

一种二阶决策的车联网边缘计算任务 动态卸载方法

巨涛, 张文金, 杨焱, 火久元

(兰州交通大学 电子与信息工程学院, 兰州 730070)

摘要:为解决车联网边缘计算中处理离散-连续混合决策问题时边缘计算系统计算性能不高的问题,基于深度强化学习提出了一种分布式的二阶决策车联网边缘计算任务动态卸载方法。通过将任务处理过程分为卸载决策和资源分配两个子问题,基于 D3DQN 网络和 TD3 网络设计了可同时处理离散和连续动作域的算法框架;同时,设计了动作空间搜索优化和动态优先级更新机制,以进一步提升对动作空间的有效搜索,提高算法性能;最终,在以上算法框架和优化机制的基础上,设计实现了二阶决策车联网边缘计算任务动态卸载算法。该算法可有效解决车联网边缘计算场景中的离散-连续混合决策问题,实现边缘计算系统资源的有效利用,保证车辆能根据当前网络状态和任务大小,以最小的时延和能耗完成计算任务卸载。仿真结果表明,与基准卸载方法相比,本文方法具有更快的收敛性和更低的时延能耗,可以较好地利用边缘系统计算资源为车载边缘任务请求提供计算服务,进一步提升用户服务质量。

关键词:车联网;移动边缘计算;深度强化学习;动态任务卸载;资源分配

中图分类号:TP311 **文献标志码:**A **文章编号:**1671-5497(2026)07-2006-14

DOI:10.13229/j.cnki.jdxbgxb.20241283

A second order decision-making dynamic offloading method for vehicle edge computing tasks

JU Tao, ZHANG Wen-jin, YANG Yao, HUO Jiu-yuan

(School of Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China)

Abstract: In response to the issue of low computational performance of edge computing systems when dealing with discrete-continuous hybrid decision-making problems in vehicular network edge computing scenarios, a distributed multi-stage decision-making strategy for dynamic offloading of vehicular network edge computing tasks based on deep reinforcement learning is proposed. By dividing the task processing procedure into two sub-problems: offloading decisions and resource allocation, an algorithm framework capable of handling both discrete and continuous action domains was designed based on the D3DQN network and TD3 network. In order to further enhance effective search in the action space and improve algorithm performance, an optimization and dynamic priority update mechanism for search in the action

收稿日期: 2024-11-29.

基金项目: 国家自然科学基金项目(61862037, 62262038); 甘肃省科技计划项目(23CXGA0028).

作者简介: 巨涛(1980-), 男, 教授, 博士. 研究方向: 并行计算, 深度学习并行优化, 边缘计算.

E-mail: jutao@lztu.edu.cn

space was designed. Finally, based on the above algorithm framework and optimization mechanisms, a multi-stage decision-making algorithm for dynamic offloading of vehicle network edge computing tasks was developed and implemented. This algorithm can effectively solve discrete-continuous hybrid decision-making problems in vehicle network edge computing scenarios, utilize edge computing system resources, and ensure that vehicles can complete the offloading of computing tasks with minimal delay and energy consumption, according to the current network status and task size. Simulation results show that, compared with baseline offloading methods, the proposed method has faster convergence and lower latency energy consumption. It can make better use of the computing resources of the edge system to provide computing services for vehicle edge task requests, thus further enhancing the quality of service for users.

Key words: internet of vehicles; mobile edge computing; deep reinforcement learning; dynamic task offloading; resource allocation

0 引言

据统计,目前车辆的保有量已经突破了10亿,并正在逐年上升,预计到2035年汽车保有量将会达到20亿^[1]。汽车保有量的增加给城市带来了诸如交通拥堵、行车安全等一系列问题^[2]。与此同时,互联网的发展使得人们对出行的服务需求更加复杂和多样化。在此背景下,车联网应运而生。车联网涵盖了车辆、路边单元(Road-side unit, RSU)与服务提供商,实现了全面的交互式通讯^[3]。

智能网联车通过集成计算和通信,可以支持各种新颖的车辆应用,但是搭载在车辆上的计算设备的计算能力有限^[4]。为解决车辆自身计算资源不足的问题,最早的解决方案是将车载用户的服务需求上传到云端进行处理,云平台处理完成后再将处理结果回传给车载用户^[5]。但将用户任务卸载到云平台处理会出现实时性不够、带宽不足、能耗较大等问题,从而导致无法高效处理车辆设备产生的数据,影响用户服务质量^[6]。在此背景下出现了车联网和边缘计算融合形成的新范式车辆边缘计算(Vehicle edge computing, VEC),其部署于路边单元的移动边缘计算(Mobile edge computing, MEC)服务器,可以完成从云端下沉的车辆计算任务,实现低延迟的边缘计算任务处理。车辆边缘计算弥补了传统云计算模式在网联车辆数据爆炸式增长中的不足^[7]。目前,车辆边缘计算引起了广泛的关注,主要的研究问题可以分为计算卸载^[8,9]、资源分配^[11,12]和边缘缓存3个方面。

1 相关工作

李智勇等^[13]通过对现有研究的分析,从系统架构模型、性能优化目标以及卸载策略等方面综述了车辆边缘计算场景中任务卸载面临的问题。边缘计算任务卸载与资源分配是紧密耦合的两个问题,目前已有诸多研究工作。许斌等^[14]针对移动边缘计算环境中任务动态变化的特点,设计了一种侧重于负载均衡的自适应启发式选择算法,防止计算资源过度分配引发信道拥塞,以避免不必要的能耗和带宽浪费。但是,传统启发式方法需要根据特定的任务调度场景和应用任务特点,针对性地设计任务调度规则,不能较好地适应动态变化的边缘计算场景,算法的可扩展性受到限制。Liu等^[15]针对车云协同环境中的计算卸载挑战,引入了基于深度确定性策略梯度(DDPG)的算法,处理了连续动作空间的问题,实现了对计算任务卸载决策的优化。但任务卸载决策是一个离散的问题,将离散问题连续化会导致算法寻优陷入次优解。孙彦景等^[16]综合考虑了车辆链路的可靠性,设计了压缩网络,利用双深度Q网络(DDQN)实现车辆到车辆(V2V)和车对基础设施(V2I)链路的智能资源分配。但是,资源分配是一个连续问题,将连续问题映射到离散空间,会降低模型的准确性。Hao等^[17]提出了一种保证用户公平性的时间连续计算卸载算法,当计算任务到达时,立即做出决策,同时在优化目标中引入公平效用函数,设计了带调整奖励的深度强化学习算法,降低了任务的决策等待时延。但没有同时考虑任务卸载决策和资源分配的连续性和离散性特点,仅使用双深度Q网络(DDQN)这一离散方

法解决混合决策问题,会影响模型的训练精度。Yan等^[18]针对在时变无线衰落信道和随机边缘任务计算场景下,如何同时确定每个任务的卸载和资源分配问题,提出一个基于 actor-critic 学习结构的深度强化学习框架,实现了终端设备的能耗时延成本最小化。Zhao等^[19]针对云-边协同的计算卸载问题,通过共同优化计算卸载和资源分配决策,设计了一种分布式计算卸载和资源分配算法,有效地提高了系统计算资源利用率和计算性能。Han等^[20]研究了动态环境下的计算任务卸载问题,提出了一种基于深度强化学习的算法,调用深度确定性策略梯度(DDPG)和决斗双深度Q网络(D3QN)分别处理连续和离散动作域,从能耗和时延两个方面最小化平均长期服务成本。但为了简化问题,上述任务模型将时间离散为多个时隙,任务只能在每个时隙才能进行卸载决策,导致任务决策等待时延增加。

以上方法虽然解决了部分车辆边缘计算环境下的任务卸载问题,但仍存在以下问题需要解决:①传统优化算法在解决边缘计算任务卸载与资源分配问题时,需要不断地调整分析模型以适应新的场景,无法适应日益复杂的高纬度状态空间和动作空间的车辆边缘计算场景;②车辆边缘计算

下的服务卸载是一个具有离散特点的任务卸载和连续特点的资源分配的混合决策问题,现有的研究只关注边缘计算领域内的任务卸载或资源分配,没有兼顾两者的相互影响,制约了车辆边缘计算系统效能的提升;③已有的模型缺乏对运动车辆在连续时间内的研究,未考虑车辆任务到达的随机性以及车辆高速移动造成的网络拓扑结构变化。

2 系统模型

2.1 多服务器多任务车辆任务卸载系统

本文面向多服务器多任务的车辆任务卸载系统网络模型如图1所示。整个系统其由1个配备边缘服务器的基站、多个配备有边缘服务器的路边单元和多个车辆组成。车辆自身具备一定的计算资源,可用于任务的本地执行。当车辆驶入基站信号范围内,基站实时监测车辆状态,做出全局化的卸载决策,确保网络资源的有效利用。本文卸载系统模型中,车辆生成任务需求后,随即向基站发出请求;随后,基站负责制定任务分配方案,并将决策结果反馈给相关车辆。在基站通信范围内有 N 个路边单元,有 M 辆车,车辆的计算能力表示为 $F_m, m \in M$ 。任务车辆随机产生任务请求,

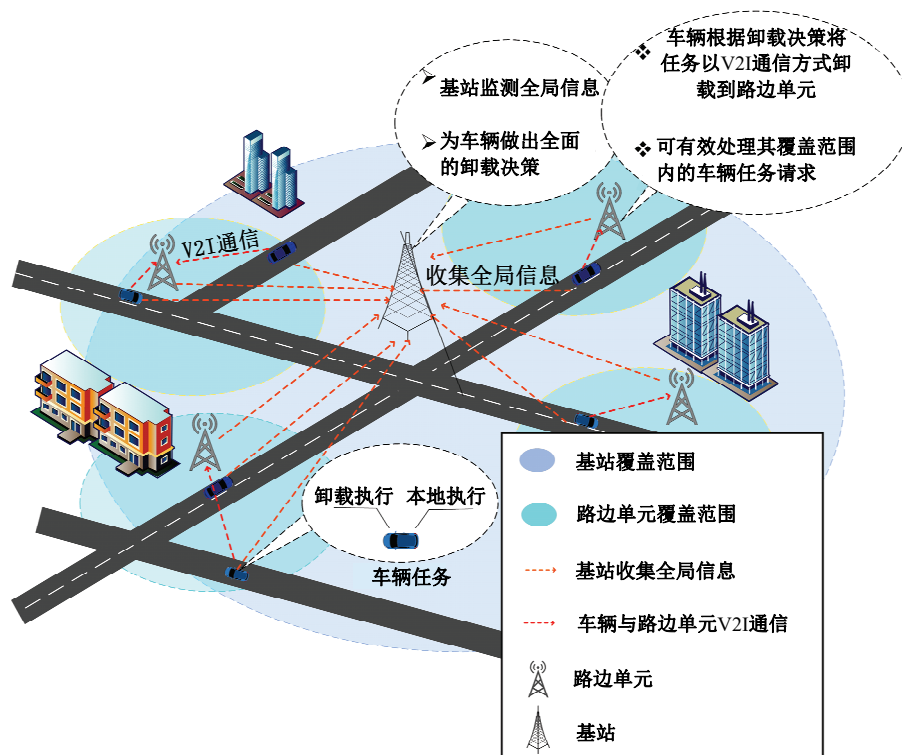


图1 多服务器多任务的车辆任务卸载系统

Fig. 1 Multi-server multi-task vehicle task offloading system

产生的任务 k_m 可以用4元组 $\{D_m, C_m, \alpha_m, \beta_m\}$ 表示,其中 D_m 表示任务大小, C_m 表示任务所需的CPU周期, α_m 表示任务延迟容忍, β_m 表示任务优先级。使用 $a_{k_m, n} \in \{0, 1\}, n \in \{0, 1, 2, \dots, N\}$ 表示任务车辆 m 的执行方式,其中 $a_{k_m, n} = 1$ 表示任务 k_m 被卸载到路边单元 n 执行, $a_{k_m, 0} = 0$ 表示任务在本地执行。

2.2 通信模型

车辆与基础设施间的通信(Vehicle-to-infrastructure, V2I)可以通过蜂窝网络实现。构建二维坐标系,以道路为 x 轴,道路到基站的垂直连线作为 y 轴。设定基站位于 $(0, h)$ 处, $|h|$ 表示基站与道路的垂直距离。车辆在道路上匀速行驶,车辆 m 的移动模式可由一个二元组 $\{(x_m, 0), v_m\}$ 表示,其中 $(x_m, 0)$ 表示车辆 m 的初始位置, $|x_m|$ 表示车辆与基站的横向距离,矢量 v_m 表示车辆 m 的速度。道路上的车辆动态连续,车辆发送任务请求时对车辆的位置信息进行更新, l' 时刻车辆 m 的位置信息为 $(x_m, 0)$, 则 $(l+1)'$ 时刻的位置为 $(x_m + v_m l, 0)$ 。

假设基站覆盖范围内的路边单元的位置信息可由 (X_n, Y_n) 表示,则车辆 m 与路边单元之间的距离定义为:

$$d_{m,n} = \sqrt{Y_n^2 + (X_n - x_m)^2} \quad (1)$$

假设V2I通信采用正交频率,根据香农定律,任务车辆 m 到路边单元 n 的数据传输速率为:

$$V_{m,n} = \omega \log_2(1 + P_m g_{m,n} d_{m,n}^{-\alpha} / N_0) \quad (2)$$

式中: ω 为分配的传输信道带宽; P_m 为任务车辆 m 的传输功率; $g_{m,n}$ 为参考距离处的信道功率增益; α 为路径损耗指数; N_0 为高斯白噪声; $d_{m,n}$ 为任务车辆 m 与路边单元 n 之间的距离。

任务车辆 m 与路边单元 n 间的通信时延表示为:

$$t_{m,n}^{\text{trans}} = D_m / V_{m,n} \quad (3)$$

2.3 计算模型

对基站覆盖范围内的 M 辆车进行分析。若车辆 m 选择本地执行,任务的计算时延仅取决于本地处理器分配给任务的计算资源 $F_m^{k_m}$,总时延为:

$$t_{m,m}^{\text{local}} = t_{m,m}^{\text{comp}} = D_m C_m / F_m^{k_m} \quad (4)$$

假设每一个装载在路边单元的边缘服务器的计算能力可以同时服务多个任务车辆。若车辆 m 选择将任务卸载到路边单元 n ,则任务在路边单元上的计算时延为:

$$t_{m,n}^{\text{comp}} = D_m C_m / F_n^{k_m} \quad (5)$$

由于结果通常很小,本文忽略反馈时延,仅考虑传输和计算过程,因此选择卸载到路边单元的总时延为:

$$t_{m,n}^{\text{edge}} = t_{m,n}^{\text{trans}} + t_{m,n}^{\text{comp}} \quad (6)$$

2.4 能耗模型

若车辆 m 在本地执行,车辆端每个CPU周期的能耗为 κ , f_m 为CPU运行频率,则本地执行任务的能耗为:

$$E_m^{\text{local}} = \kappa f_m^3 t_{m,m}^{\text{local}} \quad (7)$$

若车辆 m 将任务卸载到路边单元 n ,传输能耗为:

$$E_{m,n}^{\text{trans}} = t_{m,n}^{\text{trans}} P_m \quad (8)$$

假设边缘服务器处理一个CPU周期的能耗为 μ , f_n 为CPU运行频率,则边缘服务器处理任务的能耗为:

$$E_{m,n}^{\text{comp}} = \mu f_n^3 t_{m,n}^{\text{edge}} \quad (9)$$

因此,车辆 m 将任务卸载到路边单元 n 所需的总能耗为:

$$E_{m,n}^{\text{edge}} = E_{m,n}^{\text{trans}} + E_{m,n}^{\text{comp}} \quad (10)$$

2.5 任务效用模型

通常,车辆任务可区分为高优先级和低优先级任务。前者包括对安全性要求较高的任务,如导航辅助、环境感知及安全预警,这类任务有严格的时限要求;后者则包括如车载娱乐等可以接受较长时间延迟的服务。时间成本对于时延敏感型任务非常重要,一旦此类任务执行超出截止时间,可能会导致严重的安全风险和功能失效。任务完成效用模型定义为任务的时间成本,反映了不同任务在严格时延约束条件下耗费的总时延。具体模型抽象过程如下。

如果高优先级任务在截止时间前完成,则任务的时间成本为任务处理总时延 t_m ;如果完成时间超过截止时间,任务的时间成本为任务处理总时延 t_m 乘以惩罚因子 η_1 , η_1 取决于任务的紧急程度,通过多次的尝试与调整来确定。则高优先级任务的时间成本函数 $U_{k_m}^H$ 定义为:

$$U_{k_m}^H = \begin{cases} t_m, & t_m \leq \alpha_m \\ \eta_1 * t_m, & t_m > \alpha_m \end{cases} \quad (11)$$

如果低优先级任务在截止时间前完成,则任务的时间成本为任务处理总时延 t_m ;如果完成时间超过截止时间,则任务的时间成本将增加,具体

增加的程度由惩罚因子 η_2 决定, η_2 的取值过程同 η_1 。则低优先级任务的时间成本函数 $U_{k_m}^L$ 定义为:

$$U_{k_m}^L = \begin{cases} t_m, & t_m \leq \alpha_m \\ \eta_2 * t_m, & t_m > \alpha_m \end{cases} \quad (12)$$

由于低优先级任务不涉及安全性问题,故低优先级任务完成时间超过截止时间仍认为是可接受的,故惩罚因子 η_2 远小于 η_1 。

最终的任务完成效用模型 U_{k_m} 定义为:

$$U_{k_m} = I(\beta_m = \beta_m^H) U_{k_m}^H + I(\beta_m = \beta_m^L) U_{k_m}^L \quad (13)$$

式中: $I(\cdot)$ 为示函数,当条件为真时取值为 1,否则为 0; β_m 为第 m 个任务的优先级; β_m^H 和 β_m^L 分别为高、低优先级任务。

3 优化目标

本文目标是在基站的辅助控制下,通过不断训练决策网络,期望基站根据当前状态对到来的任务做出即时卸载决策,以实现最小的任务处理时延和最少的能量消耗。

针对每个任务车辆,任务 k_m 耗费的总时延 T_m 表示为:

$$T_m = U_{k_m} \quad (14)$$

任务 k_m 耗费的总能耗 E_m 表示为:

$$E_m = I(a_{k_m,n} = 1) E_{m,n}^{\text{edge}} + I(a_{k_m,n} = 0) E_m^{\text{local}} \quad (15)$$

由基站确定任务 k_m 执行方式和相应的资源分配,则最优化问题定义如下:

$$P1: \min \sum_{m=1}^M \varphi \times T_m + (1 - \varphi) \times E_m \quad (16)$$

$$\text{s.t. C1: } a_{k_m} \in \{0, 1\}$$

$$\text{C2: } \sum_{n=0}^N a_{k_m} = 1 \quad n \in \{0, 1, 2, \dots, N\}$$

$$\text{C3: } \sum_{m=1}^M F_n^{k_m} \leq F_0$$

式中: $\varphi \in (0, 1)$ 为对时延、能耗的偏好程度;约束条件 C1 表示任务采用整体卸载的形式;约束条件 C2 确保任务只能在本地端、边缘端中采取一种执行方式;约束条件 C3 表示路边单元边缘服务器提供的计算资源不能超过自身限制。

4 任务卸载策略

本文通过将任务处理过程分为卸载决策和资源分配两个子问题,基于 D3DQN 网络和 TD3 网络设计了可同时分别处理离散和连续动作域的算

法框架。同时,为了进一步提升对动作空间的有效搜索,提高算法性能,引入了动作空间搜索优化和动态优先级更新机制。最终,在以上算法框架和优化机制的基础上,设计实现了二阶决策车联网边缘计算任务动态卸载算法(Second order decision-making dynamic offloading, SODDO)。

4.1 马尔可夫决策过程构建

为了实现任务卸载和资源分配决策优化,首先构建马尔可夫决策过程。具体的状态空间、动作空间和奖励函数定义如下。

(1) 状态空间

基站中的智能体观测收集基站覆盖范围内的车辆信息和环境信息,收集的主要观测状态信息包括:车辆位置 $(x_m, 0)$ 、车辆可用资源 $F_m^{k_m}$ 、任务大小 D_m 、任务复杂度 C_m 、任务最大容忍时延 α_m 、任务优先级 β_m 。车辆任务的状态空间向量定义为:

$$s_m = \{(x_m, 0), F_m^{k_m}, D_m, C_m, \alpha_m, \beta_m\} \quad (17)$$

(2) 动作空间

根据系统的当前状态和观察到的环境,针对任务 k_m ,智能体选择动作,包括任务的卸载决策 a_0 和资源分配 a_1 。动作空间可以表示为:

$$a_m = \{a_0, a_1\}, a_0 \in \{0, 1, 2, \dots, N\}, \forall a_1 \in (0, 1) \quad (18)$$

式中: $a_0 = 0$ 为任务在本地处理,否则任务卸载到相应的路边单元 n 上处理; a_1 为给任务 k_m 分配资源的比例。

(3) 奖励函数

智能体观察状态 s 并执行动作 a ,针对任务 k_m ,此时的即时奖励 R 是 SODDO 算法获得最优卸载策略并分配资源后,任务效用和总能耗的加权和,奖励函数表示如下:

$$R = -\max \sum_{m=1}^M \xi \times U_{k_m} + (1 - \xi) \times E_m - \eta_3 \quad (19)$$

式中: $\xi \in (0, 1)$ 为对时延、能耗的偏好程度; η_3 为路边单元提供的计算资源超过自身限制,即不满足式(16)中约束条件 C3, 时的惩罚,该惩罚参数的设置通过多次的尝试与调整来确定。

4.2 二阶决策动态任务卸载策略

(1) D3DQN 及 TD3 算法分析

D3DQN 算法^[21]可有效处理离散动作空间的优化问题,算法处理框架如图 2 所示。该算法是一种结合了 Dueling DQN^[22] 和 Double DQN^[23] 的深度强化学习算法,它的核心在于改进了目标值

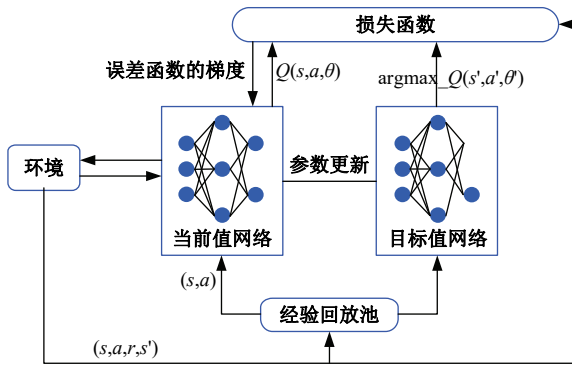


图2 D3DQN算法框架

Fig. 2 D3DQN algorithm framework

的计算方式,通过使用两个Q网络来减少目标Q值的“过估计”问题,提高决策的准确性。其中使用评估网络来选择当前状态下的最优化动作,使用目标网络来计算最优动作的价值。同时在经验回放池中根据样本的重要性进行加权采样,以更高效地利用有限的经验数据^[24]。

TD3算法^[25]是针对连续动作空间的强化学习算法,算法处理框架如图3所示。TD3算法克服了DDPG算法^[26]由于函数逼近引入的偏差和方差问题,特别是过高估计问题。采用两套Critic网络,计算目标值时取二者中的较小值,从而抑制网络过估计问题;在计算目标值时,在下一个状态的动作上加入扰动,使得价值评估更准确。同时,采用延迟更新策略,在Critic网络更新多次后,再更新Actor网络,保证Actor网络的训练更加稳定。

(2) 二阶决策动态任务卸载框架

本文在以上D3DQN及TD3算法框架的基础上,设计了分布式的二阶决策动态卸载策略SODDO,将混合决策问题解耦为任务卸载和资源分配两个子问题,智能体根据不同策略对两个子问题进行决策,最小化时延能耗加权和。SODDO算法框架如图4所示。首先,使用基于任务优

先级算法解决任务执行顺序问题,并使用基于D3DQN的一阶决策网络求解任务卸载决策子问题,最后,使用基于TD3的二阶决策网络求解边缘服务器的资源分配连续子问题,通过双层反复迭代最终求得问题的近似最优解。

4.3 动作空间搜索优化

深度强化学习方法通过智能体与车辆的不断交互获得奖励,从而不断优化自身动作策略,以期待最大化其长期效益。在车辆边缘计算场景中,智能体在不同的环境状态下搜索动作空间,探索合适的卸载决策和资源分配方式。传统D3DQN采用 ϵ -greedy算法进行动作探索,是一种按一定概率进行随机探索的盲目选择,会影响探索效率。为了优化动作空间的搜索效率,本文采用基于置信度上界(Upper-confidence-bound, UCB)的动作选择策略。

UCB策略是一种平衡搜索与利用的策略。其核心思想是,对于每个可能的动作,不仅考虑目前为止的平均奖励,还要考虑到与该动作相关的不确定性^[27]。该决策以动作价值估计置信区间的上界作为选择动作的依据,在每个决策点,算法选择具有最高UCB值的动作。其处理流程如图5所示。首先,神经网络输出价值函数 $Q(s,a)$,表示在状态 s 下采取动作 a 的预期奖励;然后,使用置信区间上界(Upper confidence bound, UCB)算法来决定下一步的动作选择,UCB算法使用式(20)选择动作;选择动作并执行后,更新UCB选择函数的参数,计算当前动作的奖励值 R ,并存储在近期奖励值序列中,当达到指定迭代数时,计算奖励值均值,并清空序列;基于收集到的奖励信息(如奖励平均值),对神经网络的参数或其他参数进行调整,以期在未来选择更优的动作,获得更高的累计奖励。通过不断更新参数和选择动作,逐

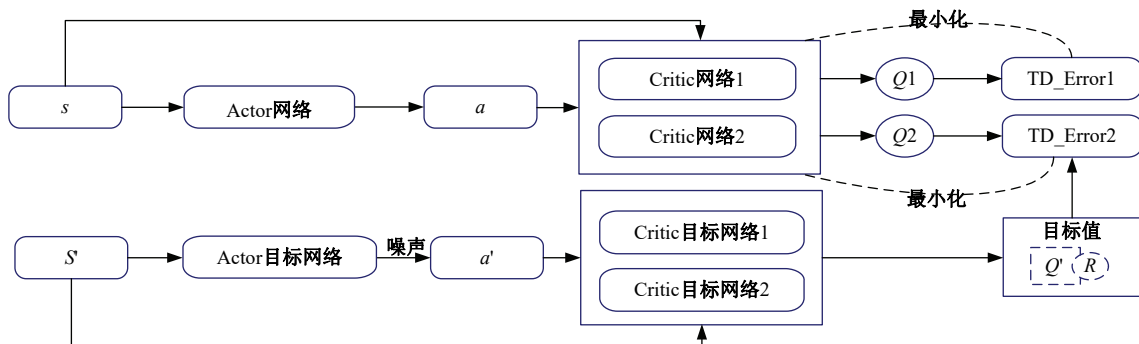


图3 TD3算法框架

Fig. 3 TD3 algorithmic framework

步优化策略。

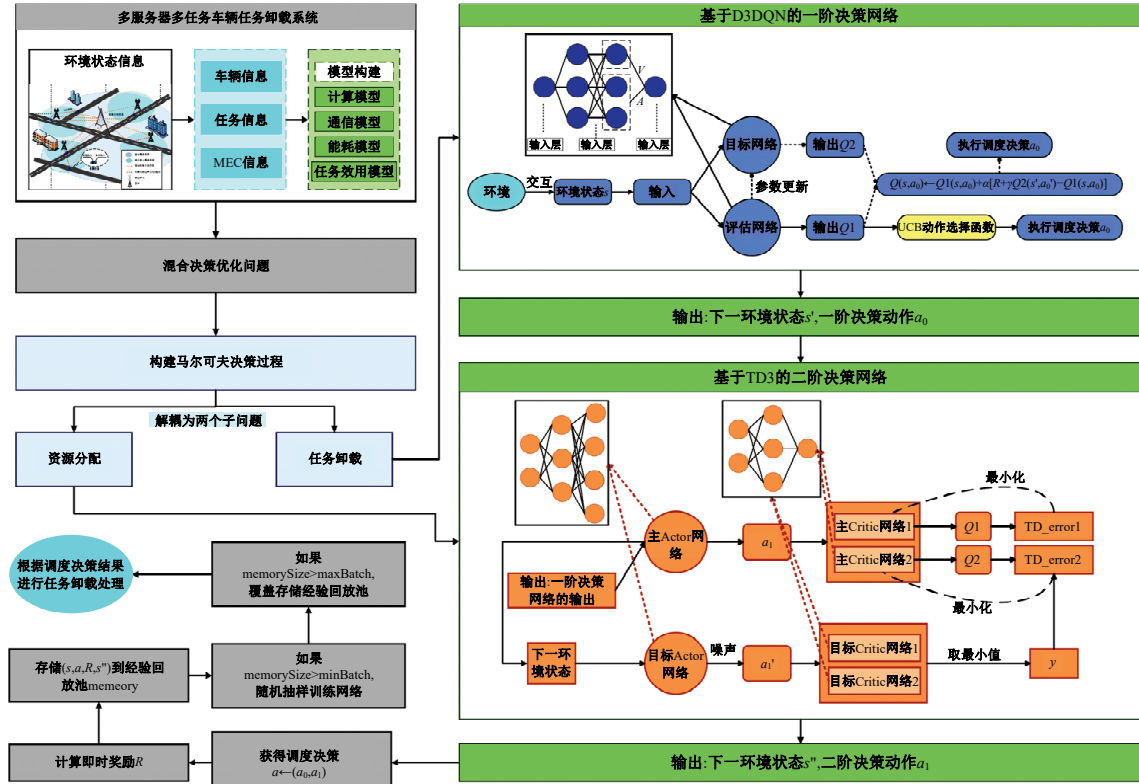


图 4 SODDO 任务卸载框架

Fig. 4 SODDO task offloading framework

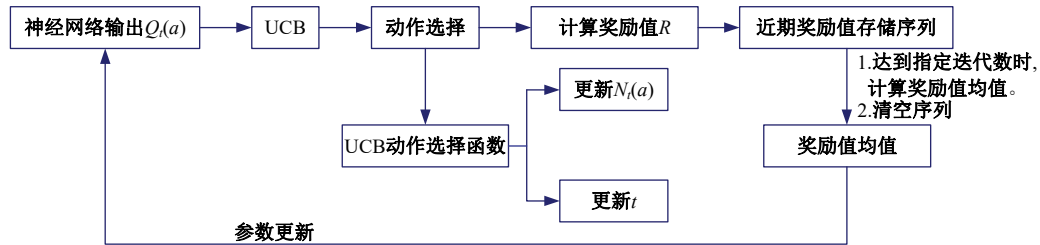


图 5 UCB 处理流程

Fig. 5 UCB work process

$$A = \arg \max [Q_t(a) + c \sqrt{\frac{\ln t}{N_t(a)}}] \quad (20)$$

式中: $Q_t(a)$ 为当前动作 a 的价值估计, 代表利用;

$\sqrt{\frac{\ln t}{N_t(a)}}$ 为估计的置信度上界, 代表探索; c 为一

个大于 0 的参数, 用于平衡利用和探索, 也可以将 c 看作是置信水平; t 为决策总数; $N_t(a)$ 为当前决策之前 a 被选择的次数, 如果 $N_t(a) = 0$, 则认为 a 就是满足最大化条件的动作。

对于每个动作 a , 每次选择该动作时, $N_t(a)$ 会增加, 不确定性会减小; 如果没有选择动作 a , 则 $\ln t$ 增加而 $N_t(a)$ 不变, 不确定性会增加。随着

训练迭代次数的增加, 具有较低价值估计或已经被选择了多次的动作, 再被选择的概率较低。奖励值可以反映神经网络的学习进度, 奖励值大则说明神经网络远未收敛, 还需对动作空间进行探索以寻找最优解。通过反复训练和调整参数, 系统可以逐渐找到最优的策略, 实现 VEC 系统资源的高效利用和任务的执行优化。

4.4 任务优先级调度优化

为了提高本文二阶决策算法的性能, 为卸载决策和资源分配提供合理的决策依据, 提出一个动态优先级资源分配算法 (Dynamic priority resource allocation, DPRA), 根据任务的特性、重要

程度和紧急程度,给予任务一个相对的优先级,以便在资源有限的情况下进行合理的任务调度、卸载和资源分配。

基站收集车辆任务请求信息后,使用优先级算法对待处理的任务按照优先级由高到低进行排序,存储在待处理队列 ϕ_{wait} 中。算法具体求解步骤如下。

(1) 计算车辆 m 的计算资源占系统总资源的比例 f_{ratio} :

$$f_{\text{ratio}} = \frac{f_{m, \text{user}}}{\sum_{m=1}^M f_{m, \text{user}}} \quad (21)$$

式中: f_{ratio} 为车辆 m 的资源充盈度,越小则本地处理能力越弱,其任务需要卸载到 MEC 进行处理的程度越大。

(2) 计算车辆 m 的本地执行时延占系统总时延的相对比例 t_{ratio} :

$$t_{\text{ratio}} = 1 - \frac{t_{\text{local}}}{\sum_{m=1}^M t_{\text{local}}} \quad (22)$$

式中: t_{ratio} 为车辆 m 的任务紧急程度,越大任务越紧急,需要卸载到 MEC 进行处理的程度越高。

(3) 根据下式初始化任务的优先级:

$$\bar{p}_m = \frac{t_{\text{ratio}}}{f_{\text{ratio}}} \quad (23)$$

对上述优先级进行归一化:

$$p_m = \frac{\bar{p}_m}{\sum_{m=1}^M \bar{p}_m} \quad (24)$$

(4) 在决策开始前根据式(25)更新所有任务的优先级。

$$Pr = \gamma p_m + \epsilon \frac{1}{e^{(t^{\text{dead}} - t^{\text{wait}})} - 1} + \lambda(e^{t^{\text{wait}}} - 1) \quad (25)$$

式中: t^{wait} 为任务的等待时间, t^{dead} 为任务的截止时间。

任务的紧迫性与任务初始优先级及其累积等待时间呈正相关,而与任务的截止期限呈负相关;遵循这一原则,具有较高优先级的任务将在调度队列中获得优先处理权,先于其他较低优先级的任务被执行。

(5) 基于以上优先级进行任务卸载与分配决策。具体步骤如算法 1 所示。

算法 1 动态优先级资源分配算法 DPRA

输入: n (任务车辆数), 待决策队列 wait .

输出: 待决策队列 wait .

1 初始化车辆任务列表 $\{D_m, C_m, \alpha_m, \beta_m\}$

2 更新车辆任务的动态优先级

3 for i in range(n)

4 while $i < n$:

5 根据式(25)更新所有任务优先级.

6 根据新的优先级重新排序车辆任务.

7 end for

8 遍历待决策队列 wait , 按照优先级高低依次检查每个任务.

9 对待决策队列 wait 中的车辆任务进行卸载决策, 对于处理完的任务, 出队列, 释放空间.

10 完成一次任务调度之后, 对决策队列 wait 中的车辆任务重新评估优先级, 并更新.

4.5 二阶决策动态任务卸载算法

由于动作 a 同时包含离散和连续空间, 使用传统深度强化学习方法很难直接计算策略的梯度^[28-30]。为了能同时处理边缘计算场景中离散-连续混合决策问题, 本文设计实现了二阶决策车联网边缘计算任务动态卸载算法, 用一阶决策网络解决任务卸载子问题, 用二阶决策网络解决资源分配子问题。算法在执行过程中, 智能体先收集当前环境状态 s 并对其进行归一化处理以确保数据的一致性和可比较性, 随后输入一阶决策网络 D3DQN, 在此网络中使用 UCB 动作选择策略得出初步的离散卸载决策 a_0 。然后, 将一阶动作 a_0 与原始状态 s 结合, 构建出新的状态 s' , 输入二阶决策网络 TD3, 得到二阶连续决策即资源分配策略 a_1 。最终卸载决策为 $a \leftarrow [a_0, a_1]$ 。

在一阶决策网络中, D3DQN 利用评估网络获取 s' 状态下最优动作价值对应的动作, 然后利用目标网络计算该动作的动作价值, 从而得到目标值, 通过两个网络的交互, 有效避免了 DQN 算法的“过估计”问题。目标值的计算方式为:

$$y_t = R' + \gamma Q(s', \arg \max_a Q(s', a; \omega_e); \omega_l) \quad (24)$$

式中: ω_e 和 ω_l 分别为评估网络和目标网络的参数。

在二阶决策网络中, TD3 算法通过 Critic 架构来评估 Q 值, 它采取两个独立的 Critic 网络对同一动作价值进行估计, 然后在更新过程中选用两者中较低的 Q 值, 有效避免策略学习中的高估问题。在更新 critic 网络的过程中, 为了防止目标值估计的不稳定性, 在预测下一个状态的动作值时, 向动作输入中添加一个扰动, 促使所学的价值函数在动作空间中展现出更佳的平滑性, 进而使

得价值评估更加准确,避免因估计波动而导致学习效率下降。由于 D3DQN 与 TD3 的目标策略与行为策略不同,为了加速训练过程并提高数据利用效率,采用经验回放机制,避免训练数据的序列依赖性问题,从而使得学习更加稳定和高效。训练过程中,智能体从达到 miniBatch + 1 的经验回放池中随机抽取 miniBatch 组数据进行训练。特别当经验回放池中的数组达到 maxBatch 时,对经验回放池进行数据覆盖操作,更新经验回放池中的数据,保证有效的特征提取和模型训练。二阶决策动态任务卸载算法 SODDO 具体实现如算法 2 所示。

算法 2 二阶决策动态任务卸载算法 SODDO

输入:待决策队列 wait
输出:车辆任务卸载策略
1 初始化经验回放池 memory、参数复制周期数 N1、最大迭代次数 I
2 for 迭代次数 iteration = 1 to I do
3 if iteration/N1 == 0
4 新网络参数
5 end if
6 if memory_size > miniBatch
7 抽取 batchsize 条数据作为样本进行学习
8 end if
9 if memory_size > maxBatch
10 覆盖经验回放池
11 end if
12 归一化环境状态信息 s
13 调用一阶决策网络获取动作 a_0
14 调用二阶决策网络获取动作 a_1
15 计算环境反馈的即时奖励 R
16 获取新状态 s'
17 添加经验(s, a, R, s')到回放池中
18 end for

5 实验仿真和结果分析

5.1 实验设置

本文采用 python3.7 和 TensorFlow1.15.0 深度学习框架进行实验仿真, TensorFlow 1 是一个广泛应用于机器学习和深度学习的开源库,支持包括 DQN、DDPG、TD3 等在内的多种强化学习算法实现。实验参数的设定参照了文献[31-33]。考虑到能源消耗约束的多样性,针对车辆端和服务端分别选用了不同的中央处理器参数配置。车辆端参考 Intel™ Core™ 系列 CPU 进行设置,服务器端参考 Intel® Xeon® 系列 CPU 进行调整,以

确保实验条件与实际应用环境相吻合;为了涵盖从简单到复杂的各类计算任务,将任务复杂程度控制在 $[50, 1\ 250]$ cycles/bit,在数据传输能力方面,设定单个车辆与路边单元之间的平均数据吞吐量为 38.5 Mbit/s,而 VEC 服务器的平均数据吞吐量为 770 Mbit/s;仿真实验具体参数如表 1 所示。

表 1 实验参数

Table 1 Experimental parameter

参 数	数 值
车辆数量	20
基站覆盖范围(直径)/m	1 200
高斯白噪声 N_0 /dBm	-110
参考信道功率增益 β_0 /dB	-30
车辆最大计算资源 $f_{\max}$ /GHz	IPC*4*1.2*3
任务复杂度 C /(cycles·bit ⁻¹)	[50, 1 250]
任务最大时延 $\hat{T}$ /s	{0.75, 1.5}
路径损耗指数 θ	4
车辆行驶速度/(km·h ⁻¹)	50
单 RSU 覆盖范围(直径)/m	400
无线通信带宽 B /MHz	5
传输功率 p /mW	100
VEC 最大计算资源 $F_{\max}$ /GHz	IPC*32*1.2*4
任务大小 D /kB	[100, 1 000]
单位 CPU 周期能耗 κ	10^{-28}
批处理尺寸 batchsize	64

为验证本文算法 SODDO 的有效性,将本文方法和以下 4 种基准算法进行比较。

(1)全部卸载方式(Edge_Only):车辆产生的所有任务均卸载到装载有边缘服务器的路边单元上处理。

(2)本地执行方式(Local_Only):车辆产生的所有任务由其自身处理。

(3)基于 DDPG 算法的卸载方式:将适用于连续动作空间的 DDPG 算法应用于本文场景中,在解决卸载决策这一离散问题时,将离散动作空间映射到一个连续空间中,为每个离散动作分配一个连续的向量。

(4)基于 Double DQN 算法的卸载方式:将适用于离散动作空间的 Double DQN 算法应用于本文场景中,为处理连续动作空间,在模拟中通过量化连续动作值,用逼近有限的离散值来模拟实际值,从而使 Double DQN 算法适应连续动作空间。

5.2 参数分析

(1)学习率分析

首先,测试了不同学习率对算法性能的影响,

所提算法在不同学习率下的收敛性能如图6所示。

当 $L_A=0.0001$ 、 $L_C=0.0002$ 、 $L_D=0.0005$ 时,本文算法收敛于局部最优解,因为较小的学习率会导致网络参数更新缓慢,增大寻优时间,还未真正找到最优解进入局部极值点就收敛。同时当 $L_A=0.01$ 、 $L_C=0.02$ 、 $L_D=0.05$ 时,由于学习率设置过大时网络更新步骤大,从而导致模型无法稳定收敛。所以将学习率分别设置为 $L_A=0.001$ 、 $L_C=0.002$ 、 $L_D=0.005$ 。

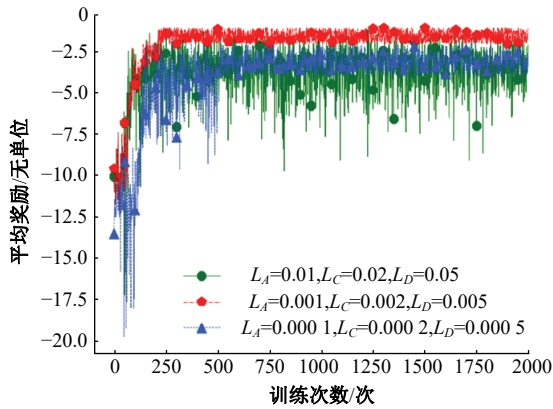


图6 不同学习率下SODDO收敛性能

Fig. 6 SODDO convergence at different learning rates

(2) 折扣因子分析

折扣因子 $\gamma \in [0, 1]$, 决定了即时奖励和未来奖励之间的相对重要性, 其数值偏向0时表明策略更侧重于最大化当前的即时奖励; 而趋向于1, 则表明策略应更多地侧重长期未来的奖励。如图7所示, 当折扣系数 $\gamma=0.9$ 时, 曲线的变化趋势较为平缓, 且经过训练的计算卸载策略具有最佳性能, 所以将算法 γ 值设定为0.9。图7比较了不同折扣因子 γ 对训练奖励的影响。

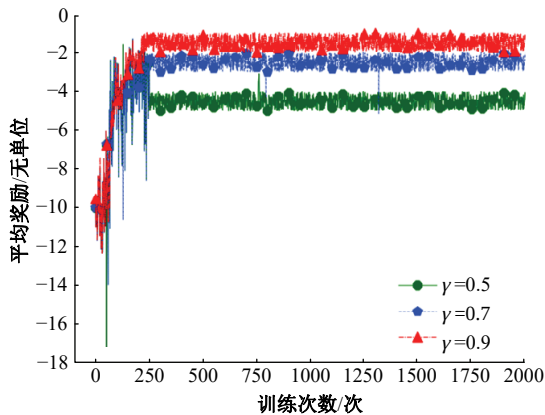


图7 不同折扣因子下SODDO收敛性能

Fig. 7 Convergence performance of SODDO under different discount factors

(3) 奖励函数加权参数分析

图8描述了将奖励函数的加权参数 ω 设置为0.2、0.5、0.8时对所提方案收敛性能影响。

奖励函数的加权参数类似于时间和能量的加权参数, ω 的值可以很容易地调整能耗和时间成本之间的权衡, 因此 ω 的取值会影响网络的收敛性能。通过试错技术, 尝试了加权参数的所有可能的组合/选项, 并取得最佳值, 以达到稳定的输出。结果表明, 在3种不同加权值下, 奖励值随着训练次数的增加而增加, 并在250次后保持稳定, 且当权重参数为0.5时, 所提方案可以获得最佳奖励值。

5.3 性能评测

(1) 算法收敛性比较

图9对比了5种算法在相同任务数量下的收敛性能, 可知, 本文方法的收敛速度最快(大约240次)且达到了最低的时延能耗加权值。相比之下, DDPG和Double DQN在350次迭代左右收

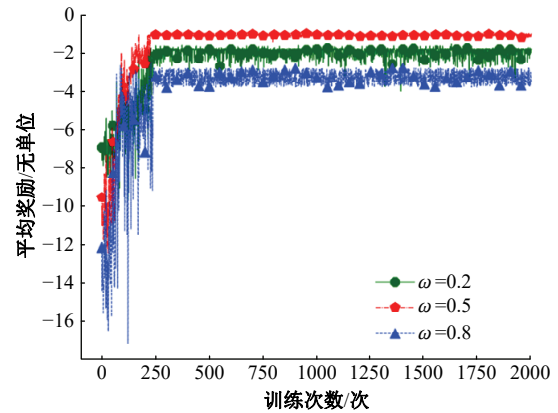


图8 不同权重下SODDO收敛性能

Fig. 8 Convergence performance of SODDO under different weights

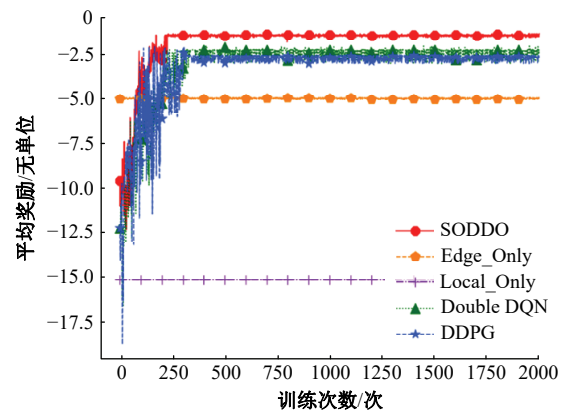


图9 不同算法收敛性能

Fig. 9 Convergence performance of different algorithms

敛,但收敛值相对于本文所提方法较高。结果表明,在前 2 000 次迭代中,本文方法可以实现比常用基线方法更低的时延能耗加权值。主要原因是本文针对任务本身构建卸载模型,不存在等待时延,且本文方法同时解决任务卸载和资源分配两个问题,大大提高了系统资源利用率,因此本文方法的收敛速度比基线算法快,且能取得更优解。

(2) 时延比较

图 10~图 12 分别对比了 5 种算法在相同任务数量下的执行时延、传输时延、总时延。

从图 10 执行时延对比可以看出,本文方法所表现的性都优于其他 4 种基线算法。主要原因是本文方法没有决策等待延迟,不存在等待时延。从图 11 的总传输时延对比可以看出,本文方法较基线算法时延更高,主要原因是本文针对任务本身建立的卸载模型考虑了任务的优先级,针对优先级高的任务,如果覆盖其任务车辆的 RSU 资源不足时,考虑将任务迁移到邻近的其他 RSU 处理,故而会增加其传输时延。

从图 12 的总时延对比中可以看出,本文方法

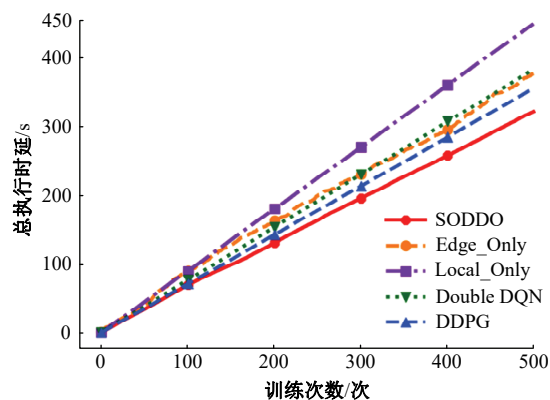


图 10 执行时延比较

Fig. 10 Total execution latency comparison

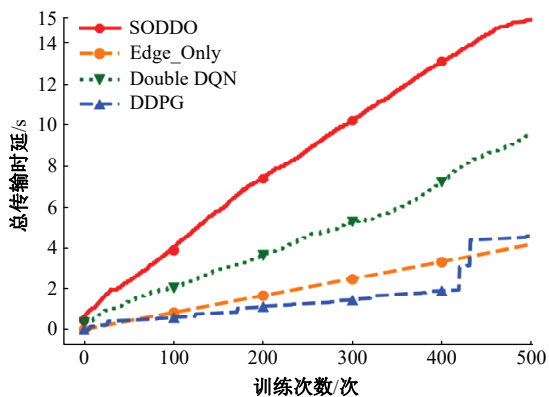


图 11 传输时延比较

Fig. 11 Total transmission delay comparison

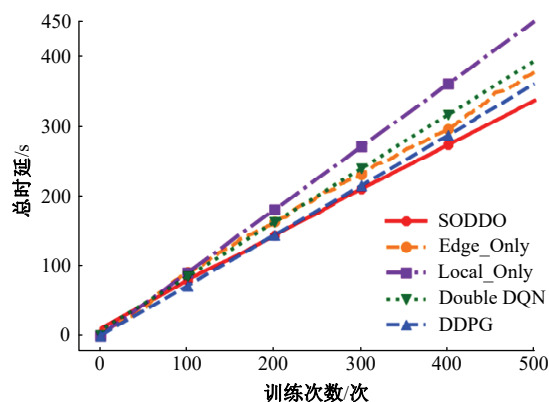


图 12 总时延比较

Fig. 12 Total latency comparison

总体时延要小于其他 4 种对比方法。尽管本文方法在传输时延上和其他 4 种方法相比开销稍大,但由于本文方法在卸载决策过程中的总时延收益大于在传输时延上的损耗,所以在整个计算任务卸载执行的整体时延上要优于其他 4 种方法。

(3) 能耗比较

图 13 对比了 5 种算法在相同任务数量下的总能耗,可以看出,随着训练迭代次数的增加,所有方法的总能耗持续增加。本文方法在达到收敛后,在总体上表现出了比其他 4 种方法更少的能耗趋势。主要原因是由于本文方法采用了二阶任务卸载和计算资源分配策略,能将计算任务的特点和边缘计算资源计算特性较好地匹配,为车载边缘计算任务分配合适的边缘系统计算资源,充分利用边缘计算资源,实现车辆边缘计算任务的高效执行。

(4) 时延能耗缩减率比较

将固定任务数量情况下不同任务卸载方法和基线卸载方法的时延能耗加权值之差与基线卸载方法的时延能耗加权值的比值,定义为时延能耗

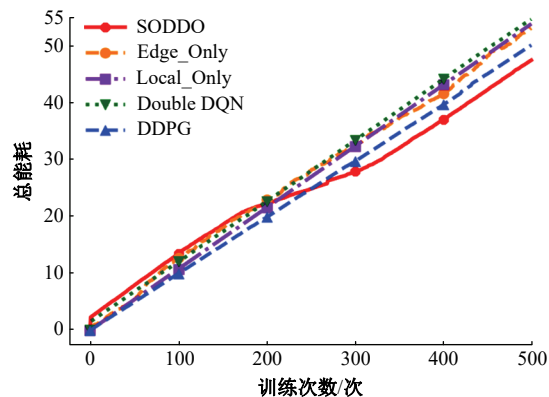


图 13 总能耗比较

Fig. 13 Comparison of total energy consumption

缩减率 (Energy and delay sum reduction ratio, EDSRR), 该指标值越大, 说明边缘计算任务卸载系统的性能越好。本文以本地执行方法作为基线卸载方法, 通过计算本文方法和其他3种方法与本地执行方法之间的时延能耗缩减率, 来进一步验证本文方法的有效性。如图14所示, Double DQN和DDPG时延能耗缩减率都达到了80%以上, 本文方法达到了90.1%, 比Double DQN提高了近6%, 优于其他调度方法。

(5) 不同车辆数量下的算法性能比较

从图15中可以看出, 随着任务车辆数量的增加, 所有方法的时延能耗加权值均呈上升趋势。本文方法在总体上表现出了比其他4种方法更好的性能。主要原因是, 本文方法能够根据任务特性最大化利用系统资源, 实现车辆边缘计算任务的高效执行。

(6) 不同RSU数量下的算法性能比较

从图16中可以看出, 随着RSU数量的增加,

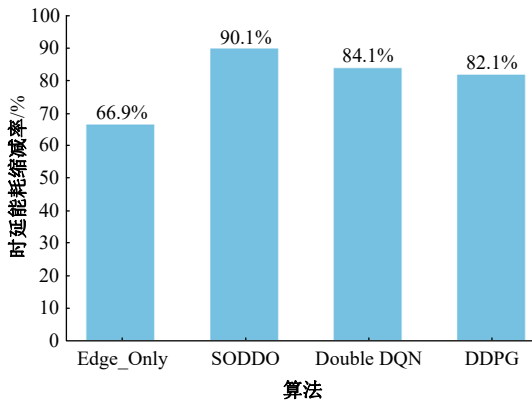


图14 时延能耗缩减率对比

Fig. 14 Comparison of latency and energy consumption reduction rates

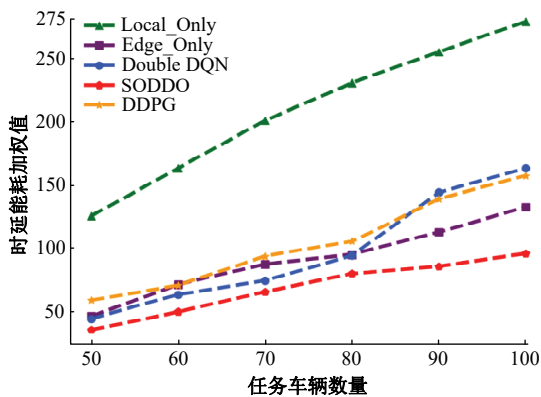


图15 不同车辆数量下的算法性能对比

Fig. 15 Comparison of algorithm performance with different number of vehicles

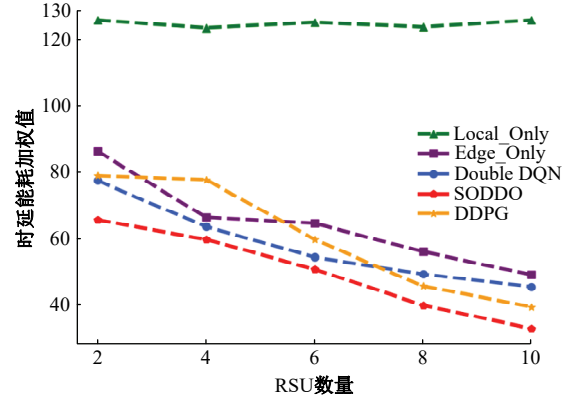


图16 不同RSU数量下的算法性能对比

Fig. 16 Comparison of algorithm performance with different number of RSUs

除全部本地处理算法外, 其余3种方法时延能耗加权值都呈下降趋势。本文方法在总体上表现出了比其他4种方法更好的性能。主要原因是, 本文方法不仅能合理分配计算资源, 最大化系统资源利用率, 还能够实现任务迁移, 进而加快对车辆任务的响应速度。

6 结束语

本文针对车联网边缘计算场景中任务卸载和计算资源分配的离散-连续混合决策问题, 以最小化设备能耗和任务时延加权和为目标, 通过将混合决策问题分解为任务卸载和资源分配两个子问题, 将任务卸载子问题建模为离散问题, 将资源分配子问题建模为连续问题, 提出了一种分布式的二阶决策车联网边缘计算任务动态卸载策略。首先, 设计了包含基于D3DQN的一阶决策网络和基于TD3的二阶决策网络, 可同时分别处理离散和连续动作域的算法框架; 同时, 引入了动作空间搜索优化和动态优先级更新机制, 以进一步提升对动作空间的有效搜索, 提高算法性能; 最终, 在所提算法框架和优化机制的基础上, 设计实现了二阶决策车联网边缘计算任务动态卸载算法。该算法对优化问题进行二阶决策, 得到最优的卸载决策和资源分配策略, 提高了系统的资源利用率。为了满足用户服务需求并尽可能提高用户服务质量, 引入了优先级更新算法, 该算法保证了时延敏感的高优先级任务能被优先调度。实验仿真结果表明: 本文方法在相同仿真实验环境下和其他基线算法相比, 在算法收敛性、时延、能耗各方面都表现出了较好的性能, 可以充分利用边缘计

算系统计算资源,提升车载边缘计算任务卸载和执行效率,更好地满足用户服务请求。

参考文献:

- [1] Sharma S, Kaushik B. A survey on internet of vehicles: applications, security issues & solutions[J]. *Vehicular Communications*, 2019, 20:100182.
- [2] 况博裕, 李雨泽, 顾芳铭, 等. 车联网安全研究综述: 威胁、对策与未来展望[J]. *计算机研究与发展*, 2023, 60(10): 2304-2321.
- Kuang Bo-yu, Li Yu-ze, Gu Fang-ming, et al. Review of internet of vehicle security research: threats, countermeasures, and future prospects[J]. *Journal of Computer Research and Development*, 2023, 60(10): 2304-2321.
- [3] Boban M, Kousaridas A, Manolakis K, et al. Connected roads of the future: use cases, requirements, and design considerations for vehicle-to-everything communications[J]. *IEEE Vehicular Technology Magazine*, 2018, 13(3): 110-123.
- [4] Yang D G, Jiang K, Zhao D, et al. Intelligent and connected vehicles: current status and future perspectives[J]. *Science China Technological Sciences*, 2018, 61: 1446-1471.
- [5] Zhang W Y, Zhang Z J, Chao H C. Cooperative fog computing for dealing with big data in the internet of vehicles: architecture and hierarchical resource management[J]. *IEEE Communications Magazine*, 2017, 55(12): 60-67.
- [6] Jaehwan L, Woongsoo N. A survey on vehicular edge computing architectures[C]//2022 13th International Conference on Information and Communication Technology Convergence(ICTC), Jeju, Island, Korea, 2022: 2198-2200.
- [7] Mao Y Y, You C S, Zhang J, et al. A survey on mobile edge computing: the communication perspective[J]. *IEEE Communications Surveys & Tutorials*, 2017, 19(4): 2322-2358.
- [8] Tang M, Wong V W S. Deep reinforcement learning for task offloading in mobile edge computing systems[J]. *IEEE Transactions on Mobile Computing*, 2022, 21(6): 1985-1997.
- [9] Zhao N, Liang Y C, Niyato D, et al. Deep reinforcement Learning for user association and resource allocation in heterogeneous cellular networks[J]. *IEEE Transactions on Wireless Communications*, 2019, 18(11): 5141-5152.
- [10] Dai Y Y, Xu D, Zhang K, et al. Deep reinforcement learning for edge computing and resource allocation in 5G beyond[C]//2019 IEEE 19th International Conference on Communication Technology, Xi'an, China, 2019: 866-870.
- [11] Raeisi-varzaneh M, Dakkak O, Habbal A, et al. Resource scheduling in edge computing: architecture, taxonomy, open issues and future research directions[J]. *IEEE Access*, 2023, 11: 25329-25350.
- [12] Feng J, Yu F R, Pei Q Q, et al. Joint optimization of radio and computational resources allocation in blockchain-enabled mobile edge computing systems[J]. *IEEE Transactions on Wireless Communications*, 2020, 19(6): 4321-4334.
- [13] 李智勇, 王琦, 陈一凡, 等. 车辆边缘计算环境下任务卸载研究综述[J]. *计算机学报*, 2021, 44(5): 963-982.
- Li Zhi-yong, Wang Qi, Chen Yi-fan, et al. A survey on task offloading research in vehicular edge computing[J]. *Chinese Journal of Computers*, 2021, 44(5): 963-982.
- [14] 许斌, 赵云凯, 朱剑鸣, 等. 移动边缘计算不确定性任务持续卸载及资源分配方法[J]. *软件学报*, 2023, 35(3): 1466-1484.
- Xu Bin, Zhao Yun-kai, Zhu Jian-ming, et al. Continuous offloading and resource allocation method of uncertain tasks in mobile edge computing[J]. *Journal of Software*, 2023, 35(3): 1466-1484.
- [15] Liu G Z, Dai F, Huang B, et al. A collaborative computation and dependency-aware task offloading method for vehicular edge computing: a reinforcement learning approach[J]. *Journal of Cloud Computing*, 2022, 11(1): 1-15.
- [16] 孙彦景, 余政达, 陈瑞瑞, 等. 车联网中基于深度强化学习的高可靠资源分配算法[J]. *重庆邮电大学学报: 自然科学版*, 2023, 35(4): 706-714.
- Sun Yan-jing, Yu Zheng-da, Chen Rui-rui, et al. Deep reinforcement learning based high reliability resource allocation algorithm for internet of vehicles[J]. *Journal of Chongqing University of Posts and Telecommunications(Natural Science Edition)*, 2023, 35(4): 706-714.
- [17] Hao H, Ding W, Zhang W. Time-continuous computing offloading algorithm with user fairness guarantee[J]. *Journal of Network and Computer Applica-*

- tions, 2024, 223: 103826.
- [18] Yan J, Bi S Z, Zhang Y J A. Offloading and resource allocation with general task graph in mobile edge computing: a deep reinforcement learning approach[J]. IEEE Transactions on Wireless Communications, 2020, 19(8): 5404-5419.
- [19] Zhao J H, Li Q P, Gong Y, et al. Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks[J]. IEEE Transactions on Vehicular Technology, 2019, 68(8): 7944-7956.
- [20] Han H, Wu D G, Zhou F H, et al. Dynamic task offloading in MEC-enabled IoT networks: a hybrid DDPG-D3QN approach[C] //2021 IEEE Global Communications Conference(GLOBECOM). Madrid, Spain, 2021: 1-6.
- [21] Guan T, Yao Y S, Chao Y, et al. Energy-efficient computing offloading policy for MEC-assisted power sensor network: a D3QN approach[C]//2023 International Conference on Cyber-Physical Social Intelligence(ICCSI), Xi'an, China, 2023: 180-185.
- [22] Wang Z Y, Schaul T, Hessel M, et al. Dueling Network Architectures for Deep reinforcement learning [C]//JMLR. org. Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48, New York, NY, USA, 2016: 1995-2003.
- [23] Hasselt H V, Guez A, Silver D. Deep reinforcement learning with double Q-learning[C]//AAAI'16. Proceeding of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, Arizona, 2016: 2094-2100.
- [24] 巨涛, 王志强, 刘帅, 等. D3DQN-CAA: 一种基于DRL的自适应边缘计算任务调度方法[J]. 湖南大学学报: 自然科学版, 2024, 51(6): 73-85.
Ju Tao, Wang Zhi-qiang, Liu Shuai, et al. D3DQN-CAA: a DRL-based adaptive edge computing task scheduling method[J]. Journal of Hunan University(Natural Sciences), 2024, 51(6): 73-85.
- [25] 张红霞, 吕智豪, 席诗语, 等. 面向绿色计算的车辆协同任务卸载方法[J]. 电子与信息学报, 2024, 48(1): 175-183.
Zhang Hong-Xia, Lv Zhi-hao, Xi Shi-yu, et al. A method for offloading vehicle collaborative tasks for green computing[J]. Journal of Electronics & Information Technology, 2024, 48(1): 175-183.
- [26] Kiran B R, Sobh I, Talpaert V, et al. Deep reinforcement learning for autonomous driving: A survey [J]. IEEE Transactions on Intelligent Transportation Systems, 2022, 23(6): 4909-4926.
- [27] Zhou D R, Li L H, Gu Q Q. Neural contextual bandits with ucb-based exploration[C] //International Conference on Machine Learning. 2020: 11492-11502.
- [28] Zhong J W, Chen X, Jiao L B. A fair energy-efficient offloading scheme for heterogeneous fog-enabled IoT using D3QN[C]//2023 25th International Conference on Communication Software and Networks(ICCSN), IEEE, 2023: 47-52.
- [29] Min M H, Xiao L, Chen Y, et al. Learning-based computation offloading for IoT devices with energy harvesting[J]. IEEE Transactions on Vehicular Technology, 2019, 68(2): 1930-1941.
- [30] Geng L W, Zhao H B, Wang J Y, et al. Deep-reinforcement-learning-based distributed computation offloading in vehicular edge computing networks[J]. IEEE Internet of Things Journal, 2023, 10(14): 12416-12433.
- [31] 郭晓东, 郝思达, 王丽芳. 基于深度强化学习的车辆边缘计算任务卸载方法[J]. 计算机应用技术, 2023, 40(9): 2803-2807, 2814.
Guo Xiao-dong, Hao Si-da, Wang Li-fang. Task offloading method based on deep reinforcement learning for vehicular edge computing[J]. Application Research of Computers, 2023, 40(9): 2803-2807, 2814.
- [32] 邝祝芳, 陈清林, 李林峰, 等. 基于深度强化学习的多用户边缘计算任务卸载调度与资源分配算法[J]. 计算机学报, 2022, 45(4): 812-824.
Kuang Zhu-fang, Chen Qing-li, Li Li-feng, et al. Multi-user edge computing task offloading scheduling and resource allocation based on deep reinforcement learning[J]. Chinese Journal of Computers, 2022, 45(4): 812-824.
- [33] 许小龙, 方子介, 齐连永, 等. 车联网边缘计算环境下基于深度强化学习的分布式服务卸载方法[J]. 计算机学报, 2021, 44(12): 2382-2405.
Xu Xiao-long, Fang Zi-jie, Qi Lian-yong, et al. A deep reinforcement learning-based distributed service offloading method for edge computing empowered internet of vehicles[J]. Chinese Journal of Computers, 2021, 44(12): 2382-2405.