计算机科学
COMPUTER SCIENCE

基于异构节点的高效任务卸载策略

范兴刚, 姜新阳, 谷文婷, 徐骏涛, 杨友东, 李强

引用本文

范兴刚, 姜新阳, 谷文婷, 徐骏涛, 杨友东, 李强. 基于异构节点的高效任务卸载策略[J]. 计算机科学, 2025, 52(8): 354-362.

FAN Xinggang, JIANG Xinyang, GU Wenting, XU Juntao, YANG Youdong, LI Qiang. Effective Task Offloading Strategy Based on Heterogeneous Nodes [J]. Computer Science, 2025, 52(8): 354-362.

相似文章推荐（请使用火狐或 IE 浏览器查看文章）

Similar articles recommended (Please use Firefox or IE to view the article)

面向多用户的任务卸载和服务缓存策略研究

Research on Multi-user Task Offloading and Service Caching Strategies

计算机科学, 2025, 52(7): 307-314. https://doi.org/10.11896/jsjkx.240500036

基于元强化学习的任务卸载优化策略

Optimization Strategy of Task Offloading Based on Meta Reinforcement Learning

计算机科学, 2025, 52(6A): 240800050-8. https://doi.org/10.11896/jsjkx.240800050

移动机器人路径规划算法综述

Review of Path Planning Algorithms for Mobile Robots

计算机科学, 2025, 52(6A): 240900074-10. https://doi.org/10.11896/jsjkx.240900074

无线可充电传感器网络中异构感知的限时移动充电调度

Time-constrained Mobile Charging Scheduling for Heterogeneous Sensing in Wireless Rechargeable Sensor Networks

计算机科学, 2025, 52(6): 355-364. https://doi.org/10.11896/jsjkx.240400186

结合开发者依赖的图神经网络缺陷预测方法

Graph Neural Network Defect Prediction Method Combined with Developer Dependencies

计算机科学, 2025, 52(6): 52-57. https://doi.org/10.11896/jsjkx.240700119

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240700088

基于异构节点的高效任务卸载策略

范兴刚[1] 姜新阳[1] 谷文婷[1] 徐骏涛[1] 杨友东[2] 李 强[1]
1 浙江工业大学计算机科学与技术学院 杭州 310023
2 浙江工业大学之江学院 浙江 绍兴 312030
(xgfan@zjut.edu.cn)

摘 要 在车联网边缘计算中,如何利用有限的网络资源实施高效的任务卸载,是近年来车联网的研究热点。通过研究异构节点模式下的任务卸载,设计了一种异构节点模式下的高效任务卸载策略 TOS-HN。当车辆产生任务时,优先考虑移动节点卸载,将任务卸载到附近空闲车辆上。若移动卸载不能满足任务需求,则采用固定节点卸载策略。在移动节点卸载模式中,先根据任务处理时延和能耗构建代价矩阵,再通过匈牙利算法确定任务车辆和处理车辆的最优匹配。仿真实验证明,TOS-HN 算法相比于其他算法具有显著优势,在时延、能耗、任务成功率和基站负载方面均具有较好的性能。

关键词:车联网边缘计算;任务卸载;异构节点模式;移动;代价矩阵
中图分类号 TP393

Effective Task Offloading Strategy Based on Heterogeneous Nodes

FAN Xinggang[1],JIANG Xinyang[1],GU Wenting[1],XU Juntao[1],YANG Youdong[2] and LI Qiang[1]
1 College of Computer Science and Technology,Zhejiang University of Technology,Hangzhou 310023,China
2 Zhijiang College of Zhejiang University of Technology,Shaoxing,Zhejiang 312030,China

Abstract In vehicular edge computing(VEC),how to use the limited network resources to implement efficient task unloading is a research hotspot in recent years. This paper focuses on task offloading in the heterogeneous node mode and designs an efficient task offloading strategy in heterogeneous node mode(TOS-HN). When a vehicle generates a task,mobile node offloading is prioritized to offload the task to a nearby idle vehicle. If the mobile offloading cannot meet the task requirements,a fixed node offloading strategy is adopted. In the mobile node offloading mode,the cost matrix is first constructed based on the task processing delay and energy consumption,and then the Hungarian algorithm is used to determine the optimal matching between the task vehicle and the processing vehicle. Simulation experiment proves that the TOS-HN algorithm has significant advantages over other algorithms,with better performance in terms of delay,energy consumption,task success rate and base station load.

Keywords Vehicular edge computing,Task offloading,Heterogeneous node mode,Mobile,Cost matrix

1 引言

近年来,车辆边缘计算 VEC 作为解决高计算需求与能源消耗问题的关键技术,正逐渐成为车联网研究领域的热点,其中任务卸载领域面临的挑战尤为突出[1-8]。Lee 等[9]利用停放车辆,将有限的雾资源分配给车辆应用,从而最大限度地减少服务延迟。Zhou 等[10]研究了动态多用户 MEC 系统中计算卸载和资源分配的联合优化问题,并提出了一种基于值迭代的强化学习方法,以确定计算卸载和资源分配的联合策略。同样,Zhang 等[11]和 Xu 等[12]将任务卸载问题视为多目标优化问题,并采用各种多目标进化算法来寻找最优卸载策略。Sun 等[13]根据任务传输路线寻找卸载位置,利用进化算法进行多目标优化,以找到最优卸载策略。Zhao 等[14]提出了一种深度确定性策略梯度(DDPG)算法,用于处理高维连续动作空间中的任务卸载与调度优化。

Zhu 等[15]提出了一种新方案,通过综合考虑雾容量、服务时延和质量损失来优化车辆雾计算(Vehicle Fog Computing,VFC)中的资源分配,从而确保满足性能指标,同时实现资源的高效利用并完成任务。Qiao 等[16]设计了任务迁移计算卸载(TMCO)算法,利用车辆移动轨迹信息进行动态评估,并选择最佳边缘服务器执行任务卸载,该方案满足时延要求且显著提升了计算效率。Zhang 等[17]提出了一种支持移动边缘计算(Mobile Edge Computing,MEC)的车联网(IoV)架构,将 MEC 服务器和车辆作为卸载节点,并联合路边单元提供低时延服务。此外,他们还提出任务卸载策略,通过优化卸载节点选择和任务分类最小化任务完成时延。为解决中心化

到稿日期:2024-07-15 返修日期:2024-10-16
基金项目:浙江省"尖兵"研发攻关计划(2023C01029,2025C01054)
This work was supported by the "Pioneer" and "Leading Goose" R&D Program of Zhejiang(2023C01029,2025C01054).
通信作者:杨友东(yydong@zjut.edu.cn)

数据处理效率低和边缘设备资源利用不足的问题，Zhang等[18]提出了一种基于自适应分层采样的边缘计算架构。该架构在确保计算效率和精度的同时，高效分析物联网数据，降低了时延并提升了资源利用率。

以上研究主要考虑任务的高可靠性和低时延，忽略了能源效率问题。事实上，能源消耗也是任务卸载过程中需要考虑的一个关键因素。

车辆的移动性使得通信环境发生动态变化，这对通信系统的稳定性和效率构成了重大挑战。因此，研究最佳卸载策略以节约能源变得尤为重要。

Jang等[19]提出了一种新的节能任务卸载策略，该策略综合考虑车辆移动性、任务截止时间和能耗因素，通过优化VEC中的位分配和卸载比例，在满足任务时延约束的同时降低了总能耗。这一策略为智能交通系统的能效管理提供了理论和实践支持。Li等[20]提出了一种将移动边缘计算与能量收集技术相结合的算法，实现了计算密集型任务的高效卸载和移动设备的充电功能并存。该算法在满足任务时延需求的同时，有效地降低了整体能耗。Lu等[21]研究了大规模网络场景，提出了一种优化任务计算时延和能源效率的策略。该策略通过优化任务分配，最大化用户卸载效用，减少了时延并提高了能源效率，从而提升了网络性能和用户体验。另外，由于现有边缘计算技术对资源（如计算、通信、缓存）的控制不够细粒度，因此无法有效支持延迟敏感的实时服务。Wang等[22]设计了一种基于软件定义的细粒度多接入边缘计算架构，能够对网络和计算资源进行精细控制和协同管理，并提出了一种基于深度强化学习Q-Learning的两级资源分配策略，以实现更高效的计算卸载和服务增强。

尽管上述工作在提升用户体验质量方面取得了显著成效，但未考虑资源有限这一根本性问题。资源的稀缺性是制约系统性能进一步提升的关键因素。在资源稀缺的网络环境中，高效的资源分配和利用成为课题研究的关键。

Zeng等[23]研究了如何有效、经济地利用志愿者车辆中的闲置资源来处理VEC服务器中的超载任务。Men等[24]提出一种基于模糊逻辑的车联网通信网络二进制卸载方案，该方案基于模糊逻辑，通过综合考虑相邻车辆之间的计算能力、移动性和链路质量，对邻近车辆进行评估，以确保二进制卸载应用任务的高完成率。Huang[25]等研究了在任务卸载过程中出现局部最优的问题。Raza等[26]分析了VEC场景中的计算效率，提出了一种基于移动性感知计算效率的任务卸载和资源分配方案。Fan等[1]提出一种异构蜂窝网络协作VEC架构，该架构将5G基站与广泛可用的4G基站相结合以支持车辆边缘计算，并设计了一种在固定节点模式下的任务卸载方案（TOS-FN），通过联合考虑任务卸载与资源分配实现最小化任务执行总时延和能耗加权和的目的。

综上所述，任务卸载领域面临的两大主要挑战是能耗管理和资源有限性。能耗问题要求在确保低时延的同时最大化能源利用率，而资源有限性迫使我们优化资源分配，以满足多样化的需求。这两方面的挑战制约了系统性能的提升，亟待解决，以实现高效和可持续的车联网任务卸载。

在车联网任务卸载计算的研究中，空闲车辆具备作为移动边缘计算节点的潜力，能够为系统提供低时延和高可靠性的服务。为应对VEC中基站高负载的问题，并进一步降低任务完成时延和总体能耗，提高基站计算效率，本文在文献[1]的基础上引入了移动节点卸载。考虑到车辆移动性，本文设计了一种在异构节点模式下的任务卸载策略，旨在充分利用道路上的空闲车辆作为卸载节点，实现资源的优化配置和高效利用。

2 系统模型

网络架构如图1所示。图1中，车辆都是移动的，产生任务的车辆称为任务车辆，空闲的车辆称为处理车辆。为便于研究，本文假设：

1）基站是异构的，5G基站与4G基站合二为一。但5G的通信范围（gNB）小于4G的通信范围（eNB），具体如图1所示。

2）行驶在城市道路上的车辆不仅可以接收LTE（Long Term Evolution，长期演进，4G通信方式）和5G NR服务，还可以进行车辆之间的相互通信。图1中，一辆车采用LTE与基站通信，另一辆车采用5G NR与基站通信。车辆之间采用LTE通信。

3）处理车辆的总数量M总是不大于任务车辆总数N，即$M \leqslant N$。例如图1中，任务车辆$N=5$，处理车辆$M=3$。

4）某一时刻内，1个任务车辆只能选择1个处理车辆进行任务卸载，1个处理车辆也只能接收1个任务车辆的卸载请求。

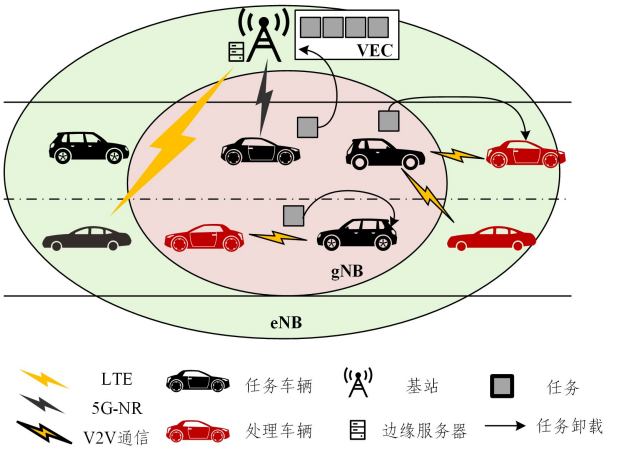

图1 异构节点卸载架构

Fig. 1 Heterogeneous node offloading architecture

定义1（任务） 任务被建模为三元组$\Phi_i=\{c_i,d_i,T_i\}$且不可分割，其中c_i表示完成任务Φ_i所需的总CPU周期数，d_i表示需要处理的输入数据大小，T_i表示Φ_i的最大容忍时延。

引入变量$\omega_{i,k}\in\{0,1\}$表示移动节点卸载模式下的任务卸载决策，当$\omega_{i,k}=1$时，表示处理车辆k将执行任务车辆i所产生的任务Φ_i，否则，Φ_i将继续在固定节点卸载模式下进一步判断其执行方式。

定义2（移动节点卸载模式） 某一时刻内，每个任务车辆只能选择1个处理车辆进行任务卸载，1个处理车辆也只能接收1个任务车辆的卸载请求。移动节点卸载模式如图2所示。

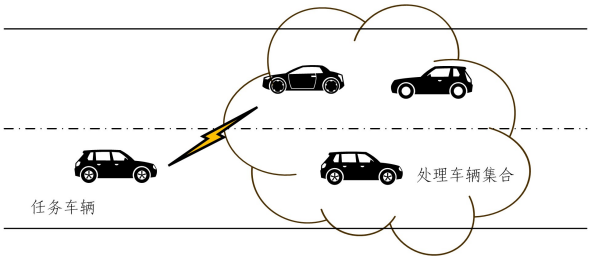

图 2 移动节点卸载架构

Fig. 2 Mobile node offloading architecture

在上述移动节点卸载模式下，可以得到本文的通信模型、时延模型以及能耗模型。

通信模型：将任务卸载到处理车辆执行，首先确保该处理车辆在其任务车辆的通信范围之内。假设处理车辆 k 具有坐标 $(x_k^{\text{pveh}}, y_k^{\text{pveh}})$，任务车辆 i 和处理车辆 k 之间的距离为：

$$g_{i,k}=\sqrt{(x_i^{\text{veh}}-x_k^{\text{pveh}})^2+(y_i^{\text{veh}}-y_k^{\text{pveh}})^2} \tag{1}$$

为了保证车辆之间传输数据的稳定性和连续性，携带任务的车辆、任务处理车辆之间的通信采用单跳自组的网络形式。任务 i 卸载到处理车辆 k 的传输速率为：

$$R_{i,k}=B_{i,k}\log_2\left(1+\frac{P_i z_{i,k}}{\sigma^2}\right) \tag{2}$$

其中，$B_{i,k}$ 为任务请求车辆 i 和处理车辆 k 之间可用频谱的带宽；P_i 表示车辆 i 的上行传输功率；$z_{i,k}$ 为任务车辆 i 和处理车辆 k 之间无线传播信道的信道增益；σ^2 为高斯白噪声功率[27]。

时延模型：当车辆产生任务时，优先考虑将其卸载到附近的处理车辆上执行。定义处理车辆 k 分配给任务 Φ_i 的计算资源为 $f_{i,k}$，假设在某一时隙内，每个处理车辆只能向 1 个任务车辆提供卸载计算服务，上行链路发送任务 Φ_i 的传输时延为：

$$D_{i,k}^t=\frac{d_i}{R_{i,k}} \tag{3}$$

任务 i 在处理车辆 k 上的执行时延为：

$$D_{i,k}^c=\begin{cases}\dfrac{c_i}{f_{i,k}}, & f_{i,k}\neq 0\\ 0, & f_{i,k}=0\end{cases} \tag{4}$$

则移动节点卸载模式下的总时延为：

$$D_{i,k}=D_{i,k}^t+D_{i,k}^c \tag{5}$$

能耗模型：当任务选择卸载到空闲车辆上执行时，任务车辆在向处理车辆传输数据的过程中以及车辆在空闲状态下都会产生一定的能量损失。车辆 i 在任务执行过程中的相应能耗定义为：

$$E_{i,k}=P_i^T D_{i,k}^t+P_i^I D_{i,k}^c \tag{6}$$

定义 3（异构节点卸载模式） 在此模式下，卸载终端包括基站以及请求车辆附近的空闲车辆，即处理车辆。当请求车辆产生任务时，为了最大化计算效率的同时有效降低基站压力，优先考虑移动节点卸载，将任务卸载到处理车辆上；若移动卸载不能满足任务需求，便结合固定节点卸载模式[1]确定该任务的最终计算方式，即判断卸载到基站执行还是在本地进行处理。

图 1 中，2 个车辆选择了固定卸载方式，2 个任务车辆选择了移动卸载，并且有 1 个任务车辆需要 2 个处理车辆。

本文在文献[1]的基础上，进一步深化研究移动节点模式下的任务卸载决策问题。目标在于通过科学合理的任务卸载策略，进一步降低任务执行的总时延和能耗，从而提升整个系统的性能与效率。

基于移动节点卸载模式下的时延模型和能耗模型，任务车辆 i 卸载到处理车辆 k 的计算代价可表示为：

$$h_{i,k}=\gamma D_{i,k}/1+(1-\gamma)E_{i,k}/1 \tag{7}$$

其中，$i(0\leqslant i\leqslant N)$ 和 $k(0\leqslant k\leqslant M)$ 分别表示任务车辆和处理车辆，$h_{i,k}$ 表示将任务卸载到处理车辆计算所产生的代价，$D_{i,k}$ 和 $E_{i,k}$ 分别表示计算任务 Φ_i 在处理车辆 k 上的执行时延和能耗，分别用式(7)和式(8)计算。之后将 $D_{i,k}$ 和 $E_{i,k}$ 规范化，即 $D_{i,k}/1$ 和 $E_{i,k}/1$ 分别表示执行时延除以单位时延、能耗除以单位能耗。γ 代表时延和能耗之间的权重参数。式中的权重参数主要用于更好地适应用户的需求。在不同的车联网背景下，时延和能耗的要求不一致。当用户电量较低时，对能耗的要求较高；当产生紧急需求时，对时延的要求较高。

定义 4（移动节点卸载问题） 此问题可建模为式(8)—式(12)。

$$\min \sum_{i=1}^{N}\sum_{k=1}^{M}\omega_{i,k}h_{i,k} \tag{8}$$

$$\text{s.t. } f_{i,k}\geqslant 0, \forall i\in N, \forall k\in M \tag{9}$$

$$\sum_{i=1}^{N}\omega_{i,k}\leqslant 1, \forall k\in M \tag{10}$$

$$\sum_{k=1}^{M}\omega_{i,k}\leqslant 1, \forall i\in N \tag{11}$$

$$\omega_{i,k}=1 \text{ 或 } 0 \tag{12}$$

根据式(7)可以得出一个 i 行 k 列的代价矩阵，$h_{i,k}$ 为该代价矩阵中的一个元素，用于表示任务车辆 i 是否与处理车辆 k 匹配成功。如果任务车辆 i 最终将任务卸载到处理车辆 k 进行计算，则 $\omega_{i,k}=1$，否则，$\omega_{i,k}=0$。式(9)定义了处理车辆 k 分配给任务车辆 i 的计算资源不能为负，$f_{i,k}$ 表示处理车辆 k 分配给任务 Φ_i 的计算资源。式(10)和式(11)分别表示在某一时间段内，每个处理车辆只能服务于 1 个任务车辆，每个任务车辆也只能选择 1 个处理车辆进行匹配。

本文所用主要符号及其定义如表 1 所列。

表 1 符号定义

Table 1 Symbol definition

参数	定义
Φ_i	任务，表示为三元组 $\{c_i, d_i, T_i\}$
$\omega_{i,k}$	任务卸载决策
d_i	任务 Φ_i 需要处理的输入数据大小
c_i	完成任务 Φ_i 所需的总 CPU 周期数
T_i	表示任务 Φ_i 的最大容忍时延
$R_{i,k}$	任务 i 卸载到处理车辆 k 的传输速率
$D_{i,k}$	计算任务 Φ_i 在处理车辆 k 上的执行时延
$E_{i,k}$	计算任务 Φ_i 在处理车辆 k 上的能耗
$h_{i,k}$	将任务卸载到处理车辆计算所产生的代价
γ	时延和能耗之间的权重参数
D_i	第 i 车辆在基站通信范围内行驶的距离
v_i	第 i 车辆的行驶速度
f_i^I	分配给任务 Φ_i 的可用 CPU 周期
P_i^{veh}	任务车辆 i 的位置
$f_{i,k}$	处理车辆 k 分配给任务 Φ_i 的计算资源
P_k^{pveh}	处理车辆 k 的位置
M	处理车辆数目
N	任务车辆数目
t_i	车辆 i 在特定基站通信范围内的驾驶时间
$g_{i,k}$	任务车辆 i 与潜在处理车辆 k 之间的距离
D_{veh}	任务车辆的通信范围
F_i	任务车辆 i 通信范围内的处理车辆集合
L_{eNB}	4G(eNB)的通信范围
L_{gNB}	5G(gNB)的通信范围

3 异构节点模式下的任务卸载策略

为了找到任务车辆与处理车辆之间的最优匹配策略，以最小化总任务时延与能耗的加权和，首先引入代价矩阵构建算法，确定每个任务的处理车辆卸载集合，并计算该集合内每个处理车辆执行该任务时的时延和能耗。通过该算法获得代价矩阵后，再利用加权匈牙利算法[28]进行匹配计算，从而得出最优的匹配决策。

完成移动节点卸载模式下任务车辆与处理车辆的最优匹配后，进一步筛选出需要在固定节点卸载模式下执行卸载的任务集合。随后，结合任务分类算法和分支定界算法，可以有效地获取全局最优的卸载决策和资源分配策略。这就是异构节点模式下的任务卸载策略(Task Offloading Strategy in Heterogeneous Node Mode，TOS-HN)。

3.1 TOS-HN 算法流程

异构节点模式下的任务卸载算法如算法 1 所示。其中，三元组$\Phi_i=\{c_i,d_i,T_i\}$表示任务，任务车辆 i 的位置表示为$P_i^{\text{veh}}=(x_i^{\text{veh}},y_i^{\text{veh}})$，$i\in N$；处理车辆 k 的位置表示为$P_k^{\text{pveh}}=(x_k^{\text{pveh}},y_k^{\text{pveh}})$，$k\in M$；$f_{i,k}$表示处理车辆 k 分配给任务Φ_i的计算资源；f_i^l为分配给任务Φ_i的可用 CPU 周期；D_i表示第 i 个车辆在基站通信范围内行驶的距离，其速度用v_i表示，在任务分类算法中用来计算车辆 i 在特定基站通信范围内的驾驶时间t_i。

算法 1 TOS-HN 算法

输入：$\Phi_i=\{c_i,d_i,T_i\}$，D_i，v_i，f_i^l，P_i^{veh}，$f_{i,k}$，P_k^{pveh}，i=1…N，k=1…M

输出：最优分配矩阵 $\boldsymbol{\omega}$，最优卸载决策 OPT

```
1. 移动节点卸载模式
2.    代价矩阵构建算法
3.       获得代价矩阵：H=(h_{i,k})_{i×k}
4.    匈牙利算法(输入代价矩阵 H)
5.       获得分配矩阵 ω=(ω_{i,k})_{i×k}
6. 固定节点卸载模式
7.    for i=1 to N do
8.       for k=1 to M do
9.          if(Σ_{i=1}^{N} Σ_{k=1}^{M} ω_{i,k}=0)
10.            获得固定节点卸载模式下的任务集合
11.            任务分类算法
12.         end
13.      end
14.   end
15.   分支定界算法
```

3.2 代价矩阵构建算法

在某一特定时隙内，首先需要识别并筛选出当前未产生任务且具备空闲计算能力的车辆，这些车辆将作为潜在的处理车辆。随后，需要遍历任务车辆集合。针对每一个任务车辆 i，需根据式(1)计算其与所有潜在处理车辆 k 之间的距离$g_{i,k}$。这一计算过程旨在判断处理车辆 k 是否在其通信范围D_{veh}之内。若满足通信条件，则将处理车辆 k 加入集合F_i中，并依据文献[1]中的式(7)和式(8)计算任务Φ_i在处理车辆 k 上的执行时延$D_{i,k}$和能耗$E_{i,k}$。最终，利用式(1)得出处理车辆 k 执行任务Φ_i的综合代价$h_{i,k}$，这一代价将作为后续决策优化的重要依据。代价矩阵构建算法如算法 2 所示。

算法 2 代价矩阵构建算法

输入：$\Phi_i=\{c_i,d_i,T_i\}$，D_{veh}，P_i^{veh}，P_k^{pveh}，$f_{i,k}$，i=1…N，k=1…M

输出：代价矩阵 $\mathbf{H}=(h_{i,k})_{i\times k}$

```
1. for i=1 to N do
2.    for k=1 to M do
3.       计算任务车辆 i 和处理车辆 k 之间的距离 g_{i,k}
4.       If(g_{i,k}<=D_veh/2)
5.          F_i←k
6.          计算 D_{i,k} 和 E_{i,k}，获得执行代价 h_{i,k}
7.       end
8.    end
9. end
```

3.3 匈牙利算法

为了找到任务车辆与处理车辆之间的最优匹配策略，以最小化总任务时延与能耗的加权和，在利用代价矩阵构建算法获得代价矩阵后，再利用加权匈牙利算法进行匹配计算，从而得出任务车辆和处理车辆的最优匹配。匈牙利算法流程如图 3 所示。

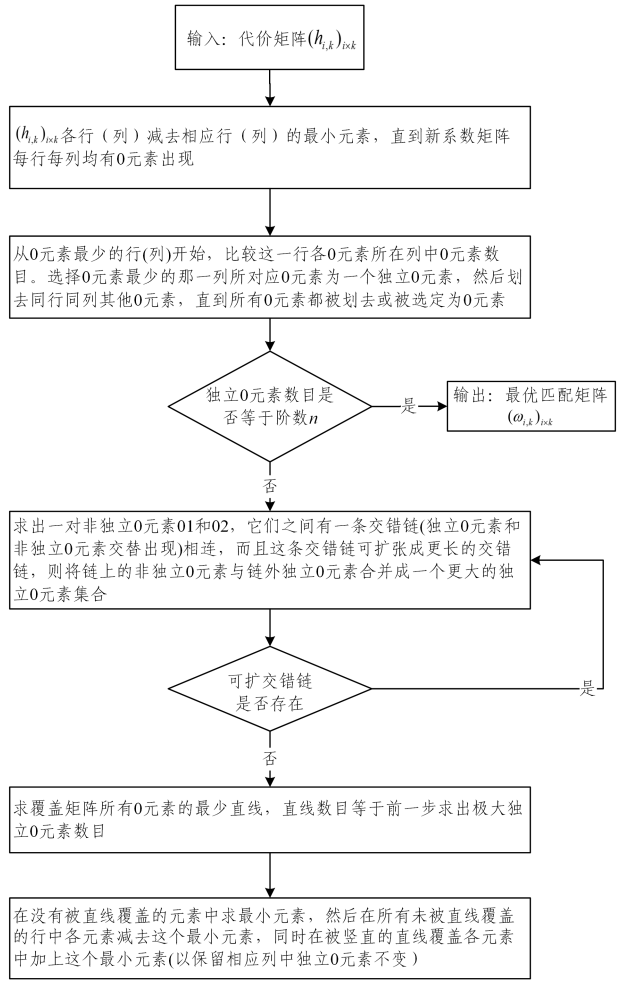

图 3 匈牙利算法流程

Fig. 3 Flowchart of Hungarian algorithm

3.4 固定节点卸载模式

对于移动节点卸载模式下没有获得资源的任务，进一步运用文献[1]提供的方法进行固定节点卸载。先利用任务分

类算法(见算法3)对任务进行分类,再运用线性松弛改进的分支定界算法找到原始问题的最优解。具体过程见文献[1]。

算法3 任务分类算法

输入:$\Phi_i=\{c_i,d_i,T_i\}$,$i=1\cdots N$,D_i,v_i,P_i^{veh}

输出:Ω_i

```
for i=1 to N do
    计算任务车辆 i 的通信时间 t_i = D_i / v_i
    if 任务车辆 i 位于 eNB 的通信范围内,而不是在 gNB 的通信范围内
        选择单基站卸载模式
        Ω_i ← Φ_i = {c_i, d_i, T_i}
    end
    if 任务车辆 i 位于 gNB 的通信范围内
        if 任务车辆的通信时间小于或等于该任务的最大容忍时延
            选择单基站卸载模式
        else
            选择双基站卸载模式
            Ω_i ← Φ_i = {c_i, d_i, T_i}
        end
    end
return Ω_i
end
```

3.5 TOS-HN 算法时间复杂度分析

在移动节点卸载模式中,首先需要针对每个任务车辆寻找处理车辆集合 F,因此时间复杂度为 $O(NM)$。匈牙利算法的时间复杂度为 $O(|V||E|)$,因为每查找一条增广链所需时间复杂度为 $O(|E|)$,对二分图 $G=(V,E)$ 的 V 个顶点依次查找增广链需要查找 V 次。映射到移动节点卸载模型中,二分图可表示为 $G=(M+N,M)$,其算法复杂度为 $O((M+N)\cdot M)$。因此移动节点卸载模式下的时间复杂度为 $O((M+N)\cdot M)$。由文献[1]可知,固定节点卸载模式下的时间复杂度为 $O(2^N)$。综合上述两种卸载模式下的算法,可知异构节点卸载模式下的时间复杂度为 $O(2^N)$。

匈牙利算法执行时间分析:本文测试了不同数量车辆对匈牙利算法的执行时间,具体硬件及软件环境见第4章。在此分析中,为了展示出采用匈牙利算法的计算成本,本文仅采用移动节点卸载,而不采用固定节点卸载[1]。

从表2可以看出,随着问题规模的增加,匈牙利算法的执行时间明显增加,这与其 $O((M+N)\cdot M)$ 的时间复杂度一致。在车辆数量(即二分图顶点数目)为10或20时,图的规模较小,匈牙利算法能够迅速找到增广链,从而使得执行时间小于20 s,满足大多数实际应用场景的需求。

表2 匈牙利算法的执行时间

Table 2 Execution time of the Hungarian algorithm

任务车辆数量	执行时间/s
10	4.98
20	18.50
30	45.25
40	85.04

随着车辆数量增加,二分图的顶点数量增多,增广链的构造变得更复杂,搜索空间扩大。这导致每次找到增广链的时间增加,且增广链数量和长度的增加进一步提高了构造和维护的时间,从而显著提升了算法的时间复杂度。可以得出,当执行时间显著增加,时间复杂度的增长呈超线性趋势。

需要注意的是,测试中仅采用了移动节点卸载模式,这是一种较为极端的边缘服务器瘫痪情况。实际应用中,大多数任务会被卸载到边缘服务器,因此上述时间消耗仍在可接受范围内。此外,实际场景中不会出现极端密集的车辆情况,本文所考虑的情况已经涵盖了大多数实际应用情境。综上所述,当车辆数量处于合理范围内时,算法仍能表现出良好的性能。

3.6 TOS-HN 算法空间复杂度分析

TOS-HN 算法输入数据包括任务 $\Phi_i=\{c_i,d_i,T_i\}$,车辆行驶距离与速度 D_i,v_i,以及其他任务车辆相关参数 f_i^l,P_i^{veh},$f_{i,k}$,P_k^{pveh}。其中 N 是任务车辆的数目,M 是处理车辆的数目。假设每个输入参数占用常数空间 $O(1)$,则存储所有输入数据所需的空间为 $O(N+M+N\times M)=O(N\times M)$(因为有些参数如 $f_{i,k}$,是按任务和处理车辆的组合计算的)。

移动节点卸载模式中的代价矩阵构建算法的目的是生成一个代价矩阵 $\boldsymbol{H}=(h_{i,k})_{i\times k}$,其中每个元素 $h_{i,k}$ 代表任务车辆 i 和处理车辆 k 之间的代价。构建该代价矩阵需存储 $O(N\times M)$ 个代价值,因此代价矩阵 $\boldsymbol{H}$ 的空间复杂度为 $O(N\times M)$。其余的临时变量如 $h_{i,k}$ 和 $g_{i,k}$ 等,均为常数空间 $O(1)$,并在每次迭代中复用这些空间。匈牙利算法主要依赖于代价矩阵 $\boldsymbol{H}$ 和增广路径等的临时存储,故匈牙利算法的空间复杂度也为 $O(N\times M)$,其主要用于存储代价矩阵和匹配的状态。

固定节点卸载模式中任务分类算法主要基于每个任务车辆的通信时间和基站信息来决定卸载模式。临时变量如 t_i,Ω_i 等,这些变量和任务车辆数量 N 成比例关系,即空间复杂度为 $O(N)$。分支定界算法的空间复杂度主要取决于问题规模,即任务车辆和处理车辆的数量。该算法需要维护一棵搜索树,每个节点代表一个可能的分配方案,因此其空间复杂度为 $O(N\times M)$,与匈牙利算法和代价矩阵构建算法的空间复杂度一致。

最后,输出数据的最优分配矩阵 $\boldsymbol{\omega}$ 以及最优卸载决策 OPT 的空间复杂度分别为 $O(N\times M)$ 和 $O(1)$。

综合上述各部分的空间复杂度可知,最主要算法的空间复杂度是由输入数据、代价矩阵 $\boldsymbol{H}$、分配矩阵 $\boldsymbol{\omega}$ 以及匈牙利算法的需要共同决定的。因此,TOS-HN 算法的总体空间复杂度为 $O(N\times M)$。

4 实验仿真

实验考虑一条长度为3000 m的双向道路,沿路设有共址基站。每个基站都配有一个 VEC 服务器。边缘服务器的计算能力至少为5 GHz,任务车辆的 CPU 频率为0.8 GHz,而处理车辆的 CPU 频率为1.5 GHz。根据系统模型,有 N 辆车同时向附近处理车辆和边缘服务器发送任务处理请求。此外,本文与文献[1]的相关参数设置相同,车辆速度为[30,80] km/h,且车辆在路上以恒定速度行驶。任务大小和所需计算资源均遵循高斯分布:$d_i\sim N(900,300)$ kB,$c_i\sim N(2000,200)$ MHz。每个计算任务的最大时延约束从均匀分布中随机生成,$T_i\sim U[1,5]$ s。在一台配备了 Intel(R) 核心(TM) i5-

13500H CPU，2.90 GHz 和 32 GB RAM 的笔记本电脑上进行仿真实验。开发工具是安装在 Windows 11 Home 64 位平台上的 MATLAB R2022b。模拟在 MATLAB 中进行，模拟使用的通信参数如表 3 所列。

表 3 实验参数

Table 3 Experimental parameter

参数	值
L_{eNB}	1 000 m
L_{gNB}	300 m
d_i	N(900,300) kB
c_i	N(2000,200) MHZ
T_i	$U[1,5]$s
$B_{i,eNB}$	1 MHz
$B_{i,gNB}$	10 MHz
P_i^T	100 mW
P_i^I	10 mW
$B_{i,k}$	1 MHz
$h_{i,k}$	4
σ^2	3×10^{-13} W
γ	0.8

4.1 对比方法

实验选取了 4 种任务卸载策略作为对比方法，其中包括固定节点模式下的任务卸载方案 TOS-FN[1] 以及 3 种异构节点模式下的任务卸载方案，分别为基于模糊逻辑的二进制卸载方案 FLBO(Fuzzy Logic Based Binary Offloading)[22]、CPU 感知任务卸载方案 CATO(CPU-aware Task Offloading)[23]，以及随机匹配任务卸载方案 RMTO(Randomized Matching Task Offloading)。RMTO 旨在通过随机机制实现任务车辆与处理车辆之间的匹配，确保每个任务车辆仅与 1 个处理车辆相对应，同时每个处理车辆也只能服务于 1 个任务车辆。

4.2 任务车辆数量的影响

图 4 展现了不同车辆数量对总任务平均时延的影响。在本次实验过程中，每个任务的最大可容忍时延被设定为 3 s，并确保了所有任务所需的计算资源都处在相同的数值范围内，以确保各方案的公平性和准确性。

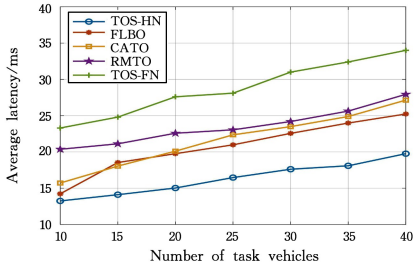

图 4 平均时延 vs. 任务车辆数量

Fig. 4 Average latency vs. the number of task vehicles

随着参与任务车辆数量的不断增加，各方案的总平均时延均呈现出逐步增加的趋势。其中，TOS-FN 由于只考虑固定节点卸载，相较于其他 4 种异构节点卸载模式，其平均时延最高，这足以说明结合车辆的计算能力是提高系统性能、减少任务处理时延的关键途径。RMTO 方案的时延上升趋势尤为显著，主要是因为 RMTO 方案中，任务车辆会随机选择附近的空闲处理车辆，而不考虑任务完成的时延要求。随着任务数量的持续增长，处理车辆难以及时完成所有任务的计算工作。而 RMTO 的这种随机选择机制，进一步导致了一部分处理车辆无法得到有效的任务分配，甚至造成计算资源的浪费。因此，在对比各方案时，RMTO 的性能表现相对较差，其总平均时延的上升趋势也最为明显。

CATO 方案在任务数量相对较少时，有较好的性能表现。其平均时延仅次于 FLBO。然而，随着任务数量的不断攀升，CATO 方案的性能表现逐渐变差，表现出一定的局限性。在任务数量较少的情况下，CATO 方案能够确保任务车辆优先选择计算能力最强的处理车辆进行任务卸载，保证该任务完成时延的最小化。然而，随着任务数量的急剧增加，由于每个处理车辆只能接收 1 个任务车辆的卸载请求，因此任务车辆在寻找合适的处理车辆时变得更加困难。故当任务车辆需要寻找在其通信范围内且处于空闲状态的处理车辆时，往往无法保证能够成功找到，进而影响了整体任务分配的效果。FLBO 的性能表现仅次于 TOS-HN，这得益于其方案在设计时充分权衡了车辆的计算处理能力、移动特性以及链路质量。相较于 CATO，FLBO 不仅能够有效确保任务的最小完成时延，而且在任务卸载的稳定性方面也展现出了显著优势。然而，无论是 FLBO 还是 CATO，它们均属于局部最优方案，尚无法保证在全局范围内实现任务的最小平均时延。

TOS-HN 方案在总体时延方面展现出了显著的优势，相较于其他两种方案，无论任务车辆的数量是多少，TOS-HN 都能保持最佳的性能。这主要得益于 TOS-HN 方案从全局角度出发，充分发掘并利用处理车辆的空闲资源，通过精确计算得出能够使总体时延达到最低的任务卸载方案。这种全局优化的策略使得 TOS-HN 在各种场景下都能实现高效的任务卸载，从而有效降低总体时延。

图 5 展示了车辆数量与总体能耗之间的关联。观察结果可以发现，随着车辆数量的增长，任务计算所需的总能耗也呈现出递增的趋势。当任务车辆数量较少时，这些车辆能够较为容易地匹配到处理车辆进行任务卸载，因此产生的能耗相对较低。然而，随着车辆数量的不断增加，部分任务车辆无法找到合适的处理车辆进行计算，只能选择将任务卸载到基站或是在本地进行处理。将任务卸载到基站意味着传输距离的延长，这会增加相应的传输能耗，而选择本地计算虽然避免了传输能耗，但本地计算能耗本身在总能耗中占有较大比例。

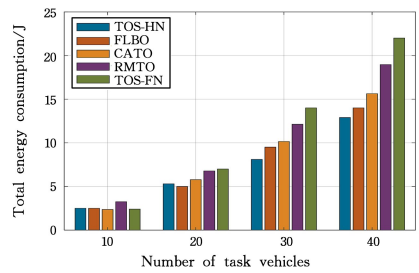

图 5 总能量消耗 vs. 任务车辆数量

Fig. 5 Total energy consumption vs. the number of task vehicles

对比不同方案，TOS-HN 方案在异构节点卸载模式中展现了显著的优势。该方案通过综合考虑任务卸载时延和能耗，能够确定任务车辆与处理车辆之间的最优匹配，使得更多的任务能够成功卸载到附近的空闲车辆中，从而有效降低总

体能耗。相比之下，RMTO 方案采用随机选择卸载节点的方式，导致空闲车辆计算资源的不合理利用，使得更多任务需要在固定节点卸载模式下寻找最优卸载方案，增加了能耗。在车辆数量较少的情况下，FLBO 和 CATO 方案尽管能够在一定程度上保证较小的任务完成时延，但在车辆数量较多的情况下，无法保证全局的最优性，导致一些任务迫不得已只能在本地进行计算，增加了设备的能耗。因此，除了本文提出的 TOS-HN 方案可以在能耗方面始终保持最优的性能表现，其他任务卸载方案性能均表现出一定的随机性，而性能的好坏主要取决于移动节点卸载模式下任务总体的完成率。

4.3 总任务量的影响

总任务量对基站负载所产生的影响如图 6 所示。为确保公平对比，所有基站在不同任务量下的计算能力均被设定为 5GHz。通过对比卸载至基站的总体任务量来说明基站负载的变化趋势。

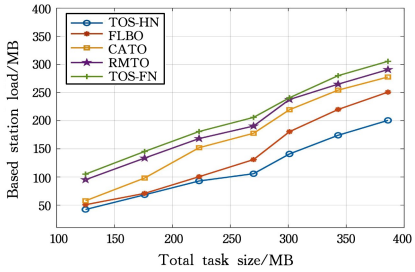

图 6 基站负载 vs. 总任务量

Fig. 6 Base station load vs. total task size

观察数据发现，随着任务量的不断攀升，卸载至基站的任务量也呈现出逐渐增加的趋势。特别地，TOS-FN 卸载至基站的任务数量最为突出，没有移动节点进行任务分流，所有的任务要么选择卸载至基站处理，要么在本地执行，因此基站承受着最大的负载压力。由于采用随机选取机制，当任务数量增加时，RMTO 会出现任务车辆无法在其通信范围内找到处理车辆的情况，使处理车辆资源闲置，大量任务不得不卸载至基站，加重了基站的负载。

尽管 FLBO 相较于 CATO 在任务完成时延上有所降低，但处理车辆的资源利用仍未能达到最佳状态。TOS-HN 方案针对每个任务车辆，均详细计算了与其通信范围内每个处理车辆的任务计算代价。基于这一代价矩阵，TOS-HN 方案能够精准选择出可使总体时延和能耗加权和达到最小的匹配方案。这种策略确保了处理车辆资源得到高效利用，从而显著减少了卸载至基站的任务量，有效减轻了基站的负载。这也进一步表明，在异构节点卸载模式中，尽管道路上的空闲车辆能够为任务车辆提供额外的计算能力，但要实现这些资源的高效利用，必须制定更为高效的卸载策略。这样的策略不仅能够平衡基站的负载，提升资源利用效率，更能够推动整个系统性能的进一步优化，从而为用户提供更为流畅和高效的服务体验。

图 7 详细展示了不同任务量对任务执行失败率的影响。在异构蜂窝网络架构下，固定节点卸载模式已展现出其在提高任务执行成功率方面的优越性。这一成效主要得益于本文所提出的异构蜂窝网络协作任务卸载的网络架构，以及针对联合任务卸载和资源分配所设计的优化算法。然而，这一方案在利用道路上空闲车辆的计算能力方面存在局限性。为此，本研究引入了移动节点卸载以辅助任务卸载和资源分配。

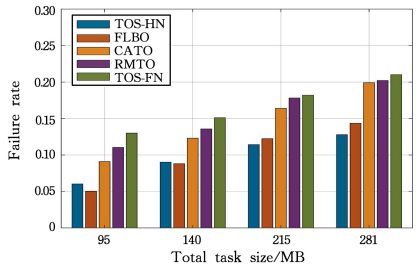

图 7 失败率 vs. 总任务量

Fig. 7 Failure rate vs. total task size

在这一模式下，采用匈牙利算法来确定任务车辆与处理车辆之间的最优匹配，从而实现任务执行总时延和能耗加权和的最小化。这一策略不仅提高了移动节点卸载模式下的任务卸载成功率，还减轻了基站的任务负担，使得固定节点卸载模式下的基站能够为卸载的任务分配更多的计算资源，从而进一步保障低时延任务的完成率。相比之下，RMTO 和 CATO 在移动节点卸载模式下分别采用随机选取机制和局部最优机制，导致仅有少数任务能够在移动节点卸载模式下成功匹配到处理车辆。因此，更多的任务仍然选择在基站或本地执行。尽管 RMTO 和 CATO 相较于仅考虑固定卸载模式的任务卸载策略实现了部分任务由处理车辆完成，降低了任务失败率，但与采用匈牙利算法的最优解相比，其任务失败率仍然较高。相较于 CATO，FLBO 在任务卸载策略的制定上更为全面，它不仅深入考虑了处理车辆的计算能力，还充分顾及了链路质量的影响。这种综合考虑使得 FLBO 在任务完成率方面表现出色。然而，当任务数量较多时，FLBO 依然面临着挑战，它无法保证在全局范围内实现最优的卸载策略。

这一结果表明，在车联网任务卸载过程中，虽然拥有更多的计算资源是至关重要的，但制定有效的策略以确保资源的合理利用同样具有重要意义。本文通过引入优化算法和移动节点卸载模式，成功地提高了任务执行的成功率，并实现了资源的高效利用。

4.4 CPU 频率的影响

图 8 与图 9 展示了在不同边缘服务器的 CPU 频率下，目标函数值与车辆数量的关系。在实验中，设置了两种不同频率的边缘服务器 CPU(5GHz 和 8GHz)。为了确保实验的公正性和准确性，本实验为每辆车设置了相同范围内的任务量，以便更准确地评估不同方案在不同 CPU 下的性能。

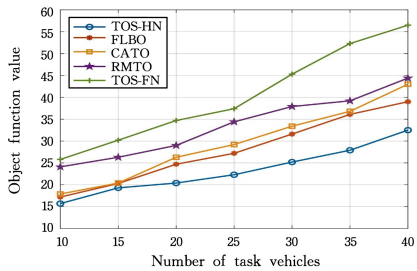

图 8 5GHz 下的目标函数值

Fig. 8 Objective function value at 5GHz

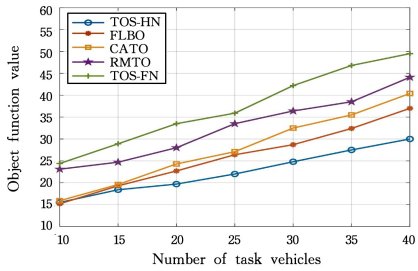

图 9 8GHz 下的目标函数值

Fig. 9 Objective function value at 8GHz

从实验结果可以明显看出，除了 RMTO 以外，其他所有方案的目标函数值都随着 CPU 性能的提高而显著下降。这一趋势表明，增强边缘服务器的计算能力有助于提升系统整体性能。特别是 TOS-FN 方案，尽管未考虑本文中引入的移动节点卸载策略，但其目标函数值仍然大幅下降，这侧面证明了边缘计算在提升系统性能方面的巨大潜力。

进一步比较不同算法的表现可以发现，未引入移动节点卸载模式的 TOS-FN 算法与其他算法之间仍然存在显著差距，这充分说明了结合闲置车辆的计算能力可以极大提升系统整体性能。对于 RMTO 方案，当 CPU 频率增加时，其性能变化最小。这是因为在 RMTO 策略下，任务车辆随机选择附近的空闲车辆进行处理，而不考虑任务的具体完成约束，导致大量计算资源的浪费。即使在 CPU 频率达到 5GHz 时，仍然存在很大的优化空间，当频率提升到 8GHz 时，这些资源仍被浪费，导致 CPU 频率的变化只带来了较小的优化效果。这表明 RMTO 方案对资源的利用不够充分，适应 CPU 频率变化的能力差，稳定性不足。

在 CPU 频率为 5GHz 时，CATO 呈现出与图 4 一致的趋势，即 CATO 方案的表现性能逐渐趋于 RMTO。但是 CPU 频率为 8GHz 时，CATO 更加靠近于 FLBO 方案。这说明 CPU 频率增大时，CATO 能够处理更多的任务车辆，虽然仍然弱于 TOS-HN 与 FLBO 方案，但是该方案本身在面对不同的 CPU 频率时，也能够表现出很好的稳定性。

FLBO 和 CATO 在任务量较少时都表现出良好的性能，且表现相似。然而，在更高的 CPU 频率和更多的任务车辆数量下，FLBO 展现出更优的性能表现。这是因为 FLBO 平衡了车辆的计算能力、移动特性和链路质量，能够在 CPU 频率和任务车辆数量变化时保持稳定的优化态势。具体来说，随着任务车辆数量的增加，FLBO 的性能逐渐超过 CATO 算法，且在更高的 CPU 频率下，这一趋势更加明显。这表明在进行优化时，充分考虑车辆的计算资源是非常重要的。由于 FLBO 方案对系统中各项指标和约束进行了全面权衡，当资源充足时，FLBO 能够更高效地利用这些资源，确保大部分任务在最大可容忍时延内完成。

在所有测试方案中，本文提出的 TOS-HN 方案表现出了最优的性能。其目标函数值始终保持在最低水平，这主要得益于 TOS-HN 方案在移动节点卸载与固定节点卸载上的巧妙分配。通过考虑场景中的移动空闲车辆，引入移动节点卸载模式，TOS-HN 方案大幅提升了整个系统的运行效率和响应速率，并成功减少了整体时延和能源消耗。

结束语 为给车联网系统提供低时延、高可靠性的服务，本文研究异构节点模式下的任务卸载，设计了一种更全面且高效的任务卸载和资源分配策略 TOS-HN。当车辆产生任务时，优先考虑移动节点卸载，将任务卸载到附近空闲车辆上，若移动卸载不能满足任务需求，则采用固定节点卸载策略。在移动节点卸载模式中，通过匈牙利算法确定任务车辆和处理车辆的最优匹配。仿真实验表明，TOS-HN 算法相比于其他算法具有显著优势，在时延、能耗、任务成功率和基站负载方面均具有较好的性能。

在车辆边缘计算中，基站间的负载平衡是确保系统高效运行的关键因素之一，这也是下一步要研究的问题。

参 考 文 献

[1] FAN X G, GU W T, LONG C Q, et al. Optimizing Task Offloading and Resource Allocation in Vehicular Edge Computing Based on Heterogeneous Cellular Networks[J]. IEEE Transactions on Vehicular Technology, 2023, 73(5): 7175-7187.

[2] LIN X Y, YAO Z W, HU S X, et al Task Offloading Algorithm Based on Federated Deep Reinforcement Learning for Internet of Vehicles[J]. Computer Science, 2023, 50(9): 347-356.

[3] ZHANG H B, ZHANG Y F, LIU K J. Task Offloading, Migration and Caching Strategy in Internet of Vehicles Based on NOMA-MEC[J]. Computer Science, 2022, 49(2): 304-311.

[4] MENEGUETTE R, DE G R, UEYAMA J, et al. Vehicular edge computing: Architecture, resource management, security, and challenges[J]. ACM Computing Surveys, 2021, 55(1): 1-46.

[5] HE S, SHI K, LIU C, et al. Collaborative sensing in internet of things: A comprehensive survey[J]. IEEE Communications Surveys & Tutorials, 2022, 24(3): 1435-1474.

[6] TANG C G, LI Z, XIAO S, et al. A Service Caching-Based Task Collaborative Offloading Algorithm for Vehicular Edge Computing[J]. Chinese Journal of Computers, 2025, 48(4): 864-876.

[7] ZHU S F, HU J M, YANG C R, et al. Optimization of offloading decision based on priority task in edge computing scenes of internet of things[J]. Journal of Jilin University(Engineering and Technology Edition), 2024, 54(11): 3338-3350.

[8] TANG L, TANG B, ZHANG L, et al. Joint optimization of network selection and task offloading for vehicular edge computing [J]. Journal of Cloud Computing, 2021, 10(1): 23.

[9] LEE S, LEE S K. Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information[J]. IEEE Internet of Things Journal, 2020, 7(10): 10450-10464.

[10] ZHOU H, JIANG K, LIU X, et al. Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing[J]. IEEE Internet of Things Journal, 2021, 9(2): 1517-1530.

[11] ZHANG Z, WANG N, WU H, et al. MR-DRO: A fast and efficient task offloading algorithm in heterogeneous edge/cloud computing environments[J]. IEEE Internet of Things Journal, 2021, 10(4): 3165-3178.

[12] XU L, LIU Y, FAN B, et al. An Improved Gravitational Search Algorithm for Task Offloading in a Mobile Edge Computing Network with Task Priority[J]. Electronics, 2024, 13(3): 540.

[13] SUN Y, WU Z, MENG K, et al. Vehicular Task Offloading and

Job Scheduling Method Based on Cloud-Edge Computing[J]. IEEE Transactions on Intelligent Transportation Systems, 2023, 24(12): 14651-14662.

[14] ZHAO X, LIU M, LI M. Task offloading strategy and scheduling optimization for internet of vehicles based on deep reinforcement learning[J]. Ad Hoc Networks, 2023, 147: 103193.

[15] ZHU C, TAO J, PASTOR G, et al. Folo: Latency and quality optimized task allocation in vehicular fog computing[J]. IEEE Internet of Things Journal, 2018, 6(3): 4150-4161.

[16] QIAO B, LIU C, LIU J, et al. Task migration computation offloading with low delay for mobile edge computing in vehicular networks[J]. Concurrency and Computation: Practice and Experience, 2022, 34(1): e6494.

[17] ZHANG R, WU L, CAO S, et al. Task offloading with task classification and offloading nodes selection for MEC-enabled IoV [J]. ACM Transactions on Internet Technology, 2021, 22(2): 1-24.

[18] ZHANGD G, YAN H R, ZHANG J. ApproxECIoT: New Edge Computing Architecture Based on Adaptive Stratified Sampling [J]. Journal of Software, 2022, 33(9): 3437-3452.

[19] JANG Y, NA J, JEONG S, et al. Energy-efficient task offloading for vehicular edge computing: Joint optimization of offloading and bit allocation[C]// 2020 IEEE 91st Vehicular Technology Conference. IEEE, 2020: 1-5.

[20] LI S, ZHANG N, JIANG R, et al. Joint task offloading and resource allocation in mobile edge computing with energy harvesting[J]. Journal of Cloud Computing, 2022, 11(1): 17.

[21] LU Y, LUO M X, WANG X. Large-scale mobile edge computing with joint offloading decision and resource allocation[C]// International Conference on Artificial Intelligence and Security. Cham: Springer, 2022: 271-286.

[22] WANG L, ZHANG J H, WANG T, et al. Fine-grained multi-access edge computing architecture for cloud network integration [J]. Journal of Computer Research and Development, 2021, 58(6): 1275-1290.

[23] ZENG F, CHEN Q, MENG L, et al. Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing[J]. IEEE Transactions on Intelligent Transportation Systems, 2020, 22(6): 3247-3257.

[24] MEN R, FAN X, SHAN A, et al. Fuzzy Logic based Binary Computation Offloading Scheme in V2X Communication Networks[J]. IEEE Access, 2024, 12: 45507-45518.

[25] HUANG J, QIAN Y, HU R Q. A vehicle-assisted data offloading in mobile edge computing enabled vehicular networks[C]// 2019 IEEE Global Communications Conference. IEEE, 2019: 1-6.

[26] RAZA S, WANG S, AHMED M, et al. Task offloading and resource allocation for IoV using 5G NR-V2X communication[J]. IEEE Internet of Things Journal, 2021, 9(13): 10397-10410.

[27] FAN W, SU Y, LIU J, et al. Joint task offloading and resource allocation for vehicular edge computing based on V2I and V2V modes[J]. IEEE Transactions on Intelligent Transportation Systems, 2023, 24(4): 4277-4292.

[28] WANG H Y, HUANG Q, et al. Graph Theory Algorithm and its MATLAB Implementation [M]. Beijing: Beihang University Press, 2010.

FAN Xinggang, born in 1974, Ph.D, professor, is a member of CCF (No. 74204M). His main research interests include sensor networks, network communication and Internet of Things.

YANG Youdong, born in 1970, Ph. D, professor. His main research interests include computer-aided design and graphics and so on.

(责任编辑:何杨)