Modeling and Simulation 建模与仿真, 2025, 14(5), 336-352
Published Online May 2025 in Hans. https://www.hanspub.org/journal/mos
https://doi.org/10.12677/mos.2025.145398

Hans汉斯

TDLB-DDPG

——基于任务依赖和负载均衡的VEC计算卸载方案

钱　振

上海理工大学光电信息与计算机工程学院，上海

收稿日期：2025年4月14日；录用日期：2025年5月7日；发布日期：2025年5月14日

摘　要

如何合理地对车辆上的计算任务进行卸载，以降低任务的响应时间、RSU的平均负载率，是车辆边缘计算(Vehicular Edge Computing, VEC)中的重要问题。深度强化学习是解决计算卸载问题的有力工具，但当前深度强化学习方法往往未考虑具有依赖关系的卸载任务之间是否可以合并以及路侧单元间的负载均衡。对此，文章提出一种基于任务依赖和负载均衡的深度确定性策略梯度(Deep Deterministic Policy Gradient, DDPG)计算卸载方案；通过任务分类合并，减小DDPG的动作空间及批量处理任务，降低卸载任务的平均响应时间；综合考虑任务群组的优先级、紧迫程度和关键路径长度来对任务群组进行排序，以提高任务的卸载成功率；设计一种RSU动态权重分配机制，辅助DDPG实现RSU之间的负载均衡。实验结果表明，相比已有基于DQN (Deep Q-Network)和DDPG的计算卸载方法，本文所提方案的平均响应时间分别降低了21.4%和22.5%，RSU平均负载率也显著低于DQN和DDPG。

关键词

计算卸载，路侧单元，深度强化学习，任务依赖，负载均衡

TDLB-DDPG

—Task Dependency and Load Balancing Based on Computation Offloading Scheme in VEC

Zhen Qian

School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai

Received: Apr. 14th, 2025; accepted: May 7th, 2025; published: May 14th, 2025

Abstract

Efficiently offloading computational tasks from vehicles to reduce task response times and improve the

文章引用: 钱振. TDLB-DDPG——基于任务依赖和负载均衡的 VEC 计算卸载方案[J]. 建模与仿真, 2025, 14(5): 336-352.
DOI: 10.12677/mos.2025.145398

average load rate of Roadside Units (RSUs) is a critical challenge in Vehicular Edge Computing (VEC). Deep Reinforcement Learning (DRL) has emerged as a powerful tool for addressing computational offloading problems. However, existing DRL-based approaches often fail to consider whether dependent offloading tasks can be merged and overlook the load balancing among roadside units. To address these limitations, this paper proposes a Deep Deterministic Policy Gradient (DDPG)-based computational offloading scheme that incorporates task dependency and load balancing. By categorizing and consolidating tasks, the action space of DDPG is reduced, and tasks are processed in batches, thereby decreasing the average response time for offloaded tasks. It further prioritizes task groups based on their priority, urgency, and critical path length to enhance offloading success rates. Additionally, a dynamic weight allocation mechanism for RSUs is designed to assist DDPG in achieving load balancing among RSUs. Experimental results demonstrate that, compared to existing Deep Q-Network (DQN) and DDPG-based offloading methods, the proposed scheme reduces the average response time by 21.4% and 22.5%, respectively, while significantly lowering the average RSU load rate.

Keywords

Computation Offloading, Roadside Unit, Deep Reinforcement Learning, Task Dependency, Load Balancing

1. 引言

随着车辆和车辆(Vehicle-to-Vehicle, V2V) [1]、车辆和基础设施(Vehicle-to-Infrastructure, V2I)之间的联通，城市智能交通系统(Intelligent Traffic System, ITS) [2]-[4]迅速发展。车载边缘计算(Vehicle Edge Computing, VEC) [5] [6]通过将算力分布放置在路侧单元(Roadside Units, RSUs) [7] [8]上，给附近车辆提供高质量的服务(Quality of Service, QoS) [9] [10]。当前 RSU 协同机制仍存在优化空间，道路车流的差异化使得同一区域内的不同 RSU 之间负载不均衡。在涉及到具有依赖关系的任务卸载时，如何构建兼顾负载均衡和任务依赖的卸载框架，已然成为挑战。传统的元启发式方法存在收敛不稳定和计算开销大的问题[11] [12]，而深度强化学习(Deep Reinforcement Learning, DRL)通过融合深度神经学习(Deep Neural Network, DNN)的感知能力与强化学习(Reinforcement Learning, RL)的决策机制[13] [14]，可以基于环境状态选择卸载动作，并通过奖励信号动态调整策略，尤其擅长处理任务依赖及 RSU 间动态均衡等问题。

目前，城市交通场景下的 VEC 卸载研究依旧存在一些困难和挑战。Fofana 等[15]提出了一种结合 Stackelberg 博弈和强化学习的 VEC，实时分析车辆状态以确定计算需求和成本函数，并设计基于经验的卸载策略。Walia 等[16]将决策树引入强化学习，优化物联网应用中的计算卸载和资源分配，提升服务质量。Pang 等[17]利用有向无环图(DAG)建模任务依赖关系，设计动态权重优化延迟和能耗，并通过双深度 Q 网络(DDQN)进行训练。Zhao 等[18]提出了一种移动感知任务依赖方案，考虑车辆移动性和任务数据依赖，采用 DRL 训练卸载策略，设计车辆移动性检测算法计算通信时间，并通过任务优先级确定方案对等待队列中的任务进行排序。在上述研究中，有的研究没有关注卸载任务之间存在的强依赖关系；有的研究即便关注到了卸载任务之间的依赖关系，也只是用有向无环图(DAG)进行标记，然后顺序地卸载到 RSU 上，没有对这些依赖关系进行深入的研究，如对这些卸载任务进行合并批量处理或者并行处理等。Zhang 等[19]提出了多类型任务平衡卸载算法，根据任务类型、大小和最大容忍延迟对任务排序，MEC 依据优先级执行不同卸载策略。Li 等[20]引入占用表动态跟踪边缘负载状态，将计算任务直接分配给资源充足

的服务节点，避免边缘服务器过载或空闲。Zheng 等[21]为 RSU 设定价格，车辆评估价格以选择任务卸载目标，并在多智能体强化学习中引入纳什均衡优化卸载决策，提升竞争环境下移动车辆的效用。在上述研究中，没有考虑任务性质和具体场景，如一些延迟敏感的紧急任务需要本地处理，不能进行卸载；也没有考虑 RSU 之间的负载均衡。

本文提出了一种基于 DDPG 任务依赖和负载均衡的计算卸载方案(Task Dependency and Load Balancing Based DDPG for Computation Offloading, TDLB-DDPG)。主要贡献如下：

1) 我们考虑了一种城市交通计算卸载的场景，将计算卸载形式化为卸载任务的平均响应时间和 RSU 平均负载率的组合优化问题，并将该问题建模为马尔可夫决策过程(Markov Decision Process, MDP)，给出了基于 DDPG 的卸载策略训练方法。

2) 提出了一种任务依赖分类合并机制，对卸载任务进行分类，并把相同类型的卸载任务进行合并，减小 DDPG 的动作空间，提高其训练效率，降低平均任务响应时间。设计了全局优先级指数来改变 RSU 队列中任务群组之间的处理顺序，重要、紧急的任务可以优先处理。

3) 提出了一种 RSU 动态权重，来帮助 DDPG 对周围的 RSU 进行合理的负载均衡，有效降低了系统中的 RSU 平均负载率。

2. 模型

2.1. 系统模型

图 1 为基于 VEC 的计算卸载系统，系统包含 N 个 RSU 和 M 辆车，分别表示为 $\mathrm{RSU}=\{\mathrm{rsu}_1,\mathrm{rsu}_2,\cdots,\mathrm{rsu}_n,\cdots,\mathrm{rsu}_N\}$ 和 $V=\{v_1,v_2,\cdots,v_m,\cdots,v_M\}$，以及一个基站(Base Station, BS)。其中，车辆和 RSU 通过 V2I 通信连接，所有 RSU 都配备了 MEC 服务器和决策模块，MEC 服务器为周围车辆提供计算资源的支持，决策模块为通信范围内的车辆提供卸载决策；RSU 之间通过光纤连接；车辆可以通过 V2I 连接到 BS，再由 BS 通过光纤上传至云端。车辆生成的任务可本地处理、卸载到 RSU 或者云端。此外，假设 MEC 系统是在离散时隙下运行，定义 T 为时隙的数量，时隙集合表示为 $\{1,2,\cdots,t,\cdots,T\}$。

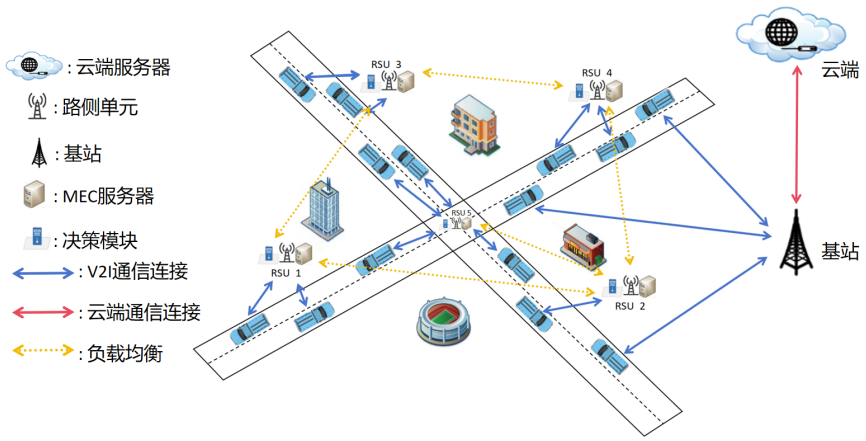

Figure 1. VEC task offloading system
图 1. VEC 计算卸载系统

2.2. 应用程序模型

在 RSU 通信范围内，车辆在每个时隙 t 都会生成多个应用，而这些应用会产生相应的具有依赖关系的任务，这些任务可以本地处理，也可以卸载到 RSU 或者云端。假设车辆 v_m 在时隙 t 生成的应用为

$A_{t,m,j}=\left\{v_m,Z,H_j,t^{deadline},type\right\}$，其中 $j\in[1,2,\cdots,J]$，Z 为应用 $A_{t,m,j}$ 生成的任务数量，H_j 为应用生成的任务之间的依赖关系，$t^{deadline}$ 为应用 $A_{t,m,j}$ 中任务的最大容忍响应时间，$type$ 为应用的类型。$A_{t,m,j}$ 可以分为若干任务 $B_{j,z}=\left\{b_{j,1},b_{j,2},\cdots,b_{j,z},\cdots,b_{j,Z}\right\}$，任务可以定义为 $b_{m,j,z}=\left\{m,j,z,datasize,t^{deadline}\right\}$，$datasize$ 为任务大小。邻接矩阵 $H_{m,j}=H_{m,j,l,r}\left(m\times m\right)$ 代表生成任务之间的依赖关系，矩阵中的参数为 0 或 1，即 $H_{m,j,l,r}=\left\{0,1\right\},1\le l<r\le m$。

2.3. 通信模型

在每个时隙 t，可以把整个系统看成静态的，系统中的每辆车都采用 V2I 通信方式与 RSU 保持连接。根据香农公式，车辆到 RSU 的传输速率可以定义为：

$$\Phi_1=W\log_2\left(1+\frac{p_m g_{m,n}}{\sigma^2}\right) \tag{1}$$

其中，W 为信道带宽，p_m 为车辆 v_m 的传输功率，σ^2 为高斯噪声。$g_{m,n}$ 为车辆 v_m 和 rsu_n 之间链路的信道增益，可以定义为：

$$g_{m,n}=\frac{g_m g_n \lambda^2}{\left(4\pi d_{m,n}\right)^2} \tag{2}$$

其中，g_m 和 g_n 是车辆的发射天线增益和 RSU 的接收天线增益，为固定值；λ 为信号波长，为固定值；$d_{m,n}$ 为车辆和 RSU 之间的距离。车辆到 BS 的传输速率 Φ_2、BS 到云端的传输速率 Φ_3 和式(1)类似，定义为：

$$\Phi_2=W\log_2\left(1+\frac{p_m g_{m,BS}}{\sigma^2}\right) \tag{3}$$

$$\Phi_3=W\log_2\left(1+\frac{p_{BS} g_{BS,cloud}}{\sigma^2}\right) \tag{4}$$

因此，任务的上传时间 $t_{m,z}^{trans}$ 可以定义为：

$$t_{m,j,z}^{trans}=\begin{cases}0, & x_{m,j,z}=0\\ \dfrac{datasize_{m,j,z}}{\Phi_1}, & x_{m,j,z}=n\\ \dfrac{datasize_{m,j,z}}{\Phi_2}+\dfrac{datasize_{m,j,z}}{\Phi_3}, & x_{m,j,z}=N+1\end{cases} \tag{5}$$

其中，$x_{m,j,z}=0$ 表示车辆的任务在本地处理，$x_{m,j,z}=n$ 表示车辆的任务卸载到 RSU 或临近的 RSU，$x_{m,j,z}=N+1$ 表示任务卸载到云端，即由车辆发给 BS，BS 再发给云端。由于车辆的传输功率和 RSU 的传输功率相差很大，因此我们只考虑任务的上传时间，忽略 RSU 和 BS 发送计算结果给车辆的时间。

2.4. 计算模型

计算卸载的计算时间可以定义为处理时间和排队时间。在时隙 t，车辆、云端或 RSU 处理任务所需的时间称为处理时间，定义为：

$$t_{m,z}^{exe}=\begin{cases}\dfrac{datasize_{m,j,z}}{f^{v_m}}, & x_{m,j,z}=0\\ \dfrac{datasize_{m,j,z}}{f^{rsu_n}}, & x_{m,j,z}=n\\ \dfrac{datasize_{m,j,z}}{f^{cloud}}, & x_{m,j,z}=N+1\end{cases} \tag{6}$$

其中，f^{v_m}、f^{rsu_n}、f^{cloud} 分别是车辆、RSU 和云端的处理能力。任务在车辆或 RSU 处理之前所等待的时间称为排队时间，因云端处理能力强大，排队时间可忽略不计。因此，排队时间可以定义为：

$$t_{m,z}^{delay}=\begin{cases}\dfrac{d_{m,j,z}^{prior\_local}}{f^{v_m}}, & x_{m,j,z}=0\\ \dfrac{d_{m,j,z}^{prior\_rsu}}{f^{rsu_n}}, & x_{m,j,z}=n\\ 0 & x_{m,j,z}=N+1\end{cases} \tag{7}$$

其中，$d_{m,j,z}^{prior\_local}$ 和 $d_{m,j,z}^{prior\_rsu}$ 分别为当任务在本地和 RSU 上处理时，队列中当前任务前面所有未处理任务的大小总和。因此，在时隙 t，车辆 v_m 处理单个任务的响应时间为：

$$T_{t,m}^{one\_task\_offload}=t^{trans}+t^{exe}+t^{delay} \tag{8}$$

车辆 v_m 在时隙 t 的平均任务响应时间为：

$$T_{t,m}^{avg\_task\_offload}=\sum_{j=1}^{J}\sum_{z=1}^{Z}\frac{1}{J\times Z}T_{t,m}^{one\_task\_offload} \tag{9}$$

2.5. 优化问题

为了使卸载策略达到最优，需要最小化卸载任务平均响应时间和平均 RSU 负载率，可以形式化为如下优化问题：

$$\begin{aligned}&\min\left(\alpha_1 T_{avg}+\beta_1 L_{avg}\right)\\ &s.t.\quad C1:\alpha_1+\beta_1=1\\ &\qquad C2:\alpha_1>0,\beta_1>0\\ &\qquad C3:x_{t,m,j,z}=\{0,n,N+1\}\end{aligned} \tag{10}$$

其中，T_{avg} 和 L_{avg} 分别为所有卸载任务的平均响应时间和 RSU 的平均负载率，α_1 和 β_1 分别是 T_{avg} 和 L_{avg} 的加权系数，在实际场景中，我们可以根据用户的喜好来调整 α_1 和 β_1 的值，$x_{t,m,j,z}=\{0,n,N+1\}$ 为时隙 t 时，对车辆 v_m 上应用程序 j 中任务 z 的卸载决策，0 表示本地处理，n 表示卸载到 RSU，$N+1$ 表示卸载到云端。

T_{avg} 定义为：

$$T_{avg}=\frac{1}{T\times M}\sum_{t=1}^{T}\sum_{m=1}^{M}T_{t,m}^{avg\_task\_offload} \tag{11}$$

其中，T 为一轮实验的时隙数。

L_{avg} 定义为：

$$L_{avg}=\frac{C}{T}\sum_{t=1}^{T}L_{avg}^{local} \tag{12}$$

其中，L_{avg}^{local} 为 rsu_n 和附近 RSU 的负载率平方和的平均值，$L_{avg}^{local}\in[0,1]$，C 为常数。

3. 基于 DDPG 的计算卸载

3.1. 任务依赖分类及合并

如表 1 所示，对车辆生成的各种任务分类，并从 5 个维度对它们进行综合的分析，并给出推荐的卸载位置。任务类型包括 9 种：自动驾驶、车联网、环境感知、车辆控制、安全与隐私、信息娱乐与交互、车辆健康监控、数据后处理与优化和其它后台任务。延迟敏感性从高到低为 L0~L3；计算需求从低到高为 C1~C4；数据量从低到高为 D1~D4；依赖性分无、弱、中、强 4 个级别；优先级从高到低有 3 个级别：P1、P2、P3。推荐卸载位置有本地处理、卸载到 RSU、卸载到云端 3 种。分析任务主要考虑 5 个维度，其中，优先级和延迟敏感性为考虑任务卸载方式的核心因素。

自动驾驶类型任务包括实时路径规划、紧急避障、编队协同控制；车联网类型任务包括紧急广播、交通流优化、充电桩同步；环境感知类型任务包括实时障碍检测、道路特征提取、天气评估；车辆控制类型任务包括底盘控制、能量管理优化、悬挂调节；安全与隐私类型任务包括实时入侵检测、动态密钥管理、隐私数据脱敏；信息娱乐与交互类型任务包括实时导航更新、多媒体、语音交互；车辆健康监控类型任务包括实时故障诊断、长期性能分析、电池健康预测；数据后处理与优化类型任务包括驾驶行为分析、路网优化建模、传感器优化；其他后台类型任务包括软件升级、日志归档、法规审计。

Table 1. Classification of vehicle-generated tasks
表 1. 车辆生成任务分类表

一级任务类型	二级任务分类数量	延迟敏感性	计算需求	数据量	依赖性	优先级	推荐卸载位置
自动驾驶	3	L0、L1	C2、C3	D1、D2	弱	P1	本地、RSU
车联网	3	L2、L3	C1、C2	D1、D2	弱	P2	RSU、云端
环境感知	3	L0、L1	C2、C3	D2、D3	弱	P2	RSU
车辆控制	3	L0、L1	C1、C2	D1	无	P1	本地
安全与隐私	3	L1、L2、L3	C1、C2	D1、D2	强、中	P1	RSU、本地
信息娱乐与交互	3	L2、L3	C1、C2	D2、D3	无	P3	本地、云端
车辆健康监控	3	L1、L2、L3	C2、C3	D2、D3	强、中	P2	云端
数据后处理与优化	3	L3	C3、C4	D4	强	P2	云端
其它后台任务	3	L3	C1、C2	D4	无	P3	云端

图 2 展示了具有依赖关系任务的合并，箭头代表任务之间的依赖关系，边上的权重代表该处理过程的重复次数。车辆 v_m 和 v_{m+1} 上两个应用各生成了一些任务，这些任务有重叠的部分，因此可以通过合并 DAG 图来优化任务之间的执行流程，合并流程如下：

$$H_{t,n,p}=H_{m,A}\cup H_{m,B}\cup H_{m+1,A}\cup H_{m+1,B} \tag{13}$$

其中，$H_{t,n,k}$ 为时隙 t 时，rsu_n 上经过合并留下来的任务群组，$k\in[1,2,\cdots,K]$。0 代表任务之间无依赖关系，1 代表相应过程执行了 1 次，2 也同理。

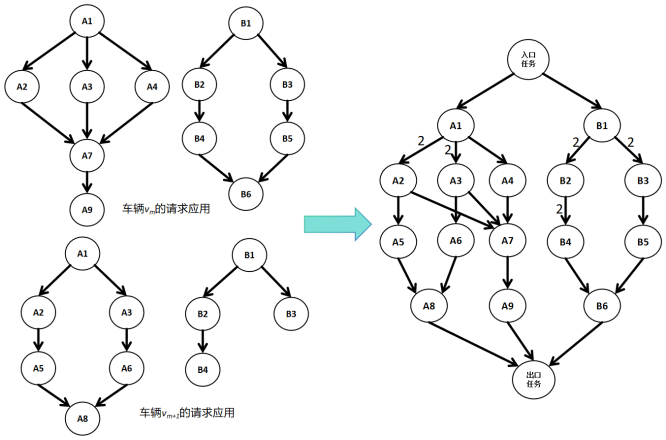

Figure 2. Merge of tasks with dependencies
图 2. 具有依赖关系任务的合并

如表 2 所示，在满足依赖关系的情况下，针对同类型任务，可以根据计算需求和数据量进行任务的合并操作。对于高计算需求的任务，无论数据量大小，它们都需要独占 CPU 资源，因此需要单独处理避免合并任务后被低优先级任务拖慢。对于中等计算需求的任务，当数据量比较大时，应单独处理，避免合并后的数据量过载；当数据量中等时，可以进行合并处理；当数据量小时，应批量合并处理。对于低计算需求的任务，当数据量比较大时，应单独处理；当数据量中等或者比较小时，应批量合并处理。

Table 2. Integration of homogeneous tasks
表 2. 同类型任务合并

计算需求\数据量	数据量大(D4)	数据量中(D3)	数据量小(D1、D2)
计算需求高(C4)	单独处理	单独处理	单独处理
计算需求中(C3)	单独处理	合并	批量合并
计算需求低(C1、C2)	单独处理	批量合并	批量合并

针对循环依赖任务组，设计 2 个机制来处理循环依赖；如果任务组中任务数据量都不是很大，可将具有循环依赖的任务组整体视为一个不可分割的原子任务，内部按初始固定顺序执行，外部调度器将其作为普通任务处理；如果任务组中任务数据量比较大，为循环依赖的任务组设置一个全局超时阈值，超时后强制终止所有关联任务，并触发异常处理，使用历史数据进行处理。

3.2. 负载均衡

3.2.1. RSU 动态权重分配

为了平衡 RSU 之间的负载，RSU 的负载率可以定义为：

$$L_n = \alpha_2 CPU_n + \beta_2 Queue + \gamma_2 Memory + \eta_2 Net \tag{14}$$

其中，$L_n \in [0,1]$，$\alpha_2 + \beta_2 + \gamma_2 + \eta_2 = 1$。$CPU_n$ 代表 rsu_n 当前的 CPU 使用率，$Queue_n$ 为当前队列等待任务大小占最大队列容量的比例，$Memory$ 为内存占用率，Net 为带宽占用率。当获取 RSU 及附近 RSU 的负载情况后，可以给它们动态分配权重来辅助 DDPG 生成对应的卸载动作。具体定义如下：

$$W_{t,n} = \begin{cases} 1.5 & L_n < 0.6 \\ 1 & 0.6 \le L_n < 0.8 \\ 0.3 & L_n \ge 0.8 \end{cases} \tag{15}$$

其中，$W_{t,n}$为时隙 t 时，选择 rsu_n 进行卸载的权重。生成 RSU 节点选择概率分布，公式如下：

$$P_{t,n}=\frac{e^{W_{t,n}}}{\sum e^{W_{t,n}}} \tag{16}$$

其中，$\sum e^{W_{t,n}}$ 为当前 RSU 与附近 RSU 的权重总和。

当 RSU 负载率 L_n 超过 0.9 或附近 RSU 接收不到该 RSU 信息时，Actor 网络会把该 RSU 列入卸载黑名单，该 RSU 通信范围内的车辆任务可以本地处理或者云端处理，直到 L_n 恢复正常或附近 RSU 接收到该 RSU 信息并检测 L_n 正常时，才会把该 RSU 从卸载黑名单去除。

3.2.2. DAG 任务群组排序

在 RSU 队列中，任务群组按照全局优先级指数(Global Priority Index, GPI)进行降序排序，随后顺序处理。定义如下：

$$GPI_{t,n,k}=+\alpha_3 Priority_{t,n,k}+\beta_3 UL_{t,n,k}+\gamma_3 CPL_{t,n,k} \tag{17}$$

其中，$Priority_{t,n,k}$ 为该任务群组的优先级，任务群组都是同一种类任务组成的，因此任务的优先级就是任务群组的优先级。$CPL_{t,n,k}$ 为时隙 t，在 rsu_n 上的 DAG 任务群组 k 的关键路径长度，即 DAG 中最长的路径。$UL_{t,n,k}$ 为任务群组的紧急程度(Urgency Level, UL)，定义如下：

$$UL_{t,n,k}=\frac{1}{T_{t,n,k}^{DAG}-T^{current}+\mu} \tag{18}$$

其中，$T_{t,n,k}^{DAG}$ 为该任务群组的截止时间，$T^{current}$ 为当前的时间，μ 为一个极小值，防止分母为 0。

3.3. 构建马尔可夫模型

本文将一个 RSU 通信范围内每辆车的计算卸载建模视为深度强化学习问题，即每辆车都被视为一个智能体，计算卸载决策可以视为 MDP。在时隙 t，rsu_n 的系统联合状态空间定义为：

$$S_t=\left\{S_{t,1},S_{t,2},\cdots,S_{t,m},\cdots,S_{t,M}\right\} \tag{19}$$

其中，$S_{t,m}=\left\{m,A_{t,m,j},info^{load}\right\}$ 表示车辆 v_m 在时隙 t 的状态，$A_{t,m,j}$ 为车辆上应用的信息，$info_n^{load}$ 为该 rsu_n 及附近 RSU 的负载信息，用以计算 RSU 的负载情况。

$info_n^{load}=\left\{n,\left(x_n,y_n\right),CPU_n,Queue_n,Memory_n,Net_n,info^{neighbour}\right\}$，$n$ 为 RSU 的索引，$\left(x_n,y_n\right)$为该 RSU 的坐标位置，CPU 为当前 CPU 的使用率，$Queue$ 为当前队列等待任务数占最大队列容量的比例，$Memory_n$ 为内存占用率，Net_n 为带宽占用率。$info^{neighbour}=\left\{g,\left(x_g,y_g\right),CPU_g,Queue_g,Memory_g,Net_g\right\}$，其中 g 为附近 RSU 的索引，后面信息是 RSU 的基本信息。

在时隙 t，rsu_n 做出的计算卸载决策可以定义为：

$$A_t=\left\{a_{t,1},a_{t,2},\cdots,a_{t,k},\cdots,a_{t,K}\right\} \tag{20}$$

其中，$a_{t,k}=\left\{k,B_{t,k}^{DAG},x_{t,k}\right\}$ 为 RSU 对 DAG 任务群组的卸载决策，k 为任务群组的索引，$B_{t,k}^{DAG}$ 记录了该任务群组中的全部任务 $b_{t,m,j,z}$。$x_{t,k}=\{0,n,N+1\}$ 表示对 DAG 任务群组的卸载决策。0 为本地处理；n 为卸载到当前 RSU 或者临近 RSU；$N+1$为卸载到云端。当 RSU 对 DAG 任务群组进行了卸载决策，那么内部任务的卸载决策也都如此。

在时隙 t，当系统做出卸载决策后，会立即获得即时奖励和下一个状态。即时奖励可以评价当前时隙所做出的卸载决策的好坏。本文的目标是通过探索最优的卸载策略，来最小化卸载任务的平均响应时间和平均 RSU 负载率。因此，在时隙 t 的即时奖励定义为：

$$r_t = -\frac{\alpha_1}{M}\sum_{m=1}^{M} T_{t,m}^{avg\_task\_offload} - C\beta_1 L_{avg}^{local} \tag{21}$$

3.4. 训练过程

TDLB-DDPG 的计算卸载框架包括 Actor 网络、Actor 的目标网络、Critic 网络和 Critic 的目标网络，如图 3 所示。

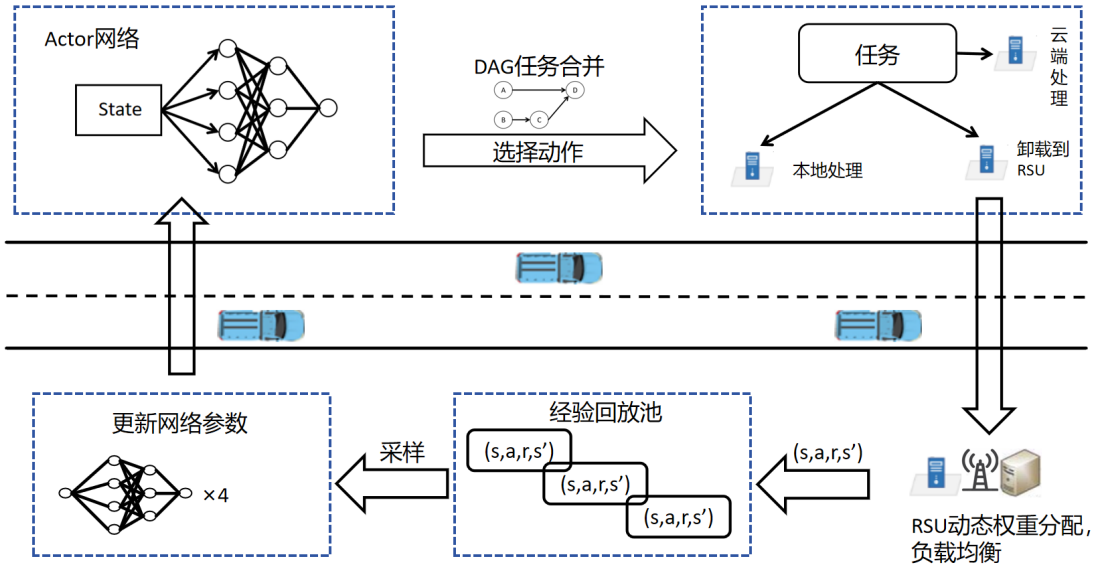

Figure 3. Framework of TDLB-DDPG task offloading scheme
图 3. TDLB-DDPG 计算卸载框架

算法 1：TDLB-DDPG 计算卸载策略

输入： VEC 网络

输出： 卸载策略

1：初始化 Actor 网络的参数 θ^{μ}，Critic 网络的参数 θ^{Q}

2：初始化 Actor 目标网络和 Critic 目标网络的参数：$\theta^{\mu'} \leftarrow \theta^{\mu}$，$\theta^{Q'} \leftarrow \theta^{Q}$

3：初始化经验池 R

4：**for** episode = 1: M **do**

5：　初始化环境、随机噪声 N

6：　获得初始状态 s_t

7：　**for** t = 1: T **do**

8：　任务分类并拒绝那些需要本地处理或云端处理的任务

9：　　DAG 任务群组合并

10：　动态权重分配，获取当前 RSU 及附近 RSU 的节点选择概率

11：　$a_t = \mu\left(s_t \middle| \theta^{\mu}\right) + N_t$

12：　if $a_{t,k} = n$

　　按照节点选择概率生成卸载到哪个 RSU 的动作值，修改 a_t

续表

13: 执行动作 a_t，RSU 按照全局优先级指数将队列中任务降序排序后处理，得到奖励 r_t 和新的环境状态 s_{t+1}

14: 将 $\{s_t, a_t, r_t, s_{t+1}\}$ 存入经验池 R

15: 从 R 中随机采样 n 条经验 (s_i, a_i, r_i, s_{i+1})

16: 计算 TD 目标值：

$$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}|\theta^{\mu'}\right)|\theta^{Q'}\right)$$

17: 最小化损失函数 L 以更新 Critic 网络：

$$\text{Loss} = \frac{1}{N}\sum_i\left(y_i - Q\left(s_i, a_i|\theta^Q\right)\right)^2$$

18: 采样策略梯度更新 Actor 策略网络：

$$\nabla_{\theta^\mu} J\left(\theta^\mu\right) \cong \frac{1}{N}\sum_i \nabla_{\theta^\mu}\mu\left(s_i|\theta^\mu\right)\nabla_a Q\left(s_i, a|\theta^Q\right)\Big|_{a=\mu(s_i)}$$

19: 更新目标网络：

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

20: **end for**

21: **end for**

TDLB-DDPG 算法训练过程的伪代码如算法 1 所示。首先初始化 4 个网络的参数以及经验池 R (第 1~3 行)。在每个 episode 开始时，初始化环境和随机噪声 N 获得初始状态 s_t (第 4~6 行)。对于每个时隙，先根据初始状态 s_t，对任务进行分类并拒绝那些需要在本地处理的任务的卸载请求(第 7~8 行)；对 DAG 任务群组进行合并操作(第 9 行)；进行动态权重分配，得到当前 RSU 及附近 RSU 的节点卸载选择概率(第 10 行)；Actor 网络生成对应的动作 a_t (第 11 行)；对于那些值为 n 的卸载动作，按照卸载节点选择概率重新生成卸载动作值，修改动作 a_t (第 12 行)。最后，执行动作 a_t，RSU 按照全局优先级指数将队列中任务降序排序后处理，获得奖励和新的状态环境，并将这些数据存入 R 中(第 13~14 行)。在每个时隙的迭代中，从 R 中随机挑选一定数量的数据来更新网络(第 15~19 行)。首先计算 TD (Temporal Difference)目标值 y_i，通过 Actor 的目标网络及下一个状态可以得到下一个状态采取的动作 $\mu'\left(s_{i+1}|\theta^{\mu'}\right)$；而通过 Critic 的目标网络以及下一个状态和下一个状态采取的动作，可以得到下一个状态的预测 Q 值 $Q'\left(s_{i+1}, \mu'\left(s_{i+1}|\theta^{\mu'}\right)|\theta^{Q'}\right)$，再加上即时奖励，可以得到 TD 目标值 y_i (第 16 行)。然后，利用 y_i 最小化损失函数 L 以更新 Critic 网络，损失函数中的 $Q\left(s_i, a_i|\theta^Q\right)$ 由 Critic 网络生成(第 17 行)。之后，利用策略梯度来更新 Actor 策略网络(第 18 行)。最后，更新 2 个目标网络(第 19 行)。

4. 实验结果与分析

4.1. 实验环境

在 64 位 Windows 11 操作系统上进行仿真实验，处理器为 13th Gen Intel(R) Core(TM) i7-13650HX 2.60 GHz，显卡为 NVIDIA GeForce RTX 4060，软件环境为 Python 3.8 和 PyTorch 2.21。从 OpenStreetMap 下载模拟地图[22]，并通过 SUMO 模拟生成交通流[23]。地图范围为 1700 m × 1900 m，包含 8 条主要道路。40 个 RSU 平均分布在每条主干道路上。实验参数如表 3 所示。

Table 3. Experimental parameters
表 3. 实验参数

参数	值
地图大小	1700 m × 1900 m
RSU 数量	40
RSU 通信半径	200 m
汽车数量	600~1000
回合数	2000
车速	0~23 m/s
应用数量	5~10
每秒所有应用生成任务数	8~20
任务大小	1 MB~2 MB
时间步长度	1 s
RSU 处理能力	2.5 GHz
汽车处理能力	0.5 GHz
最大容忍响应时间	3 s, 3.3 s, 3.6 s, 3.9 s, 4.2 s, 4.5 s
折扣因子	0.99
学习率	1e-4
批量大小	64

4.2. 基线算法

为了验证所提出算法的有效性，设计了以下几种策略作为基准。

1) 本地处理(Local Process) [24]：应用程序产生的所有任务都在车辆的处理器处理。

2) 随机卸载策略(Random Offload)：在每个时隙 t，车辆将应用程序产生的任务生成随机请求发送给 RSU，RSU 将接收到的任务按照到达时间排序。最后，RSU 将可以卸载的任务信息发送给车辆，随后车辆启动卸载。

3) 基于 DQN 的卸载策略[25]：在每个时隙 t，车辆将应用程序产生的任务利用 DQN 生成卸载策略，之后的处理流程与随机卸载策略相同。

4) 基于 DDPG 的卸载策略[26]：该卸载策略的流程类似于基于 DQN 的卸载策略。

4.3. 结果分析

4.3.1. 收敛性能对比

图 4 显示了三种深度强化学习方法训练过程中的总奖励。可以看到，随着训练轮数的增加，三种算法的总奖励都呈上升趋势，强化学习方法可以通过不断迭代来积累经验获得策略。DQN 约在 600 轮处开始收敛，而 DDPG 约在 1000 轮处开始收敛，由于计算卸载的动作空间是离散的，更适合于 DQN 处理，因此收敛速度更快一些。然而，由于 DQN 存在高估问题，因此其震荡幅度比 DDPG 大。所提出的 TDLB-DDPG 大概在 700 轮开始收敛，收敛速度比较快，而奖励最大。这是因为 TDLB-DDPG 旨在为复杂多变的城市交通场景提供可靠的计算卸载支持，关注整个系统的综合奖励包括卸载任务的平均响应时间和 RSU 平均负载率，而不仅仅只考虑卸载任务的平均响应时间。此外，任务的分类使得 TDLB-DDPG 具有

相比基线方法更小的动作空间。

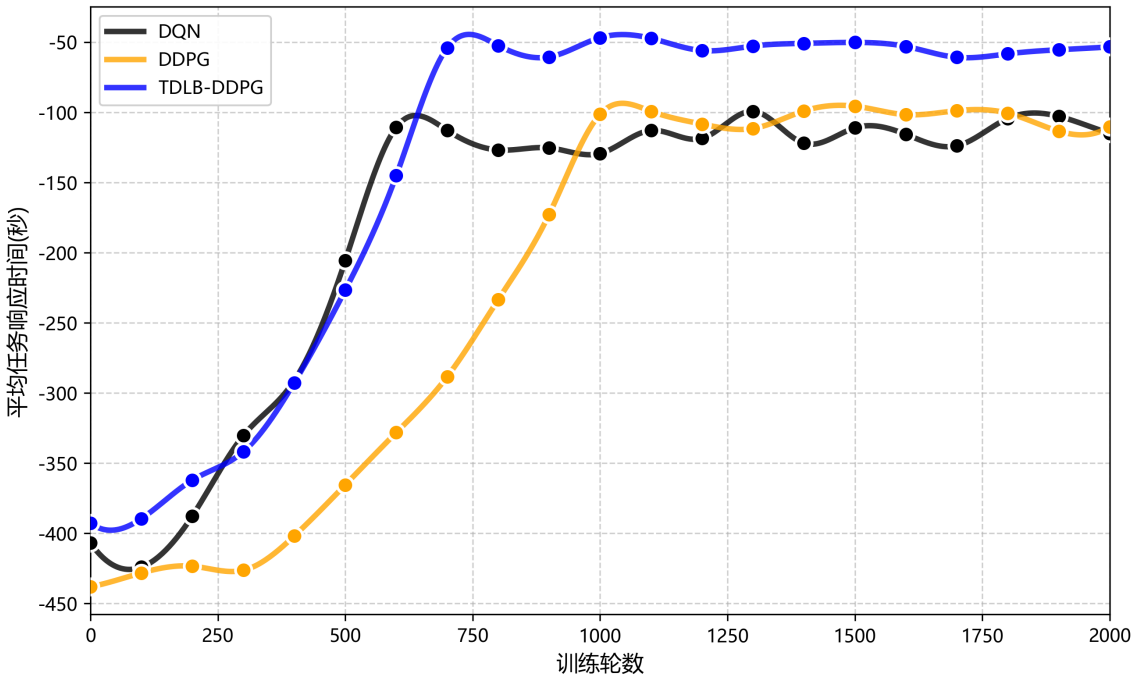

Figure 4. Total rewards for various algorithms
图 4. 各种算法的总奖励

4.3.2. 任务卸载平均响应时间对比

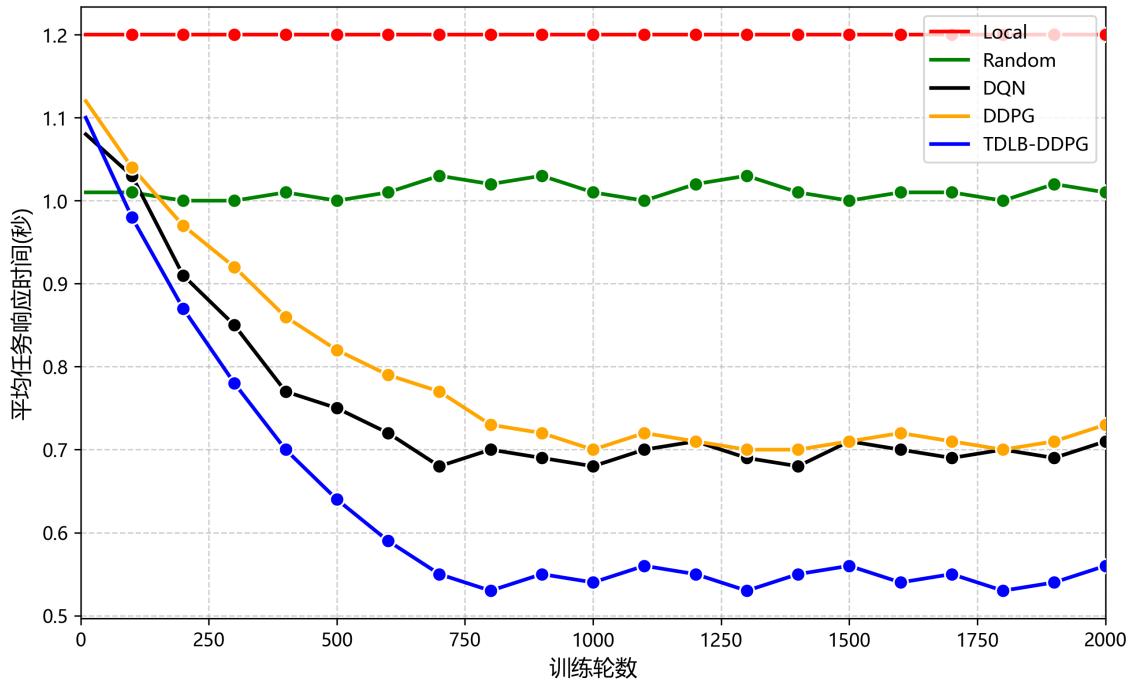

Figure 5. Comparison of average response time for task offloading across different algorithms
图 5. 各算法的卸载任务平均响应时间对比

图 5 给出了各种算法的卸载任务平均响应时间。对于本地处理策略，由于车辆的计算能力低，任务响应时间最长。随机卸载的性能优于本地处理，但由于没有任何优化策略，其平均任务响应时间类似于本地处理策略，基本保持不变。对于三种深度强化学习卸载策略，TDLB-DDPG 的性能超出 DQN 和 DDPG，算法收敛后卸载任务平均响应时间分别降低了约 21.4%和 22.5%，相比本地处理和随机卸载则分别降低了 54.1%和 45%。首先，对卸载任务进行分类，避免原本应卸载到云端的大数据量、延迟敏感性低的任务卸载到 RSU，以及避免其他种类任务长时间的等待和高负载，有效提高了 RSU 计算资源的利用率。其次，对相同种类具有依赖关系的任务进行合并，并可以对这些任务进行批量处理，有效减少卸载任务的平均响应时间。

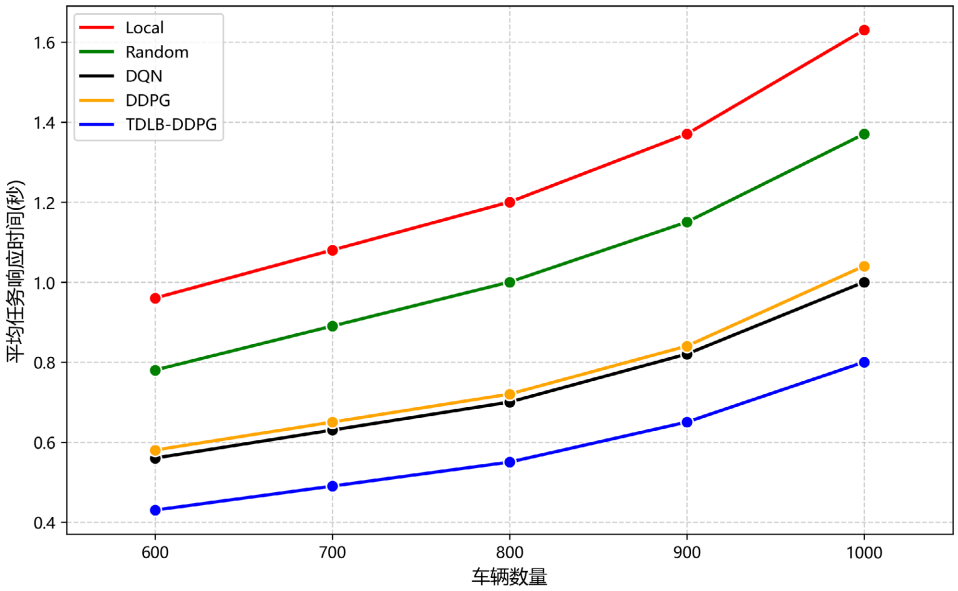

Figure 6. Comparison of average response time for task offloading under various numbers of vehicles
图 6. 不同车辆数量下的卸载任务平均响应时间对比

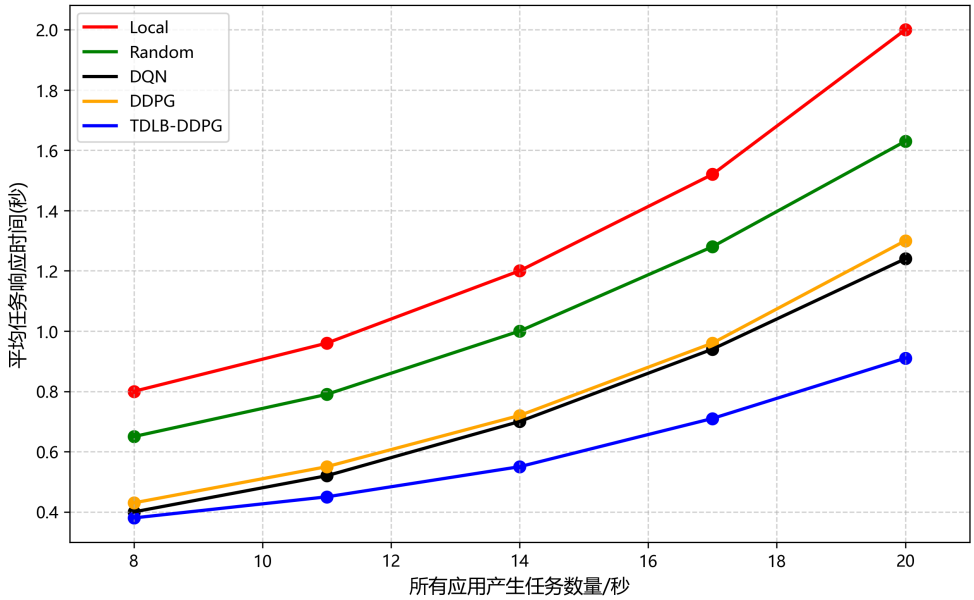

Figure 7. Comparison of average response time for task offloading under various numbers of tasks
图 7. 不同任务数量下的卸载任务平均响应时间对比

图 6 和图 7 分别给出了各种算法在不同车辆和任务数量下的卸载任务平均响应时间对比。随着车辆或任务的增加，所有卸载方案的平均响应时间都会增加，但所提出的 TDLB-DDPG 的卸载任务平均响应时间最低，优于其他 4 种方案；并且随着车辆和任务数量的增加，TDLB-DDPG 曲线的增长幅度低于其他 4 种方案。首先，基于 DAG 的依赖关系任务的合并使得许多相似的任务可以进行批量处理，任务越多，节省的时间就越多。其次，在任务大幅增加的情况下，任务的处理顺序对平均响应时间至关重要。

4.3.3. RSU 平均负载率对比

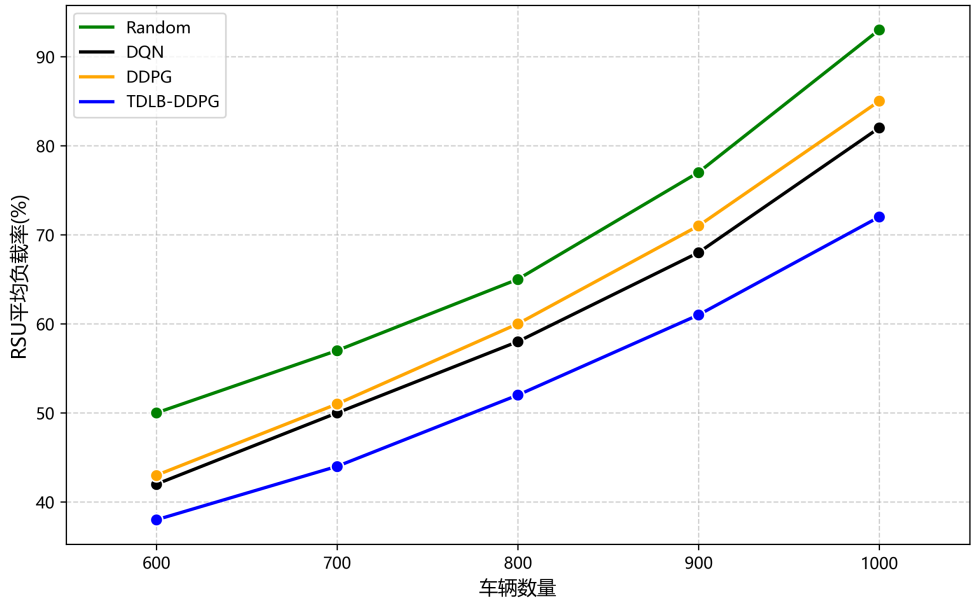

Figure 8. Comparison of the average load rate of RSUs under various numbers of vehicles
图 8. 不同车辆数量下的 RSU 平均负载率对比

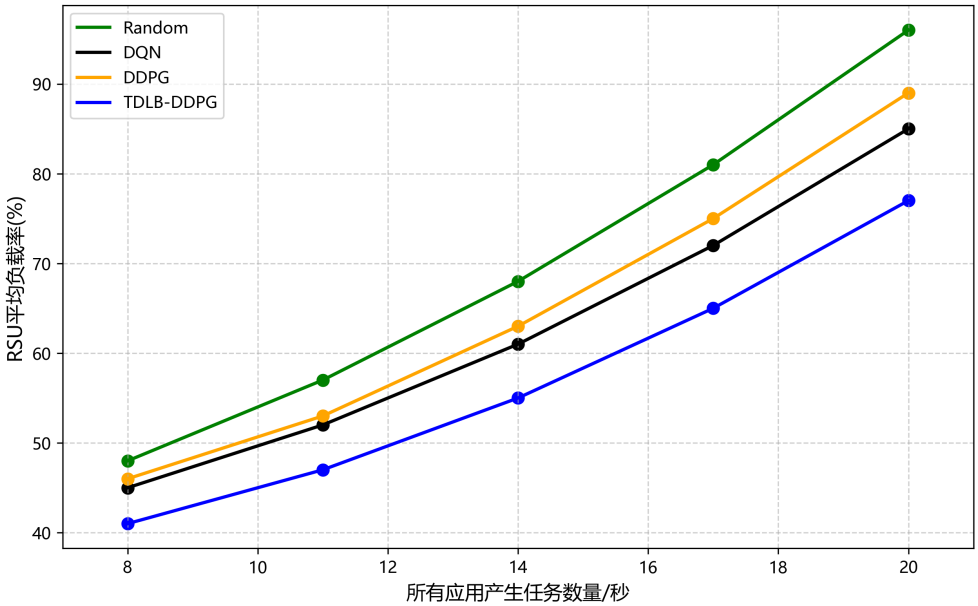

Figure 9. Comparison of the average load rate of RSUs under various numbers of tasks
图 9. 不同任务数量下的 RSU 平均负载率对比

图 8 和图 9 分别给出了各种算法在不同车辆和任务数量下的 RSU 平均负载率对比。随着车辆或任务的增加，所有卸载方案的 RSU 平均负载率都会增加，但所提出的 TDLB-DDPG 的 RSU 平均负载率最低，优于其他 4 种方案。TDLB-DDPG 会根据当前及周围 RSU 的负载情况，动态进行权重分配，使得任务会优先卸载到负载率低的 RSU 上，显著降低了 RSU 的平均负载率，缓解了 RSU 负载不均衡的情况。

4.3.4. 任务处理成功率

图 10 给出了各个算法在不同最大容忍响应时间下任务处理成功率的对比。随着最大容忍响应时间的增加，所有算法的任务处理成功率显著增加；且 TDLB-DDPG 与其他算法之间的差异也是越来越小。最大容忍响应时间与任务的紧迫性有关，说明在面临延迟敏感任务时，TDLB-DDPG 具有更好的表现。DAG 任务群组排序使得 RSU 优先处理紧迫的任务，提高了任务卸载的成功率，间接提高任务处理的成功率。

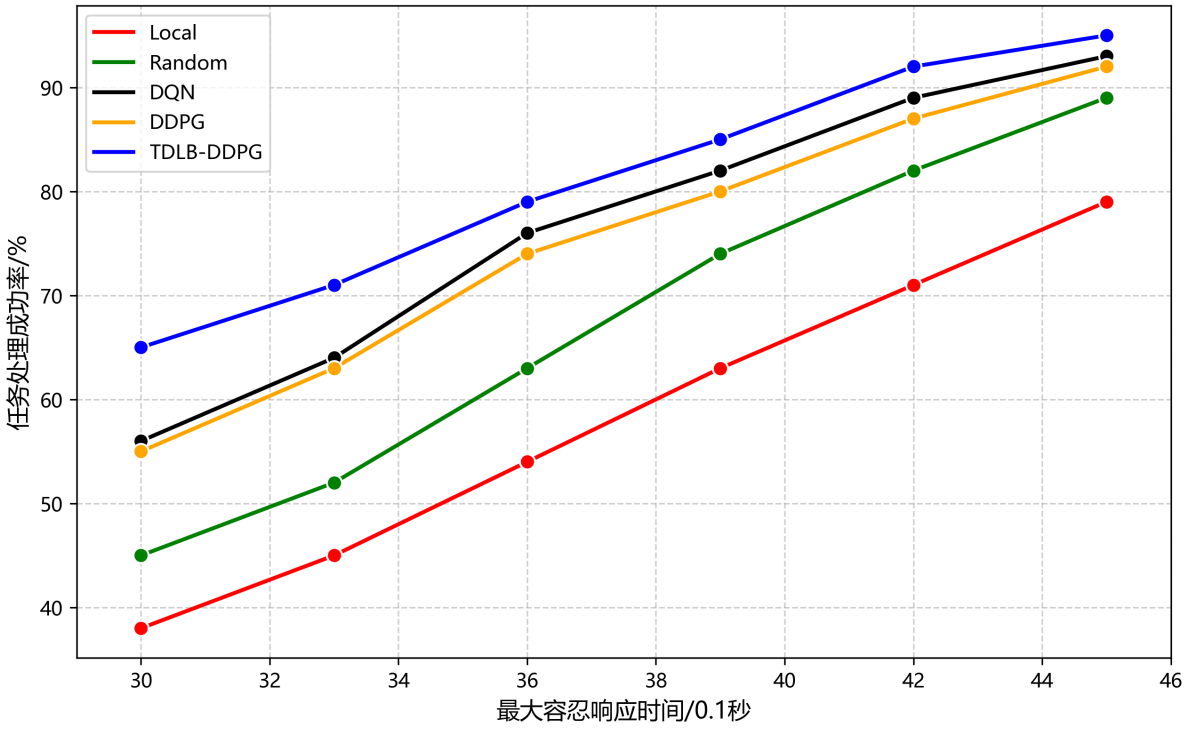

Figure 10. Comparison of task processing success rates under different maximum tolerant response time
图 10. 不同最大容忍响应时间下的任务处理成功率对比

5. 结束语

本文深入研究了车辆边缘计算中的计算卸载问题，构建了一个基于 VEC 的计算卸载模型，提出了一种基于任务依赖和负载均衡的计算卸载方案。任务依赖合并减小了 DDPG 的动作空间，显著降低了卸载任务的平均响应时间；而 RSU 动态权重分配显著降低了 RSU 的平均负载率。这些特性使得 TDLB-DDPG 相比其他卸载方案更符合应用需求。

在未来，我们可以针对交通流做宏观预测，并根据预测结果预留资源，动态划分 RSU 服务区域内的优先级区域，形成进一步的负载均衡。此外，系统的能源消耗也是一个值得研究的问题。

参考文献

[1] Zheng, D., Wang, L., Kai, C. and Peng, M. (2023) Resource Optimization for Task Offloading with Real-Time Location

Prediction in Pedestrian-Vehicle Interaction Scenarios. *IEEE Transactions on Wireless Communications*, **22**, 7331-7344. https://doi.org/10.1109/twc.2023.3250254

[2] Zabihi, Z., Eftekhari Moghadam, A.M. and Rezvani, M.H. (2023) Reinforcement Learning Methods for Computation Offloading: A Systematic Review. *ACM Computing Surveys*, **56**, 1-41. https://doi.org/10.1145/3603703

[3] Mao, M., Hu, T. and Zhao, W. (2023) Reliable Task Offloading Mechanism Based on Trusted Roadside Unit Service for Internet of Vehicles. *Ad Hoc Networks*, **139**, Article 103045. https://doi.org/10.1016/j.adhoc.2022.103045

[4] Liu, L., Chen, C., Pei, Q., Maharjan, S. and Zhang, Y. (2020) Vehicular Edge Computing and Networking: A Survey. *Mobile Networks and Applications*, **26**, 1145-1168. https://doi.org/10.1007/s11036-020-01624-1

[5] Huang, Z., Chen, Y. and Zhang, Y. (2024) Lyapunov-Guided Deep Reinforcement Learning for Vehicle Task Stable Offloading. 2024 27*th International Conference on Computer Supported Cooperative Work in Design* (*CSCWD*), Tianjin, 8-10 May 2024, 833-838. https://doi.org/10.1109/cscwd61410.2024.10580013

[6] Huang, B., Zhou, Y., Zhang, X., Chen, J. and Shang, L. (2024) Computation Offloading and Resource Allocation for Vehicle-Assisted Edge Computing Networks with Joint Access and Backhaul. *IEEE Access*, **12**, 110248-110259. https://doi.org/10.1109/access.2024.3440000

[7] Zhang, X., Wu, W., Zhao, Z., Wang, J. and Liu, S. (2023) RMDDQN-Learning: Computation Offloading Algorithm Based on Dynamic Adaptive Multi-Objective Reinforcement Learning in Internet of Vehicles. *IEEE Transactions on Vehicular Technology*, **72**, 11374-11388. https://doi.org/10.1109/tvt.2023.3270967

[8] Bi, X., Shi, J., Zhang, B., Lyu, Z. and Huang, L. (2023) An RSU-Crossed Dependent Task Offloading Scheme for Vehicular Edge Computing Based on Deep Reinforcement Learning. *International Journal of Sensor Networks*, **41**, 244-256. https://doi.org/10.1504/ijsnet.2023.130711

[9] Materwala, H., Ismail, L. and Hassanein, H.S. (2023) QoS-SLA-Aware Adaptive Genetic Algorithm for Multi-Request Offloading in Integrated Edge-Cloud Computing in Internet of Vehicles. *Vehicular Communications*, **43**, Article 100654. https://doi.org/10.1016/j.vehcom.2023.100654

[10] Liu, J., Wang, Y., Pan, D. and Yuan, D. (2024) QoS-Aware Task Offloading and Resource Allocation Optimization in Vehicular Edge Computing Networks via MADDPG. *Computer Networks*, **242**, Article 110282. https://doi.org/10.1016/j.comnet.2024.110282

[11] Li, W., Sun, X., Wan, B., Liu, H., Fang, J. and Wen, Z. (2023) A Hybrid GA-PSO Strategy for Computing Task Offloading Towards MES Scenarios. *PeerJ Computer Science*, **9**, e1273. https://doi.org/10.7717/peerj-cs.1273

[12] Zhou, W., Chen, L., Tang, S., Lai, L., Xia, J., Zhou, F., *et al*. (2021) Offloading Strategy with PSO for Mobile Edge Computing Based on Cache Mechanism. *Cluster Computing*, **25**, 2389-2401. https://doi.org/10.1007/s10586-021-03414-0

[13] Liu, J., Ahmed, M., Mirza, M.A., Khan, W.U., Xu, D., Li, J., *et al*. (2022) RL/DRL Meets Vehicular Task Offloading Using Edge and Vehicular Cloudlet: A Survey. *IEEE Internet of Things Journal*, **9**, 8315-8338. https://doi.org/10.1109/jiot.2022.3155667

[14] Zhu, Z. and Zhao, H. (2022) A Survey of Deep RL and IL for Autonomous Driving Policy Learning. *IEEE Transactions on Intelligent Transportation Systems*, **23**, 14043-14065. https://doi.org/10.1109/tits.2021.3134702

[15] Fofana, N., Letaifa, A.B. and Rachedi, A. (2025) Intelligent Task Offloading in Vehicular Networks: A Deep Reinforcement Learning Perspective. *IEEE Transactions on Vehicular Technology*, **74**, 201-216. https://doi.org/10.1109/tvt.2024.3427814

[16] Walia, G.K. and Kumar, M. (2025) Computational Offloading and Resource Allocation for IoT Applications Using Decision Tree Based Reinforcement Learning. *Ad Hoc Networks*, **170**, Article 103751. https://doi.org/10.1016/j.adhoc.2024.103751

[17] Pang, S., Hou, L., Gui, H., He, X., Wang, T. and Zhao, Y. (2024) Multi-Mobile Vehicles Task Offloading for Vehicle-Edge-Cloud Collaboration: A Dependency-Aware and Deep Reinforcement Learning Approach. *Computer Communications*, **213**, 359-371. https://doi.org/10.1016/j.comcom.2023.11.013

[18] Zhao, L., Zhang, E., Wan, S., Hawbani, A., Al-Dubai, A.Y., Min, G., *et al*. (2024) MESON: A Mobility-Aware Dependent Task Offloading Scheme for Urban Vehicular Edge Computing. *IEEE Transactions on Mobile Computing*, **23**, 4259-4272. https://doi.org/10.1109/tmc.2023.3289611

[19] Zhang, Y., He, X., Xing, J., Li, W. and Seah, W.K.G. (2024) Load-Balanced Offloading of Multiple Task Types for Mobile Edge Computing in IoT. *Internet of Things*, **28**, Article 101385. https://doi.org/10.1016/j.iot.2024.101385

[20] Li, Z., Yu, K., Zhou, H. and Wu, X. (2023) DQN-Based Collaborative Computation Offloading for Edge Load Balancing. 2023 8*th IEEE International Conference on Network Intelligence and Digital Content* (*IC-NIDC*), Beijing, 3-5 November 2023, 1-6. https://doi.org/10.1109/ic-nidc59918.2023.10390728

[21] Xie, J., Zheng, F., Wen, W. and Jia, Y. (2024) Price-Based Task Offloading for Load-Imbalance Vehicular Multi-Access

Edge Computing. 2024 *IEEE* 99*th Vehicular Technology Conference* (*VTC*2024-*Spring*), Singapore, 24-27 June 2024, 1-6. https://doi.org/10.1109/vtc2024-spring62846.2024.10683084

[22] OpenStreetMap. https://openstreetmap.maps.arcgis.com

[23] SUMO (2024) SUMO—Simulation of Urban Mobility. https://eclipse.dev/sumo/

[24] Lu, J., Li, Q., Guo, B., Li, J., Shen, Y., Li, G., *et al*. (2022) A Multi-Task Oriented Framework for Mobile Computation Offloading. *IEEE Transactions on Cloud Computing*, **10**, 187-201. https://doi.org/10.1109/tcc.2019.2952346

[25] 王锦, 张新有. 基于 DQN 的无人驾驶任务卸载策略[J]. 计算机应用研究, 2022, 39(9): 2738-2744.

[26] Wang, Y., Fang, W., Ding, Y. and Xiong, N. (2021) Computation Offloading Optimization for UAV-Assisted Mobile Edge Computing: A Deep Deterministic Policy Gradient Approach. *Wireless Networks*, **27**, 2991-3006. https://doi.org/10.1007/s11276-021-02632-z