高速铁路云边协同计算场景下任务依赖感知卸载方法

胡景天，曹源，孙永奎，王峰
（北京交通大学自动化与智能学院，北京 100044）

摘要：针对高速铁路复杂感知任务具有有向无环图依赖、高速移动和轨旁计算资源动态变化等特征，传统面向独立任务的卸载方法难以实现子任务依赖约束下的高效在线决策，提出移动性与依赖联合感知分层近端策略优化算法。首先，将复杂感知业务抽象为有向无环图任务，建立轨旁多边缘协同计算模型，统一描述输入上传、跨服务器迁移、任务排队和计算过程。在此基础上，将动态卸载问题建模为马尔可夫决策过程，构建拓扑调度和服务器决策分层卸载机制，设计联合动作掩码，通过改进的近端策略优化方法学习高速移动环境下的在线卸载策略。仿真结果表明，与对比算法相比，所提方法在不同计算负载、边缘计算资源和列车运行速度场景下均能获得较低的平均端到端时延和较高的系统计算效率，具有较好的稳定性和高速移动适应能力。

关键词：高速铁路；边缘计算；依赖感知；任务卸载；近端策略优化
中图分类号：U28
文献标志码：A

Dependency-Aware Task Offloading for Cloud-Edge Collaborative Computing in High-Speed Railways

HU Jingtian, CAO Yuan, SUN Yongkui, WANG Feng
School of Automation and Intelligence, Beijing Jiaotong University, Beijing 100044, China

Abstract: To address the characteristics of complex perception tasks in high-speed railway scenarios, including directed acyclic graph (DAG) dependencies, high-speed mobility, and dynamically varying trackside computing resources, a mobility- and dependency-aware hierarchical proximal policy optimization algorithm (MDH-PPO) was proposed, since conventional offloading methods designed for independent tasks cannot efficiently support online decision-making under subtask dependency constraints. First, complex perception services were modeled as DAG tasks, and a collaborative multi-edge computing model was established to characterize input uploading, inter-server migration, task queueing, and computation. The dynamic offloading problem was then formulated as a Markov decision process. A hierarchical offloading mechanism comprising topology scheduling and server decision-making was developed, together with a joint action mask. An enhanced proximal policy optimization method was further employed to learn an online offloading policy in high-mobility environments. In the simulations, lower average end-to-end latency and higher system computational efficiency were achieved with MDH-PPO than with the benchmark algorithms under varying computing loads, edge computing capacities, and train operating speeds. These results demonstrate the algorithm's stable training performance and strong adaptability to high-speed mobility.

Key words: high-speed railway, edge computing, dependency awareness, task offloading, proximal policy optimization

收稿日期：2026–08–16；**修回日期：**2026–09–01
通信作者：曹源，ycao@bjtu.edu.cn
基金项目：国家自然科学基金资助项目（No.U2368202, No.U2468203）
Foundation Items：The National Natural Science Foundation of China (No.U2368202, No.U2468203)

0 引言

截至2025年底，我国高速铁路营业里程已达5.0万km，形成了全球规模最大的高速铁路网，商业运营速度处于世界领先水平[1-2]。随着高速铁路建设由基础设施规模扩张逐步转向安全、高效和智能化运营，铁路行业正在加快推进智能铁路典型应用场景建设[3-5]。环境感知、状态监测、目标识别和辅助决策等智能业务需要持续处理摄像头、激光雷达、毫米波雷达及运行监测设备产生的多源异构数据[6-8]。此类任务通常具有数据规模大、计算负载高和时延约束严格等特点，单纯依赖车载计算资源难以在高业务负载条件下持续满足实时处理需求。

铁路新一代移动通信系统5G-R与边缘计算等技术的发展，为高速铁路构建感知、通信与计算融合的智能运行体系提供了技术基础[9]。移动边缘计算通过将计算、存储和网络能力部署至接入网络边缘，可缩短数据源与计算资源之间的距离，降低远程云计算带来的传输开销[10]。在高速铁路场景中，可沿线路部署轨旁边缘服务器，将列车产生的计算密集型任务卸载至邻近边缘节点执行，从而缓解车载计算压力并降低业务处理时延[11]。然而，高速列车持续运动使其接入服务器、无线链路状态及可用边缘资源不断变化，任务执行期间还可能经历覆盖切换和跨服务器数据传输，使卸载决策具有显著的时空动态性。

针对高速移动环境下的边缘任务卸载，已有学者从任务路由、移动预测、切换管理和边缘协同等多方面开展研究。文献[12]针对高速铁路多接入边缘计算网络研究任务路由与切换问题，通过联合优化提高网络任务处理能力；文献[13]进一步利用列车移动预测进行动态任务卸载与迁移，以适应高速移动引起的接入变化；文献[14]面向车载边缘计算设计分布式任务卸载机制，根据车辆及边缘节点状态动态调整卸载决策。实际上铁路复杂感知业务通常由数据预处理、特征提取、目标检测、信息融合和决策输出等多个处理环节组成，各处理环节之间存在明确的前驱与后继关系。上述研究通常将业务请求抽象为相互独立的任务或可分割计算负载，不能充分描述复杂感知任务内部不同处理环节之间的执行依赖关系。

有向无环图（directed acyclic graph, DAG）被广泛用于描述子任务间的计算依赖和数据传递关系。部分研究开始进一步考虑移动性与任务依赖之间的耦合关系。文献[15]研究图任务卸载与计算资源联合分配问题。文献[16]提出依赖感知任务卸载与服务缓存方法（dependency-aware task offloading and service caching, DTOSC），通过联合优化依赖任务卸载和服务缓存提升车载边缘计算效率；文献[17]提出可信依赖任务卸载方法（dependency-aware trustworthy task offloading, DeTTO），在子任务依赖关系基础上进一步考虑候选节点可信度；文献[18]联合考虑移动用户的依赖任务卸载与网络带宽分配，通过移动状态辅助动态卸载决策；文献[19]针对车载边缘计算研究DAG任务的在线卸载与移动感知调度。这些研究主要面向通用移动边缘计算或城市车联网场景，高速铁路场景下列车移动、边缘选择、DAG拓扑和依赖数据传输之间形成了更加紧密的时空耦合关系，现有方法难以直接适用于该类场景。

随着子任务规模和候选边缘服务器数量增加，卸载动作空间快速扩张，传统基于确定性先验信息的优化方法难以满足在线决策需求。深度强化学习能够通过持续交互学习动态环境下的长期卸载策略，已逐渐应用于复杂依赖任务的在线调度。文献[20]利用DAG描述依赖任务，并通过深度强化学习实现多边缘节点间的子任务卸载；文献[21]将任务依赖与网络流调度联合建模，并设计依赖感知强化学习方法；文献[22]利用图神经网络提取DAG拓扑特征，实现多用户多边缘环境下的依赖任务卸载；文献[23]提出准牛顿深度强化学习两阶段在线卸载方法（quasi-Newton deep reinforcement learning-based two-stage online offloading, QNRLO），利用批归一化深度网络和准牛顿优化生成在线卸载决策。文献[24]将Transformer与近端策略优化（proximal policy optimization, PPO）结合，通过限制策略更新幅度提高策略梯度训练的稳定性，适合处理此类动态序贯决策问题。然而，现有学习方法通常以任务完成时延、能耗或边缘负载作为主要决策依据，对高速铁路场景中特有的耦合关系考虑不足，未能显式约束DAG执行条件，可能产生前驱任务尚未完成而后继任务被提前调度等不可行动作，从而增加策略学习难度。

综上，现有研究已分别从高速移动环境下的边

缘任务卸载、依赖任务调度和强化学习在线决策等方面开展了较为深入的研究，但针对高速铁路场景，同时考虑列车移动与多边缘协同，以及任务之间拓扑依赖关系的在线卸载研究仍相对不足。为此，本文利用节点描述逻辑子任务的计算需求，利用有向边刻画子任务间的执行约束及中间结果迁移关系，建立高速铁路轨旁多边缘协同计算模型。在此基础上，将DAG卸载转化为拓扑约束下的序贯服务器选择过程，根据当前子任务的前驱映射、服务器负载、通信状态和列车移动状态构造候选动作特征，并通过近端策略优化学习在线卸载策略，以降低复杂感知任务的端到端完成时延。

本文的主要贡献如下：

（1）建立具有计算与数据依赖关系的DAG任务模型，将复杂感知业务由传统独立任务建模扩展为具有明确执行约束的子任务图，从而描述多阶段感知处理过程中的并行、汇聚及前后依赖关系。

（2）构建考虑高速列车连续移动与轨旁多边缘协同的任务计算模型，统一刻画列车至源服务器、边缘服务器间依赖数据传输以及任务结果返回列车当前接入服务器所产生的通信与计算开销。

（3）提出移动性与依赖联合感知分层近端策略优化算法。上层拓扑调度模块按照DAG依赖关系确定当前待卸载子任务集合，下层策略网络从候选边缘服务器中选择其执行位置；进一步将根节点输入传输、前驱结果迁移和预计计算时延构造为物理代价先验，并结合策略引导束搜索和映射安全修正实现完整DAG的在线卸载。

1　系统模型与问题建模

本文考虑云边协同三层架构下的高速铁路双向线路运行场景，如图1所示，该架构包括列车层、边缘层和云中心层。

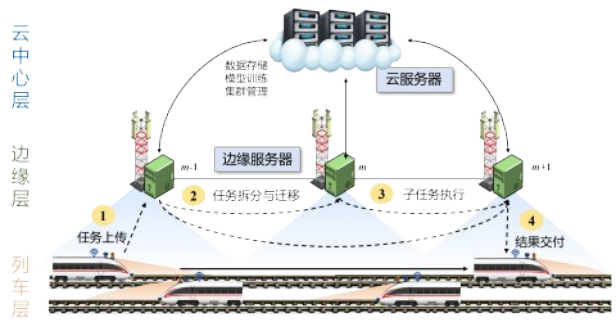

图1　高速铁路云边协同架构

1）列车层：高速列车搭载相机、激光雷达、毫米波雷达及运行状态监测设备，持续产生目标识别、环境感知和定位导航等智能业务。各业务由多个具有计算与数据依赖关系的子任务构成，列车将采集的感知数据上传至当前关联的轨旁边缘服务器进行处理。

2）边缘层：沿铁路线路部署多个具备计算、存储和通信能力的轨旁边缘服务器，并通过轨旁有线网络互联，为列车提供近端计算和任务卸载服务。列车根据当前位置动态关联相应的轨旁边缘服务器，并可利用沿线多个边缘节点的计算资源进行协同处理。

3）云中心层：云中心部署于远端数据中心，主要承担全局管理和非实时计算任务，为边缘层提供模型、策略及全局管理支持。本文研究主要针对具有严格时延要求的实时感知任务，云中心不作为此类任务的候选执行节点，因此本文后续卸载决策主要针对轨旁多边缘服务器展开。高速铁路云边协同计算模型相关符号定义如表1所示。

表1　符号说明

符号	含义
□, M	轨旁边缘服务器集合及服务器数量
□, N	运行列车集合及列车数量
□(t)	调度周期 t 内到达的铁路感知请求集合
$G_p=\left(\mathcal{T}_p,\mathcal{E}_p\right)$	请求 p 的 DAG 任务图
$T_{p,i}$	请求 p 的第 i 个逻辑子任务
$\delta_{p,j,k}$	请求 p 中子任务 $T_{p,k}$ 对 $T_{p,j}$ 的依赖指示变量
$x_{p,i}^{m}$	子任务 $T_{p,i}$ 是否映射到边缘服务器 m
$f_{p,i}^{m}(t)$	边缘服务器 m 分配给子任务 $T_{p,i}$ 的计算资源
$R_{mn}(t)$	边缘服务器 m 与 n 之间的等效传输速率
T_p,D_p	请求 p 的完成时延和最大容忍时延

轨旁边缘服务器集合为 $\mathcal{M}=\{1,2,\cdots,M\}$，运行列车集合为 $\mathcal{R}=\{1,2,\cdots,N\}$。一个调度周期内到达系统的铁路感知任务集合记为 $\mathcal{P}(t)$。对于任意任务 $p\in\mathcal{P}(t)$ 可表示为：

$$p=\left(s_r^{src},s_r^{dst},a_p,D_p\right),p\in P(t)\#(1)$$

其中，任务到达时列车关联的服务器记为 s_r^{src}，称为源服务器。源服务器负责接收列车上传的感知数据、获取当前系统状态并协调轨旁边缘节点执行逻

辑子任务。由于列车在任务执行期间持续移动，任务完成时列车可能已经切换至另一个轨旁服务器。将结果交付时列车关联的服务器记为 s_r^{dst}，称为结果交付服务器。当 $s_r^{\mathrm{src}} \neq s_r^{\mathrm{dst}}$ 时，最终结果需要通过轨旁通信网络转发至结果交付服务器，再下发给列车；a_p 和 D_p 分别表示任务到达时刻和最大容忍时延。

1.1　任务模型

本文主要考虑由多个相互关联处理环节构成的复杂铁路感知任务，其子任务之间存在明确的执行顺序和数据依赖关系[25]。典型任务结构如图 2 所示。

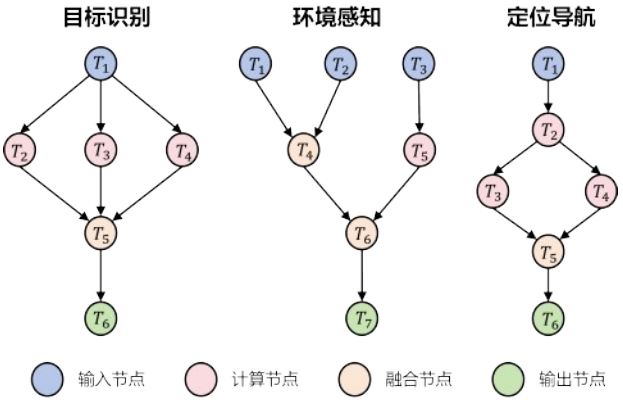

图2　高铁典型感知任务DAG

不同感知任务的处理环节和分支结构可能不同，但均包含明确的子任务执行顺序和中间结果传递关系。因此，将单个铁路感知任务 p 抽象为 DAG：

$$G_p = \left(\mathcal{T}_p, \mathcal{E}_p\right) \#(2)$$

其中 $\mathcal{T}_p = \{T_{p,1}, T_{p,2}, \cdots, T_{p,K_p}\}$ 表示逻辑子任务集合，$\mathcal{E}_p$ 表示有向依赖边集合。

每个逻辑子任务 $T_{p,k}$ 可表示为：

$$T_{p,k} = \left(c_{p,k}, d_{p,k}^{\mathrm{in}}, d_{p,k}^{\mathrm{out}}\right) \#(3)$$

其中，$c_{p,k}$ 表示逻辑子任务 $T_{p,k}$ 的计算工作量，$d_{p,k}^{\mathrm{in}}$ 和 $d_{p,k}^{\mathrm{out}}$ 分别表示其输入数据量和输出数据量。对于依赖边 $(T_{p,j}, T_{p,k})$，前驱节点向后继节点传递的中间结果数据量记为 $d_{p,j,k}$。

若有向边 $(T_{p,j}, T_{p,k}) \in E_p$，则逻辑子任务 $T_{p,k}$ 依赖逻辑子任务 $T_{p,j}$ 的输出，即 $T_{p,j}$ 为 $T_{p,k}$ 的前驱节点。只有当前驱节点完成计算且其中间结果到达后继节点的执行服务器后，后继节点才能开始执行。定义二进制依赖指示变量 $\delta_{p,j,k}$，其中第一个节点下标表示前驱节点，第二个节点下标表示后继节点，即：

$$\delta_{p,j,k} = \begin{cases} 1, \left(T_{p,j}, T_{p,k}\right) \in E_p, \\ 0, \left(T_{p,j}, T_{p,k}\right) \notin E_p. \end{cases} \#(4)$$

基于上述依赖关系，逻辑子任务 $T_{p,k}$ 的前驱集合和后继集合分别定义为：

$$\mathrm{Pre}_p(k) = \left\{ j \middle| \left(T_{p,j}, T_{p,k}\right) \in E_p \right\} \#(5)$$

$$\mathrm{Suc}_p(k) = \left\{ j \middle| \left(T_{p,k}, T_{p,j}\right) \in E_p \right\} \#(6)$$

前驱集合为空的逻辑子任务称为根节点，其集合为：

$$V_p^{\mathrm{root}} = \left\{ T_{p,k} \middle| \mathrm{Pre}_p(k) = \varnothing \right\} \#(7)$$

后继集合为空的逻辑子任务称为终端节点，其集合为：

$$V_p^{\mathrm{exit}} = \left\{ T_{p,k} \middle| \mathrm{Suc}_p(k) = \varnothing \right\} \#(8)$$

具有多个后继节点的逻辑子任务称为分岔节点，具有多个前驱节点的逻辑子任务称为汇合节点。任务 p 只有在所有必要终端节点均完成后才被视为完成。

1.2　通信模型

任务通信链路主要涉及车地无线接入链路和边缘服务器间有线链路。

任务生成时，列车首先与当前位置覆盖范围内的源服务器建立无线连接，通过车地无线链路将数据上传至源服务器。考虑到列车高速运行对车地无线传输性能的影响，本文进一步将多普勒效应引入无线链路建模。列车高速移动会产生明显多普勒频移，进而破坏正交频分复用（orthogonal frequency division multiplexing, OFDM）子载波之间的正交性[26]。设系统载波频率为 f_c，光速为 c_0，则列车 i 在时刻 t 的最大多普勒频移为：

$$f_{d,i}^{max} = \frac{v_i(t) f_c}{c_0} \#(9)$$

设 OFDM 符号持续时间为 T_s，将多普勒效应引起的归一化子载波间干扰功率比例表示为：

$$P_{\mathrm{ICI}} = 1 - \int_{-1}^{1} \left(1 - |\tau|\right) J_0\left(2\pi f_{d,i}^{\max} T_s \tau\right) d\tau \#(10)$$

其中，$J_0(\cdot)$ 为第一类零阶 Bessel 函数。列车 i 到服

务边缘服务器 s 的有效传输速率表示为：

$$R_{i,s}=\frac{W}{N}log_2\left(1+\frac{P_iH_{i,s}}{\omega_i+P_iH_{i,s}P_{ICI}}\right)\#(11)$$

其中，W为信道带宽，N为共享同一无线资源的列车数量，P_i为列车发射功率，ω_i为等效噪声与外部干扰功率。

具有前后依赖关系的逻辑子任务被分配至其他轨旁边缘服务器执行时，中间结果依据DAG拓扑结构通过轨旁有线网络传输至后继节点所在服务器 n，边缘服务器之间的有效有线传输速率为 $R_{m,n}$；若部署在同一服务器，则可在服务器内部直接传递，无需产生跨服务器通信开销。

1.3　时延模型

任务的完成时延主要由传输时延和执行时延组成，由于本文考虑计算密集型任务，计算结果回传时延忽略不计。

任务的传输时延分为上传时延和跨边缘服务器迁移时延。采集数据由列车上传至源接入服务器，输入上传时延为：

$$T_{r,k}^{up}=\frac{d_{p,k}^{in}}{R_{i,s_r^{src}}}\#(12)$$

源接入服务器接收原始数据后，根据卸载决策将逻辑子任务分配至不同轨旁边缘服务器执行。设 $\rho_{n,m}$ 表示数据由服务器 n 迁移至服务器 m 所经历的跨区域传输次数，并规定 $\rho_{n,n}=0$。对于依赖边 $(T_{p,j},T_{p,k})\in E_p$，则服务器间迁移时延为：

$$T_{p,k,n,m}^{trans}=\rho_{n,m}\frac{d_{p,j,k}}{R_{n,m}}\#(13)$$

当两个逻辑子任务部署在同一服务器时，$n=m$，不产生服务器间传输时延。定义二进制变量 $x_{p,k}^m$ 表示逻辑子任务 $T_{p,k}$ 是否在边缘服务器 m 上执行。

任务的执行时延分为处理时延和排队时延两部分，计算时延只与任务大小和分配的计算资源有关，任务 $T_{p,k}$ 的执行时延为:

$$T_{p,k,m}^{\mathrm{com}}=\frac{d_{p,k}\kappa}{f_{p,k}^m}\#(14)$$

其中，$f_{p,k}^m$ 为服务器 m 为逻辑子任务 $T_{p,k}$ 分配的计算资源，而排队时延与当前服务器的负载情况有关，设 K_m 为逻辑子任务到达服务器 m 时该服务器上尚未完成的任务集合。逻辑子任务 $T_{p,k}$ 在服务器 m 上的排队时延为：

$$T_{p,k,m}^{\mathrm{queue}}=\begin{cases}0,\sum\limits_{T_{(r,l)}\in K_m}x_{r,l}^m=0\\ \sum\limits_{T_{(r,l)}\in K_m}T_{r,l,m}^{\mathrm{com}},\text{其他}\end{cases}\#(15)$$

逻辑子任务 $T_{p,k}$ 的完成时刻可统一表示为:

$$t_{p,k}^{\mathrm{finish}}=\begin{cases}a_p+T_{p,k}^{\mathrm{up}}+\sum\limits_{m\in M}x_{p,k}^m\left(T_{p,k,m}^{\mathrm{queue}}+T_{p,k,m}^{\mathrm{com}}\right),T_{(p,k)}\in V_p^{\mathrm{root}}\\ t_{(p,j)}^{\mathrm{finish}}+\sum\limits_{m\in M}x_{p,k}^m\left(T_{p,k,j,m}^{\mathrm{trans}}+T_{p,k,m}^{\mathrm{queue}}+T_{p,k,m}^{\mathrm{com}}\right),T_{(p,k)}\notin V_p^{\mathrm{root}}\end{cases}\#(16)$$

任务 p 的DAG完成时刻由最晚完成的必要终端节点决定，一个调度周期内所有任务的总完成时延为：

$$T=\sum_{p\in\mathcal{P}(t)}\max t_{p,k}^{\mathrm{finish}}-a_p\#(17)$$

1.4　能耗模型

本文中系统的能耗模型主要包括传输、迁移与计算的使用能耗。任务 p 由列车上传至边缘服务器的传输能耗为：

$$E_p^{\mathrm{up}}=P_i\sum_{T_{(p,k)}\in V_p^{\mathrm{root}}}T_{p,k}^{\mathrm{up}}\#(18)$$

任务的迁移能耗与传输功率和时延有关，任务 p 的迁移能耗表示为：

$$E_p^{\mathrm{trans}}=P^{\mathrm{trans}}\sum_{T_{(p,k)}\in V}T_{p,k,n,m}^{\mathrm{trans}}\#(19)$$

其中，P^{trans}为边缘服务器的等效传输功率。计算能耗取决于边缘服务器的动态电压频率和计算频率，任务 p 在边缘服务器 m 上执行产生的计算能耗为：

$$E_{p,k,m}^{\mathrm{com}}=\sum_{T_{(p,k)}\in V_p}\kappa_m c_{p,k}\left(f_{p,k}^m\right)^2\#(20)$$

其中，κ_m为边缘服务器 m 的有效开关电容系数，$f_{p,k}^m$为边缘服务器分配的计算资源。一个调度周期内的系统总能耗为：

$$E=\sum_{p\in P_t}E_p^{\mathrm{up}}+E_p^{\mathrm{trans}}+E_{p,k,m}^{\mathrm{com}}\#(21)$$

1.5　优化问题

在上述模型基础上，本文以任务完成时延为主要优化指标，并兼顾任务卸载和计算产生的系统能耗，将归一化时延与能耗的加权和作为综合优化目标：

$$\min \lambda \bar{T} + (1 - \lambda)\bar{E} \#(22)$$

$$\text{s.t.} \sum_{m \in M} x_{p,k}^{m} = 1, \forall p \in P_t, T_{(p,k)} \in V_p \#(23)$$

$$0 \leqslant f_{p,k}^{m} \leqslant x_{p,k}^{m} F_m, \forall p,k,m \#(24)$$

$$\sum_{p \in P_t} \sum_{T_{(p,k)} \in V_p} f_{p,k}^{m} \leqslant F_m, \forall m \in M \#(25)$$

$$T_p \leqslant D_p, x_{p,k}^{m} \in \{0,1\}, \forall p,k,m \#(26)$$

其中 $\bar{T}$ 和 $\bar{E}$ 分别表示归一化后的系统总完成时延和系统总能耗，$\lambda \in [0,1]$ 为权重系数，式(23)保证每个逻辑子任务必须且只能选择一个执行服务器；式(24)描述卸载决策与计算资源分配之间的耦合关系；式(25)保证服务器分配的总计算资源不超过其可用计算能力；式(26)限定任务的最大容忍时延，并规定每个逻辑子任务仅在一个边缘服务器上执行。

2　基于近端策略优化的依赖感知卸载方法

随着逻辑子任务数量和候选服务器数量增加，问题(22)的卸载映射空间快速扩张，传统穷举或精确优化方法难以满足高速铁路场景的在线决策需求。为此，本文提出移动性与依赖联合感知分层近端策略优化算法(mobility- and dependency-aware hierarchical proximal policy optimization，MDH-PPO)，将完整DAG映射分解为拓扑约束下的序贯服务器选择过程。MDH-PPO由拓扑调度层和服务器决策层组成。拓扑调度层根据当前系统中各DAG的执行状态和前驱完成情况，构造就绪子任务集合，并按照给定的拓扑调度规则确定当前待卸载子任务；服务器决策层利用列车移动状态、服务器负载、根节点输入传输、前驱结果迁移和预计计算时延确定其执行服务器。MDH-PPO算法架构如图3所示。

2.1　马尔可夫决策过程

将动态卸载过程表示为马尔可夫决策过程，其状态空间、动作空间和奖励函数定义如下。状态转移由列车移动、任务执行、服务器队列变化和新请求到达共同决定，本文采用无模型PPO通过环境交互学习卸载策略。

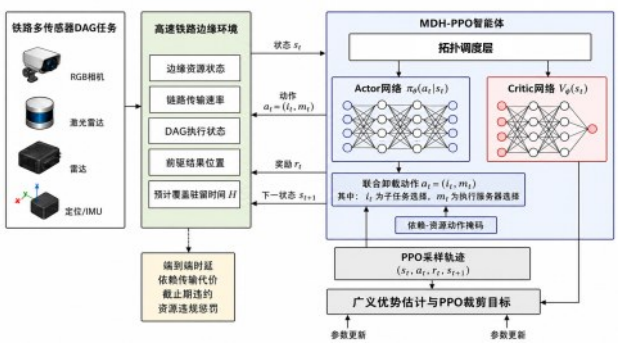

图3　MDH-PPO卸载算法框架

1) 状态空间。决策时刻 t 的系统状态由边缘服务器状态、列车状态、待调度子任务状态和依赖状态组成，即：

$$\boldsymbol{s}_t = \left(s_t^{ser}, s_t^{tra}, s_t^{task}, s_t^{dep}\right) \#(27)$$

其中，服务器状态包括各边缘服务器的可用计算资源、等待队列负载和资源利用率；列车状态包括列车当前位置、运行速度以及与当前位置源服务器的无线传输速率；子任务状态包括计算工作量、输入输出数据量、剩余容忍时延和预计关键路径长度；依赖状态包括前驱完成情况、前驱结果所在服务器、依赖数据量和已完成子任务的卸载位置。

2) 动作空间。DAG拓扑调度模块根据前驱完成状态确定当前就绪子任务，构成决策时刻 t 的就绪子任务集合 $Q^{\text{ready}}(t)$。联合动作由待调度子任务及其执行服务器组成：

$$\boldsymbol{a}_t = \left(k_t, m_t\right), T_{p,k_t} \in Q^{ready}(t), m_t \in M \#(28)$$

为降低组合动作空间的搜索复杂度，将联合策略分解为子任务选择策略和服务器选择策略：

$$\pi_\theta\left(\boldsymbol{a}_t|\boldsymbol{s}_t\right) = \pi_{\theta_1}^{sel}\left(k_t|s_t\right)\pi_{\theta_2}^{off}\left(m_t|k_t,s_t\right) \#(29)$$

其中，$\pi_{\theta_1}^{\text{sel}}$ 用于从就绪集合中选择待调度子任务，$\pi_{\theta_2}^{\text{off}}$ 用于确定其执行服务器，$\boldsymbol{\theta} = \{\theta_1, \theta_2\}$ 为Actor网络参数。

3) 奖励函数。为使策略学习目标与式(22)优化目标保持一致，采用当前卸载动作引起的归一化时延增量和能耗增量构造即时奖励：

$$r_t = -\lambda \frac{\Delta T_t}{T^{ref}} - (1 - \lambda) \frac{\Delta E_t}{E^{ref}} \#(30)$$

其中，ΔT_t 和 ΔE_t 分别表示当前动作引起的预计完成时延增量和能耗增量，T^{ref} 和 E^{ref} 分别为时延和

能耗的归一化参考值。

2.2　联合动作掩码

策略网络直接生成的动作可能不满足DAG执行顺序或服务器资源约束。对于联合动作(k,m)，定义可行性掩码为：

$$g_t(k,m)=I\left[F_m(t)>0\right]\prod_{j\in Pre_p^k}\mathbb{I}\left[R_{\left(z_{(p,j)},m\right)}>0\right]\#(31)$$

掩码依次判断子任务是否就绪、候选服务器是否具有可用计算资源以及全部前驱结果能否传输至候选服务器。对于根节点，将源接入服务器视为其虚拟前驱服务器。利用该掩码修正联合动作概率：

$$\tilde{\pi}_\theta(k,m|s_t)=\frac{g_t(k,m)\pi_\theta(k,m|s_t)}{\sum_{m'\in M}g_t(k',m')\pi_\theta(k',m'|s_t)}\#(32)$$

其中，将不可行动作的概率置为0，并对剩余动作重新归一化。若当前不存在可行动作，则暂缓调度，直至服务器释放资源。

2.3　策略更新

MDH-PPO采用Actor-Critic结构，Actor输出掩码后的联合动作分布，Critic估计状态价值$V_\varphi(s_t)$。为避免与跨区域传输次数$\rho_{(n,m)}$混淆，采用$\chi_t(\boldsymbol{\theta})$表示新旧策略的概率比：

$$\chi_t(\boldsymbol{\theta})=\frac{\tilde{\pi}_\theta(a_t|s_t)}{\tilde{\pi}_{(\theta_{old})}(a_t|s_t)}\#(33)$$

采用广义优势估计计算时序差分误差和优势函数：

$$\delta_t=r_t+\gamma V_\varphi(s_{t+1})-V_\varphi(s_t),\#(34)$$

$$\hat{A}_t=\sum_{l=0}^{L-1}(\gamma\epsilon)^l\delta_{(t+l)}\#(35)$$

其中，γ为折扣因子，ϵ为广义优势估计参数，L为采样轨迹长度。Actor的裁剪目标函数为：

$$L^c(\theta)=\mathbb{E}_t\Big[min\left(\chi_t(\theta)\hat{A}_t,clip\left(\chi_t(\theta),1\pm\alpha\right)\hat{A}_t\right)\Big]\#(36)$$

Critic的价值损失为：

$$L^V(\varphi)=\mathbb{E}_t\left[\left(V_\varphi(s_t)-V_t^{tar}\right)^2\right]\#(37)$$

综合策略目标、价值损失和策略熵，Actor和Critic网络参数通过损失函数更新，网络损失函数表示为：

$$L(\theta,\varphi)=-L^c(\theta)+c_1L^V(\varphi)-c_2H(\tilde{\pi}_\theta)\#(38)$$

其中，α为裁剪系数，V_t^{tar}为目标回报，$H(\tilde{\pi}_\theta)$为策略熵，c_1和c_2分别为价值损失和熵正则化系数。

2.4　卸载流程

MDH-PPO包括离线训练和在线决策两个阶段。离线训练阶段利用仿真环境生成列车移动、任务到达、链路变化和服务器负载过程，根据DAG前驱关系构造就绪子任务集合，并通过采样联合卸载动作更新Actor和Critic参数。在线决策阶段固定策略参数，利用策略网络搜索生成完整DAG候选映射，并依据端到端时延、依赖迁移代价和关键路径信息评价候选方案。MDH-PPO的训练及在线卸载流程如算法1所示。

算法1　MDH-PPO训练及在线卸载伪代码

输入：任务集合P、服务器集合M、训练回合数E、轨迹长度L及PPO超参数；

输出：训练后的卸载策略$\tilde{\pi}_{(\theta^*)}$

离线训练阶段

1) 初始化Actor、Critic及高速铁路边缘计算环境；

2) 在每个训练回合初始化列车位置、服务器队列、链路状态和DAG执行状态；

3) for e=1:1:E

4) 根据列车运行状态更新源服务器及车地无线传输状态，并依据通信模型计算无线传输速率；

5) 拓扑调度层根据DAG前驱完成状态构造就绪子任务集合$Q^{\text{ready}}(t)$；

6) 根据式(28)和式(29)依次确定待调度子任务和候选执行服务器，并依据联合动作掩码剔除不满足依赖和资源约束的动作；

7) Actor生成联合动作概率，构造可行性掩码，并从可行动作分布中采样$\boldsymbol{a}_t$；

8) 执行卸载动作，更新服务器队列、依赖数据位置和DAG完成状态；

9) 根据系统模型计算即时奖励r_t并存储状态转移样本(s_t,a_t,r_t,s_{t+1})；

10) 当前回合结束后，依据式(33)至式(38)计算优势函数、裁剪策略目标及价值损失，并更新Actor和Critic网络参数，保存验证性能最优的策略参数；

11) end for

在线卸载阶段

12) 加载训练后的策略参数 θ^*，初始化当前列车、服务器、通信链路及DAG执行状态；

13) 获取服务器资源变化及实时列车状态信息，依据式(27)构造系统状态 $\boldsymbol{s}_t$；

14) 固定网络参数，通过式(28)和式(29)生成分层卸载动作，并利用联合动作掩码选择可行执行服务器；

15) 执行卸载动作并更新DAG及系统状态。

结束

Actor和Critic的参数规模分别为 P_A 和 P_C，就绪子任务数为 Q，服务器数为 M，每轮采集 L 个样本并执行 K 次更新，则单轮训练复杂度约为 $O\left(KL\left(P_{\mathrm{A}}+P_{\mathrm{C}}\right)+LQM\right)$，在线单次决策复杂度约为 $O\left(P_{\mathrm{A}}+QM\right)$。联合动作掩码仅保留满足依赖与资源约束的动作，可降低无效动作采样比例。

3　仿真实验与分析

3.1　实验设置与对比算法

为验证MDH-PPO在高速铁路依赖感知任务卸载场景中的有效性，采用Python构建列车—边缘服务器协同计算仿真平台。实验选取文献[6]OSDaR23数据集中的476个有效同步帧生成连续感知请求，并依据雅万高铁线路数据构造双线对向运行轨迹。每列车在前一请求完成后生成下一请求，以模拟持续到达业务，每个参数点采用3组固定随机种子独立运行并取平均值。

感知业务依据文献[7]提出的相机与激光雷达融合铁路目标检测流程构建，由图像特征提取、轨道区域与前方列车感兴趣区域分割、激光雷达点云处理、铁路目标融合检测和结果输出5个逻辑子任务组成，DAG结构如图4所示。OSDaR23仅用于提供同步关系、输入规模和场景属性，不重新训练目标检测网络。主要仿真参数见表2。

设置以下3种对比算法。

1) DTOSC：依据文献[16]提出的依赖感知任务卸载与服务缓存方法实现，保留服务缓存、任务依赖、计算资源和截止期等决策因素，并适配本文DAG结构。

2) DeTTO：依据文献[17]提出的可信依赖任务卸载方法实现，根据依赖局部性、节点可信度和预计完成时间逐节点确定执行服务器。

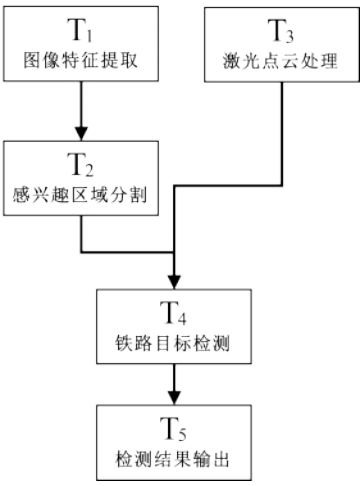

图4　相机与激光雷达融合的铁路目标检测任务流程图

表2　　仿真参数

参数	符号	数值
边缘服务器间距	d_e	10 km
无线覆盖半径	R_c	5.1 km
载波频率	f_c	2.1 GHz
车地无线传输带宽	W	20 MHz
OFDM符号时长	T_s	33.33 μs
路径损耗指数	ε	3
发射功率	P_i	200 mW
噪声干扰功率	ω_i	−94 dBm
边缘服务器数量	M	16
列车数量	N	5 - 25
列车速度	v	100 - 350 km·h^{-1}
边缘服务器计算资源	F_m	10-30 GHz
CPU周期数	κ	100 cycle·bit^{-1}

3) QNRLO：依据文献[23]提出的准牛顿深度强化学习两阶段卸载方法实现，采用批归一化深度神经网络生成候选策略，并利用有限内存拟牛顿法筛选卸载方案；本文将其动作扩展为各DAG节点的候选边缘服务器选择。

评价指标包括平均端到端时延和系统计算效率。平均端到端时延由输入上传、跨服务器迁移、排队计算、汇合等待和结果交付等时延组成，单位为s；系统计算效率定义为有效DAG计算工作量与累计完成时延之比，单位为Gcycle·s^{-1}。

3.2　收敛性分析

图5给出了MDH-PPO训练过程中的单回合平

均奖励及其10回合滑动平均值。训练初期，策略尚未形成稳定的依赖映射，奖励波动较大；随训练推进，滑动平均奖励在前100回合内持续提高并逐渐收敛，第200回合获得最优验证策略，后续未出现明显性能退化。结果表明，裁剪策略更新能够限制参数变化幅度，联合动作掩码减少了不可行动作采样，从而保持较好的训练稳定性。

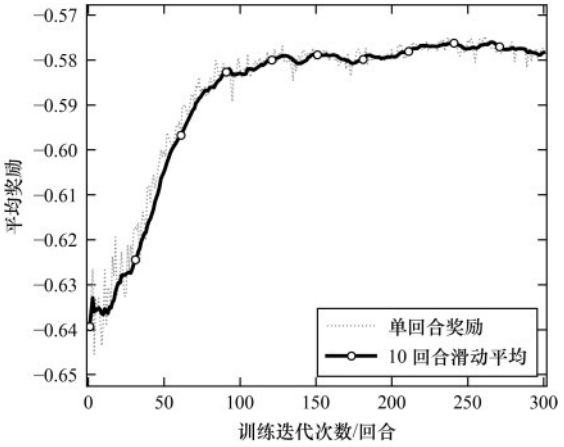

图5　MDH-PPO 训练奖励随迭代次数的变化

3.3　不同计算负载下系统性能分析

图6和图7分别给出了列车数量变化时的系统计算效率和平均端到端时延。列车数量由5列增至25列时，各算法的时延均上升、计算效率均下降，表明并发请求增加加剧了车地通信与边缘计算资源竞争。在25列场景下，MDH-PPO的平均时延为1.281 s，较DeTTO的1.605 s降低20.19%；系统计算效率为4.124 Gcycle·s^{-1}，较DeTTO的3.294 Gcycle·s^{-1}提高25.18%。MDH-PPO联合利用服务器负载、任务依赖和关键路径信息，可抑制高负载条件下的热点服务器聚集，其性能优势随并发规模增加而扩大。

3.4　不同计算资源下系统性能分析

图8和图9分别给出了单服务器计算资源变化时的系统计算效率和平均端到端时延。计算资源由10 GHz增至30 GHz时，各算法的时延持续下降、计算效率逐步提高；MDH-PPO的平均时延由1.026 s降至0.774 s，系统计算效率由5.189 Gcycle·s^{-1}增至6.893 Gcycle·s^{-1}。算力继续增加后，时延下降幅度趋缓，反映出输入上传、迁移和结果交付等通信开销逐渐成为主要限制。在30 GHz场景下，MDH-PPO较DeTTO的平均时延降低

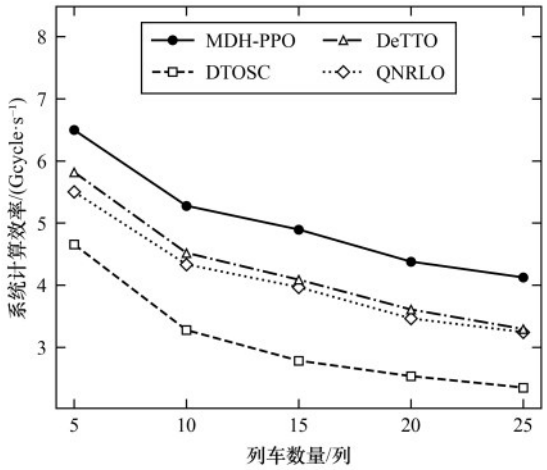

图6　系统计算效率随列车数量的变化

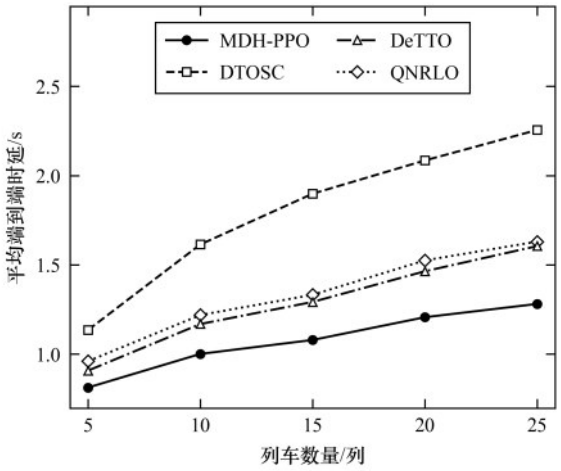

图7　平均端到端时延随列车数量的变化

3.26%，系统计算效率提高3.46%，说明依赖感知映射与动态负载协调在不同算力配置下均能保持稳定收益。

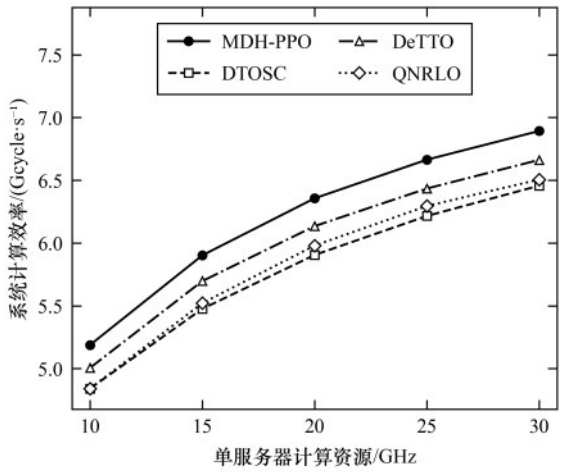

图8　系统计算效率随服务器计算资源的变化

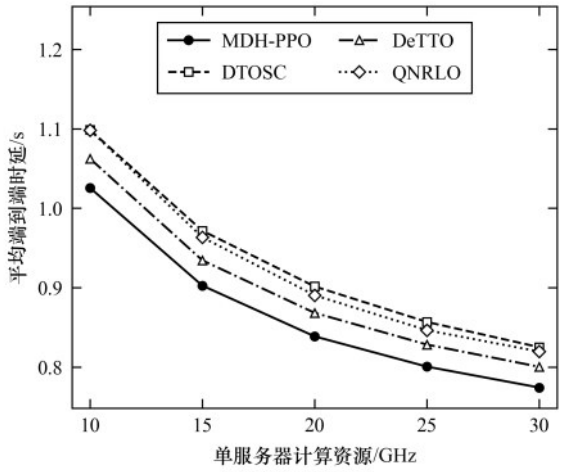

图 9　平均端到端时延随服务器计算资源的变化

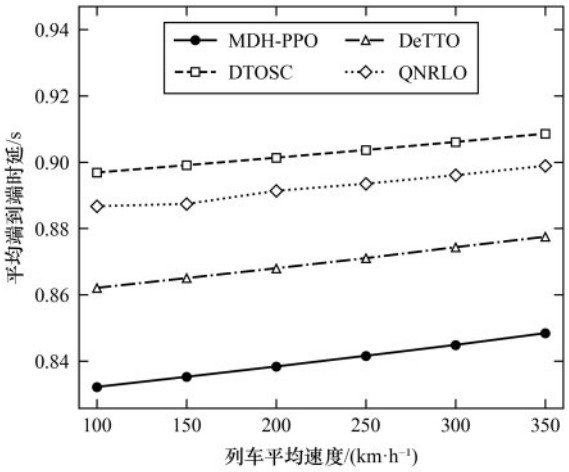

图 11　平均端到端时延随列车速度的变化

3.5　不同列车速度下系统性能分析

图 10 和图 11 分别给出了列车平均速度变化时的系统计算效率和平均端到端时延。速度由 100 km·h^{-1} 增至 350 km·h^{-1} 时，各算法的时延缓慢上升、计算效率相应下降；MDH-PPO 的平均时延由 0.832 s 增至 0.848 s，增幅为 1.95%，系统计算效率由 6.407 Gcycle·s^{-1} 降至 6.283 Gcycle·s^{-1}，降幅为 1.93%。多普勒干扰和服务区域切换随速度提高而增强，但其在总时延中的占比较低，因此速度变化对总体性能的影响有限。在 350 km·h^{-1} 场景下，MDH-PPO 较 DeTTO 的平均时延降低 3.31%，系统计算效率提高 3.51%，表现出较好的高速移动适应性。

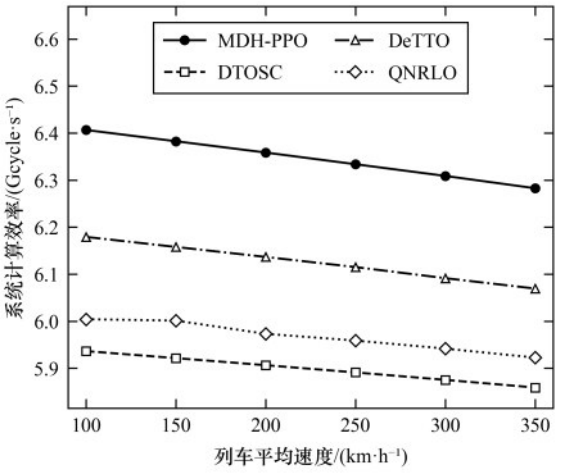

图 10　系统计算效率随列车速度的变化

4　结束语

本文研究高速铁路多传感器 DAG 任务在轨旁多边缘网络中的在线卸载问题，建立了包含输入上传、跨服务器迁移、排队和计算过程的时延—能耗模型，并提出 MDH-PPO 算法。该算法利用联合状态描述边缘资源、通信条件和 DAG 执行进度，通过子任务—服务器联合动作掩码保证决策可行，并采用 PPO 实现动态策略更新。仿真结果表明，在 25 列高负载场景下，MDH-PPO 较 DeTTO 的平均端到端时延降低 20.19%，系统计算效率提高 25.18%；在服务器算力和列车速度变化场景下，该算法均保持稳定性能优势。后续将结合真实高速铁路链路测量和边缘计算平台，进一步验证算法在通信波动与突发任务条件下的性能。

参考文献：

[1] 秦勇, 丁奥, 王彪, 等. 轨道交通列车新一代健康管理系统架构研究[J]. 机车电传动, 2024(1): 1-10.
Qin Y, Ding A, Wang B, et al. Research on the new generation framework of PHM systems for railway trains[J]. Electric Drive for Locomotives, 2024(1): 1-10.

[2] Li X H, Zhu M H, Zhang B Y, et al. A review of artificial intelligence applications in high-speed railway systems[J]. High-speed Railway, 2024, 2(1): 11-16.

[3] 龙腾, 王彧弋, 林军, 等. 轨道交通车载智能化应用技术发展展望[J]. 机车电传动, 2024(1): 11-21.
Long T, Wang Y Y, Lin J, et al. Development and prospects of intelligent technology in rail transit vehicles[J]. Electric Drive for Locomotives, 2024(1): 11-21.

[4] 朱力, 龚泰源, 梁豪, 等. 边缘智能在轨道交通中的应用：前景与展望[J]. 电子与信息学报, 2023, 45(4): 1514-1528.
Zhu L, Gong T Y, Liang H, et al. Application of edge intelligence in rail transit: Prospects and future outlook[J]. Journal of Electronics & Information Technology, 2023, 45(4): 1514-1528.

[5] 尚敬, 张慧源, 李晨. 具身智能机器人发展研究及轨道交通应用展望

[J]. 机车电传动, 2025(2): 1-14.
Shang J, Zhang H Y, Li C. Research on the development of embodied intelligence robots and their prospects for applications in rail transit[J]. Electric Drive for Locomotives, 2025(2): 1-14.

[6] Tagiew R, Köppel M, Schwalbe K, et al. OSDaR23: Open sensor data for rail 2023[C]//Proceedings of the 2023 8th International Conference on Robotics and Automation Engineering. Piscataway: IEEE Press, 2023: 270-276.

[7] Wang Z Y, Yu G Z, Wu X K, et al. A camera and LiDAR data fusion method for railway object detection[J]. IEEE Sensors Journal, 2021, 21 (12): 13442-13454.

[8] 潘文波, 李程, 袁希文, 等. 面向地铁场景的激光雷达目标检测研究[J]. 机车电传动, 2022(2): 89-96.
Pan W B, Li C, Yuan X W, et al. Research on lidar target detection for metro scene[J]. Electric Drive for Locomotives, 2022(2): 89-96.

[9] He R S, Ai B, Zhong Z D, et al. 5G for railways: Next generation railway dedicated communications[J]. IEEE Communications Magazine, 2022, 60(12): 130-136.

[10] 刘雷, 陈晨, 冯杰, 等. 车载边缘计算中任务卸载和服务缓存的联合智能优化[J]. 通信学报, 2021, 42(1): 18-26.
Liu L, Chen C, Feng J, et al. Joint intelligent optimization of task offloading and service caching for vehicular edge computing[J]. Journal on Communications, 2021, 42(1): 18-26.

[11] 李方伟, 张海波, 王子心. 车联网中基于MEC的V2X协同缓存和资源分配[J]. 通信学报, 2021, 42(2): 26-36.
Li F W, Zhang H B, Wang Z X. V2X collaborative caching and resource allocation in MEC-based IoV[J]. Journal on Communications, 2021, 42(2): 26-36.

[12] Xu J Y, Wei Z C, Lyu Z W, et al. Throughput maximization of offloading tasks in multi-access edge computing networks for high-speed railways[J]. IEEE Transactions on Vehicular Technology, 2021, 70(9): 9525-9539.

[13] Hu J T, Cao Y, An Y T, et al. Dynamic task offloading and migration algorithms for rail transit based on mobility prediction[J]. IEEE Transactions on Vehicular Technology, 2026, 75(3): 4810-4823.

[14] 丛玉良, 孙闻晞, 薛科, 等. 基于改进的混合遗传算法的车联网任务卸载策略研究[J]. 通信学报, 2022, 43(10): 77-85.
Cong Y L, Sun W X, Xue K, et al. Research on task offloading strategy of Internet of vehicles based on improved hybrid genetic algorithm[J]. Journal on Communications, 2022, 43(10): 77-85.

[15] 陈卓, 操民涛, 周致圆, 黄欣, 李彦. 移动边缘计算中基于图到序列深度强化学习的复杂任务部署策略[J]. 通信学报, 2024, 45(3): 244-257.
Chen Z, Cao M T, Zhou Z Y, et al. Graph-to-sequence deep reinforcement learning based complex task deployment strategy in MEC[J]. Journal on Communications, 2024, 45(3): 244-257.

[16] Shen Q Q, Hu B J, Xia E J. Dependency-aware task offloading and service caching in vehicular edge computing[J]. IEEE Transactions on Vehicular Technology, 2022, 71(12): 13182-13197.

[17] Dass P, Misra S. DeTTO: Dependency-aware trustworthy task offloading in vehicular IoT[J]. IEEE Transactions on Intelligent Transportation Systems, 2022, 23(12): 24369-24378.

[18] Chen X C, Cao J N, Sahni Y, et al. Mobility-aware dependent task offloading in edge computing: A digital twin-assisted reinforcement learning approach[J]. IEEE Transactions on Mobile Computing, 2025, 24(4): 2979-2994.

[19] He X, Pang S C, Gui H Y, et al. Online offloading and mobility awareness of DAG tasks for vehicle edge computing[J]. IEEE Transactions on Network and Service Management, 2025, 22(1): 675-690.

[20] Wang J, Hu J, Min G Y, et al. Dependent task offloading for edge computing based on deep reinforcement learning[J]. IEEE Transactions on Computers, 2022, 71(10): 2449-2461.

[21] Chen X C, Cao J N, Sahni Y, et al. Dynamic task offloading in edge computing based on dependency-aware reinforcement learning[J]. IEEE Transactions on Cloud Computing, 2024, 12(2): 594-608.

[22] Cao Z Q, Deng X H, Yue S, et al. Dependent task offloading in edge computing using GNN and deep reinforcement learning[J]. IEEE Internet of Things Journal, 2024, 11(12): 21632-21646.

[23] 章坚武, 芦泽韬, 章谦骅, 等. 基于拟牛顿法的深度强化学习在车联网边缘计算中的研究[J]. 通信学报, 2024, 45(5): 90-100.
Zhang J W, Lu Z T, Zhang Q H, et al. Research on deep reinforcement learning in Internet of vehicles edge computing based on Quasi-Newton method[J]. Journal on Communications, 2024, 45(5): 90-100.

[24] Gholipour N, Dias de Assuncao M, Agarwal P, et al. TPTO: A Transformer-PPO based task offloading solution for edge computing environments[C]//Proceedings of the 2023 IEEE 29th International Conference on Parallel and Distributed Systems. Piscataway: IEEE Press, 2023: 1115-1122.

[25] 贺佳. 基于三维点云与图像融合的轨道交通场景行人检测方法[J]. 机车电传动, 2024(3): 146-155.
He J. Pedestrian detection method in rail transit scenes based on fusion of 3D point clouds and images[J]. Electric Drive for Locomotives, 2024(3): 146-155.

[26] Qiao Y Y, Niu Y, Chen S, et al. Energy efficiency optimization of ultra-reliable low-latency communication for high-speed rail[J]. IEEE Transactions on Vehicular Technology, 2024, 73(11): 16638-16653.

胡景天（2000- ），男，山西晋中人，北京交通大学博士研究生，主要研究方向为云边协同计算、高速铁路任务卸载等。

曹源（1982- ），男，河南开封人，博士，北京交通大学教授、博士生导师，主要研究方向为列车运行控制系统健康管理。