一种基于深度强化学习的任务卸载方法*

高宇豆[1,2],黄祖源[1],王海燕[1],保 富[1],张 航[1],李 辉[1]
(1.云南电网有限责任公司 信息中心,云南 昆明 650214;2.西南林业大学 大数据与智能工程学院,云南 昆明 650224)

摘 要:随着车联网的快速发展,车载应用大多是计算密集和延迟敏感的。车辆是资源受限的设备,无法为这些应用提供所需的计算和存储资源。边缘计算通过将计算和存储资源提供给网络边缘的车辆,有望成为满足低延迟需求的有效解决方案。这种将任务卸载到边缘服务器的计算模式不仅可以克服车辆资源的不足,还可以避免将任务卸载到云可能导致的高延迟。提出了一种基于深度强化学习的任务卸载方法,以最小化任务的平均完成时间。首先,把多任务卸载决策问题规约为优化问题。其次,使用深度强化学习对优化问题进行求解,以获得具有最小完成时间的优化卸载策略。最后,实验结果表明,该方法的性能优于其他基准方法。

关键词:任务卸载;车联网;边缘计算;深度学习;强化学习
中图分类号:TP311 **文献标识码**:A DOI:10.16157/j.issn.0258-7998.212133

中文引用格式:高宇豆,黄祖源,王海燕,等. 一种基于深度强化学习的任务卸载方法[J].电子技术应用,2022,48(8):29-33.
英文引用格式:Gao Yudou,Huang Zuyuan,Wang Haiyan,et al. Task offloading based on deep reinforcement learning for Internet of Vehicles[J]. Application of Electronic Technique,2022,48(8):29-33.

Task offloading based on deep reinforcement learning for Internet of Vehicles

Gao Yudou[1,2],Huang Zuyuan[1],Wang Haiyan[1],Bao Fu[1],Zhang Hang[1],Li Hui[1]
(1.Center of Information,Yunnan Power Grid Co.,Ltd.,Kunming 650214,China;
2.School of Big Data and Intelligent Engineering,Southwest Forestry University,Kunming 650224,China)

Abstract:With the rapid development of Internet of Vehicular, more and more vehicles' applications are computation-intensive and delay-sensitive. Resource-constrained vehicles cannot provide the required amount of computation and storage resources for these applications. Edge computing(EC) is expected to be a promising solution to meet the demand of low latency by providing computation and storage resources to vehicles at the network edge. This computing paradigm of offloading tasks to the edge servers can not only overcome the restrictions of limited capacity on vehicles,but also avoid the high latency caused by offloading tasks to the remote cloud. In this paper, an efficient task offloading algorithm based on deep reinforcement learning is proposed to minimize the average completion time of applications. Firstly, the multi-task offloading strategy problem is formalized as an optimization problem. Secondly, a deep reinforcement learning is leveraged to obtain an optimized offloading strategies with the lowest completion time. Finally, the experimental results show that the performance of the proposed algorithm is better than other baselines.
Key words:task offloading;Internet of Vehicles;edge computing;deep learning;reinforcement learning

0 引言

车联网(Internet of Vehicle,IoV)是车载网(Vehicular Ad hoc Network,VANET)和物联网(Internet of Things,IoT)的深度融合,旨在提高车辆网络的性能,降低交通拥堵的风险[1]。在车联网中,许多车辆应用不仅需要大量的计算资源,还对响应时间有严格的要求[2]。但是,车辆是计算资源和通信能力有限的装置。对于这些计算密集、延迟敏感的应用,车辆无法提供足够的计算和存储资源[3]。

为应对车载应用所需的大量计算资源,云计算被视为一种可行的解决方案。在云计算环境下,车辆可以通过无线网络将计算密集型应用卸载到云上运行。这种端-云协作的计算模式很好地扩展了车辆的计算能力[4]。

然而,对于计算密集、延迟敏感的应用,端-云协作的计算模式是不够的。因为,远程任务卸载带来的高传输延迟会降低用户体验[3]。为解决此问题,将车联网和边缘计算相结合的车辆边缘计算,被认为是满足低延迟的更好解决方案[5]。

但是,由于边缘服务器具有的计算资源和存储资源是有限的,过多的卸载任务会导致边缘服务器过载[6]。在边缘服务器过载情况下,任务的等待时间会显著延长,

*基金项目:国家自然科学基金项目(61702442)

从而增加任务的完成时间。为此，本文提出了一种车联网中基于深度强化学习的任务卸载方法，主要贡献包括以下几点：

(1)把以最小化任务完成时间为目标的多任务卸载问题规约为优化问题；

(2)提出了一种基于深度强化学习的任务卸载算法，使用深度强化学习来求解上述优化问题；

(3)通过实验，验证了所提算法的有效性。

1 相关工作

相关学者使用动态规划方法对任务卸载进行了研究。文献[7]使用二次约束二次规划方法，提出了一种两层计算卸载框架，用于将移动用户和小基站的计算密集型任务分别卸载到移动边缘服务器和宏基站；文献[8]基于混合整数线性规划方法，提出一种对边缘云上的计算、缓存和通信进行联合优化的方法；在此基础上，文献[9]提出了一种基于混合整数非线性规划的任务卸载方法，实现了在节省用户设备电池寿命的同时最大限度地减少延迟。这些工作主要将计算卸载问题规约为凸优化问题，使用动态规划方法对问题进行求解。但是，车辆边缘计算环境下的任务卸载未必总是凸优化的。

相关学者使用博弈论对任务卸载进行了研究。文献[10]将移动设备用户间的分布式计算卸载决策问题规约为一个多用户计算卸载博弈，并设计了一个可以达到纳什均衡的分布式计算卸载算法；文献[11]使用博弈论，提出了一个移动感知的分层边缘计算框架，实现同时降低智能设备的能源成本和任务执行时间；文献[12]研究了基于端-边-云车联网下的计算卸载机制，提出了各种基于博弈论的任务卸载优化策略，可用于边缘服务器的选择和任务传输管理。这些工作仅考虑了卸载策略对系统一个快照的最优方案或接近最优方案，未考虑车辆边缘计算环境下当前策略对资源分配的长期影响。

与上述工作相比，本文工作的主要区别在于：使用深度强化学习方法来求解车辆边缘计算环境下的任务卸载问题。由于深度学习方法能提高对车辆网络的认知能力，因此该方法能更好地学习到任务卸载的复杂特征。

2 本文方法

2.1 车辆网边缘计算架构

本文将考虑多任务的车辆网边缘计算架构。此架构由 1 个基站(Base Station，BS)、k 辆车、m 个路侧单元(Road Side Unit，RSU)和 m 个边缘服务器组成，其中，车辆集表示为 $V=\{v_1,v_2,\cdots,v_K\}$，路侧单元集表示为 $R=\{r_1,r_2,\cdots,r_M\}$，边缘服务器集表示为 $S=\{s_1,s_2,\cdots,s_M\}$。在此架构下，借鉴文献中基站的思想[13]，本文将基站作为控制实体，用于获取边缘计算可用资源、信道可用资源、任务信息等；道路旁的每个路侧单元配备了一个边缘服务器；每个边缘服务器可为其覆盖范围内的车辆提供计算和存储资源；车辆是任务的产生者，通过无线网络与路侧单元进行通信。

在给定的时间间隔 t，每辆车可以产生多个任务。每个任务表示为二元组 $T_{i,j}=<d_{i,j},c_{i,j}>$，其中，$d_{i,j}$ 表示执行第 i 辆车产生的第 j 个任务所需的数据量的大小，$c_{i,j}$ 表示执行第 i 辆车产生的第 j 个任务所需的计算量的大小，$i\in[1,2,\cdots,K]$，$j\in[1,2,\cdots,J]$。这些任务通路侧单元可以被卸载到边缘服务器上执行。

2.2 本地计算模型

当车辆 v_i 产生的任务 $T_{i,j}$ 在本地执行时，则该任务的完成时间 $\mathrm{LT}_{i,j}^{\mathrm{c}}$ 由两部分组成：执行时间 $\mathrm{LT}_{i,j}^{\mathrm{exec}}$ 和等待时间 $\mathrm{LT}_{i,j}^{\mathrm{wait}}$，定义如下：

$$\mathrm{LT}_{i,j}^{\mathrm{c}}=\mathrm{LT}_{i,j}^{\mathrm{exec}}+\mathrm{LT}_{i,j}^{\mathrm{wait}} \tag{1}$$

任务的执行时间 $\mathrm{LT}_{i,j}^{\mathrm{exec}}$ 由车辆的计算能力和任务所需的计算量决定。本文将车辆的计算能力抽象为计算资源单元的集合。设每个计算单元的计算能力为 α^{l}(单位为 GHz)，本地车辆分配给任务的单元数为 β^{l}，则执行时间定义为：

$$\mathrm{LT}_{i,j}^{\mathrm{exec}}=\frac{T_{i,j}\cdot c_{i,j}}{\alpha^{\mathrm{l}}\cdot\beta^{\mathrm{l}}} \tag{2}$$

任务的等待时间 $\mathrm{LT}_{i,j}^{\mathrm{wait}}$ 由车辆拥有的 CPU 个数和在本地执行的任务数量决定。本文用 $\mathrm{AT}_{i,j}^{\mathrm{l}}$ 表示任务 $T_{i,j}$ 在本地开始执行的实际时间，$\mathrm{GT}_{i,j}$ 表示车辆产生任务 $T_{i,j}$ 的时间，则等待时间定义为：

$$\mathrm{LT}_{i,j}^{\mathrm{wait}}=\mathrm{AT}_{i,j}^{\mathrm{l}}-\mathrm{GT}_{i,j} \tag{3}$$

2.3 卸载计算模型

当车辆 v_i 产生的任务 $T_{i,j}$ 被卸载到边缘服务器执行时，则该任务的完成时间 $\mathrm{ET}_{i,j}^{\mathrm{c}}$ 由四部分组成：任务所需数据的传输时间 $\mathrm{DT}_{i,j}^{\mathrm{tra}}$、任务在边缘服务器的执行时间 $\mathrm{ET}_{i,j}^{\mathrm{exec}}$、任务在边缘服务器的等待时间 $\mathrm{ET}_{i,j}^{\mathrm{wait}}$ 和执行结果返回时间 $\mathrm{RT}_{i,j}^{\mathrm{tra}}$，定义如下：

$$\mathrm{ET}_{i,j}^{\mathrm{c}}=\mathrm{DT}_{i,j}^{\mathrm{tra}}+\mathrm{ET}_{i,j}^{\mathrm{exec}}+\mathrm{ET}_{i,j}^{\mathrm{wait}}+\mathrm{RT}_{i,j}^{\mathrm{tra}} \tag{4}$$

任务所需数据的传输时间 $\mathrm{DT}_{i,j}^{\mathrm{tra}}$ 是指任务 $T_{i,j}$ 所需的数据从车辆 v_i 传输到路侧单元 r_j 的时间，由数据传输率 $s_{i,1}$ 和数据量的大小 $T_{i,j}\cdot d_{i,j}$ 决定。其定义如下：

$$\mathrm{DT}_{i,j}^{\mathrm{tra}}=\frac{T_{i,j}\cdot d_{i,j}}{s_{i,1}} \tag{5}$$

进一步，本文考虑车辆与路侧单元间的无线传输是基于正交频分多址的，则数据传输率 $s_{i,1}$ 又由分配给任务的上行线路的带宽、车辆的传输功率 ρ_i、车辆与路侧单元间的信道增益决定 $\lambda_{i,1}$。于是，数据传输率的定义如下：

$$s_{i,1}=\frac{B}{N}\log_2(1+\rho_i\cdot\lambda_{i,1}\cdot\sigma^{-2}) \tag{6}$$

其中，B 表示上线线路的带宽，N 表示将带宽分为 N 份，σ 表示高斯噪声。

任务在边缘服务器的执行时间 $ET_{i,j}^{exec}$ 由边缘服务器的计算能力和任务所需的计算量决定。与本地执行时间的定义类似，其定义如下：

$$ET_{i,j}^{exec}=\frac{T_{i,j}\cdot c_{i,j}}{\alpha^{e}\cdot\beta^{e}} \tag{7}$$

其中，α^{e} 表示边缘服务器每个计算单元的计算能力(单位为 GHz)，β^{e} 表示边缘服务器分配车辆给任务的单元数。

任务在边缘服务器的等待时间 $ET_{i,j}^{wait}$ 由边缘服务器拥有的 CPU 个数和卸载到边缘器执行的任务数量决定。与本地等待时间的定义类似，其定义如下：

$$ET_{i,j}^{wait}=AT_{i,j}^{e}-GT_{i,j} \tag{8}$$

其中，$AT_{i,j}^{e}$ 表示任务 $T_{i,j}$ 在边缘服务器开始执行的实际时间。

文本假设 $RT_{i,j}^{tra}$ 为 0。

2.4 问题定义

在多任务的车辆边缘计算架构下，任务卸载的目标是：通过优化任务卸载决策，最小化任务的完成时间。

首先，需要定义任务卸载决策。由于每个任务要么在本地执行，要么被卸载边缘服务器执行，故而，任务卸载决策可以定义为：

$$A=\{\alpha_{i,j}|i\in K,j\in J\} \tag{9}$$

其中，$\alpha_{i,j}$ 为卸载决策变量，其定义如下：

$$\alpha_{i,j}=\begin{cases}1 & \text{任务 } T_{i,j} \text{ 在边缘服务器上执行}\\ 0 & \text{任务 } T_{i,j} \text{ 在本地执行}\end{cases} \tag{10}$$

其次，定义每个任务的完成时间。基于卸载决策变量，任务的完成时间定义如下：

$$T_{i,j}^{c}=(1-\alpha_{i,j})LT_{i,j}^{c}+\alpha_{i,j}\cdot ET_{i,j}^{c} \tag{11}$$

最后，基于上述分析，以最小化任务完成时间为目标的多任务卸载问题可以规约为一个如下所示的单目标优化问题：

$$\min=\frac{1}{K}\left[\sum_{i=1}^{K}\sum_{j=1}^{J}((1-\alpha_{i,j})LT_{i,j}^{c}+\alpha_{i,j,1}\cdot ET_{k,i}^{c})\right] \tag{12}$$

s.t. 式(9)~式(11)

3 基于深度强化学习的任务卸载

本节将提出一种基于深度强化学习的任务卸载算法，使用 DQN 来求解优化问题。该算法主要包括四部分内容：(1)设置 DQN 的配置；(2)基于 DQN 的任务卸载算法；(3)基于 ε-贪婪策略的动作选择；(4)使用经验回放来更新 DQN。

3.1 DQN 的配置

DQN 是一种基于值的深度强化学习方法，将强化学习和深度学习进行了很好的结合。其核心是：使用神经网络来近似传统 Q-Learning 算法中的 Q 函数[14]。本文选择 DQN 的原因是：在面对高维空间的复杂问题时，DQN 能够很好地学习到优化策略。

在多任务的车辆边缘计算架构下，为了利用 DQN，需要设置 DQN 的配置，包括 agent、环境、状态和奖励。

首先，在基站上部署 agent。agent 通过一系列的观察、动作和奖励对环境做出反应。具体地，当收到来自车辆的任务请求时，agent 根据当前观察(状态)选择动作，并确定卸载决策变量。然后，agent 将卸载决策变量回传给车辆，以指示车辆是否卸载该任务。任务执行结束后，再将任务的本地完成时间与边缘服务器上完成时间的差值作为奖励反馈给 agent。

其次，环境由车辆网络组成，主要包括边缘服务器的计算资源、路侧单元的带宽资源和任务信息。其定义如下：

$$env=\{\sum_{i=1}^{M}s_i\cdot c,\ \sum_{j=1}^{M}r_j\cdot b,\ \sum_{i=1}^{K}\sum_{j=1}^{J}T_{ij}\} \tag{13}$$

其中，$\sum_{i=1}^{M}s_i\cdot c$ 表示边缘服务器的计算资源，$\sum_{j=1}^{M}r_j\cdot b$ 表示路侧单元的带宽资源，$\sum_{i=1}^{K}\sum_{j=1}^{J}T_{ij}$ 表示车辆产生的所有任务信息。

然后，在时间步 τ，观察(状态)由边缘服务器的可用计算资源和路侧单元的可用带宽资源组成。其定义如下：

$$s_\tau=\{\sum_{i=1}^{M}s_i\cdot c_a,\ \sum_{j=1}^{M}r_j\cdot b_a\} \tag{14}$$

其中，$\sum_{i=1}^{M}s_i\cdot c_a$ 表示边缘服务器的可用计算资源，$\sum_{j=1}^{M}r_j\cdot b_a$ 表示路侧单元的可用带宽资源。

最后，在时间步 τ，对于任务 $T_{i,j}$，动作 a 的奖励由任务的本地完成时间与边缘服务器上完成时间的差值组成。其定义如下：

$$r_\tau=LT_{i,j}^{c}(\tau)-ET_{i,j}^{c}(\tau) \tag{15}$$

3.2 卸载算法框架

当动作和状态对空间变得高维连续时，传统的 Q-Learning 算法遍历高维 Q-table 是非常耗时的。为了解决这一问题，DQN 使用深度神经网络来近似 $Q(s,a)$。基本思想是：首先，agent 根据对当前环境观察得到状态 s；其次，将状态 s 作为深度神经网络的输入，得到神经网络输出的多个 $Q(s,a)$；然后，使用 ε-贪婪策略从这多个 $Q(s,a)$ 中选择一个动作 a，与此同时，环境将响应选定的动作，给出奖励 r，并演化到新状态 s'；最后，将 (s,a,r,s') 存入经验池 D，并使用 D 更新 DQN。

基于上述分析，算法 1 给出了基于 DQN 的任务卸载算法框架，记为 DQN。

算法 1 基于 DQN 的任务卸载(DQN)：

输入：预测网络 Q^{pre}，预测网络的参数 θ^{pre}，目标网络 Q^{tar}，目标网络的参数 θ^{tar}，经验池 D，当前状态 s_τ；

输出：预测网络和目标网络。

(1)初始化大小为 N 的经验池 D；

(2)使用随机生成的参数值 θ^{pre} 初始化预测网络 Q^{pre}；

(3)θ^{pre} 赋值给 θ^{tar},用 θ^{tar} 初始化目标网络 Q^{tar};

(4)for episode=1 to H do;

(5) for t=1 to M do;

(6) a_τ=基于 ε-贪婪策略选择动作(s_τ, Q^{pre});

(7) 计算动作 a_τ 的奖励及下一个状态 $s_{\tau+1}$;

(8) 将(s, a, r, s')存入经验池 D;

(9) 使用算法 2 和经验池 D 更新 DQN$(Q^{pre}, \theta^{pre}, Q^{tar}, \theta^{tar})$。

在动作选择上,DQN 引入了 ε-贪婪策略。具体地,agent 采用 $1-\varepsilon$ 概率来选择最大 Q 值的动作,采用概率 ε 来随机选择动作。因此,基于 ε-贪婪策略的动作选择可定义为:

$$a=\begin{cases}\underset{a_\tau}{\text{argmax}}\ Q(s_\tau, a_\tau), & \text{prob}=1-\varepsilon \\ \text{随机动作 } a_\tau, & \text{prob}=\varepsilon\end{cases} \tag{16}$$

3.3 DQN 网络更新

为了进一步打破数据间的关联性、防止过拟合,在训练过程中,DQN 使用了预测网络 Q^{pre} 和目标网络 θ^{tar}。具体地,针对当前的动作对(s, a),预测网络 Q^{pre} 使用最新参数 θ^{pre} 预测当前 Q 值,而目标网络使用很久前的参数 θ^{tar} 得到目标 Q 值。与预测网络相比,目标网络的结构完全相同,但参数不同。

DQN 的代价函数可以定义如下:

$$L_i(\theta_i)=(y_i-Q^{pre}(s_t, a_t; \theta_i^{pre}))^2 \tag{17}$$

$$y_i=\begin{cases}r_i & s_{i+1}\text{ 终止} \\ r_i+\gamma \underset{a_{i+1}}{\max}\ Q^{tar}(s_{i+1}, a_{i+1}; \theta^{tar}) & s_{i+1}\text{ 不终止}\end{cases} \tag{18}$$

其中,r_i 是当前获得的奖励,γ 是折现系数。

基于损失函数,算法 2 描述了 DQN 的更新过程。

算法 2 DQN 网络的更新:

输入:预测网络 Q^{pre},预测网络的参数 θ^{pre},目标网络 Q^{tar},目标网络的参数 θ^{tar};

输出:θ^{pre},θ^{tar}。

(1)从经验池中随机采样得到 D_τ;

(2)使用式(18)计算目标 Q 值 y_i;

(3)使用式(17)计算代价函数 $L_i(\theta_i)$;

(4)对 $L_i(\theta_i)$使用梯度下降,以更新预测网络 Q^{pre} 中的参数 θ^{pre};

(5)每 C 步后,使用参数 θ^{pre} 更新目标网络 Q^{tar} 中的参数 θ^{tar};

(6)输出 θ^{pre} 和 θ^{tar}。

4 实验

4.1 实验设置

实验考虑包含 10 个 RSU、10 辆车、30 个任务的场景。进一步,实验模拟环境所需的参数设置如表 1 所示。

实验中,本文选择下述 3 个基准算法,与本文所提 DQN 方法进行对比。

(1)本地任务卸载(LTO)。车辆产生的所有任务全部在本地执行。

(2)随机任务卸载(RTO)。车辆产生的任务随机卸载到边缘服务器执行。

(3)边缘任务卸载(ETO)。车辆产生的所有任务全部卸载到边缘服务器执行。

表 1 参数设置

参数	数值
参数功率 ρ/mW	1.8~2.2[15]
信道增益 λ/dB	129.6~158.4[16]
噪声功率 σ^2/mW	1.5×10^{-8}[15]
折现系数 γ	0.9
车辆计算能力/(Gigacycle/s)	0.3
经验池 D 的大小	500
D_τ 的大小	32
路侧单元的带宽/MHz	10~20
边缘服务器的计算能力/(Gigacycle/s)	3~6

4.2 平均完成时间的比较

图 1 直观显示了 30 个任务平均完成时间的对比。从图 1 可以看出:本文所提方法 DQN 的性能优于其他 3 种基准算法。相比于表现最差的 LTO 算法,DQN 算法的性能提升了 15.5%;相比于表现较好的 ETO 算法,TODQN 算法的性能提升了 4.1%。这是因为 DQN 算法同时充分考虑了车辆边缘计算环境下的计算资源和带宽资源,而基准算法没有考虑这些资源对任务完成时间的影响。

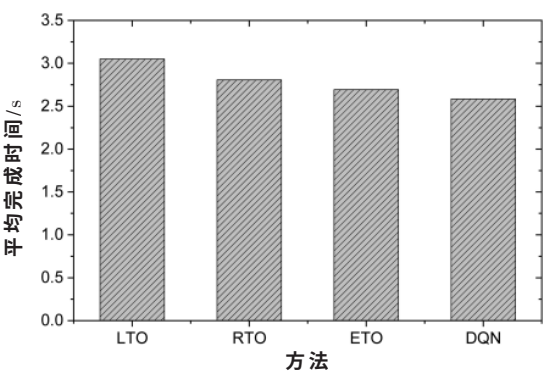

图 1 平均完成时间的对比

4.3 参数对算法性能的影响

本实验研究了各种参数对算法性能的影响,例如任务的数量、任务所需的数据量。

(1)任务的数量:本实验中,其他参数保持不变,任务数量从 30 增加到 70。从图 2 可以看出,随着任务数量的增多,任务总完成时间开始增加。相比于其他 3 种基准算法,DQN 算法的增加幅度最小。这是因为 DQN 可以合理地利用车辆的计算资源和边缘服务器的计算资源。

(2)任务计算量:本实验中,其他参数保持不变,任务所需的计算量从0.5 Gigacycle 增加到1.3 Gigacycle。从图 3 可以看出,随着任务计算量的增加,所有算法的总完成时间都在增加,尤其是 ETO 算法的性能逐渐低于 RTO

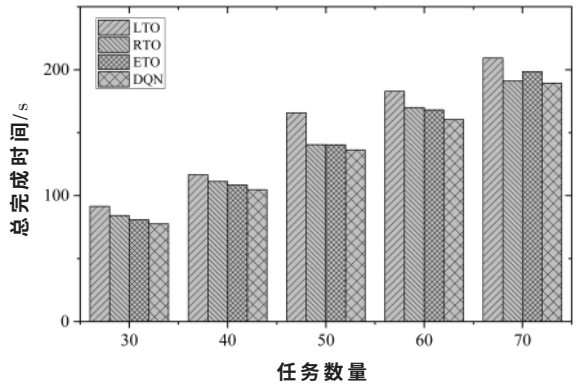

图 2 任务数量变化对总完成时间的影响

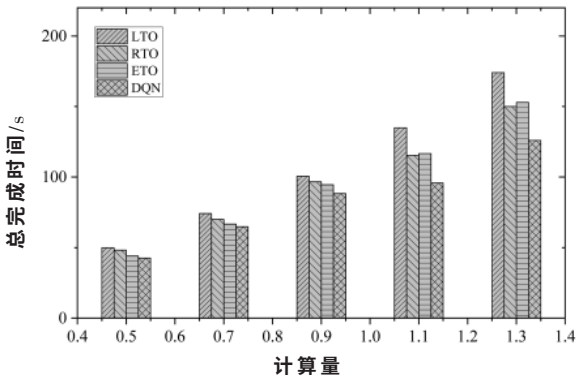

图 3 任务计算量变化对总完成时间的影响

算法。这是因为边缘服务器过载造成的。与 ETO 和RTO 相比，DQN 能很好地解决边缘服务器过载的问题。

5 结论

本文提出了一种车辆边缘计算环境下基于深度强化学习的任务卸载方法。该方法使用 DQN 对任务卸载问题进行求解，可以得到具有最小完成时间的优化卸载策略。通过实验结果表明，该方法具有良好的性能。在下一步的研究工作中，将同时考虑任务完成时间和缓冲内容的任务卸载。

参考文献

[1] ZHU L，YU F R，WANG Y，et al.Big data analytics in intelligent transportation systems：a survey[J].IEEE Transactions on Intelligent Transportation Systems，2019，20(1)：383-398.

[2] LIU L，CHEN C，PEI Q，et al.Vehicular edge computing and networking：a survey[J].Mobile Networks and Applications，2021，26：1145-1168.

[3] LIU Y，WANG S G，ZHAO Q，et al.Dependency-aware task scheduling in vehicular edge computing[J].IEEE Internet of Things Journal，2020，7(6)：4961-4971.

[4] KHAN A U R，OTHMAN M，MADANI S A，et al.A survey of mobile cloud computing application models[J].IEEE Communications Surveys & Tutorials，2014，16(1)：393-413.

[5] SHI W S，ZHANG Y Z，WANG Y F，et al.Edge computing：state-of-the-art and future directions[J].Journal of Computer Research and Development，2019，56(1)：69-89.

[6] XU X L，SHEN B W，DING S，et al.Service offloading with deep Q-network for digital twinning empowered Internet of Vehicles in edge computing[J].IEEE Transactions on Industrial Informatics，2022，18(2)：1414-1423.

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

[8] CHEN M，HAO Y，HU L，et al.Edge-CoCaCo：toward joint optimization of computation，caching，and communication on edge cloud[J].IEEE Wireless Communications，2018，25(3)：21-27.

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

[10] CHEN X，JIAO L，LI W Z，et al.Efficient multi-user computation offloading for mobile-edge cloud computing[J].IEEE/ACM Transactions on Networking，2016，24(5)：2795-2808.

[11] ZHANG K，LENG S，HE Y，et al.Mobile edge computing and networking for green and low-latency Internet of Things[J].IEEE Communications Magazine，2018，56(5)：39-45.

[12] ZHANG K，MAO Y M，LENG S P，et al.Mobile-edge computing for vehicular networks：a promising network paradigm with predictive off-loading[J].IEEE Vehicular Technology Magazine，2017，12(2)：36-44.

[13] LI Q，WANG S G，ZHOU A，et al.QoS driven task offloading with statistical guarantee in mobile edge computing[J].IEEE Transactions on Mobile Computing，2022，21(1)：278-290.

[14] MNIH V，KAVUKCUOGLU K，SILVER D，et al.Human-level control through deep reinforcement learning[J].Nature，2015，518(7540)：529-533.

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

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

（收稿日期：2021-09-05）

作者简介：

高宇豆(1995-)，女，硕士，工程师，主要研究方向：电网信息化和电力大数据分析。

黄祖源(1989-)，男，本科，工程师，主要研究方向：电力信息化。

王海燕(1987-)，女，本科，工程师，主要研究方向：电网信息化。

扫码下载电子文档

《电子技术应用》http://www.chinaaet.com

版权声明