计算机科学
COMPUTER SCIENCE

车联网中基于联邦深度强化学习的任务卸载算法

林欣郁, 姚泽玮, 胡晟熙, 陈哲毅, 陈星

引用本文

林欣郁, 姚泽玮, 胡晟熙, 陈哲毅, 陈星. 车联网中基于联邦深度强化学习的任务卸载算法[J]. 计算机科学, 2023, 50(9): 347-356.

LIN Xinyu, YAO Zewei, HU Shengxi, CHEN Zheyi, CHEN Xing. Task Offloading Algorithm Based on Federated Deep Reinforcement Learning for Internet of Vehicles [J]. Computer Science, 2023, 50(9): 347-356.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220800243

车联网中基于联邦深度强化学习的任务卸载算法

林欣郁　姚泽玮　胡晟熙　陈哲毅　陈　星
福州大学计算机与大数据学院　福州 350108
福建省网络计算与智能信息处理重点实验室　福州 350108
(linxy0826@163.com)

摘　要　随着车联网应用服务体系日益丰富,计算资源有限的车辆难以处理这些计算密集和时延敏感的车联网应用。计算卸载作为移动边缘计算中的一种关键技术可以解决这一难题。对于车联网中动态的多车辆多路侧单元的任务卸载环境,提出了一种基于联邦深度强化学习的任务卸载算法。该算法将每辆车都看作是智能体,采用联邦学习的框架训练各智能体,各智能体分布式决策卸载方案,以最小化系统的平均响应时间。设置评估实验,在多种动态变化的场景下对提出的算法的性能进行对比分析。实验结果显示,提出的算法求解出的系统平均响应时间短于基于规则的算法和多智能体深度强化学习算法,接近于理想方案,且求解时间远短于理想方案。实验结果表明,所提算法能够在可接受的算法执行时间内求解出接近于理想方案的系统平均响应时间。

关键词:边缘计算;任务卸载;车联网;深度强化学习;联邦学习
中图法分类号　TP338

Task Offloading Algorithm Based on Federated Deep Reinforcement Learning for Internet of Vehicles

LIN Xinyu,YAO Zewei,HU Shengxi,CHEN Zheyi and CHEN Xing
College of Computer and Data Science,Fuzhou University,Fuzhou 350108,China
Fujian Provincial Key Laboratory of Networking Computing and Intelligent Information Processing,Fuzhou 350108,China

Abstract　With the rapid development of the service system of Internet of Vehicles applications,vehicles with limited computational resources have difficulty in handling these computation-intensive and latency-sensitive applications. As a key technique in mobile edge computing,task offloading can address the challenge. Specially,a task offloading algorithm based on federated deep reinforcement learning(TOFDRL) is proposed for dynamic multi-vehicle multi-road-side-unit(multi-RSU) task offloading environment in Internet of Vehicles. Each vehicle is considered as an agent,and a federated learning framework is used to train each agent. Each agent makes distributed decisions,aiming to minimize the average system response time. Evaluation experiments are set up to compare and analyze the performance of the proposed algorithm under a variety of dynamically changing scenarios. Simulation results show that the average response time of system solved by the proposed algorithm is shorter than that of the rule-based algorithm and the multi-agent deep reinforcement learning algorithm,close to the ideal scheme,and its solution time is much shorter than the ideal solution. Experimental results demonstrate that the proposed algorithm is able to solve an average system response time which is close to the ideal solution within an acceptable execution time.

Keywords　Mobile edge computing,Task offloading,Internet of Vehicles,Deep reinforcement learning,Federated learning

1　引言

随着我国车联网产业的发展,智能汽车越来越普及,为了满足人们的需求,车联网应用服务体系日益丰富,常见的车联网应用有智能语音助手、自动驾驶、增强现实等[1]。这些应用通常具有计算密集和时延敏感的特征,而车辆端的计算资源通常是有限的,难以在时延约束下完成这些计算密集型任务[2]。移动边缘计算(Mobile Edge Computing,MEC)[3]是一种正在不断发展的新兴计算范式,它在移动云计算(Mobile Cloud Computing,MCC)[4]的基础上,将计算资源从距离用户遥远的云端转移到距离用户更近的边缘,为用户提供高性能和低时延的服务,其可以作为处理计算密集和时延敏感型

到稿日期:2022-08-26　返修日期:2022-12-19
基金项目:国家自然科学基金(62072108);福建省自然科学基金杰青项目(2020J06014)
This work was supported by the National Natural Science Foundation of China(62072108) and Natural Science Foundation of Fujian Province for Distinguished Young Scholars(2020J06014).
通信作者:陈星(chenxing@fzu.edu.cn)

任务的一种解决方法。近年来，有许多工作将 MEC 应用到车联网场景中，对计算密集和时延敏感的车联网应用进行处理[5-7]。计算卸载是 MEC 中的一个关键特性，如果总是将任务都卸载到边缘，则容易造成边缘端过载，浪费车辆端的计算资源，并且需要耗费额外的传输时延，造成任务响应时间的延长，从而不能达到最优的系统平均响应时间，甚至无法满足任务的时延约束。因此，将任务或其子任务卸载到哪个计算节点，以平衡计算资源的利用和通信成本，从而达到最短的系统平均响应时间是目前面临的挑战。

由于环境具有动态变化性，任务卸载问题通常可以建模为一个混合整数非线性规划问题，它是一个 NP 难（NP-Hard）的优化问题[8]，现有的研究工作广泛使用粒子群优化算法（Particle Swarm Optimization，PSO）、遗传算法（Genetic Algorithm，GA）等启发式算法进行求解[9-12]。启发式算法通常能获得一个较好的近似最优解，但它是在线算法，运行时决策需要较长的求解时间，不满足车联网应用对时延敏感的要求。近年来，已有文献表明 MEC 中的任务卸载可以被抽象成马尔可夫决策过程（Markov Decision Process，MDP）[13]。强化学习（Reinforcement Learning，RL）的数学理论基础是 MDP[14]，并且它是一种离线算法，运行时决策只需要很短的求解时间，通常能满足车联网应用对时延敏感的要求。但是，传统的强化学习无法适应巨大的状态空间和动作空间。随着深度学习（Deep Learning，DL）的发展，近年来有许多研究将深度学习应用在强化学习中，提出了深度强化学习（Deep Reinforcement Learning，DRL）的概念，并将其应用在 MEC 中的任务卸载问题[2,15-23]。

为了符合现实情况，本文不局限于单设备的任务卸载，而是考虑了多设备同时卸载的场景。在多设备同时卸载的场景中，现有方法常在边缘进行集中式调度[17-19]，这需要非常大的状态空间。本文利用了车辆在系统中是分散的特性，使用分布式方法进行任务卸载，具体来说是把每辆车都看作一个智能体，分别决策自己的卸载策略。此外，本文使用了联邦学习的框架，以缩短训练时间，并得到一个适用于所有车辆的通用模型。

针对动态变化的多车辆多路侧单元协同的车联网任务计算环境，本文提出了一种基于联邦深度强化学习的分布式任务卸载算法。它可以事先与动态变化的环境交互训练出一个模型，再将该模型使用在运行时环境中，在可接受的算法执行时间内得到一组任务卸载策略，使得整个系统的平均响应时间近似最优。本文的主要贡献如下：

（1）本文将考虑一个动态变化的多车辆多路侧单元协同的车联网任务计算环境。其中，每个车辆和路侧单元都具有计算能力，均可以被视为计算节点。假设每辆车的到达任务可以被任意划分为多个子任务，将部分任务卸载到各个计算节点上，各个计算节点可以并行处理这些子任务。优化目标是找到一组合适的卸载策略，使得整个系统的平均响应时间最小化。

（2）本文提出了一种基于联邦深度强化学习的任务卸载算法（Task Offloading algorithm based on Federated Deep Reinforcement Learning，TOFDRL）。该算法将每辆车都视为一个智能体，由车辆自身决策任务的卸载策略。对于每个智能体，使用一种经典的深度强化学习算法深度 Q 网络（Deep Q-Networks，DQN）[24]对其进行训练，该算法可以很好地适应动态变化的环境，并能快速地在特定环境中做出近似最优的决策。此外，考虑到每个智能体之间是合作博弈的关系，它们有共同的优化目标，我们引入联邦学习的框架，既可以缩短训练时间，又可以得到一个适用于所有类型车辆的通用模型。

（3）本文对所提出的 TOFDRL 算法进行了对比实验。在不同的场景中，分别从系统的平均响应时间和算法的执行时间两方面验证所提出算法的有效性。实验结果表明，本文方法能够在可接受的算法执行时间内求解出接近于理想方案的系统平均响应时间。

本文第 2 章回顾了相关工作；第 3 章对车联网中多车辆多路侧单元协同的分布式任务卸载问题进行了形式化定义；第 4 章详细描述了本文提出的 TOFDRL 方法；第 5 章设计了仿真实验，通过对比实验验证并分析了本文算法的有效性；最后总结全文并展望未来。

2 相关工作

计算卸载是 MEC 的关键技术之一，用于解决在时延约束内处理时延敏感型和计算密集型任务这一难题，在车联网中有广阔的应用前景，近年来有大量学者对其进行了广泛研究。离线的强化学习算法不仅能在运行时环境中用较短的时延求解出近似最优的卸载策略，还能适应动态变化的环境，近年来也有大量学者将强化学习用于计算卸载。本节将简要回顾并分析强化学习算法在计算卸载中的研究现状。

Zhan 等[2]考虑了高速公路场景下行驶车辆的任务卸载，使用一种近端策略优化（Proximal Policy Optimization，PPO）的强化学习算法来决策卸载的位置，以最小化单车辆的长期任务响应时间和能耗。Zhang 等[15]考虑了单车辆多路侧单元场景中的任务卸载，使用 DQN 设计最佳卸载方案，以最大化系统的效用。Min 等[16]考虑了单设备在多边缘环境中的任务卸载，使用一种深度强化学习算法决策卸载地点与卸载比例，从而降低能耗、计算时延和任务丢弃率。但是，以上研究仅考虑了单设备情况下的任务卸载，并不适用于现实中多设备需要共享与竞争资源的情况。

Huang 等[17]考虑了多设备单边缘的任务卸载环境，使用深度强化学习在线卸载框架（Deep Reinforcement learning-based Online Offloading，DROO），在边缘端集中式学习完全卸载策略，以缩短 CPU 执行时延。Luo 等[18]考虑多车辆的任务卸载环境，设计了一个具有通信、计算、缓存和协作计算的统一框架，使用增强型 DQN 集中式调度网络中的资源，以达到最佳的资源利用率。Ke 等[19]考虑多车辆单路侧单元的任务卸载环境，提出了一种基于深度强化学习的自适应计算卸载方法（Adaptive Computation Offloading Method Based on Deep Reinforcement Learning，ACORL），以平衡能耗、带宽分配和执行时延。但是，以上研究都使用了集中式的方法进行任务卸载决策，其缺陷在于，当环境中有很多设备时，需要巨大的状态空间来表示各个设备的状态，这会显著降低模型的训练速度和决策速度，并且需要消耗大量的存储资源。

Yu 等[20]提出了双时间尺度深度强化学习（Two-timescale Deep Reinforcement Learning，2Ts-DRL）模型，以适应不同时延敏感型的任务，并使用联邦学习框架分布式训练 2Ts-DRL，通过联合优化计算卸载、资源分配和服务缓存放置，来最小化总卸载时延和网络资源使用量。Tang 等[21]提出了一种基于无模型的分布式深度强化学习算法，使得每个设备可以在不了解其他设备的任务模型和卸载决策下确定自己的卸载策略，同时能显著降低任务的平均时延和任务丢弃率。Huang 等[22]提出了一种分布式多智能体深度强化学习方案，以在确保时延约束的情况下，最大限度地减少整体能耗，此外还设计了一种联邦深度强化学习算法，以降低训练过程中的计算复杂性和信令开销。但是，以上研究均假设车辆和路侧单元的计算能力等环境因素不变，这不适用于动态变化的运行时环境。在动态不确定的车联网端边协同环境下，多车辆的任务卸载仍是一个极具价值的研究方向。

受到以上研究的启发，本文考虑动态不确定的多车辆多路侧单元协同的车联网任务计算环境，并提出了一种联邦深度强化学习算法，用于分布式地决策各车辆的卸载策略，使得整个系统的平均响应时间最小化。

3 系统模型与问题定义

本节将阐述车联网端边协同环境中的任务卸载决策问题。首先，提出了一个车联网端边协同的边缘计算系统模型，如图 1 所示。然后，在此基础上，对系统中的任务建立了数学模型。最后，将车联网中的任务卸载决策问题抽象成一个优化问题。为了方便阅读，本节给出了问题定义中的主要符号，如表 1 所列。

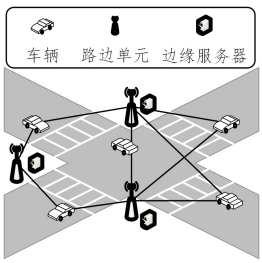


图 1 车联网中端边协同的边缘计算系统模型

Fig. 1 End-edge collaborated edge computing system in Internet of Vehicles

表 1 问题定义中的主要符号

Table 1 Main symbols in problem definition

符号	定义
N	部署的路侧单元个数
M	车辆个数
K	车辆可以连接上的路侧单元个数
$v_i \in V$	车辆节点
$e_j \in E$	路侧单元节点
λ_i	车辆 v_i 的任务到达率
μ_{v_i}	车辆 v_i 的服务率
μ_{e_j}	路侧单元 e_j 的服务率
$d_{i,j}$	车辆 v_i 和路侧单元 e_j 之间单位任务量的传输时延
$f_{i,j}$	车辆 v_i 和路侧单元 e_j 之间传输的任务量
w_{v_i}	车辆 v_i 负载的任务量
w_{e_j}	路侧单元 e_j 负载的任务量
$\rho_{\max}$	路侧单元的最大负载率

3.1 系统模型

假设在某区域内的道路上分布了 N 个路侧单元（Road Side Units，RSU），每个路侧单元都配备了一个具有计算能力的边缘服务器，可用于处理任务。路侧单元的集合记为 $E=\{e_1,e_2,\cdots,e_N\}$，其中 e_j 表示第 j 个路侧单元。该区域内有 M 辆车，记为 $V=\{v_1,v_2,\cdots,v_M\}$，其中 v_i 表示第 i 辆车。每辆车可以通过无线网络连接距离最近的 K 个路侧单元，进行通信和数据传输，并且可以使用边缘服务器的计算资源进行数据处理。

3.2 任务模型

车辆会产生任务，并且车辆与路侧单元均有计算资源可以处理任务，具有计算资源的车辆和路侧单元统称为计算节点。假设任务内部是无依赖的，在运行时可以被任意划分为多个子任务，划分任务的开销忽略不计，并且这些子任务可以卸载到 K 个路侧单元中的任意一个执行。

根据文献[19]，假设单位时间内车辆的任务到达率服从泊松分布。使用向量 $\boldsymbol{\lambda}=[\lambda_1,\lambda_2,\cdots,\lambda_M]$ 表示 M 辆车的任务到达率，即单位时间内车辆产生的任务量。使用向量 $\boldsymbol{\mu}=[\mu_{v_1},\mu_{v_2},\cdots,\mu_{v_M},\mu_{e_1},\mu_{e_2},\cdots,\mu_{e_N}]$ 表示计算节点的服务率，即单位时间内计算节点可以处理的任务量，μ_{v_i} 表示车辆 v_i 的服务率，μ_{e_j} 表示路侧单元 e_j 的服务率。M 辆车和 N 个路侧单元之间的单位任务量传输时延使用矩阵 $\boldsymbol{D}$ 表示，如式(1)所示：

$$\boldsymbol{D}=\begin{pmatrix} d_{1,0} & d_{1,1} & \cdots & d_{1,N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{M,0} & d_{M,1} & \cdots & d_{M,N} \end{pmatrix} \tag{1}$$

其中，$d_{i,0}=0$ 表示车辆 v_i 到自身的传输时延，$d_{i,j}$ 表示车辆 v_i 和路侧单元 e_j 之间的传输时延。单位时间内 M 辆车的卸载策略使用矩阵 $\boldsymbol{F}$ 表示，如式(2)所示：

$$\boldsymbol{F}=\begin{pmatrix} \boldsymbol{f}_{v_1} \\ \vdots \\ \boldsymbol{f}_{v_M} \end{pmatrix}=\begin{pmatrix} f_{1,0} & f_{1,1} & \cdots & f_{1,N} \\ \vdots & \vdots & \ddots & \vdots \\ f_{M,0} & f_{M,1} & \cdots & f_{M,N} \end{pmatrix} \tag{2}$$

其中，$f_{i,0}$ 表示车辆 v_i 在本地执行的任务量，$f_{i,j}$ 表示车辆 v_i 卸载到路侧单元的任务量。使用向量 $\boldsymbol{w}=[w_{v_1},w_{v_2},\cdots,w_{v_M},w_{e_1},w_{e_2},\cdots,w_{e_N}]$ 表示单位时间内计算节点负载的任务量，其中 w_{v_i} 表示车辆 v_i 负载的任务量，w_{e_j} 表示路侧单元 e_j 负载的任务量。假设车辆仅能将任务卸载到路侧单元或本地执行，而不能卸载到其他车辆执行，此时，车辆 v_i 负载的任务量仅与本地执行的任务量有关，即 $w_{v_i}=f_{i,0}$。路侧单元 e_j 负载的任务量与 M 辆车的卸载策略有关，因此 w_{e_j} 可以通过式(3)计算。

$$w_{e_j}=\sum_{i=1}^{M} f_{i,j} \tag{3}$$

3.3 计算模型

根据排队论中的 $M/M/n$ 排队模型，每个计算节点中单位任务量在队列中的等待时间与单位任务量所需的执行时间之和可以通过式(4)计算[25]。

$$T_a(n,\lambda,\mu)=\frac{C\left(n,\frac{\lambda}{n\mu}\right)}{n\mu-\lambda}+\frac{1}{\mu} \tag{4}$$

其中，n 表示每个计算节点拥有的服务器数量；λ 表示任务的

到达率；μ 表示计算节点中单个服务器的服务率；$C\left(n,\frac{\lambda}{n\mu}\right)$ 是 Erlang-C 公式，表示任务到达时需要排队等待处理的概率[26]，如式(5)所示：

$$C(n,\rho)=\frac{\frac{(n\rho)^n}{n!}\cdot\frac{1}{1-\rho}}{\sum_{k=0}^{n-1}\frac{(n\rho)^k}{k!}+\frac{(n\rho)^n}{n!}\cdot\frac{1}{1-\rho}} \tag{5}$$

其中，$\rho=\frac{\lambda}{n\mu}$ 表示计算节点的负载率。

本文中，每个计算节点均仅拥有一台服务器，即 $n=1$。因此，使用 $M/M/1$ 排队模型分别对每个计算节点进行建模。对于每个计算节点，单位任务量在队列中的等待时间与单位任务量所需的执行时间之和可以化简为：

$$T_a(\lambda,\mu)=\frac{1}{\mu-\lambda} \tag{6}$$

因为每个计算节点是并行处理任务量的，所以车辆 v_i 到达任务的响应时间为所有计算节点中最晚执行完成的时间，如式(7)所示：

$$T_i=\max(f_{i,0}\cdot t_{i,0},f_{i,1}\cdot t_{i,1},\cdots,f_{i,N}\cdot t_{i,N}) \tag{7}$$

其中，$t_{i,j}$ 表示将车辆 v_i 到达的任务卸载到计算节点 j。$\forall j\in\{0,1,\cdots,N\}$ 的单位任务量的平均响应时间，由单位任务量的等待与执行时间和单位任务量的传输时间组成[23]，如式(8)所示：

$$t_{i,j}=T_a(w_j,\mu_j)+d_{i,j} \tag{8}$$

其中，当 $j=0$ 时，使用车辆本地的负载和服务率进行计算，即 $w_j=w_{v_i}$，$\mu_j=\mu_{v_i}$；否则，使用对应路侧单元的负载和服务率进行计算，即 $w_j=w_{e_j}$，$\mu_j=\mu_{e_j}$。

对于整个系统，处理所有车辆到达任务的平均响应时间如式(9)所示：

$$T=\frac{1}{M}\sum_{i=1}^{M}T_i \tag{9}$$

3.4 问题定义

本文的问题定义为：在 N 个路侧单元和 M 辆车的环境中，每个路侧单元和车辆都具有计算能力，每辆车都可以连接上最近的 K 个路侧单元，并能将自身的任务部分卸载到可以连接上的路侧单元或将部分留在本地执行。每个计算节点是并行处理任务的，每辆车可以知道自身及其可以连接上的路侧单元的信息，包含任务到达率、服务率、负载率和单位任务量的传输时延。本文的优化目标是找到一组卸载策略 $\boldsymbol{F}$，以最小化整个系统中所有车辆到达任务的响应时间，如式(10)所示。其中，约束条件 C1 表示每个任务必须被完整划分并卸载到各个可用的计算节点；C2 表示卸载到路侧单元的任务总量不超过其能承受的负载量；C3 表示每辆车的任务到达率小于其服务率；C4 表示每个路侧单元的服务率均大于所有车辆服务率的最大值；C5 表示车辆和路侧单元之间的单位任务量传输时延和传输任务量必须是非负的；C6 表示车辆需要处理的任务量不大于其任务到达率。

$$\begin{aligned}&\min_{F}\ T\\&\text{s.t. C1}:\sum_{j=0}^{N}f_{i,j}=\lambda_i>0,\forall i\in\{1,2,\cdots,M\}\\&\quad\text{C2}:0\leqslant w_{e_j}\leqslant\rho_{\max}\cdot\mu_{e_j},\forall j\in\{1,2,\cdots,N\}\\&\quad\text{C3}:0<\lambda_i<\mu_{v_i},\forall i\in\{1,2,\cdots,M\}\\&\quad\text{C4}:\mu_{e_j}>\max(\mu_{v_1},\mu_{v_2},\cdots,\mu_{v_M}),\forall j\in\{1,2,\cdots,N\}\\&\quad\text{C5}:d_{i,j}\geqslant0,f_{i,j}\geqslant0,\forall i\in\{0,1,2,\cdots,M\},\\&\qquad j\in\{1,2,\cdots,N\}\\&\quad\text{C6}:0\leqslant w_{v_i}\leqslant\lambda_i,\forall i\in\{1,2,\cdots,M\}\end{aligned} \tag{10}$$

4 基于联邦深度强化学习的任务卸载算法

本节将介绍本文提出的基于联邦深度强化学习的车联网端边协同环境中的任务卸载方法。根据第 3 节构建的系统模型和问题模型，对马尔可夫决策过程进行构建，即构建状态空间、动作空间、状态转移函数和奖励函数四元组。然后，根据这个四元组，通过联邦深度强化学习对问题进行求解。

4.1 马尔可夫决策过程

MDP 是一种通过交互式学习来实现目标的理论框架，它是强化学习在数学上的理想化形式[14]。本文将每辆车都看作是一个智能体，负责与环境交互，进行学习和决策。它们之间是合作博弈的关系，每辆车之间需要博弈路侧单元的计算资源，使得自身的平均响应时间最小化，同时又要合作共享路侧单元的计算资源，使得整个系统的平均响应时间最小化。记车辆 v_i 可连接上的 K 个路侧单元为集合 $E_i=\{e_{i,1},e_{i,2},\cdots,e_{i,K}\}$，其在路侧单元集合 E 中的索引分别为 $j_1,j_2,\cdots,j_K$，即 $E_i=\{e_{j_1},e_{j_2},\cdots,e_{j_K}\}$。接着，将基于第 3 节定义的问题模型，定义车辆 v_i 的状态、动作、状态转移函数和奖励函数 MDP 四元组。

4.1.1 状态空间

状态是智能体决策的依据，通常是与优化目标相关的环境中的量。记车辆 v_i 及其可以连接上的 K 个路侧单元的服务率为 $\boldsymbol{\mu}_i=[\mu_{v_i},\mu_{e_{i,1}},\mu_{e_{i,2}},\cdots,\mu_{e_{i,K}}]$，车辆 v_i 可以连接上的 K 个路侧单元的单位任务量传输时延为 $\boldsymbol{d}_i=[d_{i,j_1},d_{i,j_2},\cdots,d_{i,j_K}]$，$t$ 时间步车辆 v_i 的到达任务的卸载方案为 $\boldsymbol{f}_i^t=[f_{i,0}^t,f_{i,j_1}^t,\cdots,f_{i,j_K}^t]$，t 时间步车辆及其可以连接上的 K 个路侧单元的负载率如式(11)所示：

$$\boldsymbol{\rho}_i^t=[\rho_{v_i}^t,\rho_{e_{i,1}}^t,\rho_{e_{i,2}}^t,\cdots,\rho_{e_{i,K}}^t]=\left[\frac{w_{v_i}^t}{\mu_{v_i}},\frac{w_{e_{i,1}}^t}{\mu_{e_{i,1}}},\frac{w_{e_{i,2}}^t}{\mu_{e_{i,2}}},\cdots,\frac{w_{e_{i,K}}^t}{\mu_{e_{i,K}}}\right] \tag{11}$$

本文将车辆 v_i 在 t 时间步的状态定义为向量 $\boldsymbol{s}_i^t=[\boldsymbol{\mu}_i,\boldsymbol{d}_i,\boldsymbol{\rho}_i^t,\boldsymbol{f}_i^t]$，分别表示车辆及其可以连接上的 K 个路侧单元的服务率、车辆到其可以连接上的 K 个路侧单元的单位任务量传输时延、车辆及其可以连接上的 K 个路侧单元当前的负载率和车辆当前的到达任务卸载方案。

不同于单车辆的环境，在多车辆的环境中，车辆的卸载决策容易受到其他车辆的影响。具体来说，当其他车辆将任务都卸载到同一个路侧单元时，当前车辆再将任务卸载到该路侧单元容易造成其过载。但是，当前车辆进行卸载决策时并不知道其他车辆的卸载策略。因此，在未知其他车辆卸载策略的条件下进行任务卸载成为了一大挑战。面对这一挑战，根据文献[27]，本文在每辆车的状态中引入了指纹。指纹是一个蕴含了其他智能体信息的低维向量，它使得各智能体可以使用其他智能体的信息进行更有效的决策，解决了在未知其他车辆卸载策略的条件下进行任务卸载的难题。该指纹由

$[e,\varepsilon]$组成，其中 e 是当前训练的轮数，ε 是 ε-greedy[28] 中的探索率。

综上所述，车辆 v_i 在 t 时间步的状态被定义为向量 $\boldsymbol{s}_i^t=[\boldsymbol{\mu}_i,\boldsymbol{d}_i,\boldsymbol{\rho}_i^t,\boldsymbol{f}_i^t,e,\varepsilon]$。

4.1.2 动作空间

动作是智能体对可优化变量的调整。本文中的智能体可调整的可优化变量是卸载策略 $\boldsymbol{F}$，记一次调整的粒度为 δ_i，即一个动作调整 δ_i 的任务量。本文将车辆 v_i 的动作空间定义为 $\boldsymbol{a}_i=[0,1,2,\cdots,K]$，车辆 v_i 在 t 时间步的动作记为 $a_i^t\in\boldsymbol{a}_i$。其中，动作 $a_i^t=0$ 表示车辆 v_i 将相应的任务量放在本地执行，有 $w_{v_i}^{t+1}\leftarrow w_{v_i}^t+\delta_i$；动作 $a_i^t=0$ 表示车辆 v_i 将相应的任务量卸载到路侧单元 e_{i,a_i^t} 上执行，有 $w_{e_{i,a_i^t}}^{t+1}\leftarrow w_{e_{i,a_i^t}}^{t+1}+\delta_i$。

4.1.3 状态转移函数

状态转移函数定义了在 t 时间步的状态 $\boldsymbol{s}_i^t$ 下执行动作 a_i^t 转移到下一状态 $\boldsymbol{s}_i^{t+1}$ 的方法。在状态 $\boldsymbol{s}_i^t$ 下执行动作 a_i^t 时，车辆及其可以连接上的 K 个路侧单元的服务率 $\boldsymbol{\mu}_i$ 和车辆到其可以连接上的 K 个路侧单元的单位任务量传输时延 $\boldsymbol{d}_i$ 不变；车辆及其可以连接上的 K 个路侧单元当前的负载率 $\boldsymbol{\rho}_i^t$ 和车辆当前的到达任务卸载方案 $\boldsymbol{f}_i^t$ 随动作 a_i^t 的变化而变化；训练的轮数 e 和探索率 ε 随训练过程而变化。具体来说，经过状态转移函数，在状态 $\boldsymbol{s}_i^t$ 下执行动作 a_i^t 可以转移到下一状态 $\boldsymbol{s}_i^{t+1}=[\boldsymbol{\mu}_i,\boldsymbol{d}_i,\boldsymbol{\rho}_i^{t+1},\boldsymbol{f}_i^{t+1},e',\varepsilon']$。其中，车辆及其可以连接上的 K 个路侧单元的负载率可通过式(12)进行转移，车辆 v_i 到达任务的卸载方案可通过式(13)进行转移。

$$\boldsymbol{\rho}_i^{t+1}=\begin{cases}\left[\dfrac{w_{v_i}^t+\delta_i}{\mu_{v_i}},\dfrac{w_{e_{i,1}}^t}{\mu_{e_{i,1}}},\dfrac{w_{e_{i,2}}^t}{\mu_{e_{i,2}}},\cdots,\dfrac{w_{e_{i,K}}^t}{\mu_{e_{i,K}}}\right] & ,a_i^t=0\\ \left[\dfrac{w_{v_i}^t}{\mu_{v_i}},\dfrac{w_{e_{i,1}}^t}{\mu_{e_{i,1}}},\cdots,\dfrac{w_{e_{i,a_i^t}}^t+\delta_i}{\mu_{e_{i,a_i^t}}},\cdots,\dfrac{w_{e_{i,K}}^t}{\mu_{e_{i,K}}}\right] & ,a_i^t\neq0\end{cases} \tag{12}$$

$$\boldsymbol{f}_i^{t+1}=[f_{i,0}^t,f_{i,j_1}^t,\cdots,f_{i,j_{a_i^t}}^t+\delta_i,\cdots,f_{i,j_K}^t] \tag{13}$$

4.1.4 奖励函数

奖励函数用于引导智能体学习在给定状态下如何执行动作才能得到最大的长期收益。由于每个智能体之间是合作博弈的关系，它们有相同的全局优化目标，因此所有智能体共享一个全局的奖励函数。通常智能体会往奖励最大化的方向学习，因此，为了使系统的平均响应时间最小化，奖励函数定义为与系统平均响应时间相关的负数，如式(14)所示：

$$r^t=r(\boldsymbol{s}_i^t,a_i^t)=T^t-T^{t+1} \tag{14}$$

其中，T^t 表示在 t 时间步系统累积的平均响应时延；T^{t+1} 表示所有智能体执行完动作 $\boldsymbol{a}^t=[a_1^t,a_2^t,\cdots,a_M^t]$ 后，在 $t+1$ 时间步系统累积的平均响应时延。T^t 和 T^{t+1} 可通过分别将 $\boldsymbol{w}^t$，$\boldsymbol{f}^t$ 和 $\boldsymbol{w}^{t+1}$，$\boldsymbol{f}^{t+1}$ 代入式(9)中计算得到。

4.2 基于联邦深度强化学习的任务卸载算法

本小节将第 4.1 节定义的 MDP 四元组应用到具体的联邦深度强化学习算法中。本文使用经典的深度强化学习算法 DQN 对智能体进行训练和运行时决策。在训练流程中，使用联邦学习框架对 DQN 进行训练；在运行时环境中，将已训练好的模型分发给各车辆进行分布式决策。算法的框架如图 2 所示，DQN 中的神经网络结构如图 3 所示。

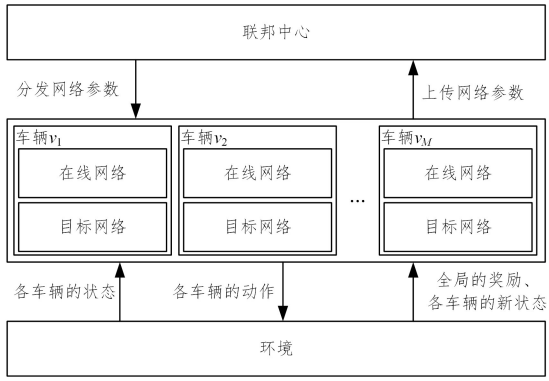


图 2 基于联邦深度强化学习的任务卸载算法框架

Fig. 2 Framework of TOFDRL

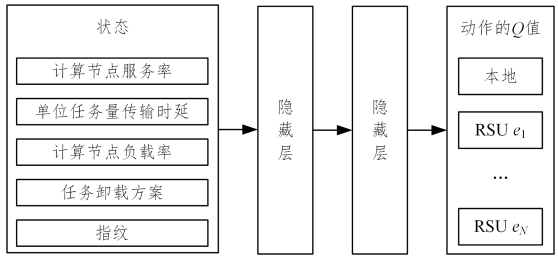


图 3 神经网络结构

Fig. 3 Structure of neural network in DQN

4.2.1 训练流程

本小节介绍了将第 4.1 节中定义的 MDP 四元组应用到具体的联邦深度强化学习算法中的训练流程，具体算法流程如算法 1 所示。

算法 1 TOFDRL 的训练流程

输入：V，E，$\boldsymbol{\lambda}$，$\boldsymbol{\mu}$，$\mathbf{D}$

输出：训练好的模型参数 θ

1. 初始化 $\varepsilon=\varepsilon_{max}$ 和 θ
2. for each $v_i\in V$ do
3. 初始化经验回放池 D_i 并设置其容量为 c
4. 初始化在线网络参数 $\theta_i=\theta$ 和目标网络参数 $\theta_i'=\theta_i$
5. end for
6. for each episode e do
7. 初始化 $T=0$，$F=O_{M,N+1}$，$w=0$
8. if e mod C=0 then
9. 根据式(15)得到中心网络参数 θ
10. 更新 $\theta_i=\theta$，$\theta_i'=\theta_i$，$\forall i\in\{1,2,\cdots,M\}$
11. end if
12. for each step t do
13. for each $v_i\in V$ do
14. 观测状态 $\mathbf{s}_i^t=[\boldsymbol{\mu}_i,\mathbf{d}_i,\boldsymbol{\rho}_i^t,\mathbf{f}_i^t,e,\varepsilon]$
15. 通过 ε-greedy 策略选取卸载动作 a_i^t
16. end for
17. 执行所有车辆的动作 $\mathbf{a}^t=[a_1^t,a_2^t,\cdots,a_M^t]$，更新 T，$\mathbf{F}$，$\mathbf{w}$
18. 所有车辆根据式(14)获得全局奖励 r^t
19. for each $v_i\in V$ do
20. 观测并转移到下一状态 $\mathbf{s}_i^{t+1}=[\boldsymbol{\mu}_i,\mathbf{d}_i,\boldsymbol{\rho}_i^{t+1},\mathbf{f}_i^{t+1},e',\varepsilon']$
21. 将经验 $(\mathbf{s}_i^t,a_i^t,r^t,\mathbf{s}_i^{t+1})$ 存入经验回放池 D_i
22. 从经验回放池中抽取经验 $(s_i,a_i,r,s_i')\sim U(D_i)$

23. 通过式(16)计算损失函数,使用 RMSprop 优化网络参数 θ_i
24. end for
25. 线性下降更新探索率 ε
26. end for
27. end for
28. 根据式(15)联邦平均网络参数,得到中心网络参数 θ
29. return 训练好的模型参数 θ

首先,初始化全局的参数,包含 ε-greedy[28] 中的探索率 ε 和联邦学习的中心网络参数 θ。然后,初始化各个智能体的参数,包含各个智能体对应的经验回放池 D_i、在线网络参数 θ_i 和目标网络参数 θ_i'。接着,开始训练。在每一轮的开始,需要初始化系统平均响应时间 T、卸载策略 F 以及车辆和路侧单元的负载 w。智能体在每 C 轮的开始需要下载最新的联邦网络参数。联邦平均网络参数如式(15)所示:

$$\theta=\frac{1}{M}\sum_{i=1}^{M}\theta_i \tag{15}$$

然后,每个智能体需要与环境进行交互学习。值得注意的是,因为智能体共享一个全局的奖励函数,所以它们需要同时执行动作以获得一个相同的全局奖励函数。每个智能体与环境交互完一步,需要将当前时间步的经验保存至对应的经验回放池 D_i 中,并从经验回放池中抽取一小批量的经验用于训练,抽取的经验记为 $(\boldsymbol{s}_i, a_i, r, \boldsymbol{s}_i')\sim U(D_i)$。参考文献[24],使用均方误差(Mean Square Error,MSE)作为损失函数,用于神经网络的训练,具体计算式如式(16)所示:

$$L(\theta_i^t)=E_{(\boldsymbol{s}_i,a_i,r,\boldsymbol{s}_i')\sim U(D_i)}[(r+\gamma\cdot\max_{a_i'}Q(\boldsymbol{s}_i',a_i';\theta_i'^t)-Q(\boldsymbol{s}_i,a_i;\theta_i^t))^2] \tag{16}$$

其中,$\gamma\in[0,1]$是对未来长期 Q 值的折扣率,γ 越接近 0,表示智能体越看重眼前的奖励,γ 越接近 1,表示智能体越看重长期的奖励;θ_i^t 是车辆 v_i 在 t 时间步时的在线网络参数;$\theta_i'^t$ 是车辆 v_i 在 t 时间步时的目标网络参数。均方根传递优化器(Root Mean Square Propagation,RMSProp)[29] 是梯度下降的一种改进算法,本文参考文献[24],应用 RMSProp 对网络参数进行优化。

4.2.2 运行时决策流程

本小节介绍了将训练好的模型使用到运行时环境的智能体决策流程,具体算法流程如算法 2 所示。首先,分发模型参数 θ,并初始化系统平均响应时间 T、卸载策略 $\boldsymbol{F}$ 以及车辆和路侧单元的负载 $\boldsymbol{w}$。然后,在每个时间步观测每辆车的状态,并使用训练好的模型根据状态决策动作,所有车辆一同执行动作,卸载各自的任务块,并更新 T,$\boldsymbol{F}$,$\boldsymbol{w}$,直到每一个任务块都被决策完毕。最后,算法返回系统的平均响应时间 T 和对应的卸载策略 $\boldsymbol{F}$。值得注意的是,在运行时环境中,不需要使用探索策略,并且只需执行一轮即可完成决策。因此,根据文献[27],状态中的 e 和 ε 均设置成最后一个训练步骤的值。

算法 2 TOFDRL 的运行时决策流程

输入:V,E,$\boldsymbol{\lambda}$,$\boldsymbol{\mu}$,$\mathbf{D}$,θ

输出:系统的平均响应时间 T 和对应的卸载策略 $\mathbf{F}$

1. 将模型参数 θ 分发给每一辆车 $\theta_i=\theta,\forall i\in\{1,2,\cdots,M\}$
2. 初始化 $T=0$,$\mathbf{F}=O_{M,N+1}$,$\mathbf{w}=0$
3. for each step t do
4. for each $v_i\in V$ do
5. 观测状态 $\mathbf{s}_i^t=[\boldsymbol{\mu}_i,\mathbf{d}_i,\boldsymbol{\rho}_i^t,\mathbf{f}_i^t,e,\varepsilon]$
6. 选取卸载动作 $a_i^t=\arg\max_a Q(\mathbf{s}_i^t,a;\theta_i)$
7. end for
8. 执行所有车辆的动作 $\mathbf{a}^t=[a_1^t,a_2^t,\cdots,a_M^t]$,更新 T,F,w
9. end for
10. return 系统的平均响应时间 T 和对应的卸载策略 $\mathbf{F}$

5 实验与分析

本节将通过仿真实验对本文算法进行性能评估。为了保证公平性,本文算法和对比算法均在 Python 3.8 的环境下实现,并在搭载了 3.00 GHz Intel Core i5-9500 CPU 芯片和 16 GB RAM 的 Windows 10 系统上运行。此外,本文的深度强化学习环境基于 TensorFlow 2.8.0 和 NumPy 1.22.3。

5.1 实验设置

在仿真实验中,考虑系统中的路侧单元数 N 为 3 或 15,车辆数 M 为 5,15 或 25,每辆车可以连接上的路侧单元个数 K 在 0 到 2 之间波动,如表 2 所列。根据文献[12,23]和对比实验验证,智能体 v_i 的动作调整粒度设置为 $\delta_i=4\%\cdot\lambda_i$。系统中车辆的任务到达率、车辆的服务率、路侧单元的服务率以及车辆与路侧单元之间的单位任务量传输时延均是动态变化的。根据香农公式,假设车辆与路侧单元之间的传输时延与它们之间的距离成正比。那么,车辆与路侧单元之间的传输时延越小,则可以认为它们之间的距离越近。假设车辆的任务到达率、车辆的服务率、路侧单元的服务率均服从正态分布,使用 $N(\mu,\sigma^2)$ 表示,其中 μ 为均值,σ^2 为方差;车辆与路侧单元之间的单位任务量传输时延的单位为 s,服从均匀分布,使用 $U(a,b)$ 表示,其中 a 为下界,b 为上界。根据文献[12,23,25],具体的模拟参数如表 2 所列。值得注意的是,根据排队论[26],车辆的任务到达率必须小于车辆的服务率,否则计算节点将无法处理完这些任务。并且,路侧单元的服务率通常大于所有车辆的服务率。为了防止路侧单元过载导致任务无法被处理完成,设置规则:当路侧单元 e_j 的负载率超过 $\rho_{\max}=85\%$ 时,路侧单元 e_j 会拒绝车辆的卸载请求,该部分任务量将放在车辆本地执行。

表 2 系统模型的模拟参数设置

Table 2 Simulation parameter settings of system model

参数	取值
路侧单元数 N	$N\in\{3,15\}$
车辆数 M	$M\in\{5,15,25\}$
车辆可以连接上的路侧单元数 K	$K\in\{0,1,2\}$
任务到达率 λ	$\lambda\sim N(10,4)<\mu_v-0.25$
车辆服务率 μ_v	$\mu_v\sim N(12,6)$
路侧单元服务率 μ_e	$\mu_v+0.25<\mu_e\sim N(15,6)$
单位任务量的传输时延 d	$d\sim U(0.1,0.2)$

5.2 实验结果

在本小节中,为了验证本文提出的 TOFDRL 算法的有效性,在不同的场景下,与 3 种已有算法从系统响应时间和算法执行时间两个方面进行对比。3 种对比算法如下。

(1)PSO-GA[12]。PSO-GA 是一种启发式算法,它在传统的 PSO 算法上,结合了 GA 的思想,引入了交叉算子和变异算子对传统的 PSO 算法中的粒子更新部分进行改进。它既有

传统 PSO 算法收敛快的特点，又结合了 GA 全局搜索的特点，可以更快地找到更优的解。PSO-GA 的参数设置如下：种群大小为 100，迭代次数为 10 000，自我学习因子 $c_1=0.5$，群体学习因子 $c_2=0.5$，惯性权重因子 $w=0.8$，变异概率 $p=0.05$。由于本文问题模型属于 NP-Hard 的问题，难以使用暴力枚举法遍历整个解空间以求得最优解。因此，PSO-GA 可以看作是理想方案。

(2)基于规则的算法(Rule-based)。基于规则的算法指根据已有的经验规则对任务进行卸载。本文参考文献[30]，采用固定百分比的规则进行任务卸载。为了避免路侧单元收到太多车辆卸载的任务量，导致其过载从而无法处理完这些任务量，制定规则如下：对于车辆 v_i 的到达任务，分别卸载 $\mu_{e_{\min}}\rho_{\max}/\sum_{i=1}^{M}\lambda_i$ 比例的任务量到可以连接上的 K 个路侧单元，剩余部分在本地执行。其中，$\mu_{e_{\min}}=\min(\mu_e)$表示 N 个路侧单元中最小的服务率。该规则满足对于任意路侧单元，M 辆车均可访问且都将其固定比例的任务量卸载到该路侧单元时，该路侧单元不会过载。

(3)多智能体深度强化学习算法(Multi-agent Deep Reinforcement Learning，MADRL)[27]。为了验证本文使用联邦学习框架进行训练的有效性，将其与不使用联邦学习框架进行训练的 MADRL 算法进行对比，各智能体仅根据局部信息进行训练和决策，无法通过联邦学习框架利用其他智能体的模型信息进行训练。

本文提出的 TOFDRL 算法中的神经网络由一层输入层、两个全连接层构成的隐藏层和一层输出层组成。其中，输入层有 13 个输入，分别对应状态中的每个元素；隐藏层中的每个全连接层都有 128 个神经元，其激活函数都是整流线性单位函数(Rectified Linear Unit，ReLU)[31]；输出层是一个有 3 个输出的全连接层，分别对应每个动作的 Q 值。算法的超参数设置如下：训练轮数 $e=3000$；根据卸载粒度 δ_i，每轮训练的步数 $t=25$；联邦平均网络参数与目标网络参数的同步频率 $C=100$；经验回放池容量 $c=65\,535$；批量大小为 32；学习率 $\alpha=10^{-3}$；折扣率 $\gamma=0.99$；探索率 ε 的初始值为 $\varepsilon_{\max}=1$，最终值为 $\varepsilon_{\min}=0.1$，在前 600 轮进行线性衰减，衰减至最终值后保持不变。

首先，在车辆所能访问的路侧单元数($K=2$)较为理想的场景下，对本文提出的 TOFDRL 算法进行评估。考虑路侧单元数 N 为 3 或 15，车辆数 M 为 5，15 或 25 的场景，在每种场景中，根据表 2 的参数设置随机生成 5 组测试数据。

图 4 给出了路侧单元数 N 为 3 时不同场景中不同卸载策略的系统平均响应时间(Average Response Time)和算法执行时间(Execution Time)。可以看出，在路侧单元数不变的情况下，随着车辆数的上升，路侧单元的计算能力有限，无法承受太多车辆卸载的任务，车辆由倾向于将任务卸载到路侧单元转变为倾向于在本地处理任务，导致系统平均响应时间也呈上升趋势。在不同车辆数的情况下，本文提出的 TOFDRL 算法得到的系统平均响应时间相比 MADRL 和 Rule-based 算法分别缩短了 36.94%和 30.03%，且接近于理想方案，仅增加了 10.63%。但是理想方案需要较长的执行时间，约为 100s 的数量级，不满足车联网中时延敏感型任务的时延约束。本文提出的 TOFDRL 算法相比理想方案仅需要很短的执行时间，仅需 0.1s 的数量级，更适合用于车联网中时延敏感型任务的卸载决策。Rule-based 算法基本不涉及数学运算，因此其算法执行时间仅需 0.001s 的数量级。但是 TOFDRL 算法得到的系统平均响应时间比 Rule-based 算法短，可以认为 TOFDRL 算法相比 Rule-based 算法是更有效的。MADRL 算法仅在训练阶段与本文提出的 TOFDRL 算法不同，在运行时环境中的卸载决策过程与 TOFDRL 算法基本相同，因此两种算法的执行时间在同一数量级，但是 TOFDRL 算法得到的系统平均响应时间短于 MADRL 算法，验证了本文使用联邦学习框架训练的有效性。

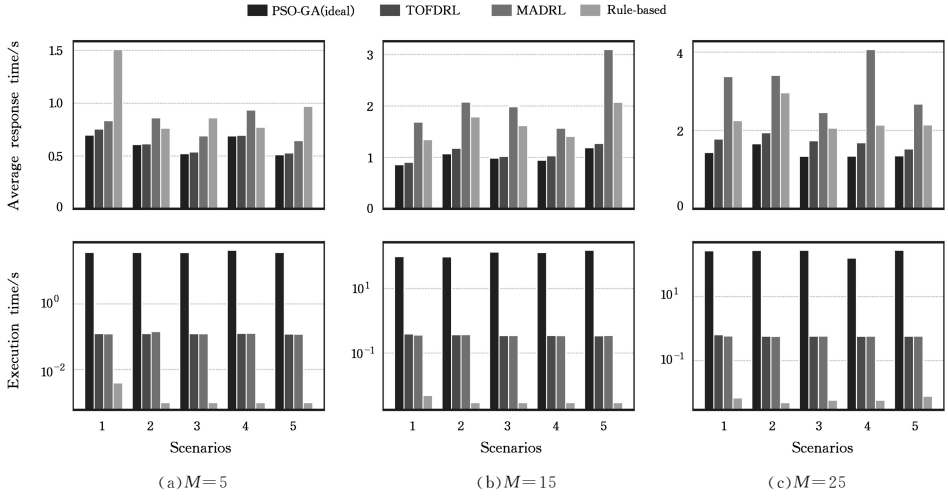


图 4 不同场景下不同卸载策略的系统平均响应时间和算法执行时间($K=2$，$N=3$)

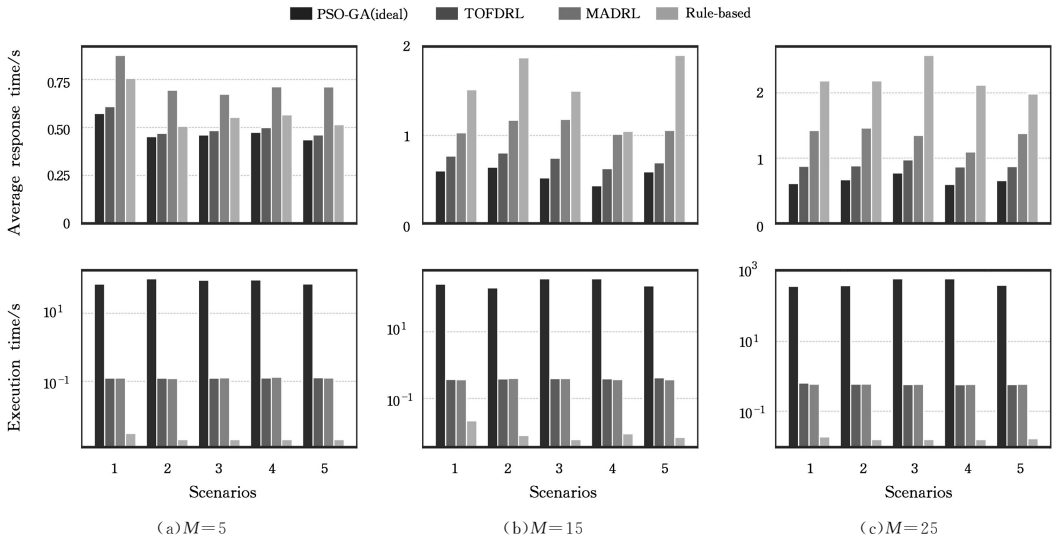
Fig. 4 Average system response time and algorithm execution time of different offloading strategies in different scenarios($K=2$，$N=3$)

图 5 给出了路侧单元数 N 为 15 时不同场景中不同卸载策略的系统平均响应时间和算法执行时间。可以看出，在不同车辆数的情况下，本文提出的 TOFDRL 算法得到的系统平均响应时间相比于 MADRL 和 Rule-based 算法分别缩短了 32.38%和 41.23%，且接近于理想方案，仅增加了 19.48%。实验结果与 $N=3$ 时类似，说明本文提出的

TOFDRL 算法同样能适用于更多路侧单元数的情况。

然后，在车辆所能访问的路侧单元在一定数目之间波动的场景($K\in\{0,1,2\}$)中，对本文提出的 TOFDRL 算法进行进一步的评估。考虑路侧单元数 N 为 3 或 15，车辆数 M 为 5，15 或 25 的场景，在每种场景中，根据表 2 中的参数设置随机生成 5 组测试数据，并且每组数据中每辆车可访问的路侧单元数在 0 到 2 之间随机波动。

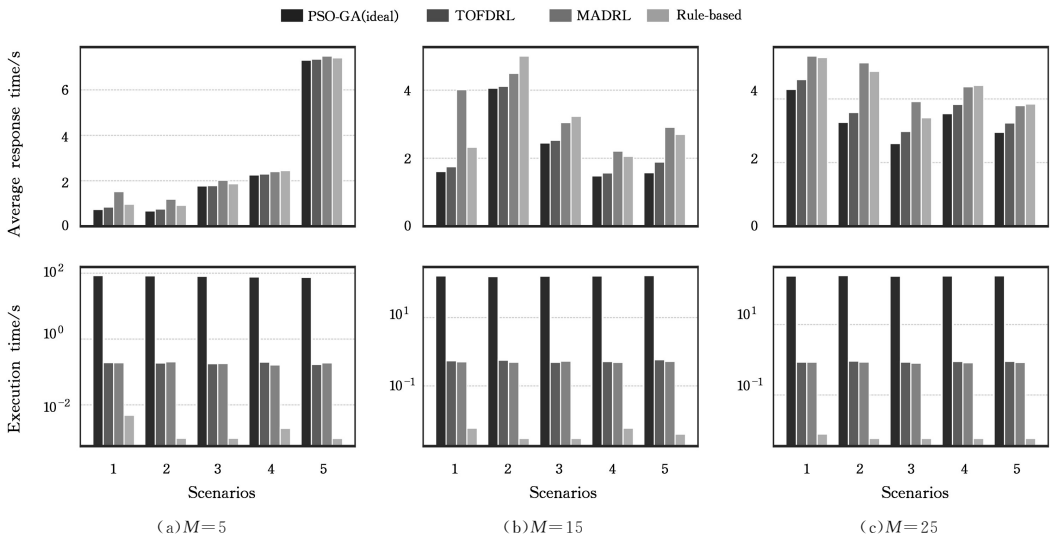
图 6 给出了路侧单元数 N 为 3 时不同场景中不同卸载策略的系统平均响应时间和算法执行时间。可以看出，在不同车辆数的情况下，本文提出的 TOFDRL 算法得到的系统平均响应时间相比于 MADRL 和 Rule-based 算法分别缩短了 22.78%和 17.01%，且接近于理想方案，仅增加了 7.69%。实验结果与 $K=2, N=3$ 时类似。不同的是，由于每辆车可访问的路侧单元在一定数目之间波动，不同车辆任务响应时间存在大范围的波动，因此不同场景下的系统平均响应时间存在大范围的波动。但是，所有算法得到的系统平均响应时间均存在波动，在不同场景中算法之间的差距基本保持不变。并且，相比 $K=2$ 的情况，在 $K\in\{0,1,2\}$的场景中，车辆可访问的边缘可能会变少，导致搜索空间减少，因此不同算法之间的差距会相对变小。



(a)$M=5$ (b)$M=15$ (c)$M=25$

图 5 不同场景下不同卸载策略的系统平均响应时间和算法执行时间($K=2, N=15$)

Fig. 5 Average system response time and algorithm execution time of different offloading strategies in different scenarios($K=2, N=15$)

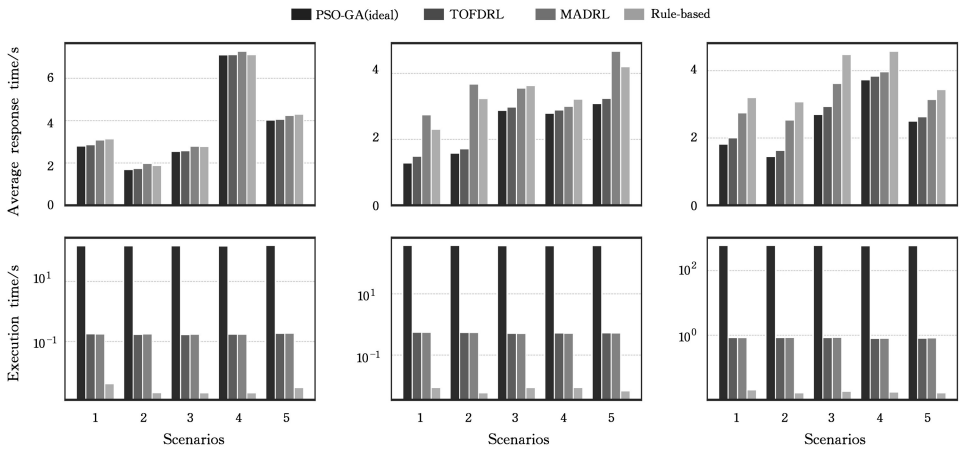


(a)$M=5$ (b)$M=15$ (c)$M=25$

图 6 不同场景下不同卸载策略的系统平均响应时间和算法执行时间($K\in\{0,1,2\}, N=3$)

Fig. 6 Average system response time and algorithm execution time of different offloading strategies in different scenarios($K\in\{0,1,2\}, N=3$)

图 7 分别给出了路侧单元数 N 为 15 时不同场景中不同卸载策略的系统平均响应时间和算法执行时间。可以看出，在不同车辆数的情况下，本文提出的 TOFDRL 算法得到的系统平均响应时间相比于 MADRL 和 Rule-based 算法分别缩短了 20.28%和 18.62%，且接近于理想方案，仅增加了5.75%。实验结果与上述实验均类似，说明本文提出的 TOFDRL 算法不仅能适应不同车辆数和不同路侧单元数的情况，还能进一步适应车辆所能访问的路侧单元在一定数目之间波动的场景。



(a)$M=5$ (b)$M=15$ (c)$M=25$

图 7 不同场景下不同卸载策略的系统平均响应时间和算法执行时间($K\in\{0,1,2\}$，$N=15$)

Fig. 7 Average system response time and algorithm execution time of different offloading strategies in different scenarios($K\in\{0,1,2\}$，$N=15$)

综上所述，在不同车辆数、不同路侧单元数和车辆所能访问的路侧单元在一定数目之间波动的场景中，本文提出的 TOFDRL 算法得到的系统平均响应时间均短于 MADRL 算法和 Rule-based 算法。虽然本文提出的 TOFDRL 算法得到的系统平均响应时间略长于理想方案，但本文方法仅需要很短的算法执行时间即可求解得到接近于理想方案的系统平均响应时间。因此，综合考虑系统平均响应时间与算法本身的开销，可以认为本文提出的 TOFDRL 算法在车联网中的时延敏感型任务卸载中是有效且优越的。

结束语 本文针对动态不确定的多车辆多路侧单元协同的车联网任务卸载问题，提出了一种联邦深度强化学习算法。它可以在未知其他车辆的任务模型和卸载决策下，分布式地训练出一个通用模型，用于各车辆分布式决策各自的卸载策略，从而最小化系统平均响应时间。并且，设计了仿真实验，验证了本文方法的有效性和优越性。实验结果表明，本文方法可以在可接受的算法执行时间内求解出接近于理想方案的系统平均响应时间。在未来的工作中，将继续研究并改进算法，以提高算法的性能，使其求解出的系统平均响应时间能够更接近于理想方案，甚至达到理想方案的水平。

参 考 文 献

[1] FENG J，LIU Z，WU C，et al. Mobileedge computing for the internet of vehicles：offloading framework and job scheduling[J]. IEEE Vehicular Technology Magazine，2019，14(1)：28-36.

[2] ZHAN W，LUO C，WANG J，et al. Deep-reinforce ment-learning-based offloading scheduling for vehicular edge computing [J]. IEEE Internet of Things Journal，2020，7(6)：5449-5465.

[3] MACH P，BECVAR Z. Mobile edge computing：a survey on architecture and computation offloading[J]. IEEE Communications Surveys & Tutorials，2017，19(3)：1628-1656.

[4] DINH H T，LEE C，NIYATO D，et al. A survey of mobile cloud computing：architecture，applications，and approaches[J]. Wireless Communications and Mobile Computing，2013，13(18)：1587-1611.

[5] RAZA S，WANG S，AHMED M，et al. Task offloading and resource allocation for iov using 5g nr-v2x communication[J]. IEEE Internet of Things Journal，2022，9(13)：10397-10410.

[6] LIN Y，ZHANG Y，LI J，et al. Popularity-aware online task offloading for heterogeneous vehicular edge computing using contextual clustering of bandits[J]. IEEE Internet of Things Journal，2022，9(7)：5422-5433.

[7] DAI P，HU K，WU X，et al. A probabilistic approach for cooperative computation offloading in mec-assisted vehicular networks [J]. IEEE Transactions on Intelligent Transportation Systems，2022，23(2)：899-911.

[8] CHEN M，HAO Y. 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.

[9] LIN B，GUO W，CHEN G. Scheduling strategy for science workflow with deadline constraint on multi-cloud[J]. Journal on Communications，2018，39(1)：56-69.

[10] HOU X，REN Z，WANG J，et al. Reliablecomputation offloading for edge-computing-enabled software-defined iov[J]. IEEE Internet of Things Journal，2020，7(8)：7097-7111.

[11] LIU Z，ZHENG H，ZHANG J，et al. Computation offloading and deployment optimization in multi-uav-enabled mobile edge computing systems[J]. Computer Science，2022，49(S1)：619-627.

[12] YAO Z，LIN J，HU J，et al. PSO-GA based approach to multi-edge load balancing[J]. Computer Science，2021，48(S2)：456-463.

[13] ALASMARI K R，II R C G，ALAM M. Mobile edge offloading using markov decision processes[C]// Edge Computing－EDGE 2018. Seattle，WA，USA，2018.

[14] SUTTON R S，BARTO A G. Reinforcement learning，second edition：an introduction[M]. Cambridge：The MIT Press，2018.

[15] ZHANG K，ZHU Y，LENG S，et al. Deep learning empowered task offloading for mobile edge computing in urban informatics [J]. IEEE Internet of Things Journal，2019，6(5)：7635-7647.

[16] MIN M，XIAO L，CHEN Y，et al. Learning-based computation

offloading for iot devices with energy harvesting[J]. IEEE Transactions on Vehicular Technology, 2019, 68(2): 1930-1941.
[17] HUANG L, BI S, ZHANG Y J A. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks[J]. IEEE Transactions on Mobile Computing, 2020, 19(11): 2581-2593.
[18] LUO Q, LI C, LUAN T H, et al. Collaborative data scheduling for vehicular edge computing via deep reinforcement learning [J]. IEEE Internet of Things Journal, 2020, 7(10): 9637-9650.
[19] KE H, WANG J, DENG L, et al. Deep reinforcement learning-based adaptive computation offloading for mec in heterogeneous vehicular networks[J]. IEEE Transactions on Vehicular Technology, 2020, 69(7): 7916-7929.
[20] YU S, CHEN X, ZHOU Z, et al. When deep reinforcement learning meets federated learning: intelligent multitimescale resource management for multiaccess edge computing in 5g ultra-dense network[J]. IEEE Internet of Things Journal, 2021, 8(4): 2238-2251.
[21] TANG M, WONG V W. Deep reinforcement learning for task offloading in mobile edge computing systems[J]. IEEE Transactions on Mobile Computing, 2022, 21(6): 1985-1997.
[22] HUANG X, LENG S, MAHARJAN S, et al. Multi-agent deep reinforcement learning for computation offloading and interference coordination in small cell networks[J]. IEEE Transactions on Vehicular Technology, 2021, 70(9): 9282-9293.
[23] CHEN X, HU J, CHEN Z, et al. A reinforcement learning-empowered feedback control system for industrial internet of things [J]. IEEE Transactions on Industrial Informatics, 2022, 18(4): 2724-2733.
[24] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529-533.
[25] JIA M, LIANG W, XU Z, et al. Qos-aware cloudlet load balancing in wireless metropolitan area networks[J]. IEEE Transactions on Cloud Computing, 2020, 8(2): 623-634.
[26] KLEINROCK L. Queueing systems, volume I: theory[M]. Hoboken: Wiley, 1975: 101-103.
[27] LIANG L, YE H, LI G Y. Spectrum sharing in vehicular networks based on multi-agent reinforcement learning[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(10): 2282-2292.
[28] WATKINS C J C H. Learning from delayed rewards[D]. Cambridge: University of Cambridge, 1989.
[29] HINTON G. Overview of mini-batch gradient descent[EB/OL]. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.
[30] WU Y, GUO K, HUANG J, et al. Secrecy-based energy-efficient data offloading via dual connectivity over unlicensed spectrums [J]. IEEE Journal on Selected Areas in Communications, 2016, 34(12): 3252-3270.
[31] NAIR V, HINTON G E. Rectified linear units improve restricted boltzmann machines[C]// Proceedings of the 27th International Conference on Machine Learning (ICML-10). Haifa, Israel, 2010: 807-814.

LIN Xinyu, born in 1999, postgraduate, is a student member of China Computer Federation. Her main research interests include computation offloading and mobile edge computing.

CHEN Xing, born in 1985, Ph.D, professor, Ph.D supervisor, is a senior member of China Computer Federation. His main research interests include software engineering, system software and cloud computing.

(责任编辑:喻藜)