·移动互联与通信技术·　　文章编号：1000-3428(2022)12-0156-09　　文献标志码：A　　中图分类号：TP391

车联网中基于DDQN的边云协作任务卸载机制

于　晶[1]，鲁凌云[2]，李　翔[1]

（1.北京交通大学 计算机与信息技术学院，北京 100044；2.北京交通大学 软件学院，北京 100044）

摘　要：面对车载终端数据计算量的爆炸式增长，计算卸载是缓解车辆资源不足的有效手段。相比于单独研究云计算或边缘计算，让两者相互协作可以实现优势互补，提高系统的整体服务质量。在车联网中，制定适应环境动态性的卸载决策存在较大困难，其中任务的紧急程度也是一个不容忽视的因素。构建一个基于软件定义网络的边云协作任务卸载架构，并设计任务优先级的度量标准，将动态环境中的任务卸载决策问题建模为马尔可夫决策过程，从而最大化由时延和成本构成的任务平均效用。为了求解任务卸载决策，提出基于双深度Q网络的任务卸载决策算法以及基于优先级的资源分配方案，并设计一种卸载比例计算方法，以保障卸载的任务量能够在通信时间内上传完成的同时最小化任务处理时延。实验结果表明，相比于全部本地、全部卸载和平均分配资源3种固定的卸载算法，该算法时延和效用性能提高了2倍以上，在车辆数目适中的情况下，任务的完成比例可以稳定保持在100%。

关键词：车联网；边云协作；任务卸载；深度强化学习；优先级；资源分配

开放科学（资源服务）标志码（OSID）：

中文引用格式：于晶，鲁凌云，李翔. 车联网中基于DDQN的边云协作任务卸载机制[J]. 计算机工程，2022，48(12)：156-164.

英文引用格式：YU J，LU L Y，LI X. Edge-cloud collaborative task offloading mechanism based on DDQN in vehicular networks[J]. Computer Engineering，2022，48(12)：156-164.

Edge-Cloud Collaborative Task Offloading Mechanism Based on DDQN in Vehicular Networks

YU Jing[1]，LU Lingyun[2]，LI Xiang[1]

（1.School of Computer and Information Technology，Beijing Jiaotong University，Beijing 100044，China；
2.School of Software Engineering，Beijing Jiaotong University，Beijing 100044，China）

【Abstract】Computation offloading is a promising scheme to alleviate the shortage of vehicle resources facing the explosive growth trend of data computation. Compared with studying cloud computing or edge computing separately，integrating with each other can realize the complementary advantages and improve the overall quality of service. In vehicular networks，a primary challenge is to make offloading decisions，which can adapt to the dynamic environment. During this process，the urgency of tasks cannot be ignored. This paper constructs a collaborative edge-cloud task offloading architecture based on Software Defined Network（SDN），where the metrics of task priority is given. The task offloading problem is then formulated as a Markov Decision Process（MDP），which aims to maximize the utility composed of delay and cost. To solve task offloading decisions，this paper puts forward a task offloading decision algorithm based on Double Deep Q Network（DDQN）and a priority-based resource allocation scheme successively. On this basis，this paper designs a method of computing offloading ratio，which aims to minimize the task processing delay while ensuring that the part of tasks can be uploaded completely within the communication time. Simulation results show that the performance of delay and utility of the proposed algorithm is more than doubled compared to other fixed offloading algorithms such as All Local，All Offloading and Allocating Resources Evenly. Under the condition of moderate numbers of vehicles，the success rate of tasks can be maintained at 100%.

【Key words】vehicular networks；edge-cloud collaboration；task offloading；Deep Reinforcement Learning（DRL）；priority；resource allocation

DOI：10. 19678/j. issn. 1000-3428. 0063739

基金项目：国家自然科学基金面上项目（61771002）；赛尔网络新一代IPv6创新项目（NGII20170636）；中央高校基本科研业务费专项资金（2021CZ102）。

作者简介：于　晶（1997—），女，硕士研究生，主研方向为移动边缘计算；鲁凌云，教授、博士；李　翔，博士研究生。

收稿日期：2022-01-11　　**修回日期：**2022-02-14　　**E-mail：**2226533775@qq.com

0　概述

随着人工智能、5G移动通信和车联网技术的发展，大量的智能车载应用不断被开发，例如自动驾驶、碰撞预警、虚拟现实等[1]。然而，计算密集型和时延敏感型的任务对于车载终端来说无疑是一个巨大的挑战。移动云计算（Mobile Cloud Computing，MCC）能够将任务卸载到资源丰富的远程云服务器上，可以解决车辆计算能力不足的问题[2]，因此成为解决上述矛盾的一个关键技术，但远距离任务传输带来的时延无法满足安全类应用对实时性的要求[3]。对此，移动边缘计算（Mobile Edge Computing，MEC）被引入到车联网中，MEC服务器被部署到道路两侧的路边单元（Road Side Unit，RSU）中，为车辆提供近距离的任务卸载服务[4]。相比于云计算卸载，边缘计算卸载不仅可以缓解车辆终端的资源不足，而且可以降低任务传输时延，缓解核心网络的压力。

目前，针对车联网中的任务卸载问题，国内外学者展开了大量研究，但大部分研究都是单独针对云计算卸载或者边缘计算卸载，很少有研究关注云计算和边缘计算的协作。此外，文献往往假设MEC服务器具有足够资源，能够满足所有任务的卸载请求，然而事实并非如此。受硬件和成本的制约，MEC服务器可提供的资源也相对有限，尤其是在车辆密集的区域，仅依靠边缘层可能无法满足海量数据的计算和存储[5]。由于云计算和边缘计算都拥有各自的优势，云服务器适合处理计算密集但对实时性要求低的任务，而边缘服务器更适合处理计算量小或者对时延敏感的任务，因此，让两者相互协作以实现优势互补逐渐吸引了大家的关注。在网络架构方面，软件定义网络（Software Defined Network，SDN）技术由于具备转控分离、集中控制、开放接口、可编程等特性，因此对于实现边云异构网络的有效协作具有重要意义[6]。

在车联网中，影响卸载决策的因素比如车辆位置、服务器资源、信道状态等都具有时变性，如果使用传统的最优化算法或者启发式算法来求解最优决策，不仅要对环境做出很多静态假设，而且每当有新任务时，都需要重新经历一次复杂度超高的计算[7]，这对于那些对时延极其敏感的安全类应用来说无疑是巨大的隐患。除此之外，现有的工作在制定任务卸载决策和资源分配方案时很少考虑车载应用的紧急程度，所以任务卸载到每个服务器的可能性是相同的，而且卸载到同一个服务器的任务分配到的资源也是相同的，这也许会导致对时延敏感的任务无法在限制时间内完成。为简化问题，很多采取部分卸载方式的方案中也会忽略车辆在上传任务过程中的移动性，并假设任务可以在剩余通信时间内上传完成，但这在实际应用中是不现实的。

为解决上述问题，本文综合考虑环境动态性以及任务优先级，提出基于双深度Q网络（Double Deep Q Network，DDQN）算法的边云协作任务卸载机制，DDQN算法在与环境交互过程中可以对未来的环境变化进行一定程度的估计，通过不断学习并调整策略实现长期效用最大化。同时，提出基于任务优先级的资源分配方案，使优先级越高的任务能分配到的资源越多。此外，给出一种基于确定卸载决策和资源分配方案的的卸载比例计算方法，在任务能够完成上传的前提下，最小化任务处理时间。

1　相关研究

针对车辆边缘计算环境中的任务卸载决策问题，目前已经有大量的研究工作。文献[8]提出一种基于遗传算法和启发式规则的混合智能优化算法，制定的卸载决策可以最小化时延和资源消耗。文献[9]对多用户多MEC服务器场景中的负载均衡和任务卸载问题进行了联合优化。文献[10-12]也基于各种数值优化理论提出相应的任务卸载和资源分配方案。强化学习作为解决复杂环境下序列决策问题的强大武器，被学者们用来解决任务卸载问题。文献[13]将计算卸载调度问题形式化为优化时延和能耗组成的长期成本问题，并将其建模为一个等价的马尔可夫决策过程（Markov Decison Process，MDP），设计的基于深度强化学习（Deep Reinforcement Learning，DRL）的算法可以得到问题的最优解。然而，上述工作均假设MEC服务器拥有足够多的计算和存储资源，但是在实际场景中，MEC的资源受限于硬件和成本，在车辆密集的区域可能无法保证服务质量。

边缘计算与云计算的融合可以构建出更加高效可靠的边云协作任务卸载环境。文献[14]为充分利用云服务器和边缘服务器的资源，提出约束随机化卸载和约束启发式贪心卸载两种方案解决协同任务卸载问题。文献[15]通过构建一个云端-边缘-车辆三层协同的网络，将任务卸载决策问题形式化为收益和服务质量最大化问题，并提出由Kuhn-Munkres算法、遗传算法、内点法和KKT条件组成的算法求解上述非确定性多项式（Nondeterministic Polynomially，NP）问题。包括上述文献在内的很多工作都假设动态的车联网环境可以被准确地建模，然而，这在实际情况下很难做到。因此，能够摆脱模型限制的DRL算法特别适合用来解决复杂动态车联网环境中的任务卸载问题。

文献[16]研究了边云协作环境中的非独立任务卸载问题，并且提出一个基于强化学习和序列二次规划的两级交替方法框架。文献[17-19]利用DRL算法解决了车联网中的任务卸载问题。近年来，利用SDN协助车联网计算卸载的研究有一定的成果[20-22]，例如，文献[21]通过设计一种集成SDN和MEC的车联网框架，实现网络、缓存和计算的动态编排。文献[22]构建一种MEC辅助的软件定义车联网架构，并提出一种带重处理机制的任务卸载方案，能够在满足时延约束的前提下最大化可靠性。

与现有工作相比，本文考虑了车联网的时变性、任务的优先级、通信链路的持续时间以及计算资源成本对任务卸载决策制定的综合影响。还提出一种基于DDQN的算法来学习任务卸载决策，该算法可以最大化由时延和成本构成的长期效用。

2 系统模型

2.1 系统架构分析

如图1所示，本文构建的基于SDN的边云协作车联网架构在逻辑上由用户层、数据层和控制层组成。

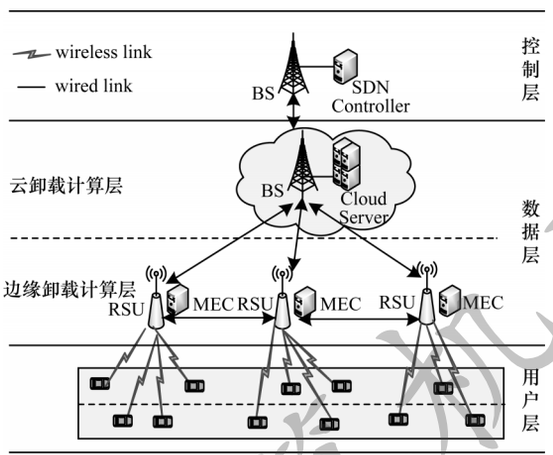

图1 基于SDN的边云协作车联网架构

Fig.1 Architecture of edge-cloud collaborative vehicular network based on SDN

用户层由行驶在一段单向双车道公路上的车辆用户构成。本文假设每个时隙最多有 n 个车辆，车辆集合记为 $N=\{1,2,\cdots,n\}$。每个车辆在同一时隙最多可以拥有一个待处理的任务，车辆 i 的任务描述为 $T_i=\{d_i,c_i,t_i^{\max},\theta_i\}$，其中：$d_i$ 是任务的总数据量，单位为比特(bit)；c_i 是完成任务 T_i 所需要的CPU周期数；$t_i^{\max}$ 是任务可以容忍的时延上限；$\theta_i(0\leqslant\theta_i\leqslant1)$ 是任务的卸载比例，为充分利用车辆本地的计算资源并进一步降低任务处理时延，本文采取部分卸载方式，将 $(1-\theta_i)$ 比例的任务留在本地处理，将车辆可用计算资源记为 $F_{\text{local}}=\{f_1^{\text{local}},f_2^{\text{local}},\cdots,f_n^{\text{local}}\}$。

数据层由边缘卸载计算层和云卸载计算层构成，负责执行用户层车辆卸载的计算任务。其中，边缘计算层是由 m 个依次部署在RSU中的边缘服务器构成，边缘服务器资源相对比较少，主要负责执行对时延要求高或者对计算资源要求少的任务。车辆会通过无线车辆到基础设施(Vehicle-to-Infrastructure，V2I)链路将任务上传到其通信范围内的中继RSU中，然后中继RSU通过有线基础设施到基础设施(Infrastructure-to-Infrastructure，I2I)链路将任务转发给目标MEC服务器执行。m 个边缘服务器当前可用的资源可以表示为 $F_{\text{mec}}=\{f_1^{\text{mec}},f_2^{\text{mec}},\cdots,f_m^{\text{mec}}\}$。云计算层由1个部署在基站(Base Station，BS)中的云服务器构成，云服务器具有充足的资源，用 f_0^{mcc} 表示。但是远距离的任务传输会产生不可预测的时延，安全和隐私也得不到保障，所以云服务器的主要作用是弥补边缘计算层资源的不足。一些计算量大且对时延要求较低的任务会被卸载到云服务器执行，中继RSU通过有线光纤链路将任务转发到云服务器中。本文构建的边云协作计算卸载环境可以通过优势互补的方式克服云计算和边缘计算各自的缺点，充分发挥各自的长处。

控制层主要是指部署在基站中的SDN控制器。逻辑上集中的SDN控制层具有瞬时获取全局网络状态的能力，可配置、可编程、对第三方开放的网络基础能力等一系列突破对于实现边缘计算和云计算的有效协作具有重要意义，能够帮助实现两者的优势互补，满足边云协作计算网络对于统一控制、任务合理卸载和资源有效分配的需求。

在本文构建的基于SDN的边云协作计算环境中，车辆的计算任务会被部分卸载到云服务器或边缘服务器中执行，所有车辆在某个时隙的卸载决策集合可以表示为 $X=\{\alpha_1,\alpha_2,\cdots,\alpha_i,\cdots,\alpha_n\}$，其中，$\alpha_i\in\{0,1,2,\cdots,m\}$，当 $\alpha_i=0$ 时，表示车辆 i 的任务将被卸载到云服务器执行；当 $\alpha_i=j,j\in\{1,2,\cdots,m\}$ 时，表示车辆 i 的任务将被卸载到编号为 j 的边缘服务器上执行。

2.2 任务优先级设定

考虑到车载应用的多样性，比如碰撞预警这类对时延要求极高的安全类应用、语音识别这类对时延不敏感的娱乐性应用等，本文将抽象出一个任务优先级的评定标准，用于指导计算资源的分配以及卸载策略的制定。除了任务的紧急程度，任务所需的计算量也是任务卸载和资源分配时必须参考的指标，任务的优先级可以定义为式(1)所示：

$$p_i=p_{\text{sf}}\frac{c_i}{t_i^{\max}} \tag{1}$$

其中：p_{sf} 是缩放系数，用来调整任务计算量或者时延对优先级的影响程度。式(1)表明，任务所需计算量越大，或者任务所能容忍的最大时延越小，任务的优先级越高。

2.3 通信模型

行驶在道路上的车辆需要将任务通过无线V2I链路传给中继RSU，链路的传输速率直接影响任务的上传时间。根据香农公式，车辆 i 与编号为 k 的中继RSU之间的无线数据传输速率 $R_{i,k}$ 可以利用式(2)计算得到：

$$R_{i,k}=B_{i,k}\times\text{lb}\left(1+\frac{P_{i,k}G_{i,k}}{\sigma^2}\right) \tag{2}$$

其中：$B_{i,k}$ 是信道带宽；$P_{i,k}$ 是车辆 i 的发射功率；σ^2 是噪声功率；信道增益 $G_{i,k}$ 与车辆和RSU之间的距离 $d_{i,k}$ 密切相关，具体关系可以表示为 $G_{i,k}=-30-$

$35\lg(d_{i,k})$[23]。此外，本文还假设车辆和RSU之间的无线通信采用了正交频分多址技术，所以忽略了信道之间的干扰。

2.4　计算模型

本文除了将任务处理时延作为评价指标，为了避免计算资源的浪费，还引入成本这一因素来指导决策的制定。本文采取部分卸载方式，将任务在本地和服务器中按比例并行处理。任务 T_i 在本地处理部分所需时间 t_i^{local} 的计算式如式(3)所示：

$$t_i^{\text{local}}=\frac{(1-\theta_i)c_i}{f_i^{\text{local}}} \tag{3}$$

任务在本地处理时不消耗服务器的资源，所以没有成本。卸载处理的那部分任务有边缘服务器和云服务器两类选择。

2.4.1　边缘卸载

当 $\alpha_i=j,j\in\{1,2,\cdots,m\}$ 时，车辆 i 的任务将被卸载到编号为 j 的MEC服务器中执行。任务的上传时间 t_i^{trans} 计算式如式(4)所示：

$$t_i^{\text{trans}}=t_{i,k}^{\text{trans}}+t_{k,j}^{\text{trans}}=\frac{\theta_i d_i}{R_{i,k}}+H_{k,j}\times\frac{\theta_i d_i}{R_{\text{fiber}}} \tag{4}$$

其中：$t_{i,k}^{\text{trans}}=\theta_i d_i/R_{i,k}$ 是任务从车辆 i 传输到编号为 k 的中继RSU所需要的时间；$t_{k,j}^{\text{trans}}=H_{k,j}\times\theta_i d_i/R_{\text{fiber}}$ 是任务从中继RSU转发到编号为 j 的目标RSU所需要的时间，$H_{k,j}$ 是两个RSU之间的跳数，R_{fiber} 是有线I2I链路的数据传输速率。任务的执行时间 t_i^{exe} 的计算式如式(5)所示：

$$t_i^{\text{exe}}=\frac{\theta_i c_i}{f_{i,j}^{\text{mec}}} \tag{5}$$

其中：$f_{i,j}^{\text{mec}}$ 是MEC服务器 j 分配给车辆 i 的计算资源。车辆 i 卸载部分任务的总处理时间 t_i^{off} 的计算式如式(6)所示：

$$t_i^{\text{off}}=t_i^{\text{trans}}+t_i^{\text{exe}} \tag{6}$$

综上所述，车辆 i 的任务总处理时间 t_i^{mec} 的计算式如式(7)所示：

$$t_i^{\text{mec}}=\max\{t_i^{\text{local}},t_i^{\text{off}}\} \tag{7}$$

在任务处理时间的基础上，结合计算资源所消耗的成本，可以得到车辆 i 的效用 u_i^e，其计算式如式(8)所示：

$$u_i^e=\delta_i\frac{t_i^{\max}-t_i^{\text{mec}}}{t_i^{\max}}-(1-\delta_i)G_{sf}C_e f_{i,j}^{\text{mec}} \tag{8}$$

其中：δ_i 用来调整时延和成本所占的权重；C_e 是MEC服务器的单位计算资源成本；G_{sf} 是缩放因子，用来将成本值调整到合适的取值范围内。

2.4.2　云卸载

当 $\alpha_i=0$ 时，车辆 i 的任务将被卸载到云服务器中执行。任务的传输时间 t_i^{trans} 计算式如式(9)所示：

$$t_i^{\text{trans}}=t_{i,k}^{\text{trans}}+t_{k,c}^{\text{trans}}+t_{c,k}^{\text{trans}}=\frac{\theta_i d_i}{R_{i,k}}+(\theta_i d_i+d_i^0)v_{\text{fiber}} \tag{9}$$

其中：$v_{\text{fiber}}=1\ \mu\text{s/bit}$，是RSU到云服务器的单位数据传输时延，$d_i^0$ 是返回的执行结果数据量，一般远小于输入的数据量，所以往往忽略其占用的传输时间。

任务的执行时间 t_i^{exe} 计算式如式(10)所示：

$$t_i^{\text{exe}}=\frac{\theta_i c_i}{f_{i,0}^{\text{mec}}} \tag{10}$$

其中：$f_{i,0}^{\text{mec}}$ 是云服务器分配给车辆 i 的计算资源。车辆 i 卸载的那部分任务的总处理时间 t_i^{off} 计算式如式(11)所示：

$$t_i^{\text{off}}=t_i^{\text{trans}}+t_i^{\text{exe}} \tag{11}$$

综上所述，车辆 i 任务的总处理时间为 t_i^{mec}，其计算式如式(12)所示：

$$t_i^{\text{mec}}=\max\{t_i^{\text{local}},t_i^{\text{off}}\} \tag{12}$$

在任务处理时间的基础上，结合计算资源所消耗的成本，可以得到车辆 i 的效用 u_i^c，其计算式如式(13)所示：

$$u_i^c=\delta_i\frac{t_i^{\max}-t_i^{\text{mec}}}{t_i^{\max}}-(1-\delta_i)G_{sf}C_c f_{i,0}^{\text{mec}} \tag{13}$$

其中：C_c 是MCC服务器的单位计算资源成本。

2.5　效用函数

基于以上定义，可以得到一个用于衡量卸载决策和资源分配方案优劣的效用函数：

$$U(X)=\frac{1}{|N|}\sum_{i\in N}p_i u_i \tag{14}$$

式(14)表示所有任务的平均效用，并且将任务的优先级作为系数进一步加强任务优先级的重要性。其中，X 是问题的解，当 $\alpha_i\in\{1,2,\cdots,m\}$ 时，$u_i=u_i^e$，当 $\alpha_i=0$ 时，$u_i=u_i^c$。因此，在求解卸载策略和资源分配方案时，优化目标就是最大化上述由时延和成本构成的效用函数。

3　边云协作任务卸载机制

车辆移动性、计算资源、信道状态的时变性等因素导致车联网环境的高度不确定，此时将任务卸载决策问题建模为MDP过程，这是一种高效的解决方案。尽管在理论上可以采用动态规划或者启发式算法求解MDP，但其超高的计算复杂度无法满足车载应用对实时性的要求。此外，MDP的状态转移概率矩阵也很难获取，在这种情况下，不基于模型的强化学习(Reinforcement Learning，RL)算法可以用来获取最优策略。RL算法不需要任何先验知识，通过不断与环境交互学习，最终获得的决策模型可以实现长期效用最大化。近年来，深度强化学习算法又利用神经网络强大的近似能力解决了RL算法状态空间有限的问题。

因此，本节首先将效用优化问题转化为马尔可夫决策过程，并采用深度强化学习方法求解最优策略。然后，阐述在任务卸载策略已知情况下的基于优先级的资源分配方案。最后，在卸载策略和资源分配方案

已知的情况下，以最小化任务处理时间和保证传输可靠性为目标，给出任务最佳卸载比例的计算方法。

3.1 基于DDQN的任务卸载决策机制

DDQN[24]是一种基于价值的深度强化学习算法，适用于车辆任务卸载这种具有大规模离散动作空间的场景。DDQN的核心思想是使用深度神经网络近似代替动作价值（以下简称“Q值”）函数，将环境状态作为输入，输出是每个动作对应的Q值，其中最大Q值对应的动作将成为当前最优策略，环境也会反馈一个奖励值来评价动作的好坏，同时也用来优化神经网络的参数，然后环境将进入下一个状态，不断循环上述过程直到收敛。

本节首先阐述MDP的3大要素，即状态空间、动作空间和奖励函数，然后介绍动作价值函数以及最优策略的定义，最后给出动作价值网络的更新过程。

3.1.1 状态空间

具有全局信息瞬时收集能力的SDN控制器恰好可以承担DRL中智能体这一角色，在每个时隙中，SDN控制器监控其通信范围内的车联网环境，并且收集状态信息，包括车辆位置、车辆任务、以及边缘服务器和云服务器的资源可用信息。定义状态空间为S，t时隙的状态$s(t)$如下：

$$\begin{aligned}s(t)=\{&x_1(t),x_2(t),\cdots,x_n(t),d_1(t),d_2(t),\cdots,d_n(t),\\&c_1(t),c_2(t),\cdots,c_n(t),t_1^{\max}(t),t_2^{\max}(t),\cdots,t_n^{\max}(t),\\&f_1^{\mathrm{mec}}(t),f_2^{\mathrm{mec}}(t),\cdots,f_m^{\mathrm{mec}}(t),f_0^{\mathrm{mec}}(t)\}\end{aligned}\tag{15}$$

3.1.2 动作空间

根据t时隙观察到的状态$s(t)$，智能体会输出一个动作作为当前时隙采取的卸载策略。定义动作空间为A，t时隙的动作$a(t)$如式（16）所示：

$$\begin{aligned}a(t)=\{&\alpha_1(t),\alpha_2(t),\cdots,\alpha_i(t),\cdots,\alpha_n(t),\phi_1(t),\\&\phi_2(t),\cdots,\phi_j(t),\cdots,\phi_m(t),\phi_0(t)\}\end{aligned}\tag{16}$$

其中：$\alpha_i(t),i\in\{0,1,\cdots,n\}$是车辆$i$在$t$时隙的卸载目标服务器，其他两类动作，即$\phi_j(t),j\in\{1,2,\cdots,m\}$和$\phi_0(t)$分别是边缘服务器和云服务器在当前时隙提供的资源比例，用来降低资源成本，尽量用更少的资源完成更多的任务。

3.1.3 奖励函数

在t时隙，智能体根据状态$s(t)$给出动作$a(t)$之后，环境立刻反馈一个奖励R_t，如式（17）所示。该奖励值可以用来评价动作$a(t)$的好坏，并指导价值网络参数的更新。

$$R_t=\frac{1}{|N|}\sum_{i\in N}p_iu_i\tag{17}$$

3.1.4 动作价值函数

强化学习算法关注的是系统长期效用最大化，即从当前时隙到结束时隙的累计效用最大。t时隙的长期累积折扣效用可以表示为$U_t=R_t+\gamma R_{t+1}+\gamma^2R_{t+2}+\cdots+\gamma^3R_{t+n}$，其中，$\gamma$是折扣因子，未来时隙的奖励对当前时隙的影响要打一个折扣。由于状态转移和动作选择的不确定性，在当前时隙，很难得到准确的U_t。在上述情况下，通过马尔可夫性假设和求期望的方式可以消除随机性，得到只与当前时隙的状态s_t和动作a_t有关的动作价值函数$Q_\pi(s_t,a_t)$，其计算式如式（18）所示：

$$Q_\pi(s_t,a_t)=E_\pi\left[U_t\middle|S_t=s_t,A_t=a_t\right]\tag{18}$$

3.1.5 最优策略的定义

解决强化学习问题意味着要寻找一个最优策略π^*，该策略使个体在与环境交互过程中获得的效用始终大于其他策略。一般来说，很难直接找到这个最优策略，只能通过比较若干不同策略的优劣来找到一个相对比较好的策略，也就是局部最优解。比较策略的优劣可以通过比较策略产生的动作价值实现，最优策略对应的动作价值优于其他策略的动作价值。所以，基于最优动作价值，可以给出最优策略的定义如式（19）所示：

$$\pi^*(s_t)=a_t,a_t=\underset{a_t\in A}{\operatorname{argmax}}\,Q^*(s_t,a_t)\tag{19}$$

从式（19）可以看出，在当前状态下，最大Q值对应的动作就是最优动作。因此，可以通过比较当前状态下所有动作的Q值来找到最优动作。

3.1.6 动作价值网络的更新

根据贝尔曼方程，可以得到t时隙的动作价值和$t+1$时隙的动作价值的递推关系，如式（20）所示，利用该公式，在理论上可以求出所有情况的Q值。

$$Q_\pi(s_t,a_t)=R_t+\gamma E_\pi\left[Q_\pi(s_{t+1},a_{t+1})\right]\tag{20}$$

然而，对于状态空间很大的复杂场景，使用动态规划或者Q-learning这种传统的方式计算Q值将产生巨大的时间和内存成本，这明显不适用于本文场景。深度强化学习解决上述问题的方式是利用神经网络近似计算Q值，参数为ω的动作价值网络将状态作为输入，输出不同动作的Q值，神经网络的正向传播所需计算量极小。所以，解决深度强化学习问题的核心变成了动作价值网络参数的训练，不断优化网络参数，使计算出的Q值尽可能接近真实值。

在深度强化学习算法中，同样使用梯度下降法更新神经网络的参数，此时需要一个真实值与神经网络的估计值进行比较，但真实值却无法获得。 根据式（20）可以得到式（21）：

$$Q_\pi(s_t,a_t)-\gamma E_\pi\left[Q_\pi(s_{t+1},a_{t+1})\right]=R_t\tag{21}$$

使用$\gamma Q_\pi(s_{t+1},a_{t+1})$近似代替$\gamma E\left[Q_\pi(s_{t+1},a_{t+1})\right]$后，式（21）的等号左边变成$Q_\pi(s_t,a_t)-\gamma Q_\pi(s_{t+1},a_{t+1})$，$Q_\pi(s_t,a_t)$和$Q_\pi(s_{t+1},a_{t+1})$都可以通过神经网络计算，而等号右边是$t$时隙获得的真实奖励，使$Q_\pi(s_t,a_t)$和$Q_\pi(s_{t+1},a_{t+1})$的差值尽量接近$R_t$是神经网络的优化目标。根据贪心策略，下一个状态$s_{t+1}$是在状态$s_t$下执行最优动作$a_t$得到的，然后将$s_{t+1}$作为输入，计算得到的$Q$值中，选取最大$Q$值用于参数更新。此时，可以得到均方差损失函数，如式（22）所示：

$$\frac{1}{q}\sum_{j=1}^{q}(R_j+\gamma\max_{a_{j+1}\in A}Q_\pi(s_{j+1},a_{j+1},\omega)-Q_\pi(s_j,a_j,\omega))^2 \tag{22}$$

其中：q是从经验池中采样得到的样本个数；$R_j+\gamma\max_{a_{j+1}\in A}Q_\pi(s_{j+1},a_{j+1},\omega)$通常被称作目标$Q$值。

原始DQN算法使用同一个神经网络计算$Q_\pi(s_t,a_t)$和$Q_\pi(s_{t+1},a_{t+1})$，导致相关性太强，不利于收敛。对此，Natural DQN算法使用两个结构相同的网络来分别计算$Q_\pi(s_t,a_t)$和$Q_\pi(s_{t+1},a_{t+1})$，并将计算当前时隙$Q_\pi(s_t,a_t)$的网络称作当前Q网络，将计算$Q_\pi(s_{t+1},a_{t+1})$的网络成为目标Q网络，目标Q网络无需训练，只是定期从当前Q网络复制参数过来，以减少目标Q值计算和神经网络参数更新的依赖关系。然而，使用贪心策略选择最大的$Q_\pi(s_{t+1},a_{t+1})$这种方式会导致过度估计的问题，使训练得到的模型有很大的偏差。对此，在Natural DQN的基础上，DDQN算法解耦了目标Q值动作的选择和目标Q值的计算，先利用当前Q网络选出下一个状态下Q值最大的动作$a'=\underset{a_{t+1}\in A}{\operatorname{argmax}}\,Q_\pi(s_{t+1},a_{t+1},\omega)$，然后从目标$Q$网络获得动作$a'$对应的$Q$值，即$Q'_\pi(s_{t+1},a',\omega')$，值得注意的是，此时的目标$Q$值未必是最大值。以上是对算法思想的介绍，基于DDQN的任务卸载决策算法具体描述见算法1。

算法1　基于DDQN的任务卸载决策算法

初始化　当前Q网络的参数ω，目标Q网络参数ω′=ω；清空经验回放集合D；

1.for each episode e do：

2.重置初始时隙t←0；

3.初始化环境状态s_t；

4.将s_t作为当前Q的输入，利用ε-greedy方法选择动作a_t；

5.在状态s_t下执行动作a_t，返回s_{t+1}状态、奖励R_t以及是否是终止状态done；

6.将$\{s_t, a_t, R_t, s_{t+1}\}$四元组存入经验回放集合；

7.从D中采样q个样本，并计算目标Q值Q'_j

$$Q'_j=\begin{cases}R_j, \text{ done is true}\\ R_j+\gamma(sQ'_{j+1}, \arg\max\limits_{a_{j+1}\in A}Q(s_{i+1}, a_{j+1}, \omega), \omega'), \text{ done is false}\end{cases}$$

8.构造均方差损失函数$\frac{1}{q}\sum_{j=1}^{q}(Q'_j-Q(s_j,a_j,\omega))^2$，利用神经网络的梯度反向传播更新当前Q网络参数ω；

9.如果t%C=0，更新目标Q网络参数ω′=ω；

10.如果s_t是终止状态，当前循环结束，e←e+1，转到步骤2；

11.$s_t=s_{t+1}$，t←t+1，转到步骤4；

12.end for

3.2　基于优先级的资源分配方案

本文提出一个基于优先级的资源分配方案，优先级越高的任务可以分配到越多的计算资源，相比于平均分配的方式，可以提高高优先级任务的完成率。通过上一节的策略模型可以得到当前时隙的卸载策略，在此基础上，本节首先根据优先级计算任务的资源分配系数，然后为其分配相应比例的计算资源。

3.2.1　边缘服务器的资源分配

如果车辆i的卸载目标服务器的编号为j，那么任务T_i的资源分配系数λ_i如式(23)所示：

$$\lambda_i=\frac{p_i}{\sum\limits_{i\in \mathrm{mec}_j}p_i} \tag{23}$$

其中：mec_j代表卸载目标为编号j的MEC服务器的车辆集合。

任务T_i分配到的资源$f_{i,j}^{\mathrm{mec}}$如式(24)所示：

$$f_{i,j}^{\mathrm{mec}}=\lambda_i\phi_j f_j^{\mathrm{mec}} \tag{24}$$

3.2.2　云服务器的资源分配

如果车辆i的卸载目标服务器为云服务器，那么任务T_i的资源分配系数λ_i如式(25)所示：

$$\lambda_i=\frac{p_i}{\sum\limits_{i\in \mathrm{mcc}_0}p_i} \tag{25}$$

其中：mcc_0代表卸载目标云服务器的车辆集合。任务T_i分配到的资源$f_{i,0}^{\mathrm{mcc}}$如式(26)所示：

$$f_{i,0}^{\mathrm{mcc}}=\lambda_i\phi_0 f_0^{\mathrm{mcc}} \tag{26}$$

3.3　卸载比例计算

采取部分卸载的方式可以充分利用本地计算资源，进而降低任务处理时延。由于任务的总处理时延$t_i^{\mathrm{total}}=\max\{t_i^{\mathrm{local}},t_i^{\mathrm{off}}\}$，所以当$t_i^{\mathrm{local}}=t_i^{\mathrm{off}}$时，$t_i^{\mathrm{total}}$最小。在卸载策略和资源分配方案已知的情况下，只有$\theta_i$是未知数，所以利用上述等式关系可以计算出一个使任务处理时延最小的卸载比例。此外，由于车辆具有移动性，因此还要考虑待卸载的部分任务能否在车辆驶出中继RSU通信范围之前上传完成，也就是说要满足$\theta_i d_i\leqslant R_{i,k}t_i^{\mathrm{rest}}$，其中$t_i^{\mathrm{rest}}$是车辆与中继RSU剩余通信时间；如果不能满足，就要选择减小卸载比例，或者等待车辆行驶到下一个RSU范围内时再上传任务，具体可以根据两种方式的任务总处理时间进行选择。综合以上考虑，任务卸载比例的计算主要有以下3种情况：

1）当$t_i^{\mathrm{local}}(\theta_i')=t_i^{\mathrm{off}}(\theta_i')$且$\theta_i' d_i\leqslant R_{i,k}t_i^{\mathrm{rest}}$时：$\theta_i=\theta_i'$。此时，卸载比例$\theta_i'$既可以取得最小任务处理时间，又能保证任务在当前RSU通信时间内完成上传。

2）当$\theta_i' d_i>R_{i,k}t_i^{\mathrm{rest}}$且$t_i^{\mathrm{total}}(\theta_i'')\leqslant t_i^{\mathrm{total}}(\theta_i''')$时：$\theta_i=\theta_i''$。此时，$\theta_i'$比例的任务无法在$t_i^{\mathrm{rest}}$时间内上传完成，需要分别计算$t_i^{\mathrm{total}}(\theta_i'')$和$t_i^{\mathrm{total}}(\theta_i''')$，如果$t_i^{\mathrm{total}}(\theta_i'')\leqslant t_i^{\mathrm{total}}(\theta_i''')$，则$\theta_i=\theta_i''$。这里给出$t_i^{\mathrm{total}}(\theta_i'')$的计算方式，首先利用等式$\theta_i d_i=R_{i,k}t_i^{\mathrm{rest}}$可以计算出能够上传完成的最大比例$\theta_i''$，在卸载$\theta_i''$比例任务的情况下，任务的总处理时间为$t_i^{\mathrm{total}}(\theta_i'')=\max\{t_i^{\mathrm{local}}(\theta_i''),t_i^{\mathrm{off}}(\theta_i'')\}=t_i^{\mathrm{local}}(\theta_i'')$。

3）当 $\theta_i' d_i > R_{i,k} t_i^{\text{rest}}$ 且 $t_i^{\text{total}}(\theta_i'') > t_i^{\text{total}}(\theta_i''')$ 时：$\theta_i = \theta_i'''$。当车辆选择驶入下一个RSU范围内再开始处理任务时，那么任务的总处理时间为 $t_i^{\text{total}} = t_i^{\text{rest}} + \max\{t_i^{\text{local}}, t_i^{\text{off}}\}$，可以计算出使得 $t_i^{\text{total}}(\theta_i''')$ 最小的任务卸载比例 θ_i'''，同样当 $t_i^{\text{local}}(\theta_i''') = t_i^{\text{off}}(\theta_i''')$ 时，$t_i^{\text{total}}(\theta_i''')$ 才能最小，此时 $t_i^{\text{total}}(\theta_i''') = t_i^{\text{rest}} + t_i^{\text{local}}(\theta_i''')$。

4 实验结果与分析

本文通过python语言对本文算法进行仿真并分析其性能。仿真场景是由1个云服务器、3个边缘服务器和若干车辆用户组成的云-边-车辆3层结构。环境参数和神经网络的参数设置如表1所示。

表1 参数设置

Table 1 Simulation parameters

仿真参数	参数值
RSU的覆盖范围/m	100
车辆的速度均值/(m·s^{-1})	20
车辆的速度方差/(m·s^{-1})	3
无线信道带宽 $B_{i,k}$/GHz	10~20
车辆传输功率 $P_{i,k}$/dBm	30~40
任务 T_i 的输入数据量大小 d_i/KB	500~1 000
任务要求的CPU处理速度/(cycles·bit^{-1})	1 000
任务 T_i 能容忍的最大时延 $t_i^{\max}$/s	0.8~4
噪声功率 σ^2/dBm	−100
车辆的CPU频率 f_i^{local}/GHz	1~2
MEC服务器的CPU频率 f_j^{mec}/GHz	20
MCC服务器的CPU频率 f_0^{mcc}/GHz	100
MEC服务器单位资源成本 C_e	0.03
MCC服务器单位资源成本 C_c	0.015
有线光纤数据传输速率 R_{fiber}/(Gb·s^{-1})	0.8
神经网络层数以及神经元个数	3,40
初始探索率	0.9
经验回放集合大小	3 000
采样个数	32

为验证本文算法的性能，选择以下5种算法进行比较。

1）Natural DQN算法。该算法是一种深度强化学习算法，在仿真实验中，除了目标 Q 值的计算不同，其他均与本文算法保持一致。

2）模拟退火算法。与本文算法的区别是，该算法在每次迭代中会随机选择任务的卸载目标服务器，并以所有任务的平均效用作为优化目标，所以只要迭代的次数足够多，在理论上可以找到使平均效用最大的策略。

3）平均分配资源算法，在资源分配时，不考虑任务的优先级。

4）全部卸载处理算法，任务全部卸载处理。

5）全部本地处理算法，任务全部留在本地处理。

图2是本文算法和Natural DQN算法的收敛过程，为更清晰地展示2个算法的差别，图2(b)进一步放大了前2 000步的迭代过程。可以看出在迭代400步以后，Natural DQN算法的波动更明显，而且均方误差大于本文算法，这恰好验证了Natural DQN算法对于目标动作价值的过渡估计。此外，本文算法在迭代2 000步左右就已经完全收敛，而Natural DQN的曲线在迭代8 000步左右才趋于一条稳定的直线算法。

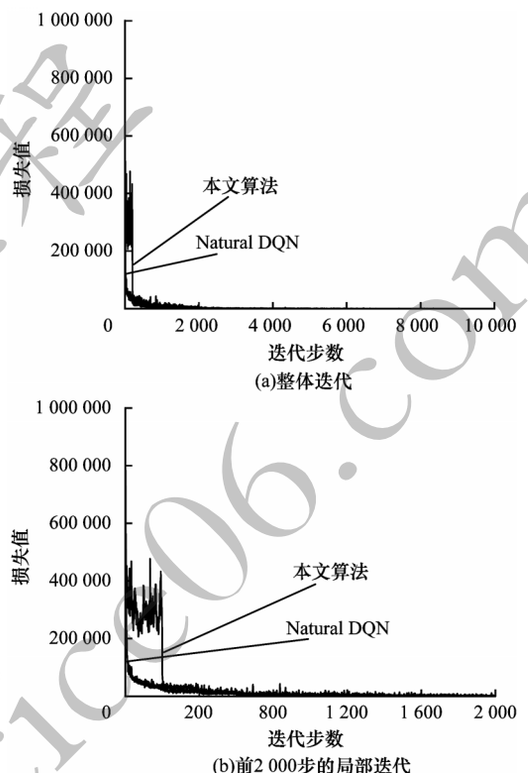

图2 不同算法的损失值变化

Fig.2 Loss value change of different algorithms

图3是不同算法产生的平均任务处理效用。将MEC和MCC服务器的数量分别固定为3和1，针对10种不同的车辆数进行对比实验。由图3可知，随着任务数量的增多，除了全部本地执行算法以外的所有算法产生的平均效用都呈现下降的趋势，这是因为分配给每个任务的计算资源逐渐减少，导致完成任务所需的时间增多。全部本地执行算法的性能最差，而模拟退火算法获得了效用最大的卸载决策，但是付出了很高的时间成本，相反，本文算法和Natural DQN算法却可以在极短的时间内获得近似最优解。此外，相比于平均分配资源这种方式，本文算法在效用方面获得了3倍以上的提升，这说明本文基于优先级的资源分配方式让资源得到了更好的利用。最后，相比于全部卸载执行算法，本文算法采取部分卸载的方式显然更加合理，效用至少提高了2倍。

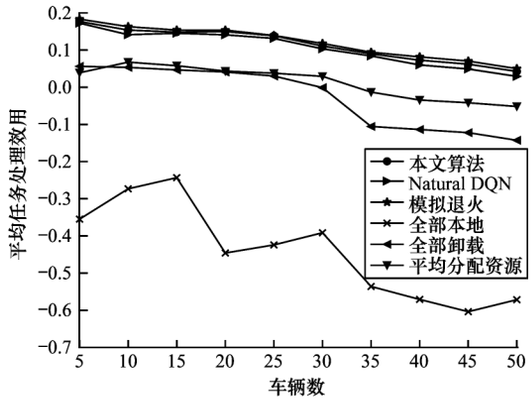

图3　不同算法的平均任务处理效用

Fig.3　Average task processing utility of different algorithms

图4是不同算法产生的平均任务处理时延。可以看出,除了全部本地执行算法,随着任务数增多,其他算法的平均时延都逐渐增多,这是因为任务分配到的资源变少,而全部本地执行算法主要与本地资源量有关,不受任务数量影响,所以变化较小。除了模拟退火算法,本文算法在时延性能方面要优于其他算法,其中,全部卸载算法卸载的任务量最大,所以产生的时延大幅增长。仔细观察还可以发现,时延与效用的关系并不绝对,因为所有算法的优化指标都是效用,而效用由时延和成本两部分构成,所以效用大不代表时延小。

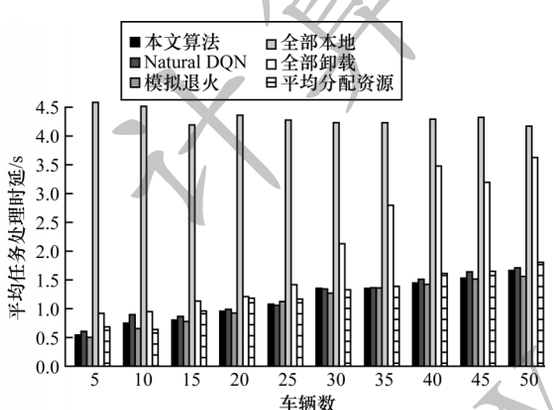

图4　不同算法的平均任务处理时延

Fig.4　Average tasks processing delay of different algorithms

图5和图6分别是任务和高优先级任务的完成比例。首先,在资源总量不变的情况下,当任务数量增加到一定程度时,任务的完成率将不可避免地下降,其中,全部卸载算法由于卸载的任务量大及浪费了本地资源,所以任务完成率很低。而对于全部本地执行算法,利用有限的本地资源,几乎很难在规定时间内完成任务,任务量大且时延敏感的高优先级任务完成比例甚至均为0。其次,本文算法的任务和高优先级任务完成比例都最接近模拟退火算法,在任务数适中的情况下,任务完成比例可以稳定维持在100%。由于在卸载决策制定和资源分配时都将优先级作为参考标准,因此相对于平均分配资源算法,本文算法能大幅提升高优先级任务的完成比例。

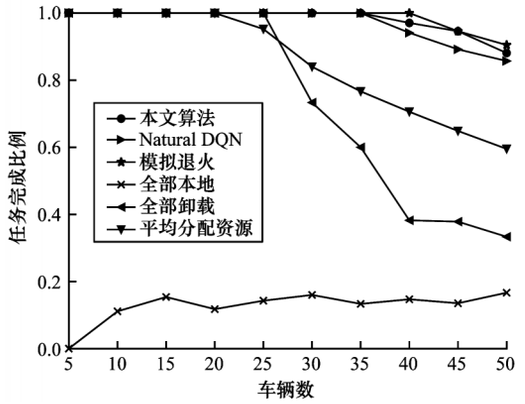

图5　不同算法的任务完成比例

Fig.5　Success rate of tasks processing of different algorithms

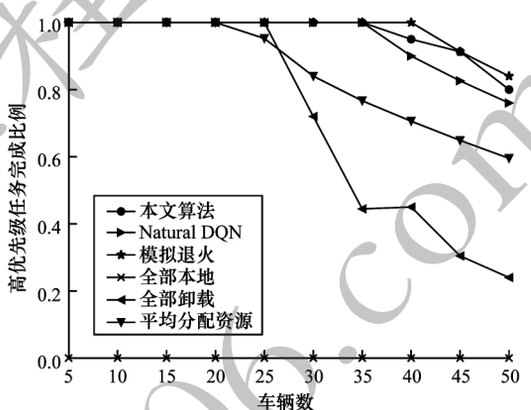

图6　不同算法的高优先级任务完成比例

Fig.6　Success rate of higher priority tasks processing of different algorithms

5　结束语

为缓解车载应用日益丰富与车辆资源有限之间的矛盾,本文构建一种基于SDN的边云协作车联网计算卸载架构,并将任务卸载问题建模为马尔可夫决策过程,把时延和计算资源成本构成的效用作为优化目标,提出基于DDQN的任务卸载机制和基于任务优先级的资源分配方案。通过在卸载决策制定和资源分配过程中考虑任务优先级因素,提高系统平均效用及降低任务处理时延。实验结果表明,本文提出的卸载比例计算方法在一定程度上提高了任务传输的可靠性,实现当前策略的时延最小化。下一步将引入无线电干扰等复杂因素,构建更加符合实际场景的通信模型。

参考文献

[1] DU J,JIANG C X,WANG J,et al. Machine learning for 6G wireless networks:carrying forward enhanced bandwidth,massive access,and ultrareliable/low-latency service[J]. IEEE Vehicular Technology Magazine,2020,15(4):122-134.

[2] VAN LE D,THAM C K. Quality of service aware computation offloading in an ad-hoc mobile cloud[J]. IEEE Transactions on Vehicular Technology,2018,67(9):8890-8904.

[3] HUANG W,HUANG Y M,HE S W,et al. Cloud and edge

multicast beamforming for cache-enabled ultra-dense networks[J]. IEEE Transactions on Vehicular Technology, 2020, 69(3): 3481-3485.
[4] ZHANG J, LETAIEF K B. Mobile edge intelligence and computing for the Internet of vehicles[J]. Proceedings of the IEEE, 2020, 108(2): 246-261.
[5] TRAN T X, HAJISAMI A, PANDEY P, et al. Collaborative mobile edge computing in 5G networks: new paradigms, scenarios, and challenges[J]. IEEE Communications Magazine, 2017, 55(4): 54-61.
[6] 李波, 侯鹏, 牛力, 等. 基于软件定义网络的云边协同架构研究综述[J]. 计算机工程与科学, 2021, 43(2): 242-257.
LI B, HOU P, NIU L, et al. Survey of cloud-edge collaborative architecture research based on software defined network[J]. Computer Engineering & Science, 2021, 43(2): 242-257. (in Chinese)
[7] ZHAN W H, LUO C B, WANG J, et al. Deep reinforcement learning-based computation offloading in vehicular edge computing[C]//Proceedings of IEEE Global Communications Conference. Washington D. C., USA: IEEE Press, 2019: 1-6.
[8] SUN J N, GU Q, ZHENG T, et al. Joint optimization of computation offloading and task scheduling in vehicular edge computing networks[J]. IEEE Access, 2020, 8: 10466-10477.
[9] DAI Y Y, XU D, MAHARJAN S, et al. Joint load balancing and offloading in vehicular edge computing and networks[J]. IEEE Internet of Things Journal, 2019, 6(3): 4377-4387.
[10] ZHAO J H, LI Q P, GONG Y, et al. Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks[J]. IEEE Transactions on Vehicular Technology, 2019, 68(8): 7944-7956.
[11] MENG X L, WANG W, WANG Y T, et al. Closed-form delay-optimal computation offloading in mobile edge computing systems[J]. IEEE Transactions on Wireless Communications, 2019, 18(10): 4653-4667.
[12] ZHOU Z Y, FENG J H, CHANG Z, et al. Energy-efficient edge computing service provisioning for vehicular networks: a consensus ADMM approach[J]. IEEE Transactions on Vehicular Technology, 2019, 68(5): 5087-5099.
[13] ZHAN W H, LUO C B, WANG J, et al. Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing[J]. IEEE Internet of Things Journal, 2020, 7(6): 5449-5465.
[14] GUO H Z, ZHANG J, LIU J J. WiFi-enhanced vehicular edge computing networks: collaborative task offloading[J]. IEEE Vehicular Technology Magazine, 2019, 14(1): 45-53.
[15] NI Z C, CHEN H L, LI Z, et al. MSCET: a multi-scenario offloading schedule for biomedical data processing and analysis in cloud-edge-terminal collaborative vehicular networks[EB/OL]. [2021-12-02]. https://ieeexplore.ieee.org/document/9628037.
[16] CHEN Q L, KUANG Z F, ZHAO L. Multiuser computation offloading and resource allocation for cloud-edge heterogeneous network[J]. IEEE Internet of Things Journal, 2022, 9(5): 3799-3811.
[17] 赵海涛, 张唐伟, 陈跃, 等. 基于DQN的车载边缘网络任务分发卸载算法[J]. 通信学报, 2020, 41(10): 172-178.
ZHAO H T, ZHANG T W, CHEN Y, et al. Task distribution offloading algorithm of vehicle edge network based on DQN[J]. Journal on Communications, 2020, 41(10): 172-178. (in Chinese)
[18] PENG H X, SHEN X M. Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks[J]. IEEE Transactions on Network Science and Engineering, 2020, 7(4): 2416-2428.
[19] KE H C, WANG J, DENG L Y, 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] ZHAO L, ZHAO W L, HAWBANI A, et al. Novel online sequential learning-based adaptive routing for edge software-defined vehicular networks[J]. IEEE Transactions on Wireless Communications, 2021, 20(5): 2991-3004.
[21] HE Y, LIANG C C, ZHANG Z, et al. Resource allocation in software-defined and information-centric vehicular networks with mobile edge computing[C]//Proceedings of the 86th Vehicular Technology Conference. Washington D. C., USA: IEEE Press, 2018: 1-5.
[22] HOU X W, REN Z Y, WANG J J, et al. Reliable computation offloading for edge-computing-enabled software-defined IoV[J]. IEEE Internet of Things Journal, 2020, 7(8): 7097-7111.
[23] YE Q, ZHUANG W H, ZHANG S, et al. Dynamic radio resource slicing for a two-tier heterogeneous wireless network[J]. IEEE Transactions on Vehicular Technology, 2018, 67(10): 9896-9910.
[24] VAN HASSELT H, GUEZ A, SILVER D. Deep reinforcement learning with double Q-learning[J]. Intelligent Control and Automation, 2016, 30(1): 129-144.

编辑 赖玉玲