计算机学报
CHINESE JOURNAL OF COMPUTERS

车联网边缘计算环境下基于深度强化学习的分布式服务卸载方法

许小龙[1),2)] 方子介[1)] 齐连永[3)] 窦万春[2)] 何强[4)] 段玉聪[5)]

[1)](南京信息工程大学计算机与软件学院，南京 210044)

[2)](南京大学计算机软件新技术国家重点实验室，南京 210023)

[3)](曲阜师范大学信息科学与工程学院，山东曲阜 273199)

[4)](斯威本科技大学计算机科学与软件工程系，墨尔本澳大利亚 3122)

[5)](海南大学计算机与网络空间安全学院，海口 570228)

摘要 边缘计算将计算、存储和带宽等资源分布到了靠近用户的一侧．通过将边缘计算引入车联网，服务提供商能为车载用户提供低延时的服务，从而提高用户出行的服务体验．然而，由于边缘服务器所配备的资源一般是有限的，不能同时支持所有车联网用户的服务需求，因此，如何在边缘服务器资源限制的约束下，确定服务卸载地点，为用户提供低时延的服务，仍然是一个巨大的挑战．针对上述问题，本文提出了一种“端-边-云”协同的 5G 车联网边缘计算系统模型，并针对该系统模型设计了深度学习和深度强化学习协同的分布式服务卸载方法 D-SOAC．首先，通过深度时空残差网络，D-SOAC 在中心云预测出潜在的用户服务需求量，协同各边缘服务器获取本地车联网边缘计算环境的系统状态，输入边缘服务器上的本地行动者网络，得到该状态下的服务卸载策略．然后，本地评论家网络基于时序差分误差，评价该服务卸载策略的优劣，并指导本地行动者网络进行网络参数的优化．优化一定步数后，边缘服务器将优化过的本地网络参数上传到位于中心云的全局网络，协同中心云进行网络参数的更新．最后，中心云将最新的参数推送回本地网络，从而不断对行动者评论家网络进行调优，获得服务卸载的最优解．基于来自现实世界的车载用户服务需求数据集的实验结果表明，在各种车联网边缘计算环境中，相比于四种现有的服务卸载算法，D-SOAC 能够降低 0.4%~20.4%的用户平均服务时延．

关键词　边缘计算；车联网；服务卸载；深度时空残差网络；异步优势行动者评论家

中图法分类号　TP311

A Deep Reinforcement Learning-BasedDistributed Service Offloading Method forEdge Computing Empowered Internet of Vehicles

XU Xiao-Long[1),2)]　FANG Zi-Jie[1)]　QI Lian-Yong[3)]　DOU Wan-Chun[2)]　HE Qiang[4)]　DUAN Yu-Cong[5)]

[1)](School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing210044)

[2)](State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023)

[3)](School of Information Science and Engineering, Qufu Normal University, Qufu 273199)

[4)](Department of Computer Science and Software Engineering, Swinburne University of Technology, Melbourne3122, Australia)

[5)](School of Computer Science and Cyberspace Security, Hainan University, Haikou 570228)

Abstract The increasing number of vehicles, along with the development of the fifth-generation (5G) wireless communication technology, has made the interconnections between vehicles and other objects (e.g., pedestrians,

本课题得到江苏省重点研发计划项目(No.BE2019104)、国家自然科学基金(No.61872219)、国家重点研发计划(No.2017YFB1400600)、新疆生产建设兵团科技计划项目(No.2020DB005)资助.许小龙，博士，教授，硕士生导师，中国计算机学会(CCF)会员(96612M)，主要研究领域为边缘计算、云计算和服务计算.E-mail:xlxu@nuist.edu.cn.方子介，学士，主要研究领域为边缘计算、深度学习.E-mail: vison307@gmail.com.齐连永(**通信作者**)，博士，教授，博士生导师，主要研究领域为服务计算、推荐系统和隐私保护.E-mail:lianyongqi@gmail.com.窦万春，博士，教授，博士生导师，主要研究领域为大数据、云计算和边缘计算.E-mail:douwc@nju.edu.cn.何强，博士，高级讲师，博士生导师，主要研究领域为边缘计算、软件工程和云计算.E-mail:qhe@swin.edu.au.段玉聪，博士，教授，博士生导师，中国计算机学会(CCF)会员(E200033712M)，主要研究领域为信息安全、人工智能和大数据.E-mail:duanyucong@hotmail.com.

infrastructures, and service platforms) become a reality, which forms a novel networking paradigm: the Internet of Vehicles (IoV). In the IoV, due to the rapid speed of the vehicles, services such as route recommendation and collision warning are required to be satisfied in time. Thanks to the birth of edge computing, which deploys resources (e.g., computation, storage, and bandwidth) at the side close to the users, thereby reducing the transmission latency and alleviating the network load, service providers can efficiently serve users with low-latency services by introducing edge computing into the IoV. Nevertheless, since the edge servers are often limited with insufficient resources, problems such as overload would occur if all the services requested by the IoV users are offloaded to the edge servers for executing, which will significantly slow down the processing speed and reduce the quality of service (QoS) provided by the edge servers. Therefore, how to allocate the limited computation and bandwidth resources of the edge servers to the IoV services and determine the offloading destinations of the services to serve the IoV users with low-latency services still remains enormous challenge. Toward this end, an end-edge-cloud collaborative computing framework for 5G-enabled IoV is proposed in this paper. Based on this framework, a distributed service offloading method with asynchronous advantage actor-critic (A3C), named D-SOAC, is developed to figure out the optimal service offloading strategy. Specifically, by leveraging the deep spatio-temporal residual network (ST-ResNet), D-SOAC predicts the future service requirements from the IoV users in each road segment firstly and sends them to the local edge server deployed in the road segment. Secondly, through combining the local future service requirements with the local network condition (e.g., transmission power and channel gain) and the local resource condition (e.g., remaining computation resources and bandwidth resources of the local edge server) into local system states, each edge server feeds the local system state into the local actor network to obtain the preliminary service offloading strategy. Technically, to avoid dimension explosion of action space in A3C, a multi-output actor network is introduced. Thirdly, based on the temporal difference (TD) error, the local critic network evaluates the preliminary offloading strategy and calculates its parameter gradient, which further guides the gradient ascent of the local actor network for gradient accumulation. After the accumulationof the parameter gradient, the local network pushes the accumulated gradient to the global network in the cloud center for parameter updating and pulls the updated global network parameters back to the local networks afterward, thereby collaborating with the global network in optimizing the preliminary service offloading strategy steadily and obtaining the optimal service offloading strategy. Eventually, extensive experimental evaluations of D-SOAC are conducted based on a real-world service requirement big dataset. The experiment results demonstrate that D-SOAC decreases the average service latency by 0.4% to 20.4% compared with four exiting service offloading methods in different IoV environments, proving the effectiveness and efficiency of D-SOAC.

Key words edge computing; Internet of vehicles; service offloading; deep spatio-temporal residual network; asynchronous advantage actor-critic

1 引言

据研究，目前全世界商用和民用车辆数之和已经超过了 10 亿．到 2035 年，预计这一数字将会达到 20 亿[1]．汽车保有量的增加，给城市带来了诸如交通拥堵、行车安全等一系列问题[2]．与此同时，互联网的迅速发展使得人们对出行的服务需求更加复杂和多样化．在此背景下，车联网（Internet of Vehicles，IoV）应运而生．基于车用无线通信技术（Vehicle to Everything，V2X），车联网将车辆、路边单元（Road-side Unit，RSU）以及服务提供商连接为一个有机的网络整体，实现了它们之间的全方位通信[3]．通过车联网，服务提供商能够获取用户服务需求和道路环境信息，基于这些数据为车载用户提供例如自动驾驶、路径规划、碰撞预警、车载娱乐等多种服务．这些服务能够有效地缓解城市中的各种道路交通问题，提高了驾驶安全性和旅途舒

适性，用户体验（Quality of Experience，QoE）也因此得以提高[4].

一般而言，搭载在车辆上的计算设备的计算能力有限，甚至某些车辆不会搭载计算设备[5]. 为此，目前最常见的解决方案是将车联网用户的服务需求卸载到云端进行处理[6]. 云平台将用户的服务需求处理完成后，再将结果传回车联网用户. 然而，由于云端和用户之间的地理距离较远，将服务卸载到云端进行处理再返回的过程往往会产生较长的时延. 与此同时，车联网中的用户通常处于高速移动的状态，这要求用户服务应在极低的时延之内完成. 如果某一些服务（例如，碰撞预警）的延迟超过了一定限度，会导致服务质量（Quality of Service, QoS）的下降，甚至交通事故的发生[7].

边缘计算作为解决此矛盾的可行方案之一，通过在路边单元（Road Side Unit, RSU）上部署边缘服务器（Edge Server，ES），将原来集中在云端的计算资源分布到靠近车联网用户一侧[8]. 因此，车联网用户和计算资源的距离得以大大缩短，所获服务的时延也得以降低. 有鉴于此，目前已有大量案例将边缘计算运用在车联网环境中. 例如，[9]中提出了一种名为 HVC（Hybrid Vehicular Edge Cloud）的分布式车联网边缘计算解决方案. 利用多路存取网络，该方法实现了路边单元和云端计算资源的有效共享. Cui 等人为车联网环境设计了一种基于区块链的容器化边缘计算平台 CUTE，该平台能够协助车联网进行资源协调和管理，从而降低用户的服务时延[10].

但是，由于边缘服务器所配备的计算、存储和带宽资源往往是有限的，很难保证将所有车联网用户的服务请求卸载到边缘服务器后，边缘服务器仍不处于过载状态. 一旦边缘服务器发生过载，服务时延将会升高，用户体验也会相应地降低[11]. 因此，某些车联网用户的服务请求仍需在云平台或者本地执行，从而保证边缘服务器资源的使用效率. 如何在满足边缘服务器资源受限的约束下，对车联网用户服务的卸载目的地进行决策，尽可能地降低服务时延，是车联网边缘计算中极具挑战性的问题之一.

此外，第五代无线通信技术（5G）的快速发展给车联网边缘计算赋予了巨大的动能. 相比于传统的第四代无线通信技术（4G），5G 能够为车联网服务提供更大的带宽、更低的延时以及更少的能耗. 更重要的是，车联网用户的移动性给网络连接的稳定性带来极大的挑战，传统的 4G 通信并不能保证用户在高速移动时网络连接依然稳定. 而如果采用基于 5G 的无线通信，即使用户的移动速度达到 500km/h，仍能保证稳定的网络连接与通信[12]. 因此，将 5G 技术引入车联网边缘计算中是十分必要的.

在车联网边缘计算环境中，网络环境、计算资源和用户服务需求等无时无刻不处在变化状态. 其中服务卸载的决策过程可以抽象为马尔科夫决策过程（Markov Decision Process，MDP）[13]. 强化学习（Reinforcement Learning，RL），作为人工智能领域的一部分，是一类通过智能体在和环境的交互过程中不断试错，学习如何得到最大收益的方法，能够有效求解马尔科夫决策问题[14]. 此外，近年来深度学习（Deep Learning，DL）的快速发展，使得计算机学习数据的高维抽象特征表示成为了可能[15]. 深度强化学习（Deep Reinforcement Learning，DRL）将深度学习和强化学习结合，较好地解决了传统强化学习无法应用于高维度状态空间和动作空间的问题，进一步提高了强化学习求解问题的能力[16].

目前为止，虽然有一些研究已将深度强化学习应用于 5G 车联网边缘计算服务卸载中，但这些研究存在着两方面的问题. 一方面是，某些研究，例如文献[17]，仅仅将强化学习作为优化目标函数，例如时延、能耗的一种手段，而并未考虑环境的动态变化，真正运用强化学习对边缘计算环境中用户服务进行长期的、动态的服务卸载决策. 另一方面，某些研究，例如文献[18]和文献[19]，虽然利用强化学习解决了服务卸载的动态决策问题，但是设计的服务卸载方法同时考虑环境中所有的边缘服务器，而没有考虑边缘服务器的分布式特征. 因此，当边缘服务器数量较多，或者服务需求量较大时，会导致状态空间和动作空间维度爆炸，造成网络参数过多，训练缓慢甚至难于训练等问题.

总的来说，如何在考虑车联网边缘计算环境动态变化、资源受限的同时，充分利用边缘计算分布式的特点，避免因边缘服务器数量或用户服务需求量较大而产生的维度爆炸问题，从而长期、稳定地为用户提供低时延的服务，是当前车联网边缘计算服务卸载的一大挑战. 为此，本文提出了一种基于深度强化学习的车联网边缘计算服务卸载方法. 特别地，考虑到边缘计算具有分布式的特征，本方法基于分布式的强化学习算法，从而提高服务卸载的

效率．本文的主要贡献包含以下三个方面：

1）提出了一种“端-边-云”协同的 5G 车联网边缘计算系统模型．该模型中，边缘服务器能够对服务卸载方法进行局部优化，而中心云负责服务卸载方法的全局优化；边缘服务器和中心云协同工作，从而实现分布式的服务卸载优化．

2）将深度时空残差网络[20]（Deep Spatio-Temporal Residual Network，ST-ResNet）和异步优势行动者评论家[21]（Asynchronous Advantage Actor-Critic，A3C）结合，提出了一种车联网边缘计算环境下分布式的服务卸载方法 D-SOAC．该方法协同了深度学习和强化学习，实验证明其能够有效地降低车联网用户长期的平均服务时延．

3）引入了一种多动作输出的行动者网络，并推导出该网络的参数梯度．解决了传统深度强化学习中，当动作由多个子动作复合而成时，动作空间维度随子动作取值数呈多项式增长，导致行动者网络参数过多、难于训练的问题．

本文其余章节的安排为：第 2 节介绍了和本文相关的研究工作，并对这些研究的成果和存在的问题做了一定分析；第 3 节对 5G 车联网边缘计算环境中的服务卸载问题建立了详细的系统模型；第 4 节提出分布式的 5G 车联网边缘计算服务卸载方法 D-SOAC；第 5 节介绍实验的参数设置和实验结果；最后总结全文，指出未来的研究工作方向．

2 相关工作

2.1 边缘计算在车联网中的应用

由于边缘计算能够很好满足车联网用户对低延迟服务的需求，近年来其在车联网中的应用得到了中外学者极大关注和广泛研究．例如，为了满足车联网环境中通信、计算和存储的多种需求，文献[22]设计了一种能耗敏感的车联网移动边缘计算（Mobile Edge Computing，MEC）调度框架．该文作者提出了一种启发式的算法，该算法同时考虑了移动边缘计算服务器的计算和网络下行能量消耗．实验表明，该方法在降低能耗、时延和任务阻塞概率方面有着极好的效果．张海波等人[23]通过整合内容分发网络（Content Delivery Network）和 MEC，利用动态信道分配算法和基于 RSU 调度的合作博弈算法，形成了一个能够合理组织中心云、边缘云和车载云资源的架构，有效降低了车联网用户的服务时延，提高了资源利用率．为了有效利用车联网中不同计算架构的优势，进而部署大规模的 IoV 系统，Wang 等人[24]提出了一种名为 CVEC 的新型车辆边缘协同计算框架．该框架能够同时为可扩展的车联网服务以及车联网应用提供水平方向和垂直方向的协作和支持．然而，这些研究在将边缘计算应用于车联网的过程中，均未考虑边缘计算的分布式特征所能给车联网带来的服务速度和灵活性方面的提升．

2.2 边缘计算中的分布式服务卸载

考虑到边缘计算环境中，一般会有多个配置在不同地理位置的边缘服务器，因此边缘计算天生具有分布式的特征，可以应用于分布式算法进行服务卸载决策．相比于传统的集中式服务卸载算法，分布式算法所要考虑的决策空间更小，因此能够更加高效、快速地解决服务卸载问题，获得了中外学者的极大关注和深入研究．文献[25]设计了一种移动边缘计算环境下的分布式服务卸载方法，该方法将卸载过程中用户和边缘云的交互过程抽象为斯塔克尔伯格博弈（Stackelberg Game）模型，旨在降低用户使用边缘服务器所产生的花费．而文献[26]则关注在服务时延约束下边缘计算系统的能耗．该文作者将卸载问题建模为一个潜在博弈（Potential Game）模型，并证明了该模型那什均衡的存在性．实验表明，该文提出的方法在环境规模增大时，能够取得较低的系统能耗．Chen 等人考虑了多用户场景中移动边缘云计算的计算卸载问题[27]．他们首先提出了一种集中式的计算卸载最优化问题，并证明了该问题是 NP-Hard．然后，他们设计了一种分布式的计算卸载模型，作为集中式计算卸载问题的替代，并使用博弈论设计了一种分布式的多用户计算卸载算法．实验表明，当用户数量增加时，该算法仍能够获得较好的性能表现．然而，上述研究均只考虑了某一时刻下的服务卸载策略，并没有考虑随着时间流逝，用户服务需求、网络环境、计算资源的动态变化导致的诸如服务质量不稳定的问题．而本文运用深度学习模型 ST-ResNet 预测用户服务需求量，协同深度强化学习方法 A3C，将车联网环境的动态变化考虑在内，构建出分布式的车联网边缘计算服务卸载方法 D-SOAC，从而为车联网环境下车载用户提供长期的高质量服务．

2.3 强化学习在车联网中的应用

近年来，将深度学习技术应用于强化学习而形成的深度强化学习方法，相比于传统的强化学习方

法有着更强的环境感知能力，在诸如电子游戏、参数优化、机器人控制等多个领域得到了广泛的研究和令人振奋的成果[28]. 如何将强化学习应用于车联网环境中，提高车联网用户的服务体验，也是当今车联网领域的一大研究方向. 文献[29]中，作者设计了一种启发式算法，来对车联网中 RSU 云资源进行有效的管理. 该文将启发式算法中选择帕累托最优解的过程，定义为马尔可夫决策过程，并将强化学习应用于此. 实验表明，该算法能够最小化长期使用的虚拟机（Virtual Machine, VM）迁移次数. 如引言所述，目前已有研究者运用深度强化学习进行车联网边缘计算服务卸载. 例如，Ning 等人[30]将车联网边缘计算环境中的通信和计算状态抽象为有限马尔科夫链，将任务规划和卸载问题表示为一个最大化用户服务体验的联合优化问题，通过运用深度强化学习方法，规划出了最佳的服务卸载和网络资源分配方案. 文献[31]在考虑车联网环境中车流量、服务需求和通信环境的变化的同时，基于 Q 学习（Q-Learning）和深度强化学习，寻找最佳的计算卸载和资源分配策略. 然而，这些研究都将车联网中所有边缘服务器和用户一次性纳入环境中. 当用户服务需求量较大，或者边缘服务器数量较多时，容易导致状态空间和动作空间维度过高，神经网络参数过多，造成训练时间的延长，甚至导致网络难于训练. 为了解决这一问题，本文基于 A3C 算法，充分发挥了边缘计算分布式的特点，每个本地网络仅负责所在环境的用户服务需求的卸载决策，最后与全局网络协同进行网络参数的更新，从而有效降低了强化学习中状态空间和动作空间的维度，减少了参数的数量，提高了训练的效率.

3 系统模型

本节首先提出了一种如图 1 所示的“端-边-云”协同的 5G 车联网边缘计算系统模型. 然后，在此基础上，对系统内的网络通信和用户服务时延建立了数学模型. 最终，我们将车联网边缘计算中的服务卸载问题抽象为一个整数规划问题. 系统模型中一些重要变量的符号表示及其含义如表 1 所示.

表 1 系统模型中的重要变量符号及其含义

变量	含义
$M(\tau)$	τ时间段内系统用户的总数量
R	道路段编号
$M_R(\tau)$	τ时间段内道路段R中用户的总数量
$U_R(\tau)$	τ时间段内道路段R的用户集
N^b	边缘服务器可分配的子信道个数
N^c	边缘服务器可分配的计算资源个数

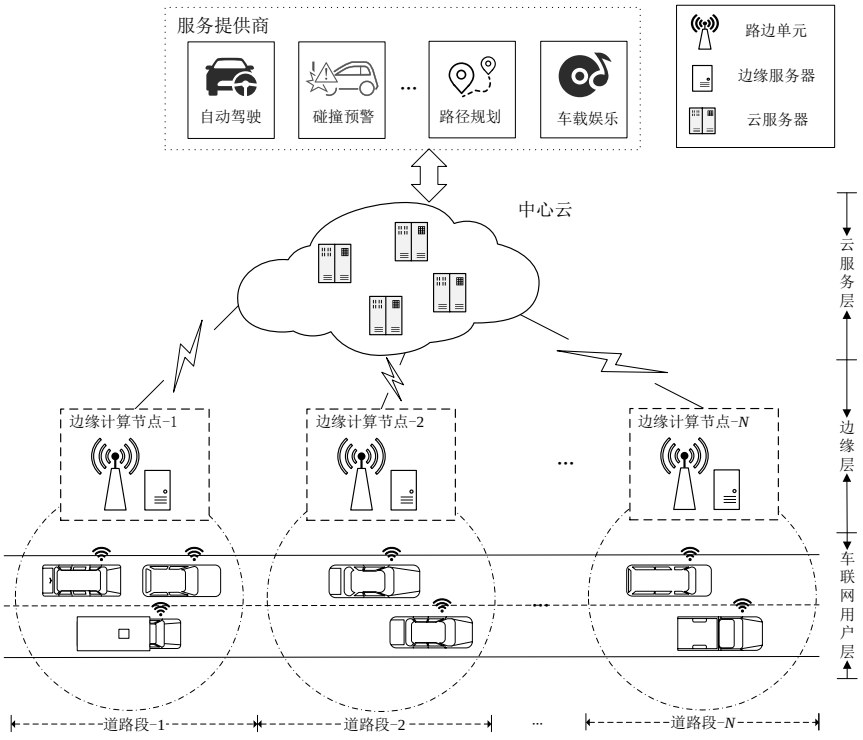

图 1“端-边-云”协同的 5G 车联网边缘计算系统模型图

3.1 “端-边-云”协同车联网边缘计算模型

车联网环境中“端-边-云”协同的 5G 边缘计算系统模型如图 1 所示．该系统分为车联网用户层、边缘层和云服务层三层．其中，车联网用户层包括了在道路上行驶的所有用户车辆，且每一用户车辆都配备了有限的计算资源．用户的服务需求可以运用本地的计算资源，在安装于本地的车载应用内满足．边缘层包括分散在道路旁的 5G 边缘计算节点，每个节点包括路边单元以及配备在该路边单元上的边缘服务器两大部分．路边单元用于收集用户服务需求、网络状况等信息，具有一定的覆盖范围，该范围将道路分为一个个不重合的道路段，车联网用户层中的用户也因此被分在唯一的道路段中．特别地，5G 边缘计算节点和用户车辆上都安装有 5G 通信设备．通过 5G 无线信道，基于毫米波（Millimeter Wave，mmWave）[32]等无线通信技术，用户车辆和相应的边缘计算节点之间形成高带宽的通信连接．在本文中，我们假设每一个路边单元均会配备一个边缘服务器，并在服务器中已经预先安装好了服务提供商提供的各类车载应用，可以满足用户卸载到边缘服务器上的服务需求．同时，用户的服务需求还可卸载到中心云执行，此时边缘计算节点会将服务需求通过有线信道转发到中心云进行计算．中心云所在的层即为云服务层，该层包含了高性能的计算资源，并且同服务提供商直接相连，能够高效、快速地满足用户的服务需求．

该系统中，时间被离散化为不同的时间段．此外，运用虚拟化技术，各个边缘服务器的计算资源可以被虚拟化为一份份独立的计算资源，分配给用户使用．在每个时间段开始时，路边单元会收集所属道路段的环境信息（例如用户服务需求、网络状况等），发送给对应的边缘服务器．边缘服务器首先根据环境信息对本地的车联网服务需求做出卸载决策．之后，其会对刚刚做出的卸载决策进行评判，学习本地的卸载策略的优化方向．当学习一定次数后，边缘服务器会将其学习到的优化知识发送给中心云．中心云对来自不同道路段的优化知识进行汇总后，可以得出优化后的卸载策略，并将其回传给各个边缘服务器进行实现．因此，卸载策略的优化过程被分配给了多个不同的本地边缘服务器同时进行处理，最后再由中心云进行处理结果的汇总．这一过程满足分布式的特征．

值得注意的是，由于车辆具有移动性，在某一时间段结束时，该车辆可能会移动到和时间段开始时不同的道路段中．我们假设在某一时间段内，用户车辆会与且仅会与时间段开始时，用户车辆所在道路段中的 5G 边缘计算节点连接．也就是说，即使在某一时刻，用户离开了在时间段开始时其所在的道路段r，但是用户车辆还会和道路段r中的 5G 边缘计算节点保持连接．因此，服务卸载决策的优化过程会付出“端-边”之间，也就是用户车辆和边缘计算节点之间稳定的通信连接的代价．考虑到 5G 技术的特性，这一代价是可以被满足的．

3.2 网络通信模型

为了更准确地对 5G 车联网边缘计算服务卸载系统中的网络通信建立计算模型，首先对车联网用户进行定义，如定义 1 所示．

定义 1. 车联网用户．车联网用户是一个三元组，记为$u_i =< \lambda_i,\ p_i,\ \delta_i >$，其中，$i$表示车联网用户的编号，$\lambda_i$是车联网用户和边缘计算节点之间的最大数据传输速率，p_i代表该用户的信号发射功率，δ_i表示该用户和边缘计算节点之间的信道增益．用$U(\tau) = \{u_1, u_2, \cdots, u_{M(\tau)}\}$表示用户集，代表车联网边缘计算环境中$\tau$时间段内所有用户的集合．

如 3.1 节所述，本系统中用户和 5G 边缘计算节点以无线方式连接．假设用户采用频分复用正交多址接入技术与边缘计算节点相连[33]，实现数据的双向传输，其中，每一个子信道的带宽为w．假设在某一道路段内，各用户与边缘计算节点之间的通信干扰可以忽略不计，则用户u_i和对应的边缘计算节点之间的最大数据传输速率为：

$$\lambda_i = n_i^b \cdot w \log_2(1 + p_i \cdot \delta_i \cdot \sigma^{-2}),\ \#(1)$$

其中，n_i^b表示分配给用户u_i的子信道数；σ代表环境中的高斯白噪声的标准差．

3.3 服务时延计算模型

在本系统中，由于用户的服务需求既可以在本地执行，亦可卸载到边缘服务器执行，亦可卸载到中心云上执行，所以需要考虑这三种不同情况下的服务时延．首先定义系统中的服务需求如下．

定义 2. 服务需求．车联网环境中的服务需求是一个三元组$s_i =< d_i, r_i, u_i >$，其中，d_i代表该服务需求所需的输入参数的数据量大小；r_i代表完成该服务需求所需的计算量；u_i表示产生该服务需求的用户．用$S(\tau) = \{s_1, s_2, \cdots, s_{M(\tau)}\}$表示车联网边缘计算环境中$\tau$时间段内所有的用户服务需求集合．

在服务需求定义的基础上，下文为不同情况下

满足用户服务需求所需的服务时延进行定义，并最终给出系统总服务时延的计算公式.

3.3.1 本地执行时用户的服务时延

当用户的服务需求在本地执行时，用户无需将该服务的输入参数数据通过无线信道上传到边缘服务器，而只需通过位于本地的车载应用，利用本地的计算资源进行处理即可. 因此，对于用户u_i和其产生的服务需求$s_i =< d_i, r_i, u_i >$，可以得到该用户在本地完成其服务需求的时延为：

$$t_i^{loc} = \frac{r_i}{f_i}, \quad \#(2)$$

其中，f_i是用户u_i所配备的本地计算资源的计算速率大小.

3.3.2 卸载到边缘服务器执行时用户的服务时延

当用户的服务需求被卸载到边缘服务器上进行处理时，需要考虑等待无线信道空闲所产生的等待时延、将服务需求输入参数从用户本地上传至边缘服务器的传输时延、在边缘服务器上处理服务需求的执行时延、以及将处理完成的服务结果返回给用户的回程时延四大部分. 考虑到服务结果相比于输入参数来说，数据量一般较小，因此回程时延一般忽略不计. 在本系统中，我们也忽略回程时延. 其余三部分中，用户$u_i =< \lambda_i, p_i, \delta_i >$将其所需求服务$s_i$的输入参数上传至边缘服务器所需的时间可由下式计算：

$$t_i^{up} = \frac{d_i}{\lambda_i}. \quad \#(3)$$

在边缘服务器上处理服务s_i的执行时延的计算公式如下所示：

$$t_i^{cal} = \frac{r_i}{n_i^c \cdot f^{MEC}}, \quad \#(4)$$

其中，n_i^c代表分配给用户u_i的计算资源数；f^{MEC}表示单个边缘服务器计算资源的计算速率.

因此，如果用户u_i的服务需求被卸载到边缘服务器执行，所产生的服务时延可被下式计算：

$$t_i^{edge} = t_i^{wait} + t_i^{up} + t_i^{cal}, \quad \#(5)$$

其中，t_i^{wait}是用户的等待时延. 对于某一个用户的服务需求来说，其等待时延即为从该用户产生该服务需求开始，直到该服务需求得到处理的时间间隔. 在本文中，由于卸载的服务需求需要等待无线信道空闲时才能进行上传，因此，服务需求开始处理的时间即为该需求被分配到无线信道的时间. 使用t_i^{start}表示用户u_i产生服务需求s_i的时间，使用t_i^{end}表示用户u_i的服务需求s_i被分配到无线信道的时间，则用户u_i的等待时延t_i^{wait}可以被计算为：

$$t_i^{wait} = t_i^{end} - t_i^{start}. \#(6)$$

3.3.3 卸载到云服务器执行时用户的服务时延

根据 3.1 节的 5G 车联网边缘计算模型，如果用户的服务需求需要被卸载到云端执行，那么该服务的输入参数首先要通过无线信道由用户车辆上传到边缘层，再由相应的边缘计算节点通过有线信道转发给云端进行处理. 考虑到云端配备的云服务器有着较强的计算能力，处理时延相较于传输时延来说可以忽略不计，因此，当用户u_i将服务s_i卸载到云端执行时产生的服务时延主要包括等待无线信道空闲的等待时延、将输入参数上传到边缘计算节点的传输时延，以及边缘计算节点和云服务器之间传输数据所产生的往返时延（Round-Trip Time，RTT）三部分，计算公式如下所示：

$$t_i^{cloud} = t_i^{wait} + \frac{d_i}{\lambda_i} + RTT. \quad \#(7)$$

由于云服务器距离边缘计算节点的地理距离较远，边缘计算节点将输入参数数据转发给云端，和云端将服务处理结果返回的过程一般会产生相近的时延，且该时延与输入参数的数据量无关. 因此，RTT可以写成：

$$RTT = 2t_{off}^{cloud}, \quad \#(8)$$

其中，t_{off}^{cloud}表示将数据从边缘服务器转发到云端所产生的时延.

3.3.4 系统总服务时延

在本系统中，每一个用户的服务需求仅可被卸载到本地、边缘服务器和云端三地中的其中一处执行. 使用 0-1 变量α_i和β_i表示某一个服务s_i的卸载情况，其中，变量α_i定义为是否在本地执行：

$$\alpha_i = \begin{cases} 1, \text{服务} s_i \text{于本地执行} \\ 0, \text{服务} s_i \text{不于本地执行} \end{cases}. \quad \#(9)$$

变量β_i定义为是否在边缘服务器执行：

$$\beta_i = \begin{cases} 1, \text{服务} s_i \text{卸载至边缘服务器执行} \\ 0, \text{服务} s_i \text{卸载至云服务器执行} \end{cases}. \quad \#(10)$$

因此，对于任意服务s_i，其服务时延为：

$$t_i = \alpha t_i^{loc} + (1-\alpha)\left[\beta t_i^{edge} + (1-\beta) t_i^{cloud}\right]. \quad \#(11)$$

所以，对于τ时间段内的所有用户来说，系统总服务时延的计算公式如下所示：

$$T(\tau)=\sum_{i=1}^{M(\tau)} t_i\ .\ \#(12)$$

3.4问题建模

在本系统中，我们要求长期一段时间内，使得系统平均时延最少的一组服务卸载策略，因此，问题可以被建模为：

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{n^b},\boldsymbol{n^c}} \lim_{\mathcal{T}\to\infty} \frac{\sum_{\tau=1}^{\mathcal{T}} T(\tau)}{\sum_{\tau=1}^{\mathcal{T}} M(\tau)},\ \#(13)$$

满足：

$$\sum_{u_i\in U_R(\tau)} n_i^b \le N^b \forall R,\forall\tau,\ \#(14)$$

$$\sum_{u_i\in U_R(\tau)} n_i^c \le N^c\ \forall R,\forall\tau,\ \#(15)$$

$$\alpha_i + \mathbb{I}_{n_i^b\ge 1} = 1\ \forall i \le M(\tau),\forall\tau,\ \#(16)$$

$$(1-\alpha_i)(1-\mathbb{I}_{n_i^c\ge 1})\beta_i = 0\ \forall i \le M(\tau),\forall\tau,\ \#(17)$$

$$\alpha_i,\beta_i \in \{0,1\}\ \forall i \le M(\tau),\forall\tau,\ \#(18)$$

其中，$\mathbb{I}_{(\cdot)}$是条件判断函数，当条件$(\cdot)$为真时，值为1，否则值为0．目标函数(13)的含义是最小化长期的用户平均服务时延．$\boldsymbol{\alpha}$，$\boldsymbol{\beta}$，$\boldsymbol{n^b}$和$\boldsymbol{n^c}$代表该最优化问题中的决策变量，即每个用户的α_i、β_i、n_i^b和n_i^c的集合．约束(14)表示在任意时间段内，分配给某道路段用户的子信道数不超过该道路段内可分配的子信道数；约束(15)表示在任意时间段内，分配给某道路段用户的计算资源数不超过该道路段边缘服务器拥有的计算资源数；约束(16)要求当分配给某一用户的子信道数为0时，该用户的服务需求必须位于本地执行；约束(17)要求当分配给某一用户的边缘服务器计算资源数为0，且不在本地执行时，该服务只能位于云端进行执行．由于该问题是一个NP-Hard非凸整数规划问题，用传统方法求其最优解的算法复杂度达到指数级．考虑到服务卸载的决策过程可以被抽象为马尔科夫决策过程，能使用深度强化学习求其近似解，因此本文接下来基于深度强化学习算法A3C求解该问题．

4 基于A3C的车联网服务卸载

本节结合ST-ResNet和A3C算法，设计了一种分布式的方法D-SOAC，来解决车联网边缘计算环境中的服务卸载问题．首先给出D-SOAC方法的总体架构，然后对其中各个部分进行详细说明，最后给出D-SOAC的完整算法．

4.1 总体架构

D-SOAC方法的总体架构主要包含两个部分，如图2所示．一部分是部署在云服务层的ST-ResNet，主要功能是根据各道路段的历史用户服务需求量预测未来的服务需求量，并将其回送给边缘计算节点供服务卸载决策时使用．另一部分包含了A3C，主要功能是根据边缘层中路边单元获取的环境信息，包括待卸载服务输入参数数据量大小、计算量大小、用户发送功率、信道增益以及未来用户服务需求量等信息，决定某用户服务需求的卸载目的地．

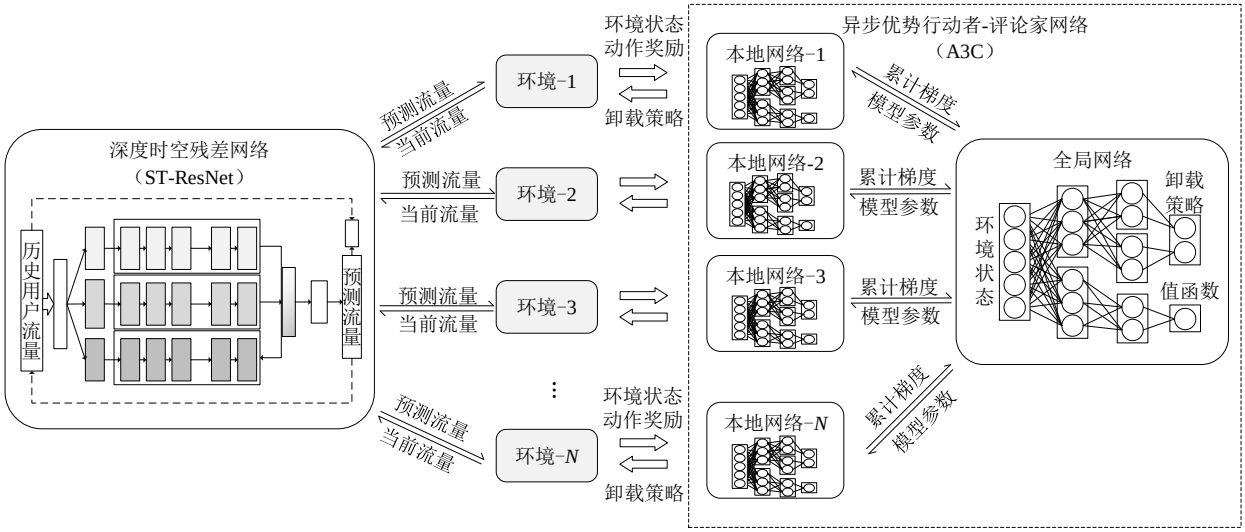

图2 D-SOAC方法总体架构

D-SOAC方法的A3C部分还能被继续分成若干个本地网络和全局网络．每个本地网络配备在边缘层中的若干5G边缘服务器上，负责本道路段中所有用户服务需求的卸载决策．而全局网络被部署

在云服务层．当本地网络做出一个卸载决策之后，边缘服务器会根据该卸载决策，返回该决策的动作奖励，本地网络再依据该动作奖励计算网络的累计梯度．当梯度累计一定次数时，边缘服务器会将自己网络的累计梯度推送到云服务器上的全局网络，用于参数更新，然后云服务器会将更新后的全局网络模型参数返回给本地网络，从而使各个本地网络不断学习得到最优的卸载策略．由于每个本地网络被部署在不同道路段中的边缘服务器上，因此该架构是一个分布式的架构，很好地适应了边缘计算分布式的特征．

4.2 基于ST-ResNet的服务需求量预测

在利用 ST-ResNet 进行服务需求量预测之前，首先对用户服务需求量进行定义．

定义 3. 用户服务需求量．车联网环境中的用户服务需求量是一个二维矩阵$\mathbf{M} \in \mathbb{R}^{\mathcal{T}\times|R|}$，其中$\mathcal{T}$表示时间段的数目，$|R|$代表系统中道路段的个数．使用$\mathbf{M}(\tau)=[M_1(\tau),M_2(\tau),\cdots,M_{|R|}(\tau)]$表示$\tau$时间段内的用户服务需求量．在某一时间段$\tau$内，某道路段$R$的服务需求量即为时间段$\tau$内道路段$R$中所有用户产生的服务需求三元组的个数，即$M_R(\tau)=\sum_{s_i\in S(\tau)}\mathbb{1}_{r_i=R}$．其中，当服务需求三元组$s_i$中的分量$r_i$和区域$R$相同时，$\mathbb{1}_{r_i=R}$的值为 1．否则其值为 0．

根据定义 3 可以看出，用户服务需求量是一种同时包含了时间属性和空间属性的典型时空数据．这种时空数据具有动态变化和随机性的特点，传统的基于统计的预测模型，例如差分整合移动平均自回归（Autoregressive Integrated Moving Average，ARIMA）模型，以及基于深度学习的模型，例如长短期记忆网络（LongShortTermMemory）已经被一系列实验证明并不能较准确地对时空数据进行预测[34][35]．但是，作为深度学习的一种方法，ST-ResNet 同时考虑数据的时间和空间特征，拥有同时提取数据之间的时间相关性和空间相关性特征的能力，已经被广泛用于预测包括用户服务需求量在内的一系列时空数据，并取得了良好的表现．例如，[36]中基于 ST-ResNet，设计了一种名为 SeqST-ResNet 的神经网络进行用户产生任务的预测．[37]中在 ST-ResNet 的基础上，设计了一种能够进行出租车服务需求预测的神经网络 DMVST-Net．综上所述，ST-ResNet 有能力完成用户服务需求预测的任务，因此在本文中，我们基于 ST-ResNet 实现服务需求量的预测，其网络结构如图 3 所示．下文对此网络结构进行具体说明．

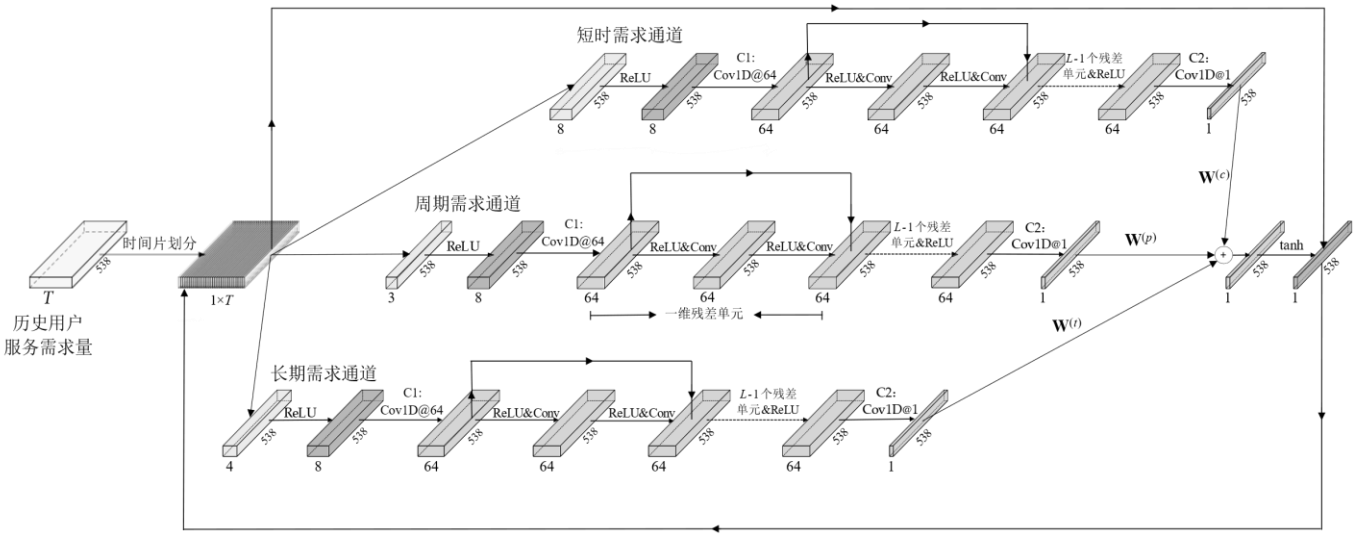

图 3ST-ResNet 的网络结构

一般来说，服务需求量具有短时不变、周期循环和长期增加的时间特征[17]．考虑到用户服务需求量的这些时间特征，ST-ResNet 从历史用户服务需求量中，提取出能表现出这三种特征的服务需求量，作为样本数据输入网络中，对网络进行训练．具体来说，对于一个标签$\mathbf{M}(\tau)$，与其对应的短时需求、周期需求和长期需求样本分别为：

$$\mathbf{M}^{(c)}(\tau)=\bigcup_{i=1}^{l_c}\mathbf{M}(\tau-i)，\ \#(19)$$

$$\mathbf{M}^{(p)}(\tau)=\bigcup_{i=1}^{l_p}\mathbf{M}(\tau-\tau_p i)，\ \#\#\#\#\#\#(20)$$

$$\mathbf{M}^{(t)}(\tau)=\bigcup_{i=1}^{l_t}\mathbf{M}(\tau-\tau_t i)\,,\quad \#\#\#\#\#(21)$$

其中，∪表示连接运算符；τ_p和τ_t分别表示周期需求和长期需求的时间跨度，l_c、l_p和l_t分别表示短时需求、周期需求长期需求的时间长度.

将具有不同的时间特征的用户服务需求量提取出来后，ST-ResNet 将它们分别输入短时需求通道、周期需求通道和长期需求通道中，提取空间特征．这三个通道是具有相同网络结构的一维卷积神经网络．特别地，为了避免网络层数过深导致的神经网络退化问题，除了第一层和最后一层之外，其余的一维卷积被替换为了一维残差单元．不失一般性，以短时需求$\mathbf{M}^{(c)}$为例进行分析．首先，$\mathbf{M}^{(c)}$作为第一个一维卷积层 C1 的输入，输出即为：

$$\mathbf{M}_{(1)}^{(c)}(\tau)=\ \mathrm{Cov1D}[\mathbf{W}_{(1)},\mathbf{M}^{(c)}(\tau)]+\mathbf{b}_{(1)},\quad \#(22)$$

其中，$\mathbf{W}_{(1)}$和$\mathbf{b}_{(1)}$分别为卷积层 C1 的权重和偏置；Cov1D表示一维卷积；下标(1)表示该变量是通道中第一层网络的参数或输出.

当短时需求通过第一个卷积层之后，会继续通过L个一维残差单元，其中第n个一维残差单元的输出可以用下式表示：

$$\mathbf{M}_{(n+1)}^{(c)}(\tau)=\mathbf{M}_{(n)}^{(c)}(\tau)+\mathrm{Res}(\mathbf{M}_{(n)}^{(c)}(\tau)),\quad \#(23)$$

其中，Res表示一维残差函数，定义为$\mathrm{Res}(x)=g(g(x))$，并且$g(x)=\mathrm{Cov1D}[\mathbf{W},\mathcal{G}(x)]+\mathbf{b}$，其中$\mathbf{W}$和$\mathbf{b}$均为可训练参数，$\mathcal{G}$表示 ReLU 函数，当自变量取值小于 0 时，值为 0，否则值等于自变量的值.

通过 L 个一维残差单元后的输出，经过 ReLU 函数激活，进入一维卷积层 C2，得到最后短时需求通道的输出，公式为：

$$\mathbf{M}_{(L+1)}^{(c)}(\tau)=\ \mathrm{Cov1D}[\mathbf{W}_{(L+1)},\mathcal{G}(\mathbf{M}_{(L)}^{(c)}(\tau))]+\mathbf{b}_{(L+1)}.\quad \#(24)$$

相似地，当周期需求和长期需求通过相应的通道之后，可以得到输出分别为：$\mathbf{M}_{(L+1)}^{(p)}(\tau)$和$\mathbf{M}_{(L+1)}^{(t)}(\tau)$．考虑到短时不变、周期循环和长期增加三种特性对未来服务需求量的影响程度不同，将这三个通道的输出分别乘以权重后进行求和，公式如下所示：

$$\mathbf{M}^{(f)}(\tau)=\sum_{i\in\{c,p,t\}}\mathbf{W}^{(i)}\odot\mathbf{M}_{(L+1)}^{(i)}(\tau)\,,\quad \#(25)$$

其中，$\mathbf{W}^{(c)}$、$\mathbf{W}^{(p)}$和$\mathbf{W}^{(t)}$分别为附加在短时需求通道输出、周期需求通道输出、以及长期需求通道输出上的权重矩阵，均为可训练参数；⊙表示逐元素乘法.

最后，经过双曲正切激活层激活，即可得到归一化后未来的用户服务需求量，如下式所示：

$$\widehat{\mathbf{M}}(\tau)=\tanh(\mathbf{M}^{(f)}(\tau))\,,\quad \#(26)$$

其中，tanh为双曲正切函数，$\tanh(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$．使用均方差函数（Mean Square Loss，MSE）作为损失函数，可以得到标签$\mathbf{M}(\tau)$的误差的计算公式为：

$$L(\theta^{st})=\frac{\left(\sum_{i=1}^{|\mathbf{M}(\tau)|}\widehat{\mathrm{M}}_i(\tau)-\mathrm{M}_i(\tau)\right)^{\frac{1}{2}}}{|\mathbf{M}(\tau)|},\quad \#(27)$$

其中θ^{st}是网络中所有可训练参数的集合，$|\cdot|$表示向量中元素的个数，在这里$|\mathbf{M}(\tau)|$即为道路段的个数$|R|$.

因此，若采用小批量梯度下降算法（Mini-Batch Gradient Descent, MBGD），一次取B个样本，并将第i个样本的误差记为$L_{(i)}(\theta^{st})$，可以使用下式对深度时空残差网络进行参数更新：

$$\theta^{st}\leftarrow\theta^{st}-\eta\cdot\frac{1}{B}\sum_{i=1}^{B}\nabla L_{(i)}(\theta^{st})\,,\quad \#(28)$$

其中η为学习率．最终，就可以找到使得$L(\theta^{st})$最小化的参数θ^{st*}．通过正向传播算法（Forward Propagation Algorithm），即可预测得到未来的用户服务需求量．若需进行多步预测，即预测未来多个时间段的用户服务需求量，则只需将预测出的单步服务需求量看作是历史需求量作为输入参数传回网络中，再进行一次正向传播即可，如算法 1 所示.

算法 1．多步用户服务需求量预测.

输入：$\mathbf{M}$、预测步数κ、τ、θ^{st*}

输出：$[\widehat{\mathbf{M}}(\tau+1),\widehat{\mathbf{M}}(\tau+2),\cdots,\widehat{\mathbf{M}}(\tau+\kappa)]$

1．$i\leftarrow 1$

2．WHILE$i\le\kappa$:

3．根据(19)~(21)计算$\mathbf{M}(\tau+i)$对应的样本$\mathbf{M}^{(c)}(\tau+i)$，$\mathbf{M}^{(p)}(\tau+i)$，$\mathbf{M}^{(t)}(\tau+i)$

4．以θ^{st*}为网络参数，将$\mathbf{M}^{(c)}(\tau+i)$，$\mathbf{M}^{(p)}(\tau+i)$，$\mathbf{M}^{(t)}(\tau+i)$输入网络，根据(22)~(26)计算得到$\widehat{\mathbf{M}}(\tau+i)$

6．$i\leftarrow i+1$

7．RETURN $[\widehat{\mathbf{M}}(\tau+1),\widehat{\mathbf{M}}(\tau+2),\cdots,\widehat{\mathbf{M}}(\tau+\kappa)]$

4.3 基于异步优势行动者评论家的服务卸载决策

强化学习是一种智能体通过和环境交互，不断试错，获得收益，从而发现能获取最大化收益的策略的机器学习方法．本文中的智能体即为各道路段

配备的边缘服务器．由于强化学习在数学上的形式一般可以用 MDP 模型进行描述，我们首先构建车联网边缘计算服务卸载的 MDP 模型，并根据该模型，将一种多动作输出的行动者网络引入 A3C 中，对传统的 A3C 算法进行了改进，最终解决了 5G 车联网边缘计算服务卸载问题．

4.3.1 MDP 模型

1）状态空间

目前，大部分利用深度强化学习进行车联网边缘计算服务卸载的工作，均以环境中某一时间段内所有车联网用户的服务需求、网络状况、以及边缘服务器的资源情况作为系统的状态，例如文献[30]、文献[31]．这种设计具有两大问题：一是在不同时间段内，用户服务需求量会发生变化，导致状态空间维度是一个不定的变量；二是，在用户服务需求量高峰时，状态空间维度较高，导致了强化学习算法难于收敛等问题．为了避免这两大问题，本文提出一种基于服务集的服务需求选择方法，使得智能体每次只固定观测到一个服务需求的状态，从而在固定状态空间维度的同时，减少了状态空间的维数．

具体来说，在边缘服务器中保存有一个服务集，在每个时间段开始时，路边单元会收集本道路段中所有用户的服务需求，并加入服务集中，同时所有用户在本地执行自己的服务需求．此时，记录当前时刻为所有用户的服务需求开始等待的时间t_i^{start}．接着，边缘服务器从服务集中随机抽取一个服务需求，提取出该服务需求的状态，交由智能体进行卸载决策，并将该服务需求从服务集中删除．此时，记录当前时刻为该被抽取出的服务需求开始进行处理的时间，即t_i^{end}．由于该服务从被加入服务集到被抽取出进行卸载决策的时间即为服务需求的等待时间t_i^{wait}，因此可以计算出$t_i^{wait} = t_i^{end} - t_i^{start}$．循环该操作，直到环境中的子信道已经被全部分配完毕．等待任一占用子信道的服务需求上传完成，空闲出子信道资源后，继续抽取服务需求，交由边缘服务器进行服务卸载决策．同时，如果在等待子信道空闲的过程中，某一在本地执行的服务需求已经执行完毕，则从服务集中去除该服务需求．采用该方法后，智能体一次只对一个服务需求进行卸载决策，这样便保证了状态空间维度较低且固定．

考虑到未来的用户服务需求量对服务卸载具有辅助作用，假设在时间步χ时环境处于时间段τ，边缘服务器选择用户u_i的服务需求进行卸载决策，那么道路段R中智能体观测到的状态$\boldsymbol{st}_R(\chi)$可表示为：

$$\boldsymbol{st}_R(\chi) = \langle \cup_{j=1}^{\kappa} M_R(\tau + j), s_i, u_i, \sigma, N^c(\chi), N^b(\chi) \rangle，\ \#(29)$$

其中κ表示用户服务需求量的预测步数；$N^c(\chi)$和$N^b(\chi)$分别表示时间步χ时可分配的计算资源数和子信道个数．简洁起见，下文在不引起歧义的情况下，将省略表示道路段的下标R。

2）动作空间

本文中，MDP 的动作空间定义为分配给用户的子信道个数和计算资源的数目，分别使用独热（one-hot）向量$(0/1)^c$和$(0/1)^b$表示．例如，假设最大子信道个数和最大计算资源个数分别为 4 和 3，那么$(0/1)^b(2) = (0,0,1,0,0)$和$(0/1)^c(1) = (0,1,0,0)$分别表示分配 2 个子信道和 1 个计算资源给当前时间步决策的用户服务需求．因此，动作空间即为两者的笛卡尔积，定义为：

$$\mathbf{A} = (0/1)^c \times (0/1)^b，\ \#(30)$$

其中，×代表笛卡尔积．我们使用$\boldsymbol{a}(\chi) \in \mathbf{A}$来表示动作空间中的在时间步$\chi$执行的一个动作．

注意到问题建模中，存在着(14)~(18)五个约束条件，但是根据(30)定义的动作空间，仅仅能保证约束(18)不被破坏．为此，本文设计了一种动作和卸载决策的映射算法，如算法 2 所示．

算法 2．动作和卸载决策的映射算法.

输入：$\boldsymbol{a}(\chi)$，剩余子信道个数$N^b(\chi)$，剩余计算资源个数$N^c(\chi)$

输出：动作对应的服务卸载策略$(\alpha, \beta, n^b, n^c)$

1．根据$\boldsymbol{a}(\chi)$计算分配子信道和计算资源个数nb、nc

2．IF$nb == 0$ OR $N^b(\chi) == 0$:

3． $\alpha \leftarrow 1$，$\beta \leftarrow 0$、$n^b \leftarrow 0$，$n^c \leftarrow 0$

4．ELSE IF $nc == 0$ OR $N^c(\chi) == 0$:

5． $\alpha \leftarrow 0$，$\beta \leftarrow 0$、$n^b \leftarrow \min(N^b(\chi), nb)$，$n^c \leftarrow 0$

6．ELSE:

7． $\alpha \leftarrow 0$，$\beta \leftarrow 1$，$n^b \leftarrow \min(N^b(\chi), nb)$，$n^c \leftarrow \min(N^c(\chi), nc)$

8．RETURN $(\alpha, \beta, n^b, n^c)$

算法 2 中，首先通过独热向量$\boldsymbol{a}(\chi)$得出该动作分配给当前服务需求的子信道个数和计算资源个数．接着进入判断，如果该动作没有为服务分配子信道，或者当前道路段的子信道已经被全部分配完，则将该服务需求卸载到本地执行，不为其分配子信道和计算资源（步骤 2、3），满足了约束

(14)(16)；如果子信道可以分配，但动作没有为当前服务需求分配计算资源，或者计算资源已经被分配完，则将该服务需求卸载到云端执行，即$\alpha=0$，$\beta=0$，为其分配的子信道个数为当前道路段中剩余的子信道个数和动作分配的子信道个数中的较小值（步骤 5），这样就满足了约束(14)和(17)；而当子信道和计算资源都能够分配时，则将该服务需求卸载到边缘服务器上执行，即$\alpha=0$，$\beta=1$，同时实际分配的子信道个数为剩余子信道个数和动作分配的子信道个数中的较小值，实际分配的计算资源数为剩余的计算资源个数和动作分配的计算资源个数中的较小值，这样就可以满足约束(14)和(15)．根据这一算法，能够保证问题中所有的约束条件均能够被满足，同时根据分配的子信道和计算资源数，即可得出决策变量α和β的取值，降低了决策变量的维度．

3）奖励函数

本文中服务卸载的目的是，使得所有用户长期的平均服务时延之和最小．定义在系统状态$\boldsymbol{st}(\chi)$时执行动作$\boldsymbol{a}(\chi)$所获得的即时奖励$\boldsymbol{r}(\chi)$为：假设$\boldsymbol{st}(\chi)$决定的是用户u_i的服务需求卸载，那么u_i产生的服务需求s_i根据动作$\boldsymbol{a}(\chi)$执行的服务卸载，相比于将s_i卸载到用户本地进行执行时能够节约的时间，即：

$$\boldsymbol{r}(\chi)=t_i^{loc}-t_i.\quad \#(31)$$

由于最大化相比于本地执行的节约时间和最小化长期用户平均服务时间本质上是一致的，因此使用(31)作为奖励函数可以求解(13).

4）MDP 过程

本文中，状态$\boldsymbol{st}(\chi)$时做出动作$\boldsymbol{a}(\chi)$的概率即服务卸载策略，记为$\pi(\boldsymbol{a}(\chi)|\boldsymbol{st}(\chi))$．例如，某边缘服务器观测到初始时间步车联网边缘计算环境处于$\boldsymbol{st}(0)$状态，并根据$\pi(\boldsymbol{a}(0)|\boldsymbol{st}(0))$选择一个服务卸载策略$\boldsymbol{a}(0)$，系统状态转移到$\boldsymbol{st}(1)$，获得奖励$\boldsymbol{r}(0)$．系统不断循环此过程，直到达到了停止条件（例如，时间段超过了上限）．整个服务卸载的过程可以表示为：

$(\boldsymbol{st}(0),\boldsymbol{a}(0),\boldsymbol{r}(0),\boldsymbol{st}(1),\boldsymbol{a}(1),\boldsymbol{r}(1),\cdots,\boldsymbol{st}(\mathrm{X}))$，

其中，$\boldsymbol{st}(\mathrm{X})$表示系统终止状态，X即终止状态的时间步．该 MDP 过程的带折扣累积奖励可以被计算为：

$$\mathcal{R}=\sum_{\chi=0}^{\mathrm{X}-1}\gamma^{\chi}\boldsymbol{r}(\chi),\quad \#(32)$$

其中γ表示折扣率，是$[0,1]$之间的常数．

深度强化学习的目的则是，寻找一个最优的策略$\pi_\theta(\boldsymbol{a}(\chi)|\boldsymbol{st}(\chi))$服从于参数$\theta$，使得 MDP 过程的带折扣累计奖励$\mathcal{R}$最大．如果在某一时间步$\chi$时，智能体观察环境正处于状态$\boldsymbol{st}(\chi)$，且根据某一策略$\pi_\theta(\boldsymbol{a}(\chi)|\boldsymbol{st}(\chi))$选择动作$\boldsymbol{a}(\chi)$，得到了一个较大的奖励$\boldsymbol{r}(\chi)$．但是该动作产生的系统状态转移使在时间步$\chi'$时智能体所能获得的奖励变得更小，即$\boldsymbol{r}(\chi')$变小，那么 MDP 过程的带折扣累计奖励$\mathcal{R}$就会相应减小，达不到最大值．因此，智能体就会改变自己的策略，执行虽然在时间步χ时获得奖励相对较低，但能使系统进行状态转移后，在时间步χ'时能获得较大奖励的动作，从而最大化带折扣累计奖励$\mathcal{R}$．因此，即使在奖励函数(31)中仅考虑某一用户节约的时间，也能达到全局时延的最优化．本文基于 A3C 来寻找能使带折扣累计奖励最大的策略，进而解决车联网边缘计算环境中的服务卸载问题．

4.3.2 基于 A3C 服务卸载方法 D-SOAC

强化学习算法一般可以分为基于值函数的方法和基于策略的方法两大类．行动者-评论家算法（Actor-Critic，AC）则将基于值函数和基于策略的方法结合起来，首先基于策略的方法，行动者观测环境产生动作，再将产生的动作送给基于值函数的评论家进行评价，从而指导智能体找到最优的动作策略．优势行动者-评论家（AdvantageActor-Critic）算法将优势函数引入了 AC 算法中，对 AC 算法进行了第一步改进．而 A3C 则在 A2C 的基础上进行了更进一步的优化，将多个不同的行动者和评论家放在不同环境中进行交互，每个行动者-评论家组合在学习一定步数后，将自己的学习经验交给一个公共的全局网络进行汇总和参数更新，然后再将全局网络中的参数返回来更新各环境中的行动者-评论家网络参数，从而指导后续的环境交互．因此，A3C 算法能够做到分布式地学习．

在 5G 车联网边缘计算环境中，原先集中在云计算中心的计算、带宽等资源被分散到了不同区域段的边缘服务器中，因此边缘服务器之间形成了一种典型的分布式架构，很好地适配了 A3C 算法的分布式并行计算特性．而 AC 和 A2C 算法均为集中式的深度强化学习算法，没有充分利用 5G 车联网边缘计算环境的分布式特征，需要同时考虑所有边缘服务器的卸载决策问题，当边缘服务器数量较多或者服务需求量较大时，会产生训练速度缓慢、状态空间、动作空间维度爆炸等问题．除此以外，AC、A2C 等集中式的强化学习算法要求服务器配备

GPU 等特殊的计算硬件，开销较大，边缘服务器也一般不会配备这些特殊的硬件.而 A3C 算法的计算强度相对较低，仅仅使用 CPU 即可完成任务，符合边缘服务器计算资源有限的特点．因此，在本文中，我们基于 A3C 算法，实现 5G 车联网边缘计算环境下的服务卸载方法 D-SOAC．

在 5G 车联网边缘计算环境中，行动者网络（策略网络）即是通过神经网络来拟合的服务卸载策略$\pi_\theta(\boldsymbol{a}(\chi)|\boldsymbol{st}(\chi))$，其中$\theta$是网络参数．对于某一行动者，它的评论家网络可以记为值函数$V^{\pi_\theta}(\boldsymbol{st};\theta^v)$，表示当使用$\pi_\theta$作为服务卸载策略时，当观察到状态$\boldsymbol{st}$后，所能获得的累计奖励的期望值，即$V^{\pi_\theta}(\boldsymbol{st})=\mathbb{E}[\sum_{i=0}^{\infty}\gamma^i\,\boldsymbol{r}(\chi+i+1)|\boldsymbol{st}(\chi)=\boldsymbol{st}]$，其中$\mathbb{E}(\cdot)$代表数学期望．在 A3C 中，评论家网络也是一个利用神经网络来拟合的函数，网络参数为θ^v．特别地，A3C 使用时序差分（Temporal-Difference，TD）误差来拟合该网络，其参数的梯度可以被下式计算：

$$\mathrm{d}\theta^v=\frac{\partial[Q(\boldsymbol{st}(\chi),\boldsymbol{a}(\chi))-V^{\pi_\theta}(\boldsymbol{st}(\chi);\theta^v)]^2}{\partial\theta^v}，\#(33)$$

其中，$Q(\boldsymbol{st}(\chi),\boldsymbol{a}(\chi))$为状态-动作值函数，一般使用单步采样近似估计，即$Q(\boldsymbol{st}(\chi),\boldsymbol{a}(\chi))=\boldsymbol{r}(\chi)+\gamma V^{\pi_\theta}(\boldsymbol{st}(\chi+1))$.

对于策略网络，它的目的是最大化带折扣累计奖励的期望．根据 4.3.1 节，策略网络决策卸载时，不仅需要决定分配给用户服务需求的子信道数量，还需要决定分配给用户服务需求的计算资源数量，因此动作空间表示为两者的笛卡尔积，这就导致了动作空间维度的复杂度为$O(N^c\cdot N^b)$，随着系统中可用子信道数量和计算资源数量的增长成平方量级增长．一旦系统中可用子信道数量和计算资源数量较多，就会导致动作空间维度过大、网络参数过多、训练缓慢等问题．为了解决这些问题，本文将传统的单动作输出策略网络替换为能够同时输出多个子动作的概率分布的策略网络，即$\pi_\theta(\boldsymbol{a}^c(\chi),\boldsymbol{a}^b(\chi)|\boldsymbol{st}(\chi))$．这样，动作空间维度的复杂度便由平方复杂度$O(N^c\cdot N^b)$变为了线性复杂度$O(N^c+N^b)$.

对于单动作输出的策略网络$\pi_\theta(\boldsymbol{a}(\chi)|\boldsymbol{st}(\chi))$，其参数的梯度可以被计算为：

$$\mathrm{d}\theta=A^\theta(\boldsymbol{st}(\chi),\boldsymbol{a}(\chi))\nabla_\theta\log\pi_\theta(\boldsymbol{a}(\chi)|\boldsymbol{st}(\chi))\,\#，(34)$$

其中，$A^\theta\big(\boldsymbol{st}(\chi),\boldsymbol{a}(\chi)\big)$是优势函数，表示在使用参数为$\theta$的策略$\pi_\theta$的情况下，执行动作的优劣性．本文中，我们使用 TD 误差衡量，即$A^\theta(\boldsymbol{st}(\chi),\boldsymbol{a}(\chi))=Q(\boldsymbol{st}(\chi),\boldsymbol{a}(\chi))-V^{\pi_\theta}(\boldsymbol{st}(\chi);\theta^v)$．假设根据策略$\pi_\theta$执行各个子动作之间的事件之间是互相独立的，即：

$$\pi_\theta(\boldsymbol{a}^c,\boldsymbol{a}^b|\boldsymbol{st})=\pi_\theta(\boldsymbol{a}^c|\boldsymbol{st},\boldsymbol{a}^b)\cdot\pi_\theta(\boldsymbol{a}^b|\boldsymbol{st})，$$

那么多动作输出策略网络的参数梯度可以由如下推导得出：

$$\begin{aligned}\mathrm{d}\theta&=A^\theta\nabla_\theta\log\pi_\theta(\boldsymbol{a}^c,\boldsymbol{a}^b|\boldsymbol{st})\\&=A^\theta\nabla_\theta\log[\pi_\theta(\boldsymbol{a}^c|\boldsymbol{st},\boldsymbol{a}^b)\pi_\theta(\boldsymbol{a}^b|\boldsymbol{st})]\\&=A^\theta\nabla_\theta[\log\pi_\theta(\boldsymbol{a}^c|\boldsymbol{st})+\log\pi_\theta(\boldsymbol{a}^b|\boldsymbol{st})]\end{aligned}$$

（方便起见，A^θ的参数和时间步χ省略不写），即多动作输出策略网络的参数梯度为：

$$\mathrm{d}\theta=A^\theta\nabla_\theta[\log\pi_\theta(\boldsymbol{a}^c|\boldsymbol{st})+\log\pi_\theta(\boldsymbol{a}^b|\boldsymbol{st})]．\#(35)$$

根据上述推导结果，可以得到基于 A3C 服务卸载方法 D-SOAC，如算法 3 所示．在 5G 车联网边缘计算环境下，每个边缘服务器均配备了行动者网络和评论家网络．路边单元通过传感器、监控摄像头等设备，收集用户服务需求量等环境信息，并将用户服务需求量发送给配置在中心云上的 ST-ResNet，预测未来的服务需求量．然后，中心云将预测的服务需求量回传给边缘服务器，边缘服务器再将其和道路段中网络发射功率、无线信道增益，以及剩余子信道和计算资源等情况组合，形成系统状态$\boldsymbol{st}(\chi)$，输入本地的行动者网络和评论家网络（步骤 7）．行动者网络负责进行卸载决策，输出动作$\boldsymbol{a}^c(\chi)$、$\boldsymbol{a}^b(\chi)$后，将该动作输入算法 2 进行服务卸载（步骤 9、10）．服务卸载完成后，智能体观测到环境的下一个状态以及该动作的奖励（步骤 11）．重复该过程若干次，直到达到了终止状态，或者超过设定的次数后，评论家网络计算该过程累积奖励的期望值（步骤 15），利用单步采样近似估计出动作-状态值函数（步骤 17），并根据(33)和(35)分别计算两个网络的累计梯度（步骤 18、19），最后将该梯度上传到位于中心云的全局网络中进行参数的更新（步骤 20）．最后，回到循环开头．云服务器将更新过后的全局网络的参数回传到边缘服务器的局部网络中进行下一幕的训练过程（步骤 5）．上述操作在所有道路段中的边缘服务器中同时、异步进行，直到超过了设定的步数后，算法终止（步骤 21），输出使(32)最大的策略网络$\pi_\theta(\boldsymbol{a}^c(\chi),\boldsymbol{a}^b(\chi)|\boldsymbol{st}(\chi))$作为服务卸载策略（步骤 22）．在算法中，使用历史的用户服务需求量来进行网络的训练过程．之后，便可以将策略网络部署在边缘服务器上，在真实环境下进行服务需求卸载．

算法 3．D-SOAC.

输入：$\mathbf{M}$，κ，N^b，N^c，χ_{max}，X_{max}，γ

输出：最优的服务卸载策略$\pi_\theta^*(\boldsymbol{a}^c(\chi),\boldsymbol{a}^b(\chi)|\boldsymbol{st}(\chi))$

1．随机初始化全局共享评论家网络参数θ^v，策略网络参数θ；初始化全局共享时间步$\mathrm{X}\leftarrow 0$

2．随机初始化各边缘服务器中的评论家网络参数$\widetilde{\theta^v}$，策略网络参数$\tilde{\theta}$；初始化边缘服务器时间步$\chi\leftarrow 1$

3．REPEAT

4．重置梯度$\mathrm{d}\theta\leftarrow 0$，$\mathrm{d}\theta^v\leftarrow 0$

5．$\tilde{\theta}\leftarrow\theta$，$\widetilde{\theta^v}\leftarrow\theta^v$

6．$\chi_{start}\leftarrow\chi$

7．根据**算法 1** 预测未来κ步服务需求量，得到$\boldsymbol{st}(\chi)$

8．REPEAT

9．根据$\pi_{\tilde{\theta}}(\boldsymbol{a}^c(\chi),\boldsymbol{a}^b(\chi)|\boldsymbol{st}(\chi))$选取动作$\boldsymbol{a}^c(\chi),\boldsymbol{a}^b(\chi)$

10．将$\boldsymbol{a}^c(\chi)$，$\boldsymbol{a}^b(\chi)$，N^b，N^c输入**算法 2** 进行服务卸载

11．获取服务卸载得到的实时奖励$\boldsymbol{r}(\chi)$，根据**算法 1** 预测未来κ步用户服务需求量，得到新状态$\boldsymbol{st}(\chi+1)$

12．$\chi\leftarrow\chi+1$

13．$\mathrm{X}\leftarrow\mathrm{X}+1$

14．UNTIL $\boldsymbol{st}(\chi)$终止状态 OR $\chi-\chi_{start}=\chi_{max}$

15．计算$Q=\begin{cases}0, & \boldsymbol{st}(\chi)\text{是终止状态}\\ V^{\pi_{\tilde{\theta}}}\left(\boldsymbol{st}(\chi);\widetilde{\theta^v}\right), & \boldsymbol{st}(\chi)\text{非终止状态}\end{cases}$

16．FOR i FROM $\chi-1$ TO χ_{start}:

17．$Q\leftarrow\boldsymbol{r}(i)+\gamma Q$

18．根据(33)计算$\mathrm{d}\theta^v$，$\Delta\theta^v\mathrel{+}=\mathrm{d}\theta^v$

19．根据(35)计算$\mathrm{d}\theta$，$\Delta\theta\mathrel{+}=\mathrm{d}\theta$

20．使用$\Delta\theta^v$、$\Delta\theta$更新全局参数θ^v、θ

21．UNTIL $\mathrm{X}>\mathrm{X}_{max}$

22．RETURN $\pi_\theta(\boldsymbol{a}^c(\chi),\boldsymbol{a}^b(\chi)|\boldsymbol{st}(\chi))$

5 实验与结果分析

本节首先对实验平台以及实验数据集进行简单说明，然后简要介绍实验参数设置．接着对 D-SOAC 方法进行了收敛性实验，证明了 D-SOAC 方法的可行性．在此基础上，针对不同的系统资源状态，我们对 D-SOAC 进行了详细的性能实验和评估，并和现有的服务卸载方法进行了对比分析．

5.1 实验平台与实验数据集

本实验的硬件环境中，CPU 使用 Intel Xeon E5-2678 v3，其包含 48 个逻辑处理器；GPU 使用 NVIDIA GeForce RTX 2080Ti．所有的实验环境均使用 Python 3.7 和 PyTorch 1.3.1 实现．

本文使用的数据集为采集自中国南京的真实车载用户服务需求数据集．数据集中包含了超过 1 亿条的车载用户服务需求信息．这些信息来自于覆盖南京市范围的 436 个路边单元，信息收集时间从 2014 年 9 月 1 日开始至 2014 年 9 月 30 日为止．数据集中每一条车载用户信息的数据格式为一个四元组：$(t,\ id, speed, color)$，其中，t表示时间，id表示覆盖该用户的路边单元编号，$speed$表示该用户车辆的速度，$color$表示车辆颜色．一条车载用户数据即代表某一用户正处于某一路边单元的覆盖范围中．在实验中，我们仅使用该四元组中前两个分量的信息，也就是时间t和路边单元编号id．运用下式，可以预处理出时间间隔τ内，在道路段R的所有用户的服务需求量为：

$$M_R(\tau)=\sum_{(t,id)\in\mathcal{D}}\mathbb{I}_{t\in\tau\wedge id\in R}\ ,\quad \#(36)$$

其中，$\mathcal{D}$表示数据集中所有的车载用户信息，(t,id)表示该数据集中某一条记录的时间和覆盖该用户的路边单元编号．$\mathbb{I}_{t\in\tau\wedge id\in R}$是一个判断函数，当记录的时间$t$处在时间段$\tau$内，并且覆盖用户的路边单元处于区域$R$中时，该函数的值为 1，否则其值为 0．

此外，数据集中，每一个路边单元编号都和一个具体的路边单元及其地理经纬度坐标对应．数据集中部分路边单元的位置分布如图 4 所示．

图 4 数据集中部分路边单元的分布位置

5.2 实验参数设置

本实验中，由于 CPU 具有 48 个逻辑处理器，因此我们从 436 个路边单元中随机抽取了 48 个路边单元，在其中配置了位于本地的行动者-评论家网络，模拟边缘服务器的工作．设置时间段长度为 2 分钟．每个用户在时间段开始时会从六种类型的服务中随机产生一种类型服务的服务需求．根据文献[18]，设置六种类型服务所需输入数据大小为[15, 25, 30, 40, 45, 60] MB，服务所需计算量大小为[1.2, 0.6, 0.7, 0.7, 0.8, 1.2] Gigacycles，产生各个服务需求的概率分别为$[0.1, 0.2, 0.3, 0.15, 0.15, 0.1]$．为了评

估 D-SOAC 方法在不同环境中的性能表现，参考文献[38]，设置不同 5G 车联网环境下每个道路段中网络的总带宽为 20GHz、30GHz 或 40GHz，而固定每一个子信道所占用的带宽为 1GHz；参考文献[39]，设置边缘服务器的计算能力为 2.5Gigacycle/s、5 Gigacycle/s 或 7.5 Gigacycle/s，而固定每份计算资源的计算能力为 0.5Gigacycle/s．因此，在不同的环境中，子信道数可能为 20、30 或 40，计算资源数可能为 5、10 或 15. 为了预测未来用户服务需求量，设置 ST-ResNet 中周期需求和长期需求的时间跨度分别为 1 天和 3 天，短时需求、周期需求和长期需求的时间长度分别为 8、3 和 4．在 ST-ResNet 的训练过程中，我们采用等间隔学习率调整策略，每隔 10 次迭代就将学习率调整为原来的 80%. 其余参数的设置如表 2 所示．

表 2 实验参数设置

变量	值
p	2 ± 0.2 W[40]
δ	144 ± 14.4 dB[41]
σ^2	1.5×10^{-8} W[40]
f	0.3 ± 0.03 Gigacycle/s
t_{off}^{cloud}	1000 ms

此外，对于 ST-ResNet 来说，由于最后的激活层为tanh函数，将输出限制在了$(-1,1)$范围内．所以需要首先对历史用户服务需求量数据进行标准化处理才能衡量误差．这里采用最大最小标准化（Min-MaxNormalization）进行数据预处理操作，定义为：

$$\mathbf{M}' = \mathrm{Nom}(\mathbf{M}) = 2\frac{\mathbf{M} - M_{min}}{M_{max} - M_{min}} - 1, \quad \#(37)$$

其中，Nom表示最大最小标准化函数，M_{min}表示$\mathbf{M}$中的最小值，M_{max}表示$\mathbf{M}$中的最大值．

在对 ST-ResNet 计算测试误差时，用均方根误差来衡量测试误差，并将误差经过最大最小标准化的逆变化，定义为：

$$RMSE = \mathrm{Nom}^{-1}\left(\frac{1}{\mathcal{V}}\sum_{i=1}^{\mathcal{V}}\sqrt{\frac{\left\|\widehat{\mathbf{M}}(i) - \mathbf{M}'(i)\right\|^2}{|R|}}\right), \quad \#(38)$$

其中，$\mathcal{V}$是验证集中的样本个数，Nom^{-1}表示最大最小标准化函数的反函数．

5.3 实验结果

实验 1．收敛性分析.

为了评价 D-SOAC 方法中 ST-ResNet 的收敛性情况，我们将 ST-ResNet 训练过程中的初始学习率分别设置为 0.0025、0.005、0.0065 和 0.007，所得到的训练误差和测试误差的收敛情况分别如图 5 和图 6 所示．

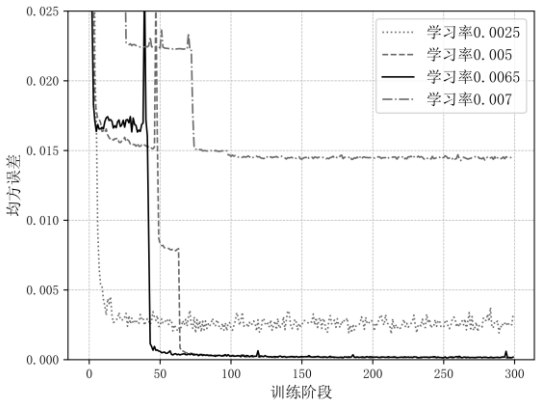

图 5 不同学习率时 ST-ResNet 的训练误差收敛情况

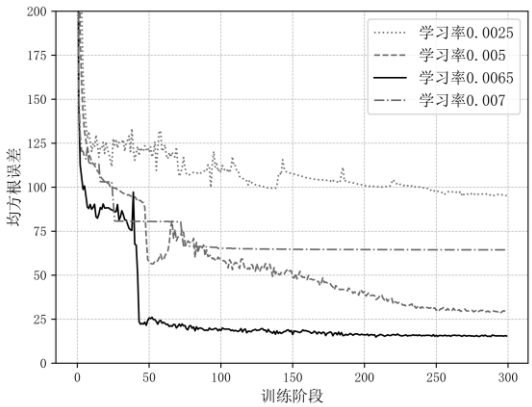

图 6 不同学习率时 ST-ResNet 的测试误差收敛情况

可以看到，当学习率为 0.007 时，由于学习率较高，ST-ResNet 不能很好的达到损失函数的极小值点，训练误差最终收敛到了 0.015 左右．相应的，在测试集上形成的测试误差也较高，均方根误差最终收敛于 70 左右．而当学习率设置较小，为 0.0025 时，虽然在训练集上损失函数收敛，但是陷入了局部的最优点处，最后的训练误差虽然优于学习率为 0.007 时，但是劣于学习率为 0.005 和 0.0065 的情况．此外，当学习率为 0.0025 时，还导致了过拟合问题的产生，虽然训练误差较低，但是测试误差却在所有学习率情况中表现最差，测试误差最终在 100 左右．当学习率设置为 0.005 或 0.0065 时，ST-ResNet 的训练表现出色，最后训练误差均收敛到了一个较小值．但是，由于 0.005 的学习率小于 0.0065，因此训练速度相较学习率为 0.0065 时更为缓慢，并且在测试集上出现了过拟合的情况，最终

测试集上的均方根误差较学习率为 0.0065 时高了 10 左右．而学习率为 0.0065 时表现最佳，在测试集和训练集上均取得了较为良好的表现．最终的测试误差收敛到了 15 左右．

和 ST-ResNet 相同，D-SOAC 方法中 A3C 网络的学习率也是影响收敛一个重要的超参数．分别设置 A3C 网络的学习率为 0.05、0.01 和 0.005，对 D-SOAC 进行训练，结果如图 7 所示．由于 D-SOAC 方法中，所有边缘服务器异步并行运行，不同区域中边缘服务器行动者-评论家网络训练速度不同，导致整个车联网边缘计算环境中所获得的移动平均奖励一开始会有一个较大的变化过程，然后开始逐步稳定并上升，最后收敛到最大的奖励处．由图可见，三种学习率情况下，D-SOAC 方法均收敛，证明了利用 D-SOAC 方法进行车联网边缘计算服务卸载的可行性．当学习率为 0.05 时，由于搜索范围较广，最终方法没有收敛至全局最优；而当学习率为 0.005 时，由于学习率较低，因此移动平均奖励收敛较慢．当学习率为 0.01 时，D-SOAC 获得了较高的累计奖励，并且收敛速度较快，达到了平衡．

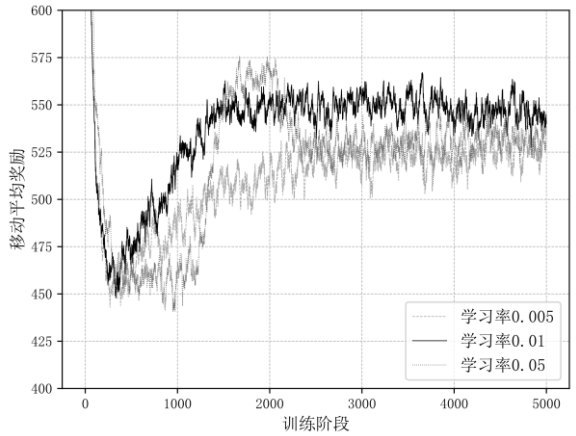

图 7 不同学习率时 D-SOAC 所获得的移动平均奖励

实验 2．超参数对服务需求预测和服务卸载性能的影响实验．

D-SOAC 方法的一大特点便是采用了 ST-ResNet 将未来的用户服务需求量作为系统状态输入 A3C，进行服务卸载的决策．此外，带折扣累计奖励中的折扣率γ影响着智能体对长期奖励或短期奖励的倾向程度．因此，预测步数κ和折扣率γ是 D-SOAC 方法中的两个重要的超参数．除此以外，ST-ResNet 中短时需求、周期需求和长期需求的时间长度设置不同会导致服务需求的预测结果产生较大差异．在本节中，我们尝试对各超参数的设置进行调整，并对其对服务需求预测和服务卸载性能的影响进行讨论．

我们首先尝试对短时需求长度l_c、周期需求长度l_p和长期需求长度l_t进行调整，测试各超参数设置下服务需求预测的准确度表现．首先固定l_p和l_t分别为 3 和 4，调整l_c的值分别为 1、2、4、6 和 8，实验结果如图 8 所示．由图可见，当短时需求长度为 6 和 8 时，服务需求预测的误差值较低，而当短时需求长度为 2 或 4 时，误差较高．该实验结果表明，某段时间用户的服务需求量和 12～16 分钟前用户服务的需求量关系密切．而和前 4～8 分钟前用户的服务需求量关系不大．

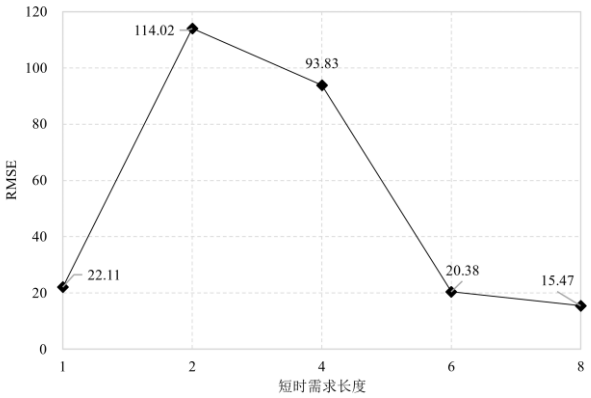

图 8 短期需求长度不同时服务需求预测误差

接着，固定l_c和l_t分别为 8 和 4，调整l_p的值为 1、2、3、4 或 5，讨论周期需求长度对服务需求预测性能的影响．实验结果如图 9 所示．实验结果表明，1 天到 3 天之前的用户历史服务需求和当前的服务需求有极大关联，而更久远之前的用户服务需求和当前用户的服务需求关系不大，甚至会导致服务需求预测准确度的降低．例如，周期需求长度为 4 时的预测误差是长度为 1 时预测误差的 6 倍多．当周期需求长度为 1 或 3 时，服务需求预测的准确度较高，分别取得了 15.14 和 15.47 的均方根误差．

然后，固定l_c和l_p分别为 8 和 3，设置l_t的值从 1 到 5 依次递增，考察长期需求长度对服务需求量预测准确度的影响情况．实验结果如图 10 所示．结果显示，当长期需求长度设置为 1 时，未来用户服务需求预测的误差较大，达到了 439.33，这说明较少数量的长期历史用户服务需求对未来用户服务需求的预测没有帮助，甚至会导致预测准确度下降．而将更多的长期历史用户服务需求考虑在内，则能达到较高的预测准确度．特别地，当长期需求长度设置为 4 时，达到了最低的均方根误差

（15.47），相较于长期需求长度为 1 时的误差减少了 28 倍.

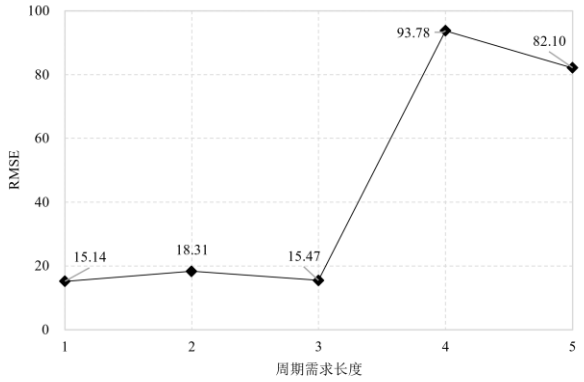

图 9 周期需求长度不同时服务需求预测误差

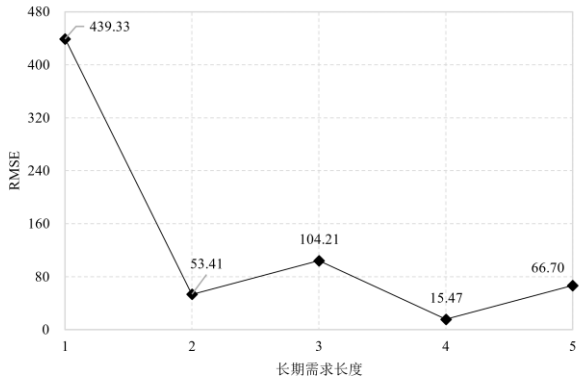

图 10 长期需求长度不同时服务需求预测误差

为了证明将未来用户服务需求量作为系统状态一部分的合理性，我们固定 A3C 的学习率为 0.01，采用学习率为 0.0065 时训练完成的 ST-ResNet 模型，分别对不预测（预测步数为 0）、预测未来一步和预测未来两步用户服务需求量的 D-SOAC 方法所获得的累计奖励进行对比，结果如图 11 所示. 可以见到，不论是否进行用户服务需求量预测，D-SOAC 均收敛. 但是不进行预测时，D-SOAC 收敛到一个较小的累计奖励上. 当预测步数为 1 时，D-SOAC 获得了最高的累计奖励，为 550 左右，说明未来用户服务需求量的预测对于服务卸载决策有着很大的作用. 而当预测步数为 2 时，系统所获累计奖励与预测步数为 1 时相差不大. 说明未来两步的用户服务需求量对于智能体决策有一定的作用. 但是预测步数为 2 时的移动平均奖励收敛速度较预测步数为 0 和预测步数为 1 时都更慢. 综上所述，预测步数设置为 1 可以对车联网用户服务卸载起到较好的指导作用.

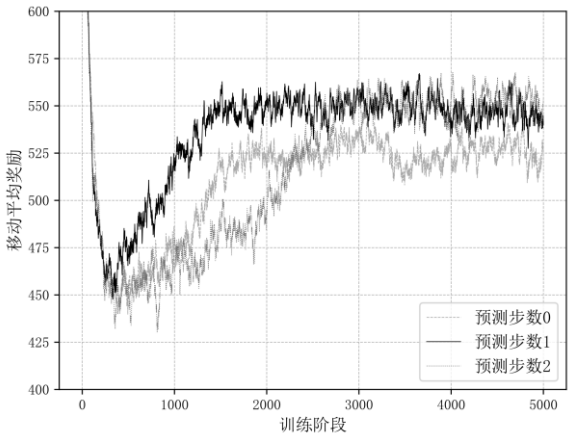

图 11 不同预测步数时 D-SOAC 所获得的移动平均奖励

最后，考察折扣率γ对 D-SOAC 方法的性能影响. 设置折扣率γ取值为$[0.9, 0.95, 0.99]$，固定 D-SOAC 学习率为 0.01，实验结果如图 12 所示. 由图可见，当折扣率取值为 0.9 和 0.99 时，D-SOAC 所得到的移动平均奖励相差无几，最后收敛到 460 左右；而当折扣率为 0.95 时，D-SOAC 取得了较好的性能表现，最后移动平均奖励能够收敛到 550 左右. 总的来说，折扣率为 0.95 时所获的奖励相比于折扣率为 0.9 或 0.99 时获得的移动平均奖励提高了 20%左右. 由此实验结果可以推断出，仅考虑短期的奖励，即折扣率较小（为 0.9）时，并不能取得全局的最优解，无法缩减全局的时延；而当折扣率设置过大，例如折扣率为 0.99 时，智能体会将长期之前时间步所获得的奖励也考虑在服务卸载决策过程中，也并不能达到较好的缩小全局时延的效果.

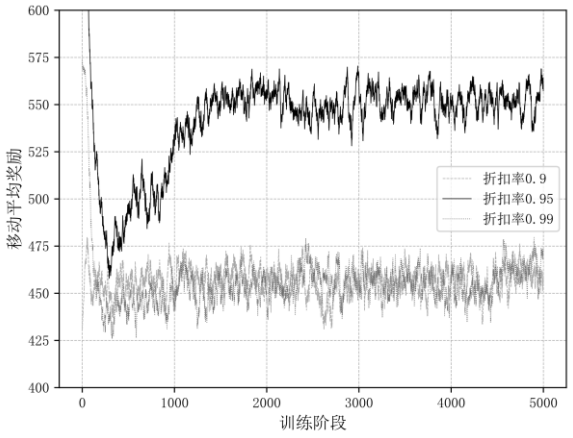

图 12 不同折扣率时 D-SOAC 所获得的移动平均奖励

实验 3. 与其他服务需求预测方法的对比实验.

本实验中，我们选择以下四种不同的服务需求

预测方法，与 ST-ResNet 方法进行服务需求量预测准确度的对比实验，以证明 ST-ResNet 在用户服务需求预测任务上的能力和先进性.

（1）HA（HistoryAverage），历史平均．将历史一段时间内某一区域的平均用户服务需求量作为未来该区域中的服务需求量.

（2）LSTM（Long-Short Term Memory），长短期记忆网络．作为递归神经网络的一种，LSTM 具有强大的序列数据处理能力，其已在自然语言处理等领域取得了广泛的应用.

（3）Bi-LSTM（BidirectionalLSTM），双向长短期记忆网络．由前向 LSTM 和后向 LSTM 共同组成，解决了 LSTM 中只能单方向处理序列数据的问题.

（4）CNN（ConvolutionalNeuralNetwork），卷积神经网络．利用卷积运算来提取数据之间的空间关系．在图像处理、模式识别等领域，CNN 已展现出强大的性能.

本实验设置 ST-ResNet 的学习率为 0.0065，短时需求长度、周期需求长度和长期需求长度分别为 8、3 和 4．实验结果如表 3 所示.

表 3 不同服务需求预测方法的训练误差和测试误差

	训练误差（MSE）	测试误差（RMSE）
HA	N.A.	84.09
LSTM	1.953×10^{-3}	23.79
Bi-LSTM	1.760×10^{-3}	23.22
CNN	1.617×10^{-3}	22.57
ST-ResNet	$\mathbf{0.188 \times 10^{-3}}$	**15.47**

从实验结果可以看出，不管在训练误差还是在测试误差方面，ST-ResNet 的表现均优于其他四种对比方法．例如，ST-ResNet 的训练误差相比于 LSTM 方法降低了 10 倍以上.即使和在对比算法中表现最好的 CNN 方法相比，ST-ResNet 的训练误差水平也仅为 CNN 的 11.6%左右．对于测试误差，ST-ResNet 的测试误差分别仅为 HA、LSTM、Bi-LSTM 和 CNN 方法的 18.4%、65.0%、66.6%和 68.5%，取得了较好的表现．该实验结果证明，ST-ResNet 由于同时考虑了用户服务需求量这一时空数据的时间相关性和空间相关性，相比于传统的仅能单一地提取时间特征（例如 HA、LSTM 和 Bi-LSTM）或者空间特征（例如 CNN）的服务需求预测方法来说，有着更高的预测准确性.

实验 4．与其他车联网服务卸载方法的对比实验.

本实验中，我们选择以下五种车联网边缘计算环境下服务卸载的方法，与 D-SOAC 方法进行对比：

（1）LO（Local Offloading），全本地卸载．车联网用户的服务需求全部由本地的车载应用执行.

（2）RO（Random Offloading），随机卸载．当车联网用户产生服务请求时，边缘服务器随机给用户分配子信道和计算资源进行相应的服务卸载操作.

（3）ECO（Edge Computing Enabled Computation Offloading Method with NSGA-II），使用进化算法 NSGA-II 寻找车联网服务卸载问题的最优解，从而进行服务卸载决策的服务卸载方法[42]．由于 ECO 在每次服务卸载决策时，需要通过迭代的方式找到最优解，因此 ECO 算法的迭代时间即为所有用户的等待时间.

（4）BDR（Big Data Deep Reinforcement Learning Approach）[43]．该方法和 D-SOAC 方法一样，也采用深度强化学习进行服务卸载决策．不同的是，BDR 方法中采用深度 Q 学习（Deep Q-Learning）作为深度强化学习的方法.

（5）Optimal，最优算法．由于服务卸载问题是一个 NP-Hard 问题，求解该问题的时间复杂度是关于用户服务需求数M和时间段数量$\mathcal{T}$的指数函数，在可接受时间内无法完成该任务．因此，我们首先对服务卸载问题进行简化，不考虑 5G 车联网边缘计算环境中可用计算资源和通信资源的动态变化，认为每个时间段内边缘服务器提供固定数量的计算资源数和子信道个数．然后，使用动态规划（DynamicProgramming，DP）技术，求得该简化的服务卸载问题的最优解.

在实验 4 中，我们从 436 个道路段中随机抽取 50 个道路段，计算 1 小时 20 分钟内经过这 50 个道路段的所有用户产生的平均用户服务时延．此外，通过改变系统中子信道和计算资源的个数，比较 D-SOAC 和对比算法在不同车联网边缘计算环境中的表现．其中，D-SOAC 方法使用学习率为 0.0065 时训练得到的 ST-ResNet 进行用户服务需求量的一步预测，使用学习率为 0.01 的 A3C 网络进行用户服务卸载决策.

当边缘服务器的计算能力较弱（可分配的计算资源数为 5）时，不同服务卸载方法在不同网络环境下的表现如图 13 所示，各算法求解所用的时间在表 4 中列出．实验结果表明，D-SOAC 方法仅花费了最优算法 7.1%～26.7%的执行时间，便达到了 62.5%～65.4%的最优解近似率．此外，D-SOAC 在

各种网络环境下的服务卸载表现均优于除最优算法以外的其他方法．当系统中子信道个数分别为20、30 或者 40 时，D-SOAC 的用户平均服务时延分别为 2.541s、2.547s 和 2.542s，相较于表现最差的 ECO 算法，D-SOAC 方法的性能提升了 5.12%～5.70%不等，将服务时延降低了 0.15s 左右．同时，D-SOAC 求解的时间仅为 ECO 算法的约 1/20．而和表现最好的 BDR 和 RO 算法比较，D-SOAC 也降低了 0.4%～2.5%的用户平均服务时延，并且求解时间仅为它们的 59.2%～67.3%．该实验结果证明 D-SOAC 方法在边缘服务器计算能力较弱时，为用户提供低时延服务的能力．

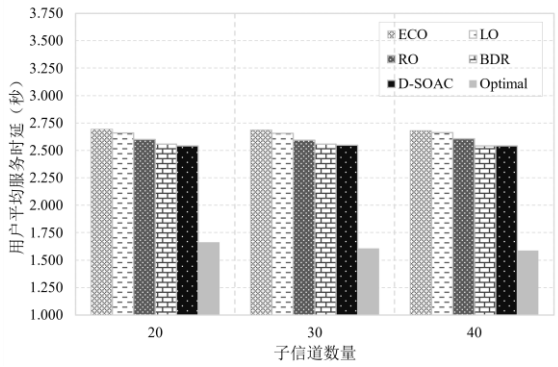

图 13 计算资源数为 5 时用户的平均服务时延

表 4 计算资源数为 5 时不同服务卸载方法求解的时间（单位：秒）

子信道数量	20	30	40
BDR	116.02	122.21	129.66
RO	121.15	116.27	113.95
LO	190.78	191.25	191.72
ECO	1485.88	1494.48	1494.31
Optimal	288.65	620.20	1081.99
D-SOAC	**77.17**	**77.24**	**76.72**

除此以外，我们随机抽取了环境中的一个道路段，以之为例对道路段内使用不同的服务卸载方法所产生的用户服务总时延和服务需求量之间的变化关系进行了实验，实验结果如图 14 至图 16 所示．由图可见，所有的方法在用户服务需求量增加时，用户服务的总时延也会相应增加．并且用户的总服务时延相比服务需求量的增加有一定的延迟，特别是当服务数量激增后，总服务时延还会在较高的水平保持一段时间．这是符合实际的，因为用户服务需求量越大，那么车联网就要消耗更多的计算资源和带宽资源，以完成车联网用户产生的各种服务，导致用户服务时延的提升．同时，车联网用户服务需求的增加导致了在一个时间段内这些用户的服务需求不能全部完成，因此占用了下一个时间段的计算资源，使边缘服务器可用资源动态变化，进而导致下一时间段的服务时延相应升高．实验结果表明，当边缘服务器配备计算资源数为 5 时，我们所提出的 D-SAOC 方法在车联网用户产生不同数量的服务需求时，为用户提供服务的总时延较其他方法均有降低．除此以外，在用户服务数量激增时，例如在第 13 个时间段，在配备了不同子信道个数的 5G 车联网边缘计算环境中，我们所提出的 D-SOAC 方法相较于其他的对比方法来说，降低了 2s～4s 不等的用户总服务时延．由此可见，D-SOAC 方法在边缘服务器配备计算资源数为 5 时，有降低用户的服务总时延的激增程度的能力．

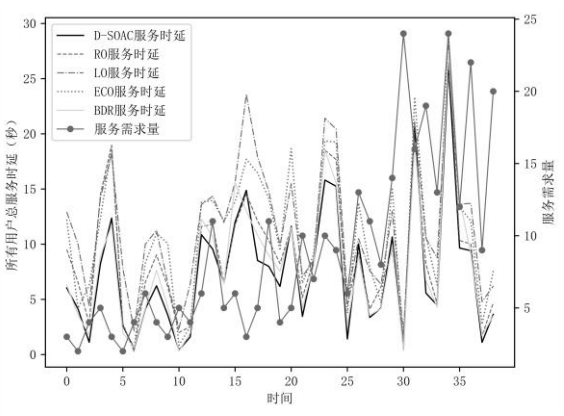

图 14 计算资源为 5、子信道数为 20 时用户服务总时延和服务需求量的变化关系

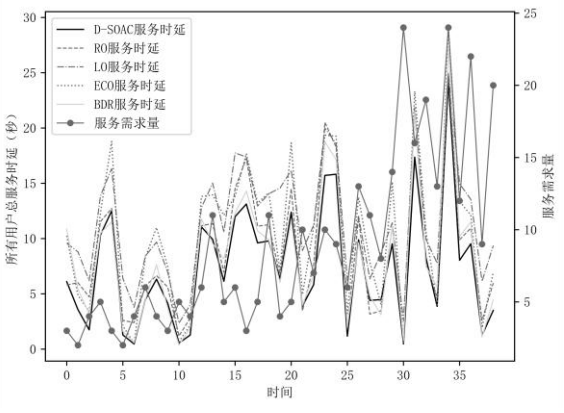

图 15 计算资源为 5、子信道数为 30 时用户服务总时延和服务需求量的变化关系

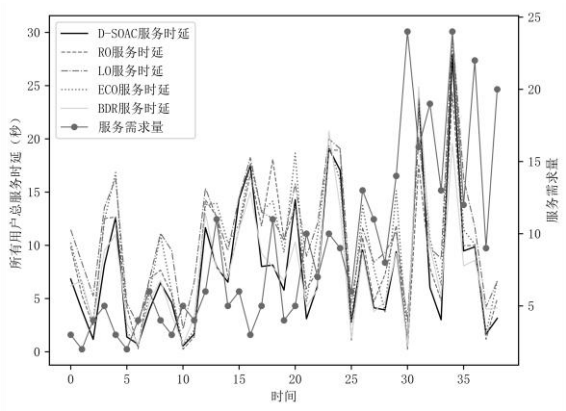

图 16 计算资源为 5、子信道数为 40 时用户服务总时延和服务需求量的变化关系

当系统中每个道路段所配备的边缘服务器计算资源数为 10 时，用户平均服务时延和卸载策略求解时间的实验结果分别如图 17 和表 5 所示．从中可见，D-SOAC 方法相较于最优算法的近似比在 46.9%～55.9%左右，但求解时间仅为最优算法的 2.6%～8.9%．相比 ECO 和 LO 方法，D-SOAC 方法均较大幅度地减小了用户的平均服务时延．例如，当子信道个数为 20 时，使用 ECO 和 LO 作为服务卸载方法所产生的用户平均服务时延分别为 2.676s 和 2.673s．而使用 D-SOAC 方法，用户平均服务时延仅为 2.455s，平均为每个用户降低了超过 0.2s 的服务时延．和 RO 方法和 BDR 方法相比，使用 D-SOAC 方法产生的用户平均服务时延下降约有 0.4%～2.2%左右的降低，表现相近．但是，考虑到在车联网环境中 1 个小时 20 分钟内，用户就产生了超过 1.2 万个用户服务需求，因此考虑用户服务时延总量的情况下，仍然有较客观的降低．除此以外，D-SOAC 方法求出服务卸载策略所花费的时间也是最少的，相较于 RO 和 BDR 方法，D-SOAC 方法将服务卸载卸载方法的求解时间降低了近 40s．

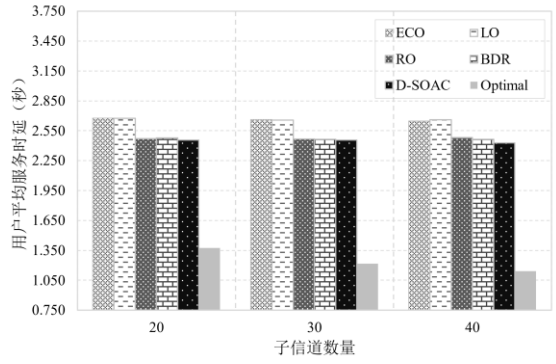

图 17 计算资源数为 10 时用户的平均服务时延

表 5 计算资源数为 10 时不同服务卸载方法求解的时间（单位：秒）

子信道数量	20	30	40
BDR	124.35	105.75	122.08
RO	127.99	128.73	125.27
LO	191.11	191.11	190.87
ECO	1490.35	1483.50	1468.31
Optimal	907.36	1912.98	3325.47
D-SOAC	**81.10**	**80.13**	**87.06**

当 5G 车联网边缘计算环境中边缘服务器配备计算资源数为 10 时，不同的服务卸载方法产生的用户服务总时延和服务需求量的变化关系如图 18 至图 20 所示．从中可见，我们所提出的 D-SOAC 方法在边缘服务器配备不同数量的子信道资源时，均能降低用户服务总时延的激增程度．例如，当边缘服务器配备计算资源数为 10，子信道数为 30 时，在第 3 个时间段用户服务需求量激增时，采用 D-SOAC 方法时用户总服务时延的增加程度相较于其他四种对比方法程度是最小的．

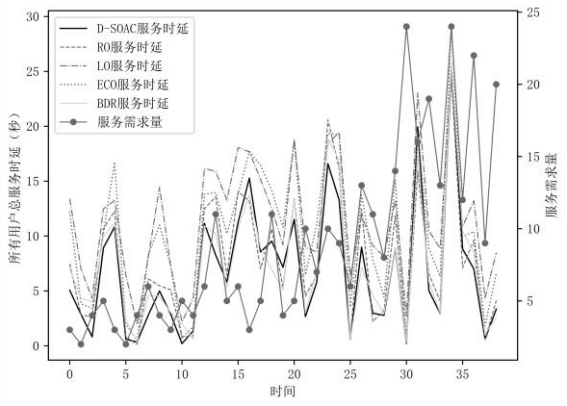

图 18 计算资源为 10、子信道数为 20 时用户服务总时延和服务需求量的变化关系

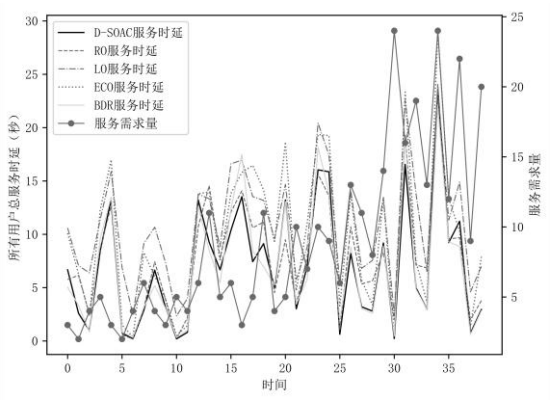

图 19 计算资源为 10、子信道数为 30 时用户服务总时延和服务需求量的变化关系

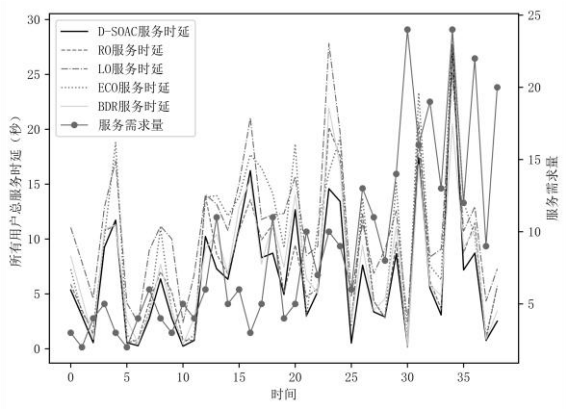

图 20 计算资源为 10、子信道数为 40 时用户服务总时延和服务需求量的变化关系

当边缘服务器中的计算资源较为充足（计算资源数为 15）时，用户能把更多的服务需求卸载到边缘服务器上来完成，因此能够获得更低的用户服务时延．实验结果也支持该结论，如图 21 和表 6 所示．和最优算法相比，D-SOAC 使用了最优算法 1.4%～5.8%的求解时间，便达到了 50%左右的最优解近似率．除此以外，当计算资源数为 15 时，不论子信道个数多少，D-SOAC 方法表现均优于对比算法．例如当子信道个数为 20 时，D-SOAC 方法相比于 ECO、LO、RO 和 BDR 算法，用户平均服务时延降低了 20.4%、20.1%、20.0%和 11.5%．当子信道个数为 30 和 40 时，采用 D-SOAC 方法能降低 5.0%～15.0%和 4.3%～14.6%的用户平均服务时延．同时，D-SOAC 方法在求解服务卸载策略的时间方面，相较于其他四种对比算法表现也是最优的．例如当边缘服务器配备计算资源数为 15，子信道数量为 40 时，D-SOAC 方法仅使用了 BDR 方法 67.8%的时间，即达到了相比于 BDR 方法更优的性能表现．总的来说，当 5G 车联网边缘计算系统配备较为充足的资源时，D-SOAC 方法的表现会更加出色．

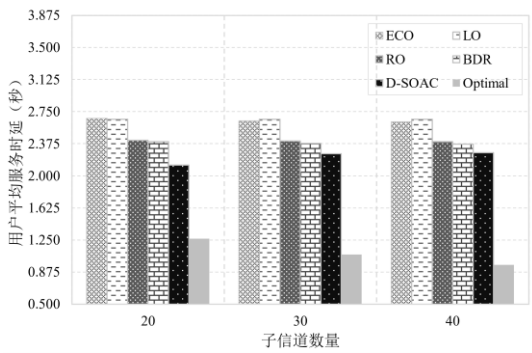

图 21 计算资源数为 15 时用户的平均服务时延

表 6 计算资源数为 15 时不同服务卸载方法求解的时间（单位：秒）

	20	30	40
BDR	141.03	130.34	136.91
RO	139.02	134.13	130.59
LO	190.26	189.85	190.39
ECO	1521.31	1524.71	1528.89
Optimal	1798.18	3848.36	6657.72
D-SOAC	**103.95**	**104.10**	**92.88**

当边缘服务器配备计算资源数为 15，且配备子信道数量为 20、30 或 40 时，不同的车联网服务卸载方法产生的用户服务总时延和服务需求量的变化关系如图 22 至图 24 所示. 由实验结果可以看出，当边缘服务器计算资源数较为充沛的时候，D-SOAC 方法相比于其他的服务卸载方法，能够大幅降低用户服务需求激增时的用户服务时延，而不论系统中配备的子信道数量多少．例如，当子信道数为 20 时，使用 D-SOAC 方法在任一时间段内产生的用户服务总时延和其他方法相比都是最小的．除此以外，当服务需求量在第 30 个时间段由 14 左右激增到 25 时，采用 D-SOAC 方法产生的用户服务总时延仅增加了 10s 左右，而其他四种方法用户总服务时延的激增均超过 20s，甚至达到 25s．当子信道数量为 30 或 40 时，也有类似的结论．比如，当边缘服务器配备子信道个数为 40 时，在第 22 至第 23 个时间段用户需求数量增加时，使用 D-SOAC 方法产生的用户服务总时延约为 13s，而使用 BDR 算法则会产生 15s 左右的用户服务总时延，使用 RO 或者 LO 算法所产生的用户服务总时延甚至会达到 20s.

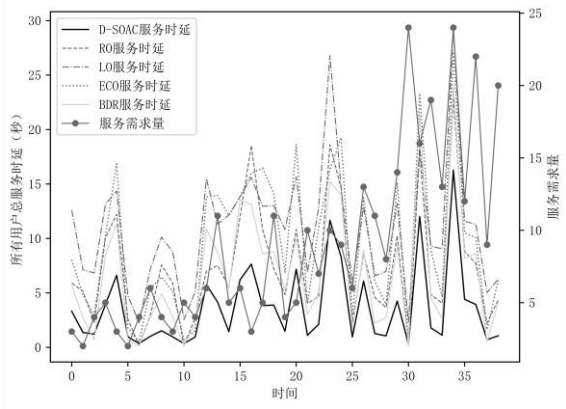

图 22 计算资源为 15、子信道数为 20 时用户服务总时延和服务需求量的变化关系

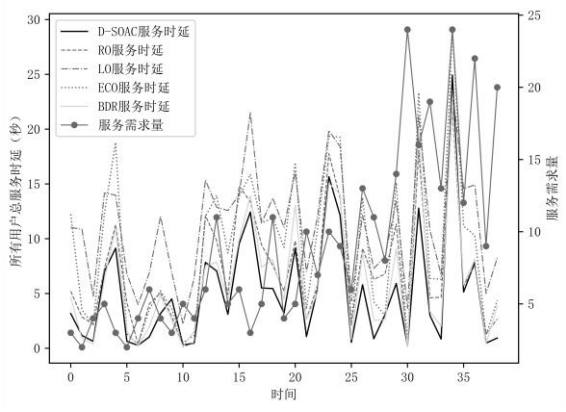

图 23 计算资源为 15、子信道数为 30 时用户服务总时延和服务需求量的变化关系

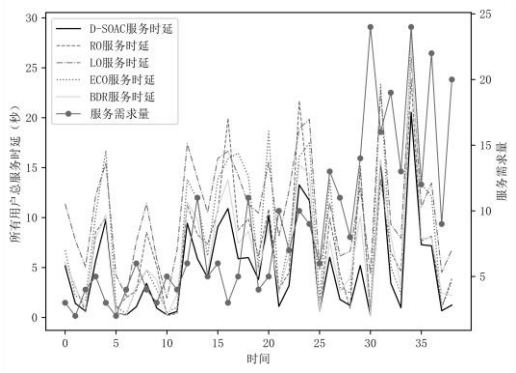

图 24 计算资源为 15、子信道数为 40 时用户服务总时延和服务需求量的变化关系

综上所述，本文提出的 D-SOAC 方法在配备不同计算资源和带宽资源的 5G 车联网边缘计算环境中，均能在较短的执行时间内，降低用户平均服务时延，相比于其他四种对比算法，有着 0.4%~20.4%不等的性能提升．由此可以证明，D-SOAC 方法能够在不同 5G 车联网边缘计算环境下降低用户服务时延，提高车联网用户的服务体验．

6 结论与展望

为满足车联网用户对低时延服务的需求，本文提出了一种分布式的基于深度强化学习的 5G 车联网边缘计算服务卸载方法 D-SOAC，以降低车联网用户长期的平均服务时延，从而为用户提供高质量的服务体验．特别地，我们利用 ST-ResNet 预测未来车联网中用户的服务需求量，将其作为系统状态的一部分输入到 A3C 中，辅助智能体进行卸载决策．同时，为了解决传统深度强化学习中，当动作包含多个子动作时，动作空间复杂度过高的问题，本文引入了一种多动作输出的行动者网络，并推导了该网络的梯度．最后，本文基于现实世界的车联网用户服务需求数据集，将 D-SOAC 与其他四种现有方法进行了对比实验．实验表明，D-SOAC 方法能够有效降低用户平均服务时延，验证了本文方法的有效性．下一步，我们将在考虑用户的服务时延需求的同时，研究如何进一步降低服务的能耗，并提高边缘服务器的资源利用率和负载均衡性．

参考文献

[1] Sharma S, Kaushik B. A survey on internet of vehicles: applications, security issues & solutions. Vehicular Communications, 2019, 20: 100182

[2] Vegni A M, Loscri V. A survey on vehicular social networks. IEEE Communications Surveys & Tutorials, 2015, 17(4): 2397-2419

[3] Boban M, Kousaridas A, Manolakis K, et al. Connected roads of the future: use cases, requirements, and design considerations for vehicle-to-everything communications. IEEE Vehicular Technology Magazine, 2018, 13(3): 110-123

[4] Ning Z, Xia F, Ullah N, et al. Vehicular social networks: enabling smart mobility. IEEE Communications Magazine, 2017, 55(5): 16-55

[5] Song H M, Kim H R, Kim H K. Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network//Proceedings of the International Conference on Information Networking. Kota Kinabalu, Malaysia, 2016: 63-68

[6] Zhang W, Zhang Z, Chao H-C. Cooperative fog computing for dealing with big data in the internet of vehicles: architecture and hierarchical resource management. IEEE Communications Magazine, 2017, 55(12): 60-67

[7] He X, Ren Z, Shi C, Fang J. A novel load balancing strategy of software-defined cloud/fog networking in the internet of vehicles. China Communications, 2016, 13(Supplement2): 140-149

[8] Shi Wei-Song, Sun Hui, Cao Jie, Zhang Quan, Liu Wei. Edge computing-an emerging computing model for the internet of everything era. Journal of Computer Research and Development, 2017, 54(5): 907-924 (in Chinese)
(施巍松, 孙辉, 曹杰, 张权, 刘伟. 边缘计算: 万物互联时代新型计算模型. 计算机研究与发展, 2017, 54(5): 907-924)

[9] Feng J, Liu Z, Wu C, Ji Y. Mobile edge computing for the internet of vehicles: offloading framework and job scheduling. IEEE Vehicular Technology Magazine, 2018, 14(1): 28-36

[10] Cui L, Chen Z, Yang S, et al. A blockchain-based containerized edge computing platform for the internet of vehicles. IEEE Internet of Things Journal, doi: 10.1109/JIOT.2020.3027700

[11] Wan S, Li X, Xue Y, et al. Efficient computation offloading for internet of vehicles in edge computing-assisted 5G networks. The Journal of Supercomputing, 2019, 76(4): 2518-2547

[12] Wang C, Haider F, Gao X, et al.Cellular architecture and key technologies for 5G wireless communication networks. IEEE Communications Magazine, 2014, 52(2): 122-130

[13] Alasmari K R, Green R C, Alam M. Mobile edge offloading using Markov decision process//Proceedings of the International Conference on Edge Computing. Seattle, USA, 2018: 80-90

[14] Kiumarsi B, Vamvoudakis K G, Modares H, Lewis F L. Optimal and autonomous control using reinforcement learning: a survey. IEEE Transactions on Neural Networks and Learning Systems, 2018, 29(6): 2042-2062

[15] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553): 436-444

[16] Arulkumaran K, Deisenroth M P, Brundage M, Bharath A A. Deep reinforcement learning: a brief survey. IEEE Signal Processing Magazine, 2017, 34(6): 26-38

[17] Dinh T Q, La Q D, Quek T Q S, Shin H. Learning for computation offloading in mobile edge computing. IEEE Transactions on Communications, 2018, 66(12): 6353-6367

[18] Dai Y, Xu D, Maharjan S, et al. Artificial intelligence empowered edge computing and caching for internet of vehicles. IEEE Wireless Communications, 2019, 26(3): 12-18

[19] Qi Q, Wang J, Ma Z, et al. Knowledge-driven service offloading decision for vehicular edge computing: a deep reinforcement learning approach. IEEE Transactions on Vehicular Technology, 2019, 68(5): 4192-4203

[20] Zhang J, Zheng Y, Qi D. Deep spatio-temporal residual networks for citywide crowd flows prediction//Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. San Francisco, USA, 2017: 1655-1661

[21] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning//Proceedings of the International Conference on Machine Learning. New York, USA, 2016: 1928-1937

[22] Ning Z, Huang J, Wang X, et al. Mobile edge computing-enabled internet of vehicles: toward energy-efficient scheduling. IEEE Network, 2019, 33(5): 198-205

[23] Zhang Hai-Bo, Cheng Yan, Liu Kai-Jian, He Xiao-Fan. The mobility management strategies by integrating mobile edge computing and cdn in vehicular networks. Journal of Electronics & Information Technology, 2020, 42(6): 1444-1451 (in Chinese)
(张海波，程妍，刘开健，贺晓帆．车联网中整合移动边缘计算与内容分发网络的移动性管理策略．电子与信息学报，2020，42(6): 1444-1451)

[24] Wang K, Yin H, Quan W, et al. Enabling collaborative edge computing for software defined vehicular networks. IEEE Network, 2018, 32(4): 112-117

[25] LiuM, Liu Y. Price-based distributed offloading for mobile-edge computing with computation capacity constraints. IEEE Wireless Communications Letters, 2018, 7(3):420-423.

[26] Chen W, Zhang Z, Hong Z, et al. Cooperative and distributed computation offloading for blockchain-empowered industrial internet of things. IEEE Internet of Things Journal, 2019, 6(5): 8433-8446.

[27] Chen X, Jiao L, Li W, Fu X. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Transactions on Networking, 2015, 24(5): 2795-2808.

[28] Liu Quan, Zhai Jian-Wei, Zhang Zong-Zhang, et al. A survey on deep reinforcement learning. Chinese Journal of Computers, 2018, 41(1): 1-27 (in Chinese)
(刘全，翟建伟，章宗长等．深度强化学习综述．计算机学报，2018，41(1): 1-27)

[29] Salahuddin M A, Al-Fuqaha A, Guizani M. Software-defined networking for rsu clouds in support of the internet of vehicles. IEEE Internet of Things Journal, 2014, 2(2): 133-144

[30] Ning Z, Dong P, Wang X, Rodrigues J, Xia F. Deep reinforcement learning for vehicular edge computing: an intelligent offloading system. ACM Transactions on Intelligent Systems and Technology, 2019, 10(6): 1-24

[31] Liu Y, Yu H, Xie S, Zhang Y. Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks. IEEE Transactions on Vehicular Technology, 2019, 68(11): 11158-11168

[32] Rangan S, Rappaport TS, Erkip E. Millimeter-wave cellular wireless networks: potentials and challenges. Proceedings of the IEEE, 2014, 102(3): 366-385

[33] Morelli M, Kuo C-C J, Pun M-O. Synchronization techniques for orthogonal frequency division multiple access (ofdma): atutorial review. Proceedings of the IEEE, 2007, 95(7): 1394–1427

[34] Pan Z, Liang Y, Wang W, Yu Y, Zheng Y, Zhang J.Urban traffic prediction from spatio-temporal data using deep meta learning//Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.Anchorage, USA, 2019: 1720-1730.

[35] Zhang J, Zheng Y, Qi D, Li R, Yi X. DNN-based prediction model for spatio-temporal data//Proceedings of the 24th ACM International Symposium on Advances in Geographic InformationSystems. New York, USA,2016: 1-4.

[36] Zhai D, Liu A, Chen S, et al. SeqST-ResNet: asequential spatial temporal resnet for task prediction in spatial crowdsourcing// Proceedings of the International Conference on Database Systems for Advanced Applications. Jeju, Korea, 2019: 260-275

[37] Yao H, Wu F, Ke J, et al. Deep multi-view spatial-temporal network for taxi demand prediction//Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence. New Orleans, USA, 2018: 2588-2595

[38] Zhang N,ChengNan, Gamage A, et al. Cloud assisted hetnets toward 5G wireless networks. IEEE Communications Magazine, 2015: 53(6): 59-65

[39] Dai Y, Xu D, Maharjan S, Qiao G, Zhang Y. Mobile edge intelligence and computing for the internet of vehicles. Proceedings of the IEEE,

2020, 108(2): 246-261

[40] Chen X, Zhang H, Wu C, et al. Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning. IEEE Internet of Things Journal, 2019, 6(3): 4005-4018

[41] Sun Y, Zhou S, Xu J. EMM: energy-aware mobility management for mobile edge computing in ultra dense networks. IEEE Journal on Selected Areas in Communications, 2017, 35(11): 2637-2646

[42] Xu X, Xue Y, Qi L, et al. An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles. Future Generation Computer System, 2019, 96: 89-100

[43] He Y, Yu F R, Zhao N, Leung V C M, Yin H. Software-defined networks with mobile edge computing and caching for smart cities: abig data deep reinforcement learning approach. IEEE Communications Magazine, 2017, 55(12): 31-37

XU Xiao-Long, Ph.D., professer, master supervisor. His research interests include edge computing, cloud computing, and service computing.

FANG Zi-Jie, B.S. His research interests include edge computing and deep learning.

QI Lian-Yong, Ph.D., professor, Ph.D. supervisor. His research interests include service computing,recommendation system, and privacy protection.

DOU Wan-Chun, Ph.D., professor, Ph.D. supervisor. His research interests include big data,cloud computing, and edge computing.

HE Qiang, Ph.D., senior lecturer, Ph.D. supervisor. His research interests include edge computing, software engineering, and cloud computing.

DUAN Yu-Cong, Ph.D., professor, Ph.D. supervisor. His research interests include information security, artificial intelligence, and big data.

Background

Due to the development of the Internet of Vehicles (IoV) and the modern wireless communication technology such as the fifth-generation network, drivers and passengers can be served with numerous services by the service providers while traveling on vehicles, thereby enhancing users' riding safety and travel experience. However, since most vehicles are not equipped with sufficient computation resources, these services are traditionally offloaded to the cloud center for execution, leading to high service latency because of the long distance between the users and the cloud center. Edge computing solves this problem by distributing edge servers equipped with resources, including computing power, storage space, and bandwidth, near to the users. However, the resources deployed on the edge servers are often limited, which makes it impossible for the edge servers to support all the latency-sensitive services requested by the users at the same time. Therefore, how to build a service offloading method that selects the appropriate offloading destination for services and minimizes the service latency remains a challenge. Recently, there have been some service offloading methods for 5G-enabled IoV with edge computing based on game theory, convex optimization, and evolutionary algorithm. However, these studies mainly had two significant drawbacks. Firstly, some of the studies did not consider the dynamic changing of the IoV environment, including but not limited to the users' service requirements, the network condition, and the amount of available resource. Secondly, most of the existing service offloading methods were centralized algorithms, neglecting the distributed feature of edge computing. To handle these problems, we propose a distributed service offloading method with asynchronous advantage actor-critic (A3C). By utilizing a big real-world dataset from Nanjing, China, the experimental results show the effectiveness of the proposed method. Our method reduces the average service latency by 0.4% ~ 20.4% compared with four existing service offloading methods in different 5G-enabled IoV environments, where edge servers are deployed with different computing and network capabilities.

This research is supported by the Key Research and Development Program of Jiangsu Province under grant no. BE2019104, the National Natural Science Foundation of China under grant no.61872219, the National Key Research and Development Program of China (No.2017YFB1400600), the Financial and Science Technology Plan Project of Xinjiang Production and Construction Corps under grant no.2020DB005, and the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) fund. This work aims to reduce the service latency and improve the QoS of edge computing in 5G-enabled IoV by leveraging

deep reinforcement learning.

Before this work, our study group has already published a number of papers in the field of smart city, intelligent transportation, edge computing, and cloud computing.