基于隐式分位数网络的车联网任务卸载策略

王聪[1]，刘帅[2]，左明敏[2]

（1.东北大学秦皇岛分校计算机与通信工程学院，河北 秦皇岛 066004；
2.东北大学计算机与通信工程学院，辽宁 沈阳 110819）

摘　要：随着物联网（IoT）和无线技术的迅猛发展，车辆面临着前所未有的计算资源需求挑战。为了应对这些挑战，研究车辆边缘计算（VEC）场景中的任务卸载问题，提出一种基于隐式分位数网络（IQN）的动态任务卸载策略。首先，对 VEC 系统进行建模，将任务卸载决策问题构建为一个马尔可夫决策过程（MDP）；然后，引入一种融合时间优先和噪声增强策略的双分位数强化学习算法，以实现更加精准的任务卸载。该算法利用 IQN 对值函数的完整概率分布进行估计，进而实现对回报分布的连续参数化估计，有效提升预测和决策的准确性。同时，算法整合了时序优先经验回放机制和噪声网络，前者优先重放对学习更有价值的经验，后者通过引入随机性增强了探索效率。实验结果表明，与传统算法相比，该算法能够显著降低整体任务的完成时延，同时提升任务卸载决策的准确性和系统资源的利用率。研究表明，通过引入 IQN 和双分位数强化学习算法，可以在动态且复杂的车联网（IoV）环境中实现高效的任务卸载。

关键词：车联网；边缘计算；任务卸载；深度强化学习；隐式分位数网络

中图分类号：TP391　　**文献标志码：**A　　**DOI：**10.19678/j.issn.1000-3428.0069929

Task Offloading Strategy for Internet of Vehicles Based on Implicit Quantile Network

WANG Cong[1], LIU Shuai[2], ZUO Mingmin[2]

(1.School of Computer and Communication Engineering, Northeastern University at Qinhuangdao, Qinhuangdao 066004, Hebei, China;
2.School of Computer and Communication Engineering, Northeastern University, Shenyang 110819, Liaoning, China)

【Abstract】 With the rapid development of the Internet of Things (IoT) and wireless technology, vehicles are facing unprecedented challenges in terms of computing resource demands. To address these challenges, task offloading in Vehicle Edge Computing (VEC) scenarios is studied, and a dynamic task offloading strategy based on an Implicit Quantile Network (IQN) is proposed. First, the VEC system is modeled, and the task offloading decision problem is constructed as a Markov Decision Process (MDP); Subsequently, a binary reinforcement learning algorithm that integrates time-first and noise enhancement strategies is introduced to achieve more accurate task offloading. This algorithm utilizes an IQN to estimate the complete probability distribution of the value function, thereby achieving a continuous parameterized estimation of the return distribution and effectively improving the prediction and decision-making accuracy. Simultaneously, the algorithm integrates a temporal-priority experience replay mechanism and a noise network. The former prioritizes replaying experiences that are more valuable for learning, whereas the latter enhances exploration efficiency by introducing randomness. The experimental results show that, compared with traditional algorithms, this algorithm can significantly reduce the overall task completion delay while improving the accuracy of task offloading decisions and the utilization of system resources. Research has shown that by introducing IQN and binary reinforcement learning algorithms, efficient task offloading can be achieved in dynamic and complex Internet of Vehicles (IoV) environments.

【Key words】 Internet of Vehicles (IoV); edge computing; task offloading; Deep Reinforcement Learning (DRL); Implicit Quantile Network (IQN)

0　引言

在车联网（IoV）环境中，传统云计算和本地计算的模式虽然为车辆间的数据交换和处理提供了基础[1]，然而，随着 IoV 技术的飞速发展和智能交通需求的不断增长，这些计算模式开始显露出一系列的局限性和挑战[2]。首先，传统的云计算模式依赖于中心化的数据中心进行数据处理和存储，虽然这

收稿日期：2024-05-28　　**修回日期：**2024-07-24

基金项目：河北省自然科学基金（F2022501025）；河北省重大科技支撑计划（242Q1602Z）。

通信作者 E-mail：congw@neuq.edu.cn

种方式在处理大量数据时相对高效，但在 IoV 的应用场景中，由于数据传输涉及远距离的通信，容易导致延迟问题[3]；其次，本地计算虽然能够在车辆端进行快速处理，降低对中心化云服务的依赖，减少数据传输延迟，但它受限于车辆的计算资源。此外，完全依赖本地计算的模式也无法实现车辆与周围环境的有效数据共享，这对于构建协同和智能化的 IoV 生态系统来说是一个重大缺陷[4]。

在基于移动边缘计算(MEC)的 IoV 系统中，车辆通过车到基础设施(V2I)无线通信，将任务卸载到附近的 MEC 服务器执行，降低网络延迟并提升数据处理效率。近年来，边缘计算的理念被成功引入 IoV，并发展为车辆边缘计算(VEC)[5]，具备云-边-车 3 层架构。VEC 主张将数据处理任务从云计算中心迁移到网络边缘，即靠近数据产生点的位置，如车辆附近的路边单元(RSU)等[6]。

在 VEC 系统中，进行有效的卸载决策是优化资源和提升系统性能的关键[7]。卸载决策需要考虑多种因素，包括任务处理要求、网络条件、计算资源的可用性及任务特点等，目标是使 VEC 系统智能、灵活地进行卸载决策，提高 IoV 系统的性能和效率[8]。

本文提出了一种时序优先与噪声增强的双分位数强化学习卸载算法(TPNE-DQRLD)。TPNE-DQRLD 利用了 2 个隐式分位数网络(IQN)，用期望的分布来预测未来奖励，从不同的分位数角度估计值函数，从而更准确地匹配不同任务的需求。同时，整合时序优先经验回放机制和噪声网络，其中，前者优先重放对学习更有价值的经验，后者通过引入随机性来增强探索效率，两者共同促进了更高质量的决策制定和性能优化。

1 相关工作

在 VEC 的任务卸载中，引入多层网络架构可以增强系统的灵活性和适应性，从而更好地面对 IoV 的动态性。VEC 架构如图 1 所示(彩色效果见《计算机工程》官网 HTML 版，下同)。

文献[9]详细讨论了一个具有 3 层架构的任务卸载模型，包括车辆用户设备层、边缘计算层和云层。这种模型能够更好地平衡计算资源，并根据任务需求和网络环境动态地调整任务的卸载决策。文献[10]在此基础上，提出了一种基于负载均衡的任务卸载方案。该方案通过分析车辆网络中的数据流量和任务需求以及 MEC 服务器的计算能力，动态调整任务卸载的决策，以最大化系统的性能和效率。

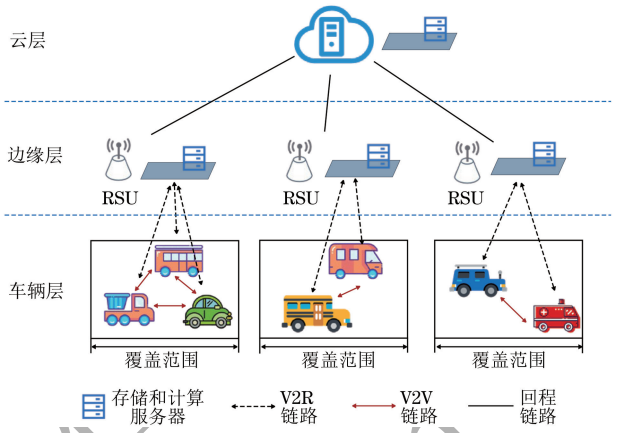

图 1 VEC 架构

Fig. 1 VEC architecture

这种方法虽然有效提高了任务卸载的效率，但是在面对网络动态性和处理延迟问题时仍存在一定的不足。

文献[11]引入了一种基于 Lyapunov 在线强化学习的任务分配方法，其研究了在雾辅助移动物联网中的智能分配任务，以最小化平均任务完成时间，并设计了一种在线强化学习算法，利用 Lyapunov 优化技术分析其计算复杂性和性能界限。

深度强化学习(DRL)算法可以更有效地降低任务卸载时的延迟、能耗和一些其他代价，当然，DRL 算法也存在一些困难和挑战。文献[12]提出了一种双重深度 Q 网络(DDQN)的动态分帧卸载算法，在 VEC 系统中处理任务卸载的问题。这项研究将任务划分为连续的子任务，执行 N 个连续子任务的分段，并考虑等待时间和传输延迟。尽管该方法提高了任务的执行效率，但在处理大规模计算任务时仍然需要改进。

文献[13]提出了一种协同博弈的 DRL 方法，同时在动态边缘计算市场中处理计算卸载问题。这项研究利用近端策略优化(PPO)来解决卸载问题，并通过设置 MEC 计算服务价格来协调多用户之间的协同博弈，确定卸载数据的大小。然而，该方法主要关注特定环境中的计算卸载，而在车载环境中，需要进一步优化。

文献[14]提出了一种 DRL 的调度方法，通过最小化任务延迟和能量消耗来最小化长期成本。该文采用 PPO 算法实现训练，利用卷积神经网络(CNN)提取任务队列的代表性特征。文章提出的 DRLOSM(DRL-based Offloading Scheduling Method)可以有效地估算未知的系统动态。然而，该方法主要关注整体的任务卸载，没有考虑单独任务的卸载决策。

文献[15]提出了一种多车智能协同计算策略，以最小化智能车辆的总系统延迟。然而，这种策略

在处理车载网络中的动态变化时可能面临挑战，如车辆的移动性和网络的不稳定性，可能会对协同计算策略的效果产生影响。此外，该方法训练过程的收敛性能仍有待提升。

文献[16]提出了一种结合 DRL 和博弈论的方法。通过设计基于纳什均衡和强化学习的计算卸载算法，旨在最大限度地减少系统的总体开销。

文献[17]专注于减少应用程序的完成时间和用户设备在卸载过程中的能量消耗，将这一计算卸载问题形式化为一个能量和时间优化的问题，并提出了基于深度 Q 网络(DQN)的优化策略。

文献[18]设计了一个基于 5G 的车辆感知多接入边缘计算网络(VAMECN)，并提出了一种基于 DRL 的联合计算卸载和任务迁移优化(JCOTM)算法，旨在最小化总系统成本。

文献[15]开发了一种多车辆智能协同计算策略(MV-ICCS)，利用基于 DDQN 的算法来最小化总系统延迟。该策略综合考虑了本地执行和多车协作，以确定最佳任务分配比例。

针对移动设备自身处理性能和电源容量的限制，文献[19]设计了一种基于 A3C 的依赖任务卸载与资源分配(DTORA)算法。该算法通过定义状态空间、动作空间和奖励函数，将依赖任务卸载问题转化为马尔可夫决策过程下的最优策略问题，并采用异步并发求解得到了高效的任务卸载和资源分配策略。

文献[20]研究了基于 MEC 的车辆众包服务场景，提出了一个数据驱动的任务卸载问题，并采用异步 DQN 来快速确定卸载决策，同时针对剩余资源分配问题提出了基于凸理论的最优解析公式。

文献[21]针对自适应服务迁移问题，提出了一种 DQN 服务迁移决策算法，并开发了一个由神经网络组成的服务迁移框架，提高了算法的适应性和实时性。

文献[22]提出了一种基于图表示的联邦学习(FedSTN)算法，用于在边缘计算环境中进行交通流量预测。该方法利用历史交通数据，通过 3 个主要模块捕捉长短期时空信息，并采用加性同态加密的方法共享短期时空隐藏信息。

文献[23]提出了一种低成本的联邦广泛学习(FBL)框架。该框架设计了一个广泛全连接模型(BFCM)用于本地模型训练，并提出了 DDFP 算法来解决资源调度问题。

在支持设备高速移动的场景下，文献[24]研究了结合设备移动性、计算与通信资源、信道状态及任务需求的计算卸载方案。该方案将计算卸载问题设计为混合整数非线性规划问题，并采用凸优化技术及改进的 Kuhn-Munkres 算法进行求解。

传统的强化学习算法通常仅关注奖励的期望值，忽视了奖励分布的其他统计信息，从而导致即使在相同的状态下采取相同的动作，在下一状态获得的奖励也可能发生变化。在这样的随机环境中，用分布来预测未来奖励比用单个数字预测未来奖励更加准确。

2 系统建模

2.1 系统模型

本文考虑一个 3 层的车载边缘计算系统，该系统由车辆用户设备层、RSU 层、云服务器层组成。如图2所示，在这个系统中，车辆可以通过无线链

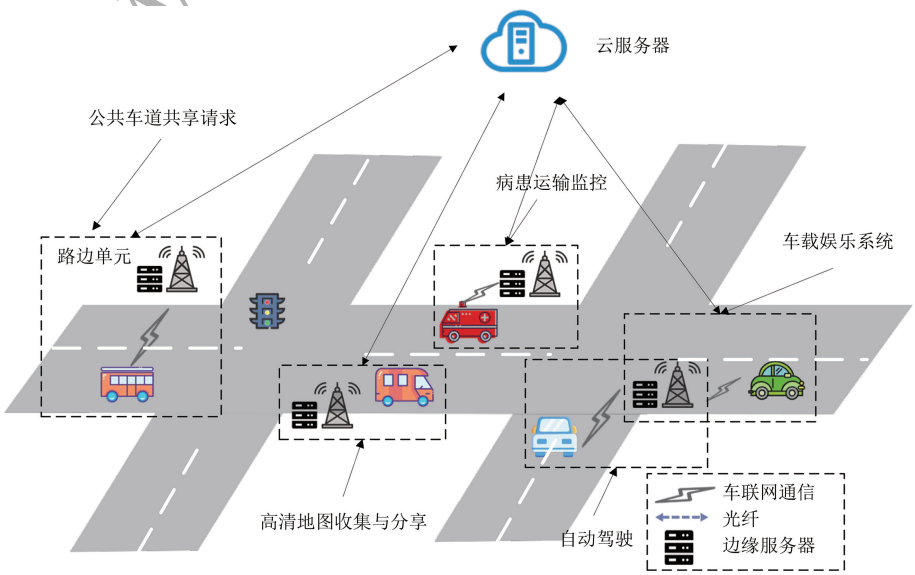

图 2 系统模型

Fig. 2 System model

路,将任务卸载到部署在路侧的边缘服务器上,对于需要更高计算能力或存储资源的任务,系统可以选择将这些任务从边缘服务器进一步卸载到云服务器上进行处理,以利用服务器的强大能力。

系统需要综合考虑各种因素,如任务的紧急性、数据的安全性需求、计算和传输的成本等,从而决定由哪一层次的节点处理车辆生成的每个任务。为方便起见,将本文的主要符号和相应描述总结在表 1 中。

表 1 符号描述

Table 1 Symbols description

符号	描述
$\mathcal{R},M$	RSU 集合,RSU 数量
$\mathcal{V},N$	车辆集合,车辆数量
$\mathcal{T},T$	任务集合,任务数量
k	车辆索引,$k\in\mathcal{V}$
j	MEC 索引,$j\in\mathcal{R}$
$T_{k,i}$	车辆 k 的第 i 个任务
$\lambda_{k,i}$	任务 $T_{k,i}$ 的卸载决策
$r_{k,j}$	车辆 k 与 MEC 服务器 j 之间的数据传输率
f_k^l	车辆 k 的计算能力
f_j^e	MEC 服务器 j 的计算能力
$r_{j,c}$	MEC 服务器 j 与云服务器之间的数据传输率
$t_{k,i}^{\text{local,completion}}$	任务 $T_{k,i}$ 的本地完成时间
$t_{k,i}^{\text{edge,completion}}$	任务 $T_{k,i}$ 卸载到边缘服务器所需要的完成时间
$t_{k,i}^{\text{cloud,completion}}$	任务 $T_{k,i}$ 卸载到云服务器所需时间

对于每个 RSU,都连接一个 MEC 服务器,为附近资源受限的车辆提供计算服务。同时,每个 MEC 服务器通过高速光纤连接到云服务器。设连续道路上有 N 台车辆,$\mathcal{V}=\{1,2,\cdots,N\}$表示车辆集合。在每一时刻,每个车辆都有多个计算密集型的任务需要处理。假设每个任务不能被分解成更小的子任务。设 $\mathcal{T}=\{1,2,\cdots,T\}$表示任务集。每个任务都可以建模为 $T_{k,i}=<c_{k,i},d_{k,i}>$,其中,$T_{k,i}$ 代表车辆 k 第 i 个任务,$k\in\mathcal{V}$ 表示车辆,i 是任务的索引,$c_{k,i}$ 表示完成任务 $T_{k,i}$ 所需的 CPU 周期数,$d_{k,i}$ 表示任务 $T_{k,i}$ 的输入数据大小。

在本系统中,每辆车都从 3 个任务卸载决策中选择一个。具体来说,$\lambda_{k,i}^{\text{local}},\lambda_{k,i}^{\text{edge}},\lambda_{k,i}^{\text{cloud}}\in\{0,1\}$分别描述任务 $T_{k,i}$ 是否在车辆本地、MEC 服务器、云服务器上处理。此外,任务 $T_{k,i}$ 的卸载决策 $\lambda_{k,i}$ 受如下约束:

$$\lambda_{k,i}^{\text{local}}+\lambda_{k,i}^{\text{edge}}+\lambda_{k,i}^{\text{cloud}}=1 \tag{1}$$

2.2 通信模型

因为整个卸载过程是毫秒级别的,所以假设车辆场景是准静态的,即移动设备的环境在计算卸载过程中不会发生太大变化。在 3 层 VEC 中存在 2 种通信模式,即 V2I 通信和基础设施到基础设施(I2I)通信。

1)V2I 通信。

当车辆 k 通过无线网络与 MEC 服务器通信时,使用 V2I 通信。假设在数据上传的过程中无线网络连接保持静态,那么任务 $T_{k,i}$ 从车辆 k 传输到 MEC 服务器 j 的传输速率 $r_{k,j}$ 为:

$$r_{k,j}=\beta_{k,j}\text{lb}\left(1+\frac{\rho_k\cdot\lambda_{k,j}}{\sigma^2+\sum\limits_{i'\in N'}\rho_{i'}\cdot\lambda_{i',j}}\right) \tag{2}$$

式中:$\beta_{k,j}$ 表示车辆 k 与 MEC 服务器 j 之间的带宽和距离;ρ_k 表示车辆 k 的发射功率;$\lambda_{k,j}$ 表示车辆 k 与 MEC 服务器 j 之间的信道增益;σ^2 表示周围噪声功率;$\sum\limits_{i'\in N'}\rho_{i'}\cdot\lambda_{i',j}$ 表示来自 MEC 服务器 j 覆盖范围内的其他车辆的无线干扰,N'表示这些其他车辆的集合。

VEC 系统将其计算任务传输到附近的 MEC 服务器,然后,服务器执行任务并将结果返回给车辆,车辆和边缘服务器之间的传输时延 $t_{k,j}^{\text{Tr}}$ 由下式给出:

$$t_{k,j}^{\text{Tr}}=\frac{d_{k,i}^{\text{input}}}{r_{k,j}}+\frac{d_{k,i}^{\text{output}}}{r_{k,j}} \tag{3}$$

式中:$d_{k,i}^{\text{input}}$ 为任务 $T_{k,i}$ 的输入数据量大小;$d_{k,i}^{\text{output}}$ 为任务 $T_{k,i}$ 的输出数据量大小。

2)I2I 通信。

MEC 服务器通过高速光纤与云服务器通信时,采用 I2I 通信。假设数据上传时光纤链路的上行和下行数据速率不变,设 $r_{j,c}$ 表示 MEC 服务器 j 与云服务器之间的数据传输速率,那么 MEC 服务器 j 与云服务器之间的传输时延 $t_{j,c}^{\text{Tr}}$ 由下式给出:

$$t_{j,c}^{\text{Tr}}=\frac{d_{k,i}^{\text{input}}}{r_{j,c}}+\frac{d_{k,i}^{\text{output}}}{r_{j,c}} \tag{4}$$

因此,将任务卸载到云服务器的总传输时延 $t_{k,c}^{\text{Tr}}$ 表示为:

$$t_{k,c}^{\text{Tr}}=t_{k,j}^{\text{Tr}}+t_{j,c}^{\text{Tr}}=\frac{d_{k,i}^{\text{input}}}{r_{k,j}}+\frac{d_{k,i}^{\text{output}}}{r_{k,j}}+\frac{d_{k,i}^{\text{input}}}{r_{j,c}}+\frac{d_{k,i}^{\text{output}}}{r_{j,c}} \tag{5}$$

2.3 计算模型

1)本地计算。

车辆 CPU 的每个核一次执行一个任务。在本地计算的情况下,任务需要按照先进先出的原则在任务队列中等待,直到计算资源可用为止。VEC 可以将任务放在本地计算,定义计算任务在车辆本地的完成时间为:

$$t_{k,i}^{\text{local, completion}}=t_{k,i}^{\text{local, execute}}+t_{k,i}^{\text{local,queue}} \tag{6}$$

$$t_{k,i}^{\text{local,execute}}=\frac{c_{k,i}}{f_k^l} \tag{7}$$

式中：$t_{k,i}^{\text{local,execute}}$ 表示任务 $T_{k,i}$ 在车辆 k 上执行所需的时间；$t_{k,i}^{\text{local, queue}}$ 表示任务 $T_{k,i}$ 在任务队列中等待的时间；f_k^l 代表车辆 k 的计算能力，即每秒可执行的 CPU 周期数。

2)边缘计算。

在边缘计算环境中，边缘服务器通常配备多个 CPU 核心，以增强其并行处理任务的能力。VEC 系统可以将任务 $T_{k,i}$ 卸载到 MEC 服务器上执行，计算任务在 MEC 服务器上的完成时间为：

$$t_{k,i}^{\text{edge, completion}}=t_{k,i}^{\text{edge, execute}}+t_{k,i}^{\text{edge, queue}} \tag{8}$$

$$t_{k,i}^{\text{edge,execute}}=\frac{c_{k,i}}{f_j^e} \tag{9}$$

式中：$t_{k,i}^{\text{edge, execute}}$ 表示任务 $T_{k,i}$ 在 MEC 服务器上执行所需的时间；$t_{k,i}^{\text{edge,queue}}$ 表示任务 $T_{k,i}$ 在边缘服务器的任务队列中等待的时间；f_j^e 代表第 j 个边缘服务器的计算能力。$t_{k,i}^{\text{edge, queue}}$ 是指从任务到达服务器开始到任务实际执行之前所需等待的时间，这个时间段的长短取决于服务器的当前负载情况以及任务的优先级等因素。

3)云服务器计算。

假设云服务器资源充足，以至于任务的排队时间相对执行时间可以忽略不计。因此，在将任务卸载到云服务器时，主要考虑任务在云服务器执行的时间。定义 $t_{k,i}^{\text{cloud,completion}}$ 为车辆 k 的计算任务 $T_{k,i}$ 在云服务器上的执行时间：

$$t_{k,i}^{\text{cloud,completion}}=\frac{c_{k,i}}{f_c} \tag{10}$$

式中：f_c 为云服务器的计算能力。

假设 $\lambda_{k,i}^{\text{local}}=1$、$\lambda_{k,i}^{\text{edge}}=1$、$\lambda_{k,i}^{\text{cloud}}=1$ 分别表示任务在本地执行、将任务卸载到边缘服务器执行以及将任务卸载到云端服务器执行。因此，计算任务 $T_{k,i}$ 所需要的执行时间为：

$$t_{k,i}=\begin{cases}t_{k,i}^{\text{local,completion}}, & \lambda_{k,i}^{\text{local}}=1\\ t_{k,i}^{\text{edge,completion}}+t_{k,j}^{\text{Tr}}, & \lambda_{k,i}^{\text{edge}}=1\\ t_{k,i}^{\text{cloud,completion}}+t_{k,c}^{\text{Tr}}, & \lambda_{k,i}^{\text{cloud}}=1\end{cases} \tag{11}$$

3 问题描述

在系统模型中，将整个任务卸载过程定义为一个 MDP，其中状态集合代表车辆当前的网络环境和计算资源状态，动作集合包括各种可能的任务卸载决策，转移概率反映了环境状态在卸载动作后的变化，奖励函数旨在量化不同卸载决策下的延迟性能。考虑到车辆的移动和边缘计算环境的动态性，卸载决策需要快速响应。因此，系统目标是找到计算复杂度较低的卸载方案，以最小化动态边缘计算环境中的累积处理延迟。每个任务 $T_{k,i}$ 的不同卸载决策都会导致不同的延迟性能。计算任务 $T_{k,i}$ 所需要的执行时间 $t_{k,i}$ 已经给出。因此，可以将任务卸载问题表述为约束优化问题，即在计算资源和通信资源的限制下使累积的处理延迟最小化，定义为：

$$\text{(P1)}t_{\text{total}}=\min_{\{\alpha_{k,i},\chi_{k,i}\}}\sum_{k\in V}t_{k,i} \tag{12}$$

$$\text{s. t. C1}:\lambda_{k,i}^{\text{local}}+\lambda_{k,i}^{\text{edge}}+\lambda_{k,i}^{\text{cloud}}=1 \tag{13}$$

$$\text{C2}:\sum_{k\in V}\beta_{k,j}\leqslant B \tag{14}$$

$$\text{C3}:\sum_{k\in V}\sum_{i\in I_k}\alpha_{k,i}^{\text{local}}\cdot f_{k,i}\leqslant f_{\max}^{\text{local}} \tag{15}$$

$$\text{C4}:\sum_{k\in V}\sum_{i\in I_k}\alpha_{k,i}^{\text{edge}}\cdot f_{k,i}\leqslant f_{e_j} \tag{16}$$

式中：$x_{k,i}=\{\beta_{k,j},f_{k,i}\}$ 为分配的通信和计算资源；C1 为卸载决策约束；C2 为给定的最大通信资源；C3 为本地设备的计算资源限制；C4 为边缘服务器的计算资源限制；$f_{k,i}$ 表示任务 $T_{k,i}$ 所需的计算资源；f_{e_j} 表示边缘服务器 j 的计算资源容量；$f_{\max}^{\text{local}}$ 表示本地设备的最大计算资源容量。

4 TPNE-DQRLD 算法

本文提出一种名为 TPNE-DQRLD 的算法，这是一种基于改进 IQN 的任务卸载算法。TPNE-DQRLD 算法通过时序经验优先回放机制，优先处理那些预期能提供更大学习价值的经验，从而加速学习过程。同时，算法引入噪声增强技术，通过在决策过程中加入随机噪声，增加探索性，帮助算法避免陷入局部最优解，提高对环境变化的适应性和策略的鲁棒性。

4.1 马尔可夫模型

鉴于优化问题符合马尔可夫性质，首先将其建模为一个 MDP。状态 s、动作 a 和奖励 r 的设置分别如下：

1)状态 s 表示为：

$$s=[r_{k,j},r_{j,c},d_{\text{local}}^{\text{queue}},d_{\text{edge}}^{\text{queue}},f_k^l,f_j^e,f_c,T_{k,i}]$$

式中：$r_{k,j}$ 表示车辆与 MEC 服务器之间的数据传输率；$r_{j,c}$ 表示 MEC 服务器与云服务器之间的数据传输率；$d_{\text{local}}^{\text{queue}}$、$d_{\text{edge}}^{\text{queue}}$ 分别表示车辆本地、MEC 服务器的任务队列长度。

2)动作空间为 $a=\{0,1,\cdots,R,R+1\}$，动作为

离散动作，例如，$a=0$ 意味着将计算任务卸载到远程云端执行，$a=1\sim R$ 表示将计算任务卸载到对应编号的 MEC 服务器执行，$a=R+1$ 表示计算任务将在本地执行。

3）转移概率 $P(s'|s,a)$ 表示在状态 s 下采取动作 a 之后转移到状态 s' 的概率。在车载边缘计算系统中，状态 s 到 s' 的转变受多个因素影响，包括车辆移动、任务执行与完成、通信条件变化以及计算资源的动态分配和使用。

4）奖励函数 $R(s,a,s')$ 表示在状态 s 下采取动作 a 并转移到状态 s' 时所获得的即时奖励。在此系统中，奖励通常与最小化处理延迟有关。因此，奖励可以设置为负的任务完成时间，即 $R(s,a,s')=-t_{k,i}$。根据定义，整个车载边缘计算任务卸载问题可以建模为寻找一个策略 π^*，它能最大化期望的累积奖励，即：

$$\pi^*=\underset{\pi}{\operatorname{argmax}}\,\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t,s_{t+1})\right] \tag{17}$$

式中：γ 是折扣因子，表示未来奖励的当前价值。

4.2 算法框架

本文所提 TPNE-DQRLD 算法的架构如图 3 所示。

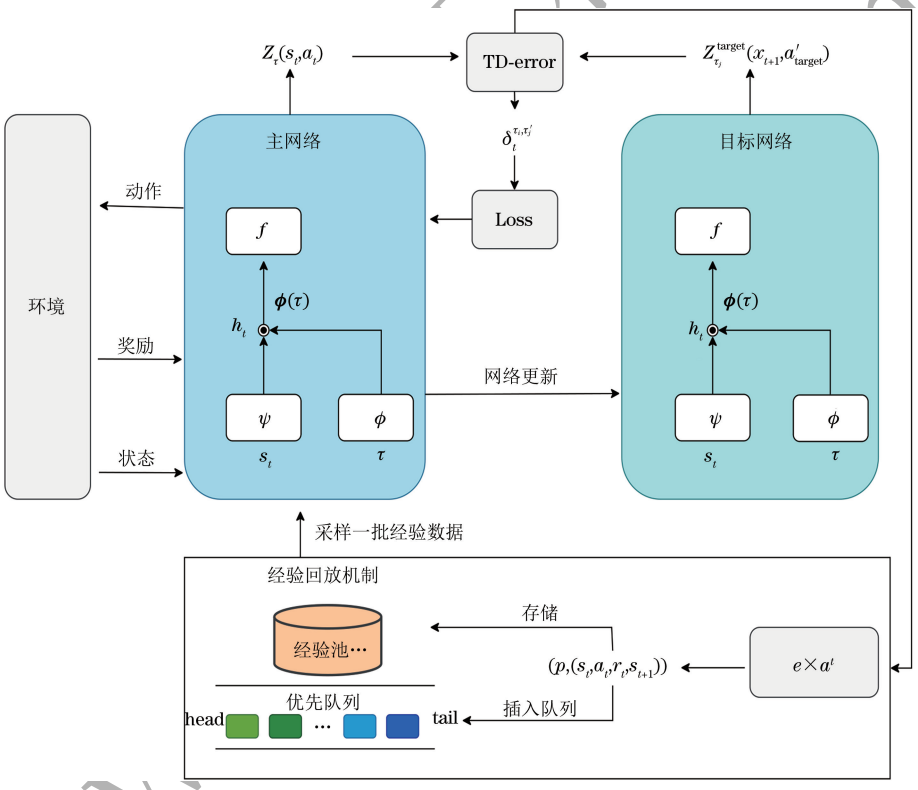

图 3 TPNE-DQRLD 算法架构

Fig. 3 TPNE-DQRLD algorithm architecture

1）IQN。

IQN 与传统的分位数网络的主要区别在于它们对值函数分布的近似方式。传统分位数网络通常估计分布的离散分位点，通过固定数量的分位点来近似值的分布。而 IQN 采用参数化的方法来表示分位函数，从而估计连续的分位点，这种方式使得 IQN 能够更灵活地学习整个值分布，不受限于预定的分位点数量。IQN 通过随机地从分位点的概率分布中采样，并将这些采样作为网络的输入，来隐式地近似整个值分布情况。这个方法提升了网络的近似分布能力，能够更全面地捕捉到底层数据分布的不确定性。由于 IQN 在每次更新时都随机生成分位点，这使得其能够探索并逼近值函数的连续分布，而不仅仅局限于特定的、固定的分位点集合。

在 TPNE-DQRLD 中，维护了 2 个 IQN：一个用于估计当前的行为值函数；另一个用于估计目标行为值函数。

每个 IQN 主要包括 3 个部分，即 ψ 卷积网络、ϕ 嵌入网络和 f 网络：

（1）ψ 卷积网络。首先，输入观测值 s_t 进入 ψ 网络，ψ 网络是一个前馈神经网络，包括一个线性层和激活函数。ψ 网络的目的是将原始观测值转换为隐藏表示 h_t。

（2）ϕ 嵌入网络。ϕ 网络包括一个余弦基线性层和激活函数。该网络将每个分位数值 τ 映射到一个由余弦函数构成的高维空间，这样做的目的是

利用余弦函数的周期性和平滑性，有效地捕获不同分位数之间的关系，从而为隐式分位数样本生成一个具有丰富表征能力的嵌入向量 $\boldsymbol{\phi}(\tau)$。其具体形式如下：

$$\boldsymbol{\phi}_j(\tau)=\mathrm{ReLU}\left(\sum_{i=0}^{n-1}\cos(\pi i\tau)w_{i,j}+b_j\right) \tag{18}$$

式中：n 表示嵌入维度，值为 64；τ 是从均匀分布 $U[0,1]$ 中抽样得到的；$\cos(\pi i\tau)$ 中的 i 与 π 相乘，表示不同的频率组成；$\omega_{i,j}$ 和 b_j 是线性层操作的权重和偏差。使用余弦变换的方法，能够有效地对每一个可能的 τ 值编码一个独特的表示，这些表示随后通过网络其余部分进行处理，以预测对应于该 τ 的值的分位数。最后，经过 ReLU 激活函数的输出为 $\boldsymbol{\phi}_j(\tau)$。

(3)f 网络。将隐藏表示 h_t 和分位数嵌入表示 $\boldsymbol{\phi}(\tau)$ 一起输入 f 网络中，这个过程可以表示为：

$$Z_\tau(s_t,a_t)\approx f(\psi(x)\odot\boldsymbol{\phi}(\tau))_a \tag{19}$$

式中：≈表示通过 f 网络的前向传播估计给定状态 s_t 和动作 a_t 在特定隐式分位数样本 τ 下的价值分布。这里的运算包括通过卷积函数 ψ 得到的结果和隐式分位数样本 τ 的嵌入函数 ϕ 得到的结果按元素相乘，然后通过全连接层进一步处理得到最终的分位数价值估计。

TD-error 是强化学习中的一种重要反馈信号，用于指导代理更新其策略和价值函数估计，以减少预测与实际奖励之间的偏差，TPNE-DQRLD 的 TD-error 定义为：

$$\delta_t^{\tau_i,\tau'_j}=r_t+\gamma Z_{\tau_j}^{\text{target}}(s_{t+1},a'_{\text{target}})-Z_{\tau_i}(s_t,a_t) \tag{20}$$

式中：$\delta_t^{\tau_i,\tau'_j}$ 表示时间步 t 的 TD-error，它是估计的分位数 Q 值与实际观察到的奖励加上未来奖励的折现值之间的差异；r_t 是即时奖励；γ 是折扣因子；$Z_{\tau_i}(s_t,a_t)$ 表示在状态 s_t 和动作 a_t 下的估计分位数 Q 值；$Z_{\tau_j}^{\text{target}}(s_{t+1},a'_{\text{target}})$ 表示在下一个状态 s_{t+1} 和动作 a'_{target} 下的目标分位数 Q 值。

在 TPNE-DQRLD 中，采用分位数 Huber 损失函数评估预测分位数与实际分位数之间的偏差。损失函数如下：

$$\mathcal{L}(s_t,a_t,r_t,s_{t+1})=\frac{1}{N'}\sum_{i=1}^{N}\sum_{j=1}^{N'}\rho_{\tau_i}^{\kappa}(\delta_t^{\tau_i,\tau'_j}) \tag{21}$$

式中：$\mathcal{L}(s_t,a_t,r_t,s_{t+1})$ 表示损失函数；N 和 N' 分别表示当前和目标网络的分位数样本数量，适当调整 N 和 N' 可以帮助找到性能和计算效率之间的平衡；$\rho_{\tau_i}^{\kappa}(\delta_t^{\tau_i,\tau'_j})$ 是 Huber 损失函数，κ 为损失平滑的阈值，它决定了误差何时被认为是大到足以从平方损失转变为线性损失，以减少异常值的影响；$\delta_t^{\tau_i,\tau'_j}$ 是 TD-error；τ_i 和 τ'_j 是从分布 $\beta(\cdot)$ 中采样得到的分位数样本，分布 $\beta(\cdot)$ 是在训练过程中定义的，它决定了分位数样本的取值范围和概率。

最后，使用如下公式进行动作选择：

$$a^*\leftarrow\underset{a'}{\operatorname{argmax}}\frac{1}{N}\sum_{i=1}^{N}Z_{\tilde{\tau}_k}^{\text{target}}(s',a'),\ \tilde{\tau}_k\sim\beta(\cdot) \tag{22}$$

式(22)表示从动作空间中选择使得预期奖励最大的动作作为最佳动作，这里的预期奖励是在所有采样的分位数 $\tilde{\tau}_k$ 上取平均后得到的。在式(22)中，a^* 是选择的动作，$Z_{\tilde{\tau}_k}^{\text{target}}(s',a')$ 是在状态 s' 和动作 a' 下的目标网络的分位数 Q 值，$\tilde{\tau}_k$ 是从分位数分布 β 中采样得到的分位数样本。

2)时序经验优先回放。

优先经验回放(PER)增强了强化学习中的传统经验回放机制，赋予智能体与环境交互过程中获得的经验不同的重要性级别。每个经验的重要性是通过其时间差异(TD)误差来量化的，这一误差反映了智能体对未来奖励的预测与实际观察到的奖励之间的差距。较高的 TD 误差意味着较大的预测误差，表明从这一经验中学习可能会带来显著的知识增益。对于每个经验，都会计算一个 TD 误差 δ_t，即：

$$\delta_t=R_t+\gamma\max_a Q(s_{t+1},a)-Q(s_t,a_t) \tag{23}$$

式中：R_t 是获得的奖励；γ 是折扣因子；$Q(s_t,a_t)$ 是当前状态-动作对的价值估计；$Q(s_{t+1},a)$ 是下一状态的最大动作价值估计。然而，这种基于 TD 误差的优先级分配策略并未充分考虑经验的时序性。时序性强调了经验之间的因果连续性，即当前的动作和结果如何影响未来的状态和奖励。对于学习算法来说，理解并利用这种连续性是理解复杂决策过程的关键，尤其是在处理具有高度时序依赖的任务时。

TPNE-DQRLD 不仅考虑经验的价值，也考虑经验的时间戳。对于最近的经验，会给予更高的优先级，因为认为最近的经验更能反映当前环境的状态。在时序优先级经验回放策略中，一个新经验的初始优先级 p 被定义为其 TD 误差 δ_t 乘以一个基于经验时间戳的衰减因子，即：

$$p=\delta_t\times\alpha^{t_{\text{current}}-t_{\text{experience}}} \tag{24}$$

式中：α 是时间衰减率；t_{current} 是当前时间步；$t_{\text{experience}}$ 是经验被记录的时间步。

3)噪声网络。

噪声网络通过在输入层之前添加一个噪声模块，以增加探索性。这个噪声模块生成一个与输入具有相同维度的噪声向量 $\boldsymbol{n}$，这个向量的元素是从某种分布中随机生成的，在实践中，使用高斯分布来生成噪声向量 $\boldsymbol{n}$。然后，将输入 $\boldsymbol{x}$ 和噪声向量 $\boldsymbol{n}$ 相加，得到新的输入：

$$\boldsymbol{x}_{\text{new}}=\boldsymbol{x}+\boldsymbol{n} \tag{25}$$

然后，将 $\boldsymbol{x}_{\text{new}}$ 作为下一层的输入。这样，噪声网络引入了随机性，使得代理不会过于依赖当前的策略，从而提高了探索性，可以更好地探索环境，避免陷入局部最优解，从而提高了学习的效果。

综上所述，TPNE-DQRLD 算法描述如下所示：

算法 1 TPNE-DQRLD 算法

输入 状态 s 的初始集合，动作空间 a，奖励函数 r，折扣因子 γ，学习率 α，噪声参数 σ

输出 为每个输入状态 s 选择的最优动作 a

1. 初始化 2 个隐式分位数网络，即当前网络 Q 和目标网络 Q′，两者具有相同的架构但参数独立

2. 初始化重播记忆 D 为空

3. FOR 每一个回合 DO

4. 初始化状态 s

5. WHILE s 不是终止状态 DO

6. 对于当前状态 s：

7. 从分布 β 中采样 N 个 τ 值

8. 对每个可能的动作 a 执行以下步骤：

9. 对每个 τ，通过主网络计算 $\phi_j(\tau)$ 和 $Z_\tau(s_t,a_t)$

10. 计算动作 a 的平均分位数 Q 值：$\overline{Q}(s_t,a_t)=\frac{1}{N}\sum_{\tau}Z_\tau(s_t,a_t)$

11. 选择动作 $a^*=\underset{a}{\operatorname{argmax}}\,\overline{Q}(s_t,a_t)$

12. 执行动作 a^*，观察奖励 r 和新状态 s_{t+1}

13. 将转移(s_t,a^*,r,s_{t+1})存储到重播记忆 D 中

14. FOR 每个抽取的经验(s_t,a^*,r,s_{t+1}) DO

15. 对当前网络和目标网络采样不同的 τ 值

16. 计算当前网络的分位数 Q 值和目标网络的分位数 Q 值 $Z_{\tau'}(s_{t+1},a^*)$

17. 通过目标网络计算 TD 目标：$y_t=r+\gamma Z_{\tau'}(s',\underset{a'}{\operatorname{argmax}}\,\overline{Q}(s',a'))$

18. 使用 $Z_\tau(s,a)$ 和 y_t 计算分位数 Huber 损失 $\mathcal{L}$

19. 使用梯度下降法，根据损失 $\mathcal{L}$ 更新当前网络 Q 的参数

20. 每 N 步后，更新目标网络 Q′的参数，使其等于当前网络 Q 的参数

21. 更新状态

22. END WHILE

23. END FOR

5 实验结果与分析

在本研究中，为了验证算法的性能，在配备有 Intel E5-2640 处理器、4 块 NVIDIA TITAN X 显卡、64 GB 内存的服务器上，基于 ChainerRL 强化学习框架实现了 TPNE-DQRLD 算法，并与算法 DFO-DDQN[12]、TODQN[9]进行对比。

1)DFO-DDQN 是一种针对移动边缘计算的任务卸载策略，利用强化学习中的 DDQN 进行决策优化。

2)TODQN 是一种基于深度强化学习构建的高效任务卸载算法，特别考虑了移动边缘计算的需求。该算法采用 DQN 结构，通过不断更新深度神经网络来学习最优卸载策略。

5.1 实验设置

在仿真实验设置中，考虑一个 3 层的 VEC，传输功率 $\rho=2$ W±0.2 W 表示传输功率在 2 W 的基础上存在±0.2 W 的浮动范围，模拟了实际环境中的功率波动情况。信道增益 $\lambda=144$ dB±14.4 dB 表示信道增益在 144 dB 的基础上存在±14.4 dB 的浮动范围，模拟了实际环境中的信道条件变化。这些参数在仿真中是随机变化的，以更真实地反映车联网环境中的不确定性。实验设计包括 30 万个训练步骤，确保算法有充分地学习和适应环境变化的机会。经验回放机制在 8 万步时启用，旨在利用历史经验来加速学习过程。目标网络的更新间隔设置为 5 000 步，以保持学习的稳定性。采用 ε-贪婪策略进行环境探索，其中，$\varepsilon=0.1$，以平衡探索和利用。折扣因子 $\gamma=0.995$，以强调长期奖励的重要性。参数设置如表 2 所示。

表 2 模拟实验参数

Table 2 Simulation experiment parameters

参数	设置
传输功率 ρ/W	2±0.2
信道增益 λ/dB	144±14.4
噪声功率 σ^2/W	1.5×10^{-8}
折扣因子 γ	0.995
经验池大小	80 000
训练步数	300 000
小批量大小 D_τ	32
MEC 服务器的通信能力/MHz	20
光纤链路的数据速率 μ/(Gb·s^{-1})	1
IQN 的隐藏层数/层	64
余弦基线性层数/层	128

5.2 结果分析

TPNE-DQRLD 与其他算法的收敛性能比较结果如图 4 所示。

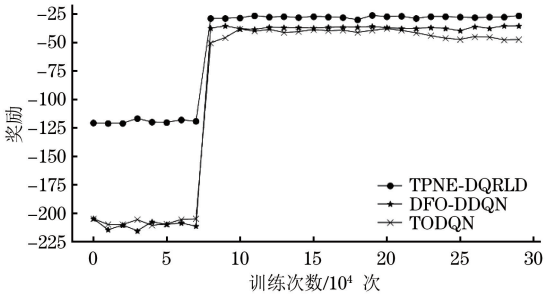

图 4 算法的收敛性能对比

Fig. 4 Comparison of convergence performance of algorithms

在这项实验中，实验结果的奖励值为负数，代表算法在交通网络环境模拟中产生的平均时延，因此，更低的奖励值意味着算法在减少平均时延方面的性能更优。实验结果表明，TPNE-DQRLD、DFO-DDQN 和 TODQN 算法的最终奖励值分别稳定在 −41.5、−48.7 和 −31.2。因此，与 DFO-DDQN 和 TODQN 相比，TPNE-DQRLD 实现了最低的平均时延，展现出最好的效果。此外，注意到 TODQN 在实验后期的波动较大，这可能是由于其探索策略或参数调整机制在面对环境的某些动态变化时较为敏感，导致学习过程出现短暂的不稳定。

TPER 消融实验特别排除了 TPER 策略的使用，即对比算法使用 PER。在实验过程中，将平均奖励滑动窗口调整为 5 000，使之更加平滑，实验结果如图 5 所示。在不使用 TPER 的情况下，TPNE-DQRLD 的性能有所下降，这进一步验证了 TPER 在所提算法中的重要性。相反，集成了 TPER 的 TPNE-DQRLD 表现出了更高的奖励值和更好的收敛性能。实验结果表明，采用时序优先经验回放的 TPNE-DQRLD 算法在快速适应环境变化和提高长期奖励方面比优先经验回放策略更为有效，在实现相同学习目标的步骤数上减少了 5 万步。

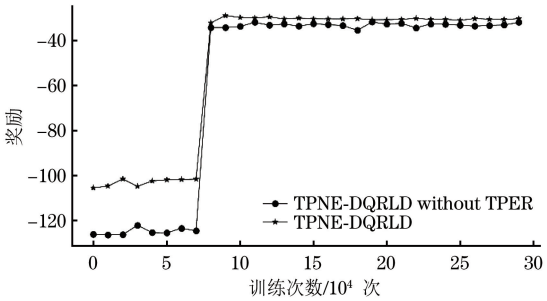

图 5 TPER 消融实验结果

Fig. 5 Results of TPER ablation experiment

如图 6 所示，随着车辆数量的增加，所有算法的系统时延都呈上升趋势，这意味着系统规模的扩大导致处理延迟增加。TPNE-DQRLD 算法在各个实验下的时延均低于其他 2 种算法，表现出了更优的性能。TODQN 和 DFO-DDQN 在车辆数量较少时时延相近，但随着车辆数量的增加，TODQN 的时延增长速度高于 DFO-DDQN，这表明在处理更大规模的系统时，TODQN 可能面临性能瓶颈。

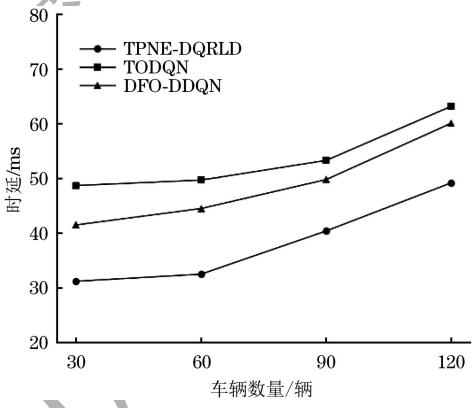

图 6 车辆数量对时延的影响

Fig. 6 The impact of vehicle quantity on latency

TPNE-DQRLD 算法和 DoubleIQN、IQN[25] 算法的收敛性能对比如图 7 所示，经验池达到容量后 TPNE-DQRLD、DoubleIQN、IQN 算法均开始收敛。然而，DoubleIQN 与 IQN 算法在收敛过程中均表现出一定程度的性能下滑，尤其是 IQN 算法在后期波动更为显著。对比结果表明，DoubleIQN 的收敛性略优于 IQN。在收敛速度和稳定性方面，TPNE-DQRLD 算法表现出更优的性能。

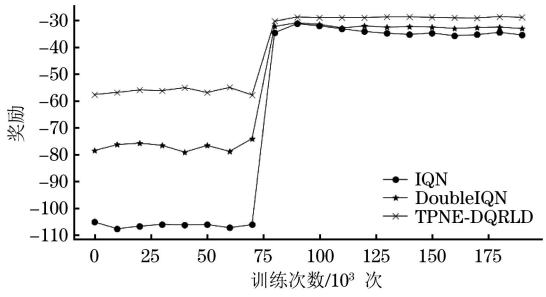

图 7 3 种算法的收敛性能对比

Fig. 7 Comparison of convergence performance of three algorithms

为了全面评估和比较算法性能，设置了 6 种应用任务，每种任务代表了不同的挑战，有的要求处理大量的输入数据，有的需要生成大量的输出结果，还有的则对计算能力有着严格的要求。这些应用涵盖了从数据密集型到计算密集型的多个方面，以确保测试结果的广泛适用性。不同应用的设置如表 3 所示。

表 3 应用设置

Table 3 Application settings

名称	数据输入大小/10^3 bit	数据输出大小/10^3 bit	处理成本/(CPU cycles•bit^{-1})	平均生成周期/ms
应用 1	3 600	100	1 500	100
应用 2	1 000	80	100	100
应用 3	100	20	100	100
应用 4	200	100	1 000	100
应用 5	50	10	50	100
应用 6	50	5	10	10

图 8 提供了一个直观的性能比较，它反映了 TPNE-DQRLD、DFO-DDQN 和 TODQN 这 3 种算法在各种应用场景下的时延表现。在所有应用场景中，TPNE-DQRLD 表现出最低的延迟，而 DFO-DDQN 和 TODQN 有相对较高的延迟。在应用场景 5 中，TPNE-DQRLD 和 DFO-DDQN 的延迟相似且明显低于 TODQN，这表明在处理需要生成较小数据输出的任务时，TODQN 可能不是最佳选择。在应用场景 3 中，3 种算法的延迟差异不明显，这意味着它们可能在处理均衡输入输出大小的任务时具有相似的性能。在其他应用任务中，特别是在计算需求和数据传输需求相互冲突的情况下，算法之间的时延差异变得更加显著。TPNE-DQRLD 在需要同时考虑即时反应和长期规划的应用任务中，特别是当计算和传输需求存在巨大差异时，能够更好地平衡响应时间与决策质量，从而保持了低时延的优势。

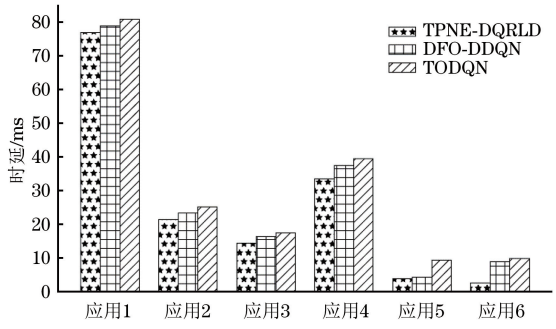

图 8 3 种算法的时延比较

Fig. 8 Time delay comparison of three algorithms

6 结束语

在 VEC 系统中，有效的任务卸载决策是实现资源优化和系统性能提升的关键。为此，本文提出了一种基于 IQN 的车联网任务卸载策略和 TPNE-DQRLD 算法，以应对现有深度强化学习算法在收敛训练复杂性上的挑战。该算法通过估计值函数分布来增强探索效率，提升计算性能和决策质量，实验显示，其能带来更稳定的学习效果同时降低任务完成时延。未来可以将时延与能耗作为一个联合优化目标，以实现更全面的资源管理策略。此外，空闲车辆的潜在作用也不容忽视，它们可以被视为移动的小型路侧单元，为邻近车辆提供额外的计算资源。这些因素的考量将有助于进一步完善车联网环境中的计算资源分配与优化。

参考文献

[1] 刘雷，陈晨，冯杰，等. 车载边缘计算卸载技术研究综述[J]. 电子学报，2021，49(5)：861-871.
LIU L，CHEN C，FENG J，et al. A survey of computation offloading in vehicular edge computing networks[J]. Acta Electronica Sinica，2021，49(5)：861-871.(in Chinese)

[2] 李萌，司鹏搏，孙恩昌，等. 基于车联网和移动边缘计算的时延可容忍数据传输[J]. 北京工业大学学报，2018，44(4)：529-537.
LI M，SI P B，SUN E C，et al. Delay-tolerant data transmission based on Internet of Vehicles and mobile edge computing[J]. Journal of Beijing University of Technology，2018，44(4)：529-537.(in Chinese)

[3] 李子姝，谢人超，孙礼，等. 移动边缘计算综述[J]. 电信科学，2018，34(1)：87-101.
LI Z S，XIE R C，SUN L，et al. A survey of mobile edge computing[J]. Telecommunications Science，2018，34(1)：87-101.(in Chinese)

[4] 曹敦，张应宝，邹电，等. V2X 多节点协同分布式卸载策略[J]. 通信学报，2022，43(2)：185-195.
CAO D，ZHANG Y B，ZOU D，et al. Multi-node cooperative distributed offloading strategy for V2X [J]. Journal of Communications，2022，43(2)：185-195.(in Chinese)

[5] LEE J，NA W. A survey on vehicular edge computing architectures[C]// Proceedings of the 13th International Conference on Information and Communication Technology Convergence (ICTC). Washington D.C.，USA：IEEE Press，2022：2198-2200.

[6] 戚可寒. 面向车辆边缘计算的任务卸载关键技术研究[D]. 杭州：杭州电子科技大学，2023.
QI K H. Research on key technologies of task unloading for vehicle edge computing[D]. Hangzhou：Hangzhou Dianzi University，2023.(in Chinese)

[7] YUAN S，FAN Y F，CAI Y. A survey on computation offloading for vehicular edge computing[C]//Proceedings of the 7th International Conference on Information Technology：IoT and Smart City. New York，USA：ACM Press，2019：107-112.

[8] AHMED M，RAZA S，MIRZA M A，et al. A survey on vehicular task offloading：classification，issues，and challenges[J]. Journal of King Saud University - Computer and Information Sciences，2022，34(7)：4135-4162.

[9] DAI F，LIU G Z，MO Q，et al. Task offloading for vehicular edge computing with edge-cloud cooperation[J]. World Wide Web，2022，25(5)：1999-2017.

[10] ZHANG J，GUO H Z，LIU J J，et al. Task offloading in vehicular edge computing networks：a load-balancing solution[J]. IEEE Transactions on Vehicular Technology，2020，69(2)：2092-2104.

[11] YAO J J，ANSARI N. Task allocation in fog-aided mobile IoT by Lyapunov online reinforcement learning[J]. IEEE Transactions on Green Communications and Networking，2020，4(2)：556-565.

[12] TANG H J, WU H M, QU G J, et al. Double deep Q-network based dynamic framing offloading in vehicular edge computing[J]. IEEE Transactions on Network Science and Engineering, 2023, 10(3): 1297-1310.
[13] LI S Y, HU X H, DU Y W. Deep reinforcement learning and game theory for computation offloading in dynamic edge computing markets[J]. IEEE Access, 2021, 9: 121456-121466.
[14] 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.
[15] CUI Y P, DU L J, HE P, et al. Multi-vehicle intelligent collaborative computing strategy for Internet of vehicles[C]// Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC). Washington D.C., USA: IEEE Press, 2022: 1647-1652.
[16] LIANG S N, WAN H B, QIN T F, et al. Multi-user computation offloading for mobile edge computing: a deep reinforcement learning and game theory approach[C]// Proceedings of the 20th IEEE International Conference on Communication Technology (ICCT). Washington D.C., USA: IEEE Press, 2020: 1534-1539.
[17] ZHU A Q, GUO S T, MA M F, et al. Computation offloading for workflow in mobile edge computing based on deep Q-learning[C]//Proceedings of the 28th Wireless and Optical Communications Conference (WOCC). Washington D.C., USA: IEEE Press, 2019: 1-5.
[18] WU Z Y, YAN D F. Deep reinforcement learning-based computation offloading for 5G vehicle-aware multi-access edge computing network[J]. China Communications, 2021, 18(11): 26-41.
[19] 李强, 仪晋辉, 杜婷婷, 等. 移动边缘计算中基于 A3C 的依赖任务卸载与资源分配[J]. 计算机工程, 2023, 49(6): 42-52.
LI Q, YI J H, DU T T, et al. Dependent task offloading and resource allocation based on A3C in mobile edge computing[J]. Computer Engineering, 2023, 49(6): 42-52. (in Chinese)
[20] DAI P L, HU K W, WU X, et al. Asynchronous deep reinforcement learning for data-driven task offloading in MEC-empowered vehicular networks[C]//Proceedings of the IEEE INFOCOM Conference on Computer Communications. Washington D.C., USA: IEEE Press, 2021: 1-10.
[21] WANG C L, PENG J, JIANG F, et al. An adaptive deep Q-learning service migration decision framework for connected vehicles[C]// Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC). Washington D.C., USA: IEEE Press, 2020: 944-949.
[22] YUAN X M, CHEN J H, YANG J Y, et al. FedSTN: graph representation driven federated learning for edge computing enabled urban traffic flow prediction[J]. IEEE Transactions on Intelligent Transportation Systems, 2023, 24(8): 8738-8748.
[23] YUAN X M, CHEN J H, ZHANG N, et al. Low-cost federated broad learning for privacy-preserved knowledge sharing in the RIS-aided Internet of vehicles[J]. Engineering, 2024, 33: 178-189.
[24] 班玉琦, 段利国, 温昊宇, 等. 面向移动感知的计算卸载及资源分配策略研究[J]. 计算机工程, 2023, 49(8): 163-173.
BAN Y Q, DUAN L G, WEN H Y, et al. Research on mobility-aware computation offloading and resource allocation strategy[J]. Computer Engineering, 2023, 49(8): 163-173. (in Chinese)
[25] DABNEY W, OSTROVSKI G, SILVER D, et al. Implicit quantile networks for distributional reinforcement learning[C]// Proceedings of International Conference on Machine Learning. New York, USA: ACM Press, 2018: 1096-1105.

编辑 吴云芳