \sum_{i=1}^{N} \frac{\pi_i^t + I_i - O_i}{\mu_i - (\pi_i^t + I_i - O_i)} + \frac{\tau \sum_{i=1}^{N} I_i}{1 - \tau \sum_{i=1}^{N} I_i}$$

$$\text{s.t. } \pi_i^t + I_i - O_i \geqslant 0, i \in \mathcal{N}$$

$$-\sum_{i=1}^{N} I_i + \sum_{i=1}^{N} O_i = 0$$

为了求解上述优化问题，本文首先将 RSU 处理计算任务分为 3 种模式：接收模式、平衡模式、卸载模式。接收模式表示 RSU 接收来自其他 RSU 的计算任务；平衡模式表示 RSU 不接收其他 RSU 的任务也不发送计算任务给其他 RSU；卸载模式表示 RSU 发送自身的计算任务给其他 RSU。RSU 在处理计算任务时，只能选择一种模式。同时，本文定义 2 个辅助函数来进行优化问题求解，一是边界计算时延函数

$$d_i(\omega_i^t) \triangleq \frac{\partial}{\partial \omega_i^t}(T_i^{f,t}) = \frac{\mu_i}{(\mu_i - \omega_i^t)^2} \tag{15}$$

它表示当 RSU 处理任务的计算时延的边界值；二是边界网络拥塞时延

$$g(\lambda^t) \triangleq \frac{\partial}{\partial \lambda^t}(T^{g,t}) = \frac{\tau}{(1 - \tau\lambda^t)^2} \tag{16}$$

它表示局域网在处理任务时网络时延的边界值。定义函数 $\xi_i^t \triangleq d_i(\pi_i^t)$ 表示当所有计算任务都由 RSU 进行本地处理的计算消耗。根据 KKT 条件有定理 2 成立，从而进行优化问题最优解的搜索。

定理 2 RSU 在 t 时刻的任务处理模式和最优协同计算卸载策略如下所示。

1) 当 $\xi_i^t < \alpha$ 时，RSU 处于接收模式，它最终的任务处理量为 $\omega_i^{t*} = d_i^{-1}(\alpha)$。

2) 当 $\alpha < \xi_i^t < \alpha + g(\lambda^{t*})$ 时，RSU 处于平衡模式，它最终的任务处理量为 $\omega_i^{t*} = \pi_i^t$。

3) 当 $\xi_i^t > \alpha + g(\lambda^{t*})$ 时，RSU 处于卸载模式，它最终处理任务量为 $\omega_i^{t*} = [d_i^{-1}(\alpha + g(\lambda^{t*}))]^+$。

$$\underbrace{\sum_{i \in \mathcal{R}} (d_i^{-1}(\alpha) - \pi_i^t)}_{\text{网络中所有RSU接收的任务量}} = \underbrace{\sum_{i \in \mathcal{F}} (\pi_i^t - [d_i^{-1}(\alpha + g(\lambda^{t*}))]^+)}_{\text{网络中所有RSU卸载的任务量}} \tag{17}$$

其中，λ^{t*}和α 是式(17)的解，λ^{t*} 表示最优网络拥塞时延，α 表示拉格朗日乘子，$\mathcal{R}$和$\mathcal{F}$ 分别表示网络中处于接收模式和卸载模式的 RSU 集合。

由于直接通过求导来求解 α 存在较大困难，因此本文采用一种二分迭代搜索的思路来通过工作量负载等式寻找最优解。在每一次迭代中，首先通过初始参数 α 确定处于接收模式的 RSU 以及其接收的任务量 λ_R。然后，令 $\lambda = \lambda_R$ 来确定网络中处于平衡模式和卸载模式的 RSU，并计算卸载的任务量 λ_F。如果 $\lambda_R = \lambda_F$，此时的 α 为最优解；否则，算法更新参数 α 并进入下一轮迭代。求解算法流程如算法 3 所示。

算法 3 二分迭代协同计算卸载算法

输入 RSU 接受任务量 $\pi_i^t, i \in \mathcal{N}, t \in \mathcal{T}$，RSU 服务速率 $\mu_i, i \in \mathcal{N}$，网络通信时间 τ

输出 ω_i^{t*}

初始化 $\omega_i^t \leftarrow \pi_i^t, i \in \mathcal{N}$， $\xi_i^t = d_i(\pi_i^t)$

1) 令 $\xi_{\max}^t = \max_{i \in \mathcal{N}} \xi_i^t$， $\xi_{\min}^t = \min_{i \in \mathcal{N}} \xi_i^t$

2) if $\xi_{\min}^t + g(0) \geqslant \xi_{\max}^t$:

3) 不需要协同计算卸载

4) $a \leftarrow \xi_{\min}^t, b \leftarrow \xi_{\max}^t$

5) while $|\lambda_R(\alpha) - \lambda_F(\alpha)| \geqslant \upsilon$

6) $\lambda_R(\alpha) \leftarrow 0, \lambda_F(\alpha) \leftarrow 0$

7) $\alpha \leftarrow \frac{1}{2}(a + b)$

8) 根据变量 α 确定处于接收模式的 RSU 以及其接收任务的数量

9) 根据变量 α 确定处于卸载模式的 RSU 以及其卸载任务的数量

10) if $\lambda_R(\alpha) > \lambda_F(\alpha)$

11) $b \leftarrow a$

12) else

13) $a \leftarrow \alpha$

14) end if

15) end while

16)end if

根据定理 2 确定每一个 RSU 的 ω_i^{t*}。

4　实验结果和分析

本文利用杭州真实的交通流数据模拟任务的产生，验证本文提出的车联网系统的有效性。系统中由一个云服务器和 9 个 RSU 组成（如图 4 所示），每一个 RSU 的覆盖区域为 200 m×200 m，且为车辆提供 8 种类型不同的服务。为了适应不同规模的任务量，RSU 布置在杭州市中心如图 4 所示的 9 个十字路口，且每一个车辆产生任务服从速率为[0,4]任务/2 分钟的泊松分布。其他参数设定如表 2 所示。

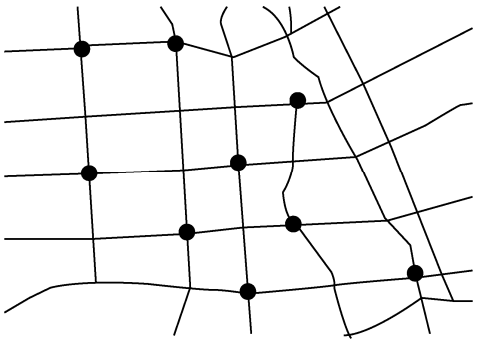

图 4　车联网系统布置说明

表 2　　参数设定

参数	值
任务所需 CPU 周期数 h/个	2×10^7
RSU i 的计算能力（CPU 周期数） F_i / GHz	3
缓存服务 s 大小 p_s/MB	0.2
RSU i 的服务缓存能力 C_i/MB	0.6
网络传输一个计算任务的时延 τ/ms	200
计算任务上传到云端的时延 T_{cloud}/ms	800

本文与 3 种基准算法进行比较。1) 非协作卸载算法：RSU 完成缓存分配后，只有 RSU 本地处理或上传到云服务器处理 2 种情况。2) 单智能体缓存算法：采用单智能体强化学习进行缓存分配，然后采用协同计算卸载进行计算任务分配。3) 贪婪缓存策略：每个 RSU 缓存最流行的服务，然后计算任务将在本地处理或上传到云服务器进行处理。

4.1　系统表现

图 5 为不同智能体（9 个本地智能体和一个元智能体）获得的奖励值，本文计算每 50 次迭代的平均奖励值。在 MAMRL 框架中，所有的智能体经过 1 000 轮迭代都会得到一个收敛的奖励值，其中，元智能体拥有最高的奖励值（大约为 90），所有的本地智能体（RSU）在收敛时大约为 80 到 85，只有一个 RSU 6 收敛时奖励值在 70 左右。不同 RSU 上的奖励值变化是由 RSU 上处理任务的不同和分配资源的不同所导致的，同时也由强化学习中各个本地智能体求解资源分配方案过程中的探索和利用权衡所决定。

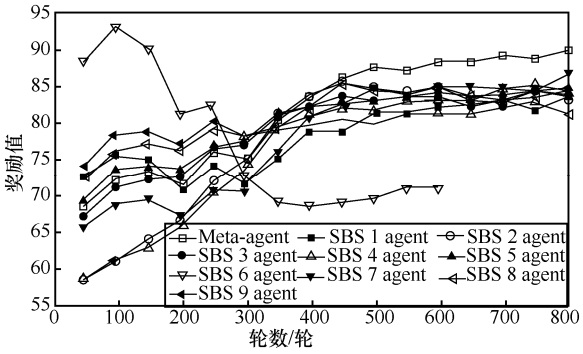

图 5　多智能体元强化学习中不同智能体的奖励值

图 6 为本文提出的缓存分配策略在不同探索衰减下的系统表现。探索率表示强化学习过程中对于动作空间的探索概率，从而在探索和利用之间进行权衡搜索到最优的缓存策略。初始情况下，探索率大可以快速探索动作空间中的优秀策略，探索率逐渐衰减可以平衡动作选择过程中的探索利用效率，从而使智能体逐渐搜索到最佳策略（动作）。较大的探索衰减强调动作选取中的探索过程，可以获得更快的收敛速度，但是可能导致智能体对于动作空间探索不够完全；较小的探索衰减则强调动作选取中的利用过程，可能会获得更好的收敛结果，但是不断减小该值可能会影响收敛速度。因此在仿真实验中，本文将探索衰减设置成不同值来观察收敛效果，可以看出，当探索衰减大于10^{-4} 时，1 000 轮迭代内可以收敛；当探索衰减为 0.1 时，收敛时的奖励值较低。

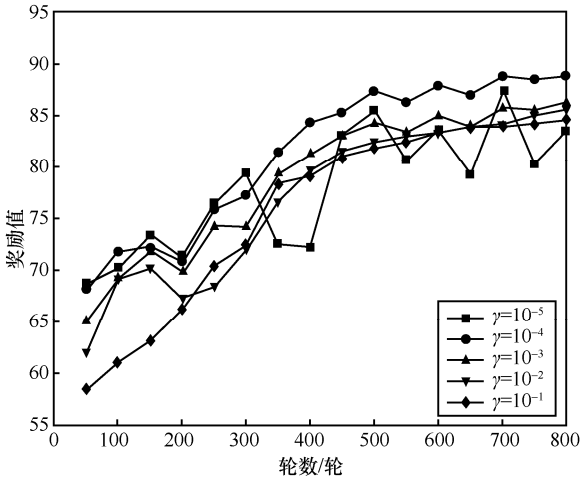

图 6　不同探索衰减的收敛结果

4.2　性能对比

和其他 3 种基准算法（非合作卸载、单智能体缓存、贪婪缓存）相比，本文提出的多 RSU 协同缓存算法表现最优，能够得到最低的系统时延。图 7 给出了不同缓存大小下不同算法的网络时延。缓存大小对于 RSU 缓存内容的选择具有重要的影响，缓存大小为 0 则表明所有处理任务的服务需要从云服器上下载，因此所有的缓存策略失去意义；如果缓存大小超过任务所需的服务个数，服务器可以缓存所有的服务，缓存策略也失去意义。因此，本文只讨论缓存大小在区间内不同算法的表现情况。从图 7 中可以看出，随着缓存大小的增加，所有算法的系统时延都会有所下降，但是本文提出的算法相比较其他 3 种算法表现更加优异；在缓存空间明显不足时，本文提出的算法相较其他 3 种算法提升更明显，这说明本文提出的算法能够更好地应对资源有限情况下的缓存分配问题，即能够在有限的资源情况下，最小化系统时延。

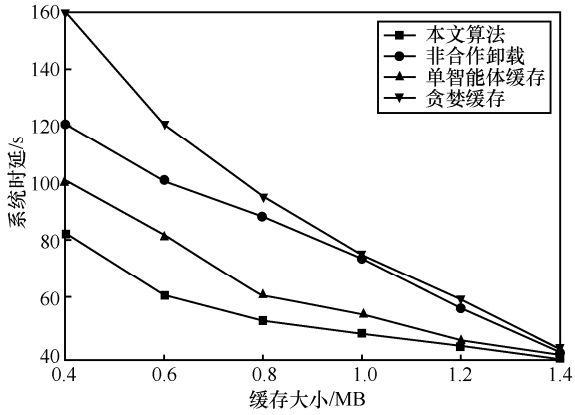

图 7　不同缓存空间下系统时延对比

图 8 说明了不同缓存大小下不同方法中从 RSU 输出到云服务器处理的任务大小。当 RSU 之间无法通过协同卸载来处理用户任务时需要将任务传输到云服务器来处理，这一过程需要相对高的传输时延。因此，输出到云服务器的任务量可以表明不同方法的协同卸载处理情况。从图 8 中曲线可以看出，随着 RSU 缓存空间的增加，更多的任务可以在 RSU 集群内进行处理，因此传输到云服务器的任务数会越来越小，相比其他 3 种算法，本文提出的算法因为具有协同计算卸载机制会在 RSU 之间进行处理任务，而不是直接上传到云处理器，因此会大幅减少输出任务量；相比单智能体缓存算法，多智能体算法在缓存空间相对不足的情况下，表现更佳，这说明本文提出的算法在受限资源情况下表现更佳。

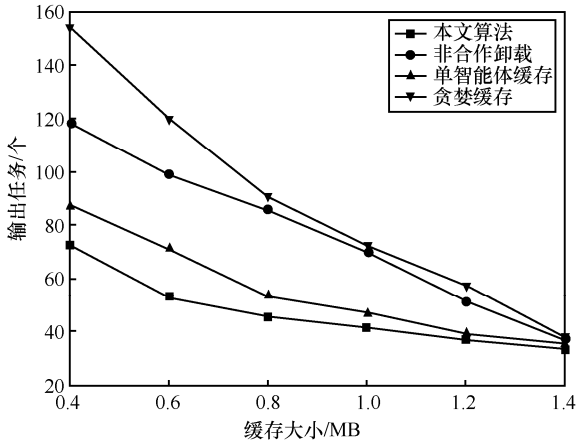

图 8　不同缓存空间下输出任务情况对比

图 9 和图 10 分别表明了在用户产生不同数量任务的情况下不同方法的系统时延和输出到云服务器的任务数量分布。当用户产生的任务数量增加时，系统的计算压力会增大，导致系统时延增加，同时也需要将更多的任务上传到云服务器上进行处理。如图 9 所示，用户产生的任务越多，所有算法的系统时延都呈上升趋势，其中贪婪算法和非协作卸载算法上升趋势明显，因为在缓存资源有限的情况下，依靠单独 RSU 来处理对应任务效率低，只有通过多 RSU 合作来处理相应任务才能获得更低的系统时延，相对于单智能体缓存算法，多智能体算法同时考虑每一个 RSU 的任务情况，因此能够获得更低的系统时延。图 10 表明用户产生任务的增加导致更多任务需要上传到云服务器处理。从图 10 可以看出，用户任务在 3 到 6 时增长趋势较缓慢，而当用户任务继续增加后，由于 RSU 缓存空间有限，所有算法都需要将大量的任务上传到云服务器，因此输出任务数量增长趋势会更加明显。

5　结束语

本文在车联网边缘计算系统中联合考虑了用户任务缓存和边缘任务调度问题，并将其建模成一个混合整数非线性规划问题，从而最小化系统时延。为了降低问题求解的计算复杂度，本文提出一种双层的多 RSU 协同缓存框架将问题进行解耦，其中外层采用多智能体元强化学习方法，在每个本地智能体进行决策学习的同时，采用

LSTM 作为元智能体来平衡本地决策并加速学习，从而得到最优的 RSU 缓存策略；在缓存策略确定后，内层采用拉格朗日乘子法求解最佳协同卸载策略，实现 RSU 间的任务分配。基于杭州真实交通数据的实验表明，本文提出的算法具有很好的能效性能，并且能够在大规模任务流下保持网络稳健性。

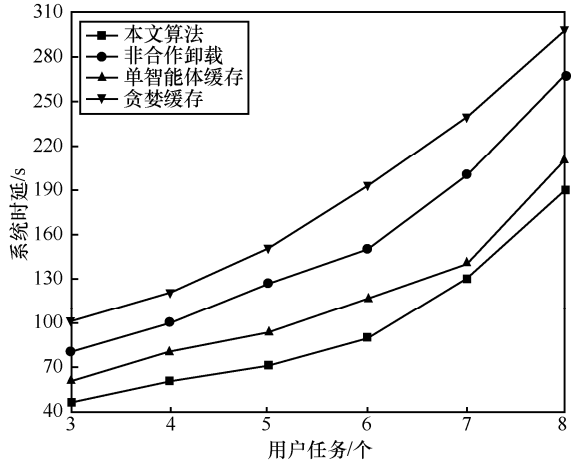

图 9　不同任务数下系统时延对比

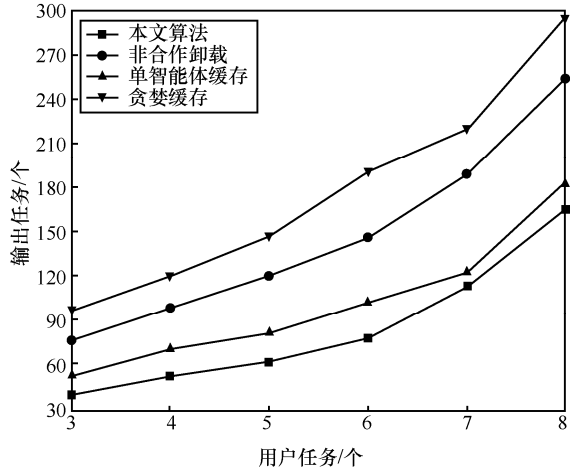

图 10　不同任务数下输出任务情况对比

参考文献：

[1] NING Z L, DONG P R, WANG X J, et al. Mobile edge computing enabled 5G health monitoring for Internet of medical things: a decentralized game theoretic approach[J]. IEEE Journal on Selected Areas in Communications, 2021, 39(2): 463-478.

[2] 谢人超, 廉晓飞, 贾庆民, 等. 移动边缘计算卸载技术综述[J]. 通信学报, 2018, 39(11): 138-155.
XIE R C, LIAN X F, JIA Q M, et al. Survey on computation offloading in mobile edge computing[J]. Journal on Communications, 2018, 39(11): 138-155.

[3] ZHUANG W H, YE Q, LYU F, et al. SDN/NFV-empowered future IoV with enhanced communication, computing, and caching[J]. Proceedings of the IEEE, 2020, 108(2): 274-291.

[4] WANG X J, NING Z L, GUO S. Multi-agent imitation learning for pervasive edge computing: a decentralized computation offloading algorithm[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(2): 411-425.

[5] 刘雷, 陈晨, 冯杰, 等. 车载边缘计算中任务卸载和服务缓存的联合智能优化[J]. 通信学报, 2021, 42(1): 18-26.
LIU L, CHEN C, FENG J, et al. Joint intelligent optimization of task offloading and service caching for vehicular edge computing[J]. Journal on Communications, 2021, 42(1): 18-26.

[6] CHEN Z Q, DUAN L Y, WANG S Q, et al. Toward knowledge as a service over networks: a deep learning model communication paradigm[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(6): 1349-1363.

[7] 张彦, 张科, 曹佳钰. 边缘智能驱动的车联网[J]. 物联网学报, 2018, 2(4): 40-48.
ZHANG Y, ZHANG K, CAO J Y. Internet of vehicles empowered by edge intelligence[J]. Chinese Journal on Internet of Things, 2018, 2(4): 40-48.

[8] WANG X J, NING Z L, GUO S, et al. Imitation learning enabled task scheduling for online vehicular edge computing[J]. IEEE Transactions on Mobile Computing, 2020, PP(99): 1.

[9] NING Z L, DONG P R, WANG X J, et al. When deep reinforcement learning meets 5G-enabled vehicular networks: a distributed offloading framework for traffic big data[J]. IEEE Transactions on Industrial Informatics, 2020, 16(2): 1352-1361.

[10] NING Z L, DONG P R, WANG X J, et al. Deep reinforcement learning for vehicular edge computing: an intelligent offloading system[J]. ACM Transactions on Intelligent Systems and Technology, 2019, 10(6): 60.

[11] TRAN T X, HAJISAMI A, PANDEY P, et al. Collaborative mobile edge computing in 5G networks: new paradigms, scenarios, and challenges[J]. IEEE Communications Magazine, 2017, 55(4): 54-61.

[12] 亓晋, 孙海蓉, 巩锟, 等. 移动边缘计算中基于信誉值的智能计算卸载模型研究[J]. 通信学报, 2020, 41(7): 141-151.
QI J, SUN H R, GONG K, et al. Research on intelligent computing offloading model based on reputation value in mobile edge computing[J]. Journal on Communications, 2020, 41(7): 141-151.

[13] XU J, CHEN L X, ZHOU P. Joint service caching and task offloading for mobile edge computing in dense networks[C]//IEEE INFOCOM 2018 - IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2018: 207-215.

[14] WANG X J, NING Z L, WANG L. Offloading in Internet of vehicles: a fog-enabled real-time traffic management system[J]. IEEE Transactions on Industrial Informatics, 2018, 14(10): 4568-4578.

[15] CHEN M, HAO Y X. Task offloading for mobile edge computing in software defined ultra-dense network[J]. IEEE Journal on Selected Areas in Communications, 2018, 36(3): 587-597.

[16] TAN L T, HU R Q, HANZO L. Twin-timescale artificial intelligence aided mobility-aware edge caching and computing in vehicular networks[J]. IEEE Transactions on Vehicular Technology, 2019, 68(4): 3086-3099.

[17] DAI Y Y, XU D, MAHARJAN S, et al. Joint computation offloading and user association in multi-task mobile edge computing[J]. IEEE Transactions on Vehicular Technology, 2018, 67(12): 12313-12325.

[18] TAO O Y, ZHI Z, XU C. Follow me at the edge: mobility-aware dynamic service placement for mobile edge computing[C]//2018 IEEE/ACM 26th International Symposium on Quality of Service. Piscataway: IEEE Press, 2018: 1-10.
[19] NING Z L, DONG P R, KONG X J, et al. A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things[J]. IEEE Internet of Things Journal, 2019, 6(3): 4804-4814.
[20] QIAO G H, LENG S P, ZHANG K, et al. Collaborative task offloading in vehicular edge multi-access networks[J]. IEEE Communications Magazine, 2018, 56(8): 48-54.
[21] NEWELL G F. Simple queueing systems[M]. Dordrecht: Springer Netherlands, 1982.
[22] HERINGS J J, PEETERS R. Stationary equilibria in stochastic games: structure, selection, and computation[J]. Research Memorandum, 2000, 118(1): 32-60.

[作者简介]

宁兆龙（1986− ），男，辽宁沈阳人，博士，重庆邮电大学教授，主要研究方向为边缘计算、车联网、网络优化等。

张凯源（1994− ），男，黑龙江哈尔滨人，大连理工大学硕士生，主要研究方向为人工智能、边缘计算。

王小洁（1988− ），女，河北张家口人，博士，重庆邮电大学特聘教授，主要研究方向为物联网、人工智能、边缘计算。

郭磊（1980− ），男，四川眉山人，博士，重庆邮电大学教授，主要研究方向为网络优化、网络通信、光网络等。