•物联网•

DOI:10.15961/j.jsuese.202200955

本刊网刊

面向车辆边缘计算的任务合作卸载

鲁蔚锋[1,2], 印文徐[1], 王　菁[1], 费汉明[3], 徐　佳[1,2*]

(1.南京邮电大学 计算机学院, 江苏 南京 210023; 2.江苏省大数据安全与智能处理重点实验室, 江苏 南京 210023;
3.国铁吉讯科技有限公司, 北京 100081)

摘　要: 随着物联网技术和人工智能技术的飞速发展, 车辆边缘计算越来越引起学者的关注。车辆如何有效地利用其周边的各种通信、计算和缓存资源, 结合边缘计算系统模型将计算任务迁移到离车辆更近的路边单元, 已经成为目前车联网研究的热点。由于车辆应用设备计算资源的有限性, 车辆用户的任务计算需求无法满足, 需要提升车辆周边计算资源的利用率来完成计算任务。本文研究了车辆边缘计算中任务的合作卸载机制, 以最小化车辆任务的计算时延。首先, 考虑周边停泊车辆以及路边单元的计算资源, 设计了由云服务器层、停泊车辆合作集群层和路边单元合作集群层组成的任务合作卸载3层系统架构, 通过路边单元合作集群和停泊车辆合作集群的合作卸载, 充分利用系统的空闲计算资源, 进一步提高了系统的资源利用率。然后, 基于k-聚类算法的思想提出了路边单元合作集群划分算法对路边单元进行合作集群的划分, 并采用块连续上界最小化的分布式迭代优化方法设计了任务合作卸载算法, 对终端车辆用户的任务进行卸载计算。最后, 通过将本文算法和其他算法方案进行实验仿真对比, 仿真结果表明, 本文算法在系统时延和系统吞吐量方面具有更好的性能表现, 可以降低23%的系统时延, 并且能提升28%的系统吞吐量。

关键词: 边缘计算; 车辆边缘计算; 合作卸载; k-聚类; 计算卸载
中图分类号: TP393.1　　**文献标志码:** A　　**文章编号:** 2096-3246(2024)01-0089-10

近年来, 物联网相关技术和自动驾驶技术已经成为研究的一大热点[1]。随着车联网相关技术的不断更新和发展, 终端车辆具备了计算、通信和缓存的功能, 这些技术在智能车辆网络方面发挥着关键性的作用[2]。考虑到车联网中实际的资源拥有情况, 单一的路边单元可能无法满足终端设备的各项资源需求, 造成了不必要的时延和能量的损耗, 通过结合车联网中的合作任务卸载机制可以有效地提高系统的资源利用率, 减轻网络流量负担, 进一步降低时延。

目前国内外有很多研究车联网中通信、计算和缓存资源的联合分配问题[3−5]以及由云边端组成的3层分布式系统中的卸载问题[6−8]。Fan等[9]结合了基站的缓存和计算资源, 设计了一种资源管理算法, 指导基站联合调度计算卸载和数据缓存分配。Zhou等[10]结合车联网的计算、通信和缓存资源, 设计了一种新颖的以信息为中心的异构网络框架, 该框架支持边缘计算中的内容缓存和计算。Zhou等[11]研究了动态多用户移动边缘计算系统中计算卸载和资源分配的联合优化, 不仅考虑到任务的延迟约束还考虑异构计算任务的不确定资源需求。Xue等[12]将非正交多址(NOMA)技术引入MEC系统中, 提出了一种用于子信道分配的低复杂度次优匹配算法。Zhao等[13]考虑时延敏感型任务的卸载问题, 将该问题建模为传输功率、子载波和计算资源分配联合优化, 通过一个迭代算法顺序处理, 通过等价参数凸规划获得最优解。Kuang等[14]研究MEC中协同计算任务卸载和资源分

收稿日期:2022 – 09 – 05
基金项目:国家自然科学基金项目(62372249; 62072254; 62272237; 62171217; 62372250; 62302236); 中国铁道科学研究院集团有限公司院基金课题重点项目(2022YJ302)
作者简介:鲁蔚锋(1979—), 男, 副教授, 博士生. 研究方向: 边缘计算与网络安全. E-mail: luwf@njupt.edu.cn
* **通信作者:**徐　佳, E-mail: xujia@njupt.edu.cn
网络出版时间:2023 – 12 – 29 10 : 40 : 09　　**网络出版地址:**https://kns.cnki.net/kcms/detail/51.1773.TB.20231228.1307.001.html

配联合问题，通过对计算卸载决策、协同选择、功率分配进行分别建模从而最小化系统时延。Dong等[15]设计了一种新型节能的主动复制机制，采用延迟率计算跟随车辆作为主车辆的备份，保证了系统的可靠性。Zhu等[16]采用多智能体深度强化学习的方案，考虑了多车辆环境的不确定性，使车辆能够做出卸载决策以获得最优的长期奖励。Zhu等[17]提出了一种新型的移动边缘服务器机制，把边缘服务器部署在移动车辆上形成车载边缘，共同优化车辆的最优路径规划和系统的资源分配方案。Liu等[18]用博弈论的方法解决时延敏感型任务的传输调度优化问题，在不同终端用户之间设计一个合理的非合作博弈，来找到最优的卸载方案。

目前已有的研究通常以时延和能耗作为约束条件来最小化系统总成本或最大化系统效用从而提出各种方案来解决优化问题，没有以最小化系统时延为目标，不能满足低时延的需求。考虑到车联网中实际的资源拥有情况，路边单元和边缘服务器相对于云服务器来说，计算能力和存储能力不足。单一的路边单元可能无法满足终端设备的各项资源需求，导致不必要的时延和能量损耗。本文通过结合车联网中的合作任务卸载机制可以有效地提高系统的资源利用率，减轻网络流量负担来降低时延，不仅充分研究了车辆边缘计算系统中的通信模型和计算资源分配问题，还考虑到车联网中停泊车辆和路边单元实际的资源拥有情况，设计了对应的合作集群来协作终端设备完成任务的计算卸载，利用系统的空闲资源提高系统计算能力，以减少系统时延为目标设计算法。本文的主要贡献包括3个方面：

1）设计了一种改进k聚类算法来划分不同的路边单元合作集群，不仅根据地理位置来划分，还考虑了实际的计算负载状况；

2）设计了一种任务合作卸载算法，充分利用了路边单元和停泊车辆空闲的可用资源，实现了系统的负载均衡；

3）应用不同数据集的实验结果表明，本文提出的算法可以找到有效的任务卸载方案。

1 系统模型

1.1 3层系统架构

如图1所示，将车联网中的任务合作卸载系统模型分为3层，分别为云服务器层、路边单元合作集群层和停泊车辆合作集群层。云服务器层位于远距离的云端数据中心，与边缘服务器和路边单元之间通过有线链路进行通信。路边单元合作集群层位于更加靠近终端设备的网络边缘，位于同一合作集群的路边单元之间可以进行X2通信[19]。停泊车辆合作集群层主要位于现实场景中的一些停车场，通过与边缘服务器之间的无线通信协助完成任务的计算，来拓展边缘服务器的计算能力。

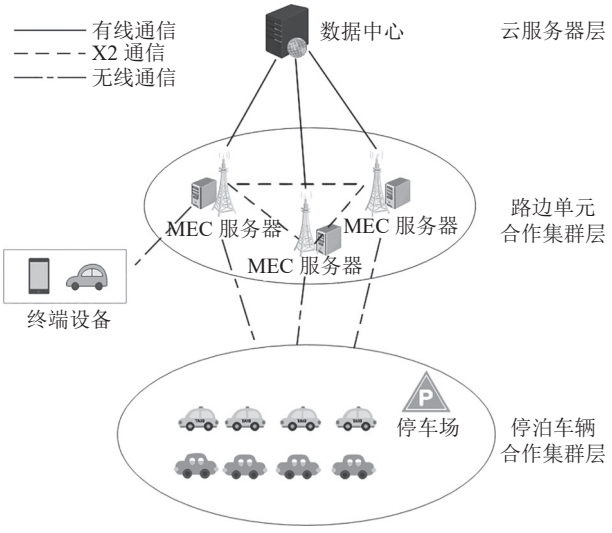


图 1 3层系统架构

Fig. 1 Three-tier system architecture

系统中所有路边单元的集合为G，将所有路边单元划分为ϑ个合作集群，一个路边单元的合作集群表示为：$M \subseteq G$。为了最小化路边单元之间的通信时延，通过距离来进行集群的划分。同时为了提高合作集群的高效性，允许一个路边单元属于多个不同的合作集群。每个路边单元m的计算资源为f_m，代表路边单元m的计算能力，整个合作集群M的计算资源为$F_{\mathrm{M}} = \sum_{m \in M} f_m$。停泊的车辆集合为$P_{\mathrm{V}}$，每个停泊车辆$i \in P_{\mathrm{V}}$的空闲计算资源为$f_i$，则整个停泊车辆合作集群的计算资源为$F_{\mathrm{V}} = \sum_{i \in V} f_i$。

车联网中终端设备的集合为J，每一个终端设备$j \in J$会通过无线链路连接到与它最近的路边单元，连接到同一路边单元m的终端设备集合表示为$J_m \in J$。对于每一个终端设备j的计算任务用三元组表示$a_j = (z_j, c_j, b_j), \forall j \in J$，其中，$z_j$、$c_j$、$b_j$分别表示任务数据量大小、计算任务所需要CPU圈数和任务时延限制。

1.2 合作集群

本文利用改进的k聚类算法[20]来进行路边单元合作集群的划分，提出了路边单元合作集群划分算法，将路边单元集合G划分为ϑ个合作集群，使各集群的聚类平方和最小，即以下目标函数最小：

$$L(\{M_d\}_{d=1}^{\vartheta}) = \sum_{d=1}^{\vartheta} \sum_{m \in M_d} \|m - \phi(m)\|^2 \qquad (1)$$

式中，M_d为第d个路边单元合作集群，每个路边单元

m至少属于一个合作集群，并且$\cup_{d=1}^{\vartheta} M_d = G$，$\phi(m)$为路边单元$m$所属合作集群的平均聚类中心，可表示为：

$$\phi(m) = \frac{\sum_{m_d \in A_m} m_d}{|A_m|} \tag{2}$$

式中，A_m为路边单元m所在的合作集群集合，m_d为其中每个合作集群的聚类中心。

一些固定停靠车辆的区域都可以看作是一个停泊车辆合作集群，比如路边的停车位、商场的停车场等。所有停泊车辆的空闲资源共同构成整个合作集群的可用资源。当多个终端设备接入到同一路边单元时，用$\varsigma_j^m(p_j^m, r_j^m)$来表示每个终端设备$j$在路边单元$m$中的资源分配情况，其中，$p_j^m$为计算资源分配，$r_j^m$为通信资源分配。

1.3　通信模型

终端设备的任务可以在本地计算也可卸载计算，系统计算卸载模型一共分为4种情况，如图2所示：

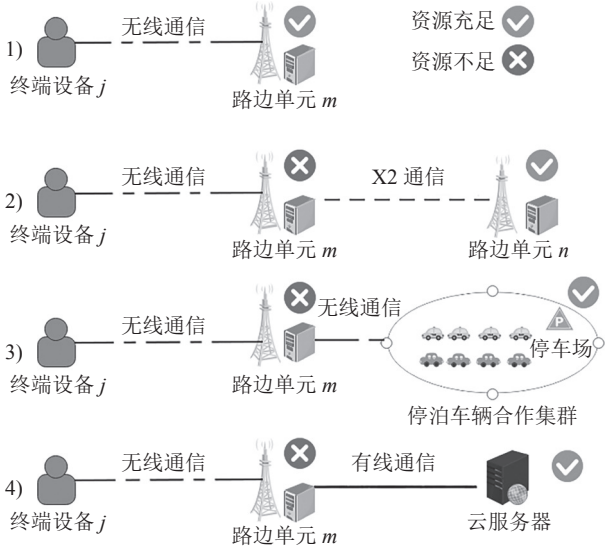


图 2　计算卸载模型

Fig. 2　Computational offload model

1）当终端设备关联的路边单元m资源充足时，终端设备j会通过无线通信将任务卸载到路边单元m计算。终端设备j到路边单元m的无线通信速率为：

$$r_j^{\mathrm{m}} = \rho_j^m B \lg\left(1 + \frac{P_j}{L_0 d_{j,m}^{\alpha} P_{\mathrm{w}}}\right) \tag{3}$$

式中：ρ_j^m为终端设备j所分得路边单元m的带宽比例，$0 \le \rho_j^m \le 1$；B为信道带宽；P_j为终端设备j的传输功率；L_0为信道路径损失；$d_{j,m}$为终端设备j和路边单元m之间的距离；α为路径损失指数；P_{w}为高斯白噪音功率。

由此可得，终端设备j卸载任务到路边单元m的传输时延为：

$$t_j^{j\to m} = \frac{x_j^m z_j}{r_j^m},\ \forall j \in J_m \tag{4}$$

式中：$x_j^m \in \{0,1\}$代表计算卸载决策变量，当$x_j^m = 1$时，表示终端设备j的任务a_j卸载到路边单元m计算；当$\forall m \in M, x_j^m = 0$时，表示终端设备$j$的任务$a_j$在本地计算。

2）当路边单元m的资源不足无法满足终端设备的需求时，路边单元m会将任务a_j通过X2通信传输给资源充足的路边单元n，m和n位于同一路边单元合作集群，可以协同完成任务的计算。

路边单元m到n之间的传输时延为：

$$t_j^{m\to n} = \frac{\sum_{j\in J_m} y_j^{m\to n} z_j}{\eta_m^n},\ \forall m,n \in M \tag{5}$$

式中：η_m^n为路边单元m和n之间的X2通信速率；$y_j^{m\to n} \in \{0,1\}$代表决策变量，当$y_j^{m\to n} = 1$时，表示终端设备j的任务a_j从路边单元m传输到路边单元n计算，当$y_j^{m\to n} = 0$时，表示其他情况。

3）当整个路边单元合作集群M的可用资源无法满足终端设备的需求时，即任务在每个路边单元合作集群的计算时延都大于任务的时延限制，路边单元m可以把终端设备j的任务通过无线通信传输到停泊车辆合作集群计算。

路边单元m到停泊车辆合作集群的传输时延为：

$$t_j^{m\to P_{\mathrm{V}}} = \frac{\sum_{j\in J_m} y_j^{m\to P_{\mathrm{V}}} z_j}{r_m^{P_{\mathrm{V}}}},\ \forall m \in M \tag{6}$$

式中：$r_m^{P_{\mathrm{V}}}$为路边单元m和停泊车辆合作集群之间的无线通信速率；$y_j^{m\to P_{\mathrm{V}}} \in \{0,1\}$代表决策变量，当$y_j^{m\to P_{\mathrm{V}}} = 1$时，表示终端设备$j$的任务$a_j$从路边单元$m$传输到停泊车辆合作集群计算；当$y_j^{m\to P_{\mathrm{V}}} = 0$时，表示其他情况。

4）当整个停泊车辆合作集群的可用资源无法满足终端设备的需求时，即任务在每个停泊车辆的计算时延都大于任务的时延限制，路边单元m还可以把终端设备j的任务通过有线通信传输到云服务器计算。

路边单元m到云服务器的传输时延为：

$$t_j^{m\to DC} = \frac{\sum_{j\in J_m} y_j^{m\to DC} z_j}{\pi_m^{DC}},\ \forall m \in M \tag{7}$$

式中：π_m^{DC}为路边单元m和云服务器之间的有线通信速率；$y_j^{m\to DC} \in \{0,1\}$代表决策变量，当$y_j^{m\to DC} = 1$时，为终端设备$j$的任务$a_j$从路边单元$m$传输到云服务器计算，当$y_j^{m\to DC} = 0$时，为其他情况。

1.4　计算模型

相比于边缘服务器，终端设备的计算能力有限，

在选择任务的计算卸载决策时要考虑终端设备的资源拥有情况。终端设备j将任务在本地计算的时延为：

$$t_j = \frac{c_j}{f_j} \tag{8}$$

式中，f_j为终端设备j的CPU本地计算能力（即每秒CPU的圈数）。

本地计算的CPU能耗为：

$$E_j = c_j v f_j^2 \tag{9}$$

式中，v为一个常量参数，跟终端设备的CPU硬件结构相关[21]。

终端设备的实际可用能量为$\widetilde{E}_j$，当$t_j > b_j$或者$E_j > \widetilde{E}_j$时，代表终端设备j没有足够的能量或者计算资源来满足任务的时延限制，此时本地计算无法满足任务的需求。

则任务a_j在终端设备j总共的本地计算时延为：

$$t_j^{\text{loc}} = \begin{cases} t_j, \text{if } \varpi_j = 1, x_j^m = 0; \\ t_j + \tau_j, \text{if } \varpi_j = 0, x_j^m = 0; \\ 0, \text{if } x_j^m = 1 \end{cases} \tag{10}$$

式中：τ_j代表当终端设备j资源不充足时，任务在设备本地等待处理的平均等待时间。$\varpi_j \in \{0,1\}$代表终端设备j资源是否充足的标识变量，即当$t_j > b_j$或者$E_j > \widetilde{E}_j$时，$\varpi_j = 0$；当资源充足时，$\varpi_j = 1$。

任务进行计算卸载时首先会卸载到离终端设备最近的路边单元m，再根据当前路边单元资源是否充足来决定任务在路边单元m计算还是借助路边单元合作集群计算。

当多个终端设备接入到同一路边单元时，路边单元m分配给终端设备j的计算资源为：

$$p_j^m = f_m \frac{c_j}{\sum_{g \in J_m} c_g}, \forall j \in J_m, m \in M \tag{11}$$

式中，f_m为路边单元m可用的计算资源。

对于每个路边单元m的资源分配要满足路边单元m总计算资源的限制：

$$\sum_{j \in J_m} x_j^m p_j^m y_j^{j \to m} \le f_m, \forall m \in M \tag{12}$$

式中：决策变量$y_j^{j \to m} \in \{0,1\}$，当$y_j^{j \to m} = 1$时，表示终端设备$j$的任务$a_j$卸载到路边单元$m$计算；当$y_j^{j \to m} = 0$时，表示其他情况。

则任务a_j在路边单元m的计算时延为：

$$t_{jm} = \frac{c_j}{p_j^m} \tag{13}$$

任务a_j卸载到路边单元m的总执行时延为传输时延与计算时延之和：

$$t_{jm}^e = t_j^{j \to m} + t_{jm}, \forall j \in J_m, m \in M \tag{14}$$

当$t_{jm}^e > b_j$时，表示路边单元m没有足够的资源完成任务a_j的计算，此时路边单元m会将任务传输到同在一个合作集群的路边单元n计算，并且n有着充足的计算资源。任务在路边单元n的计算时延t_{jn}可根据式（13）同理计算，所以任务a_j在路边单元n的总执行时延为：

$$t_{jmn}^e = t_j^{j \to m} + t_j^{m \to n} + t_{jn}, \forall j \in J_m, m, n \in M \tag{15}$$

当$\forall \mathrm{m} \in M, t_{jm}^e > b_j$时，整个路边单元合作集群都没有足够的资源，任务会传输到停泊车辆合作集群中，充分利用停泊车辆的空闲计算资源来完成任务的计算。任务a_j在停泊车辆合作集群的计算时延为：

$$t_{jP_{\text{V}}} = \frac{c_j}{F_{\text{V}}} \tag{16}$$

则任务a_j卸载到停泊车辆合作集群的总执行时延：

$$t_{jmP_{\text{V}}}^e = t_j^{j \to m} + t_j^{m \to P_{\text{V}}} + t_{jPV}, \forall j \in J_m, m, n \in M \tag{17}$$

当$t_{jmP_{\text{V}}}^e > b_j$时，停泊车辆合作集群也没有足够的资源，任务会传输到云服务器中计算，由于云服务器的计算资源十分充足，因此可忽略任务在云服务器的计算时延，任务a_j卸载到云服务器的总执行时延为：

$$t_{jmDC}^e = t_j^{j \to m} + t_j^{m \to DC}, \forall j \in J_m, m \in M \tag{18}$$

综上所述，任务a_j的卸载计算总时延可表示为：

$$t_j^{off} = y_j^{j \to m} t_{jm}^e + \sum_{n \in M} y_j^{m \to n} t_{jmn}^e + y_j^{m \to P_{\text{V}}} t_{jmP_{\text{V}}}^e + y_j^{m \to DC} t_{jmDC}^e, \quad \forall j \in J_m, m, n \in M \tag{19}$$

1.5 问题的定义

本文的优化目标是最小化任务执行的总时延，任务执行的总时延包括任务本地计算时延和任务卸载计算时延，故可以定义系统目标函数：

$$\min_{x,y} \sum_{m \in M} \sum_{j \in J_m} (1 - x_j^m) t_j^{loc} + x_j^m t_j^{off} \tag{20}$$

式中，loc为本地计算，off为卸载运算。

满足：

约束1 $$\sum_{j \in J_m} x_j^m \rho_j^m \le 1, \forall m \in M,$$

约束2 $$\sum_{j \in J_m} x_j^m p_j^m y_j^{j \to m} \le f_m, \forall m \in M,$$

约束3 $$(1 - x_j^m) + x_j^m \left(y_j^{j \to m} + \sum_{n \in M} y_j^{m \to n} + y_j^{m \to PV} + y_j^{m \to DC} \right) = 1,$$

约束4 $$\max(y_j^{j \to m}, y_j^{m \to n}, y_j^{m \to P_{\text{V}}}, y_j^{m \to DC}) \le x_j^m。$$

约束1代表路边单元分配给接入终端设备的通信资源限制; 约束2代表每个路边单元m的计算资源限制; 约束3和约束4代表任务只能在一个地方执行。

目标函数和约束条件中包含多个非连续的0~1决策变量$\{x,y\}$, 并且多个0~1变量之间存在耦合关系, 导致了这个优化问题是混合整数非线性规划问题,并且这个问题是NP难的。

定理1 优化问题是NP(非确定性多项式时间)难的。

假设目标函数只包含一个0~1决策变量x, 则可看作是一个0~1背包问题[22], 由于0~1背包问题是一个典型的NP难问题, 而优化问题比0~1背包问题更加复杂, 所以优化问题也是个NP难问题。因此, 在第1.3节中设计了具体的算法来求解这个优化问题。

2 算法设计

2.1 路边单元合作集群划分算法

路边单元合作集群划分算法是一个不断迭代的过程, 首先选取$\mathcal{G}$个合作集群中心, 然后根据路边单元和这些合作集群中心的欧式距离, 找到每个路边单元所属的合作集群。再根据式(1)更新合作集群的中心, 不断重复上述步骤, 直到满足某个终止条件。

算法1 路边单元合作集群划分算法。

输入: 所有路边单元的集合G, 最大的迭代次数$t_{\max}$, 阈值ε;

输出: 最终的路边单元合作集群$\{M_d^{(t+1)}\}_{d=1}^{\vartheta}$。

1. 随机选择$m\in\mathcal{G}$个合作集群中心$m\in G$来初始化合作集群;
2. for each $m\in G$ do
3. 根据路边单元m和$\{m_d^{(0)}\}_{d=1}^{\vartheta}$的距离来确定$m$所在的合作集群$A_m^{(0)}$;
4. 得到初始路边单元合作集群$\{M_d^{(0)}\}_{d=1}^{\vartheta}$, 其中, $M_d^{(0)}=\{m|m_d^{(0)}\in A_m^{(0)}\}$;
5. end for
6. 初始化$t=0$;
7. 对于每个合作集群$M_d^{(t)}$, 重新计算新的集群中心$m_d^{(t+1)}$;
8. 对于每个路边单元m和$A_m^{(t)}$, 计算新的集群集合$A_m^{(t+1)}$, 得到新的$\{M_d^{(t+1)}\}_{d=1}^{\vartheta}$;
9. if式(1)没有收敛或者$t_{\max}>t$或者$L(\{M_d^{(t)}\}_{d=1}^{\vartheta})-L(\{M_d^{(t+1)}\}_{d=1}^{\vartheta})>\varepsilon$ then:
10. $t=t+1$;
11. 回到步骤7;
12. else
13. return最终的路边单元合作集群$\{M_d^{(t+1)}\}_{d=1}^{\vartheta}$;
14. end if

算法1给出了路边单元合作集群划分算法的详细步骤, 步骤1随机选取ϑ个初始合作集群中心; 步骤2~5对于每个单元m, 计算其与每个合作集群中心的距离, 距离哪个集群中心近, 就划分到对应的合作集群中, 得到初始的合作集群$\{M_d^{(0)}\}_{d=1}^{\vartheta}$; 步骤7重新计算$\vartheta$个集群的中心; 步骤8根据新的集群中心重新归类每个单元m属于的集群; 步骤9-14是看算法是否满足指定的终止条件, 如果满足3个终止条件中的任意一个, 则直接返回最终的路边单元合作集群算法结束, 如果不满足则回到步骤7继续迭代。

2.2 任务合作卸载算法

为了方便描述, 将优化目标函数表示为$\min F(x,y)$的形式。考虑到目标函数求解的复杂性, 采用块连续上界最小化(BSUM)[23]的分布式迭代优化方法来求解, BSUM方法的核心思想是将整个优化变量拆分为多个变量块, 通过迭代的方法, 在固定其他变量块的同时, 来优化单一的变量块, 直到达到预定的精度阈值(上限)。利用BSUM方法求解优化问题首先要解决两个问题, 一个是目标函数的非凸问题, 一个是0~1变量的非连续问题。

为了解决目标函数的非凸问题, 通过添加平方惩罚项定义目标函数的近似上界函数, 在每次迭代优化过程中, 不直接对目标函数进行求解, 而是对它的近似上界函数$\overline{F}$求解。文献中[23]已经证明优化近似上界函数的方法是合理的, 优化原始目标函数的近似上界函数可以保证原始目标函数是严格下降的。其中对应变量x和y的近似上界函数分别为$\overline{F}_x$和$\overline{F}_y$:

$$\overline{F}_x=F(x,y)+\frac{\xi}{2}\|x-x^*\|^2 \tag{21}$$

$$\overline{F}_y=F(x,y)+\frac{\xi}{2}\|y-y^*\|^2 \tag{22}$$

式中, ξ为一个大于0的惩罚参数, x^*代表在当前迭代中固定变量y而得到的最优解, y^*代表在当前迭代中固定变量x而得到的最优解。

为了解决0~1变量的非连续问题, 将变量x和y线性松弛成0到1之间的连续变量, 则可以定义2个变量的可行解空间为:

$$X\triangleq\left\{x:\sum_{m\in M}\sum_{j\in J_m}x_j^m=1,x_j^m\in[0,1]\right\},$$

$$Y\triangleq\left\{y:\sum_{m\in M}\sum_{j\in J_m}y_j^{j\to m}+y_j^{m\to n}+y_j^{m\to PV}+y_j^{m\to DC}=1,\right.$$

$$\left.y_j^{j\to m},y_j^{m\to n},y_j^{m\to PV},y_j^{m\to DC}\in[0,1]\right\}。$$

在每次迭代t时, 通过求解目标函数的近似上界

函数来得到当前的最优解，即在第$t+1$次迭代时用以下方式来更新x和y的值：

$$x^{(t+1)} \in \arg\min_{x\in X} \overline{F}_x(x;x^{(t)},y^{(t)}) \tag{23}$$

$$y^{(t+1)} \in \arg\min_{y\in Y} \overline{F}_y(y;x^{(t+1)},y^{(t)}) \tag{24}$$

由于最终求解出的优化变量是0到1之间的连续变量，所以需要把它们重新变为0~1变量，借助阈值舍入技术[24]来将松弛后的变量重新变回一个0~1变量。这里举例说明阈值舍入技术，假定一个求解后的优化变量$x_j^{m^*} \in X$和一个舍入阈值$\sigma \in (0,1)$，则对应变量的取值为：

$$x_j^{m^*} = \begin{cases} 1, \text{if } x_j^{m^*} \geq \sigma; \\ 0, \text{otherwise} \end{cases} \tag{25}$$

上述的阈值舍入技术同样也可应用于变量y，然而通过阈值舍入后得到的解可能会超过系统的通信资源和计算资源限制。所以将近似上界函数变成为新的舍入问题$\overline{F}+w\gamma$，即对式（20）的约束条件2和3进行以下调整：

$$\sum_{j\in J_m} x_j^m \rho_j^m \leq 1+\gamma_\rho, \forall m \in M \tag{26}$$

$$\sum_{j\in J_m} x_j^m p_j^m y_j^{j\to m} \leq f_m+\gamma_f, \forall m \in M \tag{27}$$

式中，γ_ρ和γ_f分别为通信资源和计算资源可容忍的最大误差值，总误差$\gamma=\gamma_\rho+\gamma_f$，$w$为总误差的权重参数，$\gamma_\rho$和$\gamma_f$分别表示为：

$$\gamma_\rho = \max\left\{0, \sum_{j\in J_m} x_j^m \rho_j^m - 1\right\}, \forall m \in M \tag{28}$$

$$\gamma_f = \max\left\{0, \sum_{j\in J_m} x_j^m p_j^m y_j^{j\to m} - f_m\right\}, \forall m \in M \tag{29}$$

对应问题$\overline{F}$和它对应的舍入问题$\overline{F}+w\gamma$，可以根据整性间隙的值来评估舍入技术的优劣，根据文献[24]中对整性间隙的定义和证明，可以得出以下结论：

定理2（整性间隙）给定一个问题$\overline{F}$它对应的舍入问题$\overline{F}+w\gamma$，对应的整性间隙为：

$$\beta = \min_{x,y} \frac{\overline{F}}{\overline{F}+w\gamma} \tag{30}$$

整性间隙是用来衡量松弛后的优化问题和原优化问题之间的相关性。当整性间隙$\beta(\beta \leq 1)$越接近1时，说明对应的舍入技术越好。

算法2 路边单元合作集群划分算法。

输入：所有路边单元的计算资源f_m，所有车辆的计算资源f_i；

输出：最优决策变量x^*，y^*。

```
for each j ∈ J do
  if ϖ_j = 1 then
    任务在终端设备j本地计算；
  else
    终端设备j把任务卸载到最近的路边单元m(m ∈ M)计算；
  end if
end for
for each 卸载任务a_j到路边单元m计算do
  初始化t = 0, ε > 0;
  找到初始可行解(x^(0), y^(0));
  repeat:
    x^(t+1) ∈ arg min_{x∈X} F̄_x(x; x^(t), y^(t));
    更新每个终端设备的决策x_j^{t+1} = x_j^t;
    y^(t+1) ∈ arg min_{y∈Y} F̄_y(y; x^(t+1), y^(t));
    t = t + 1;
  until ||(F̄^(t) − F̄^(t+1)) / F̄^(t)|| ≤ ε;
    通过阈值舍入技术求解舍入问题F̄+wγ，得到x^(t+1), y^(t+1);
    计算整性间隙β;
    if β ≤ 1 then
      return x* = x^(t+1), y* = y^(t+1);
    end if
  end for
```

算法2展示了任务合作卸载算法的具体步骤和过程。步骤1~7表示每个终端设备$j \in J$首先会根据自身资源的拥有情况来判断任务是在本地计算还是卸载计算。对于每个卸载到路边单元m的计算任务a_j，执行步骤8~22来找到最优的卸载决策。步骤9表示初始化$t=0$和ε，ε是一个很小的正数，用来保证算法收敛到ε最优解[24]，具体解释在定理2中做出说明。步骤10表示找到一个初始可行解，步骤11~16代表算法的迭代过程，在每次迭代的过程中通过求解式（23）和（24）来不断更新x和y，直到$\left\|\frac{\overline{F}^{(t)}-\overline{F}^{(t+1)}}{\overline{F}^{(t)}}\right\| \leq \varepsilon$，来保证收敛到$\varepsilon$最优解。步骤17表示通过阈值舍入技术求解舍入问题$\overline{F}(x,y)+w\gamma$，得到对应的0~1决策变量$x^{(t+1)}, y^{(t+1)}$。步骤18~20表示计算当前问题的整性间隙$\beta$，如果$\beta \leq 1$代表当前的舍入方案是可行的，输出最优的卸载决策$x^*$，$y^*$。

定理3（收敛性）BSUM算法收敛到ε最优解的时间复杂度为$O(\lg(1/\varepsilon))$，是一个次线性收敛。

算法ε的最优解$x^{\varepsilon}\in X$代表：$x^{\varepsilon}\in\{x|x\in X,F(x,y)-F(x^{*},y)\le\varepsilon\}$，其中，$F(x^{*},y)$是目标函数针对于变量$x$的全局最优值。

3　实验仿真

3.1　实验参数设置

基于Python3.0的环境进行实验仿真，为了验证本文路边单元合作集群划分算法的有效性，采用全球基站开放数据库中的基站位置真实数据[25]，共包括全球5万多个基站的相关数据，模拟现实场景中路边单元合作集群的划分。聚类算法的精度阈值设为0.005，最大的迭代次数设为3 000。为评估任务合作卸载算法的各项性能，假设每个路边单元附近有20~50 个终端用户，每个终端用户大约每分钟产生一个计算任务，车辆的计算资源在0.5到1.0 GHz之间，路边单元的计算资源在2到5 GHz之间，云层的计算资源认为是无穷大的，任务的数据量大小在2到6 GB之间，任务的时延限制在0.5到10 s之间。

3.2　实验结果分析

图3是250个基站聚类之前的实际位置。图4是通过路边单元合作集群划分算法聚类之后的效果。

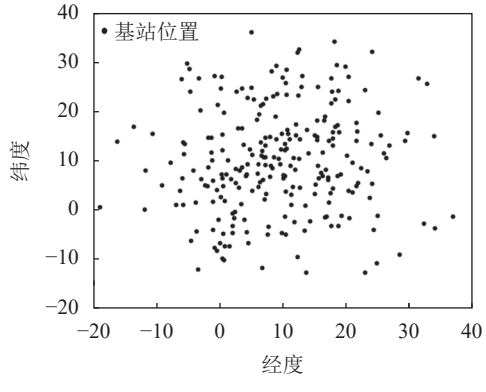


图 3　基站位置

Fig. 3　Base station location

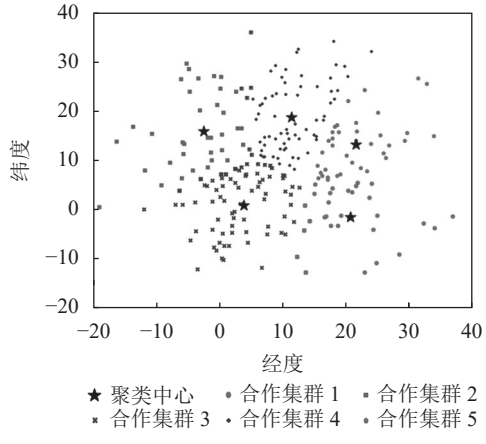
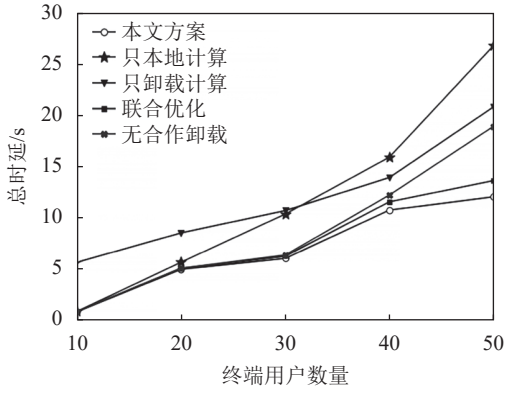


图 4　聚类效果

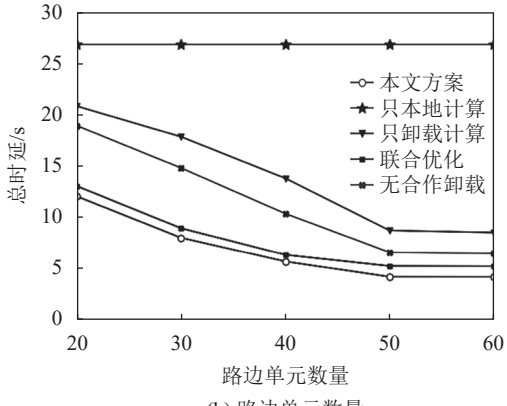
Fig. 4　Clustering effect

从图4可以看出，本文提出的算法将所有的基站划分为了5个不同的合作集群，每个聚类中心一般都位于所有合作集群的中心，并且集群之间可以相互覆盖，证明了本文算法的有效性。

为了证明提出的任务合作卸载算法的优越性，将本方案的实验结果与只本地计算、只卸载计算、无缓存队列、联合优化方案[9]和无合作卸载进行对比，无合作卸载方案是指路边单元之间不存在卸载任务的合作计算；联合优化方案是指考虑到本地处理能力和卸载能力，集中优化移动边缘计算中的联合资源分配。图5展示了在不同终端用户数量和不同路边单元数量下系统总时延的变化曲线，这里的系统总时延就是优化的目标函数。从图5(a)可以看出，本文提出的合作任务卸载方案在不同终端用户数量下系统总时延都要低于其他4种方案，可以降低23%的系统时延，说明本文方案可以有效地降低系统任务处理的总时延。从图5(b)可以看出，在不同路边单元数量下，本文方案也能发挥出很好的性能。当路边单元数量达到50左右时，本文方案、只卸载计算和无合作卸载的总时延变化都趋于稳定，说明在当前的系统状态下，此时的路边单元数量刚好达到系统平衡点，路边单元合作集群有着充足的计算资源，可以很好



(a) 终端用户数量



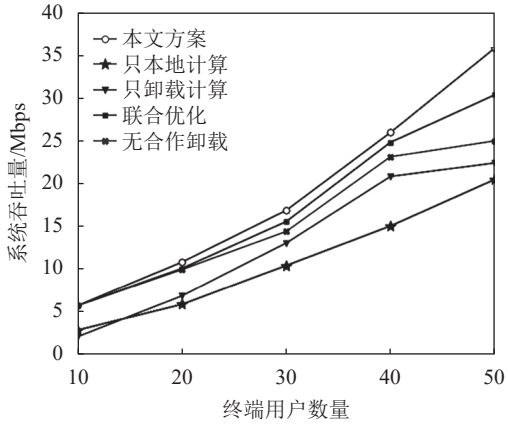
(b) 路边单元数量

图 5　不同情况下总时延的变化

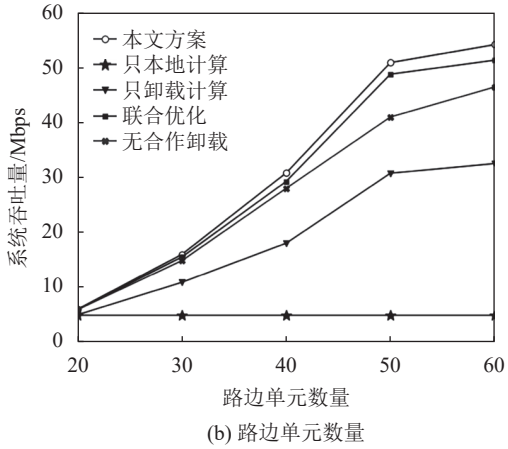
Fig. 5　Variation of total delay in different situations

地完成所有任务的卸载计算。

图6展示了在不同终端用户数量和不同路边单元数量下系统吞吐量的变化曲线，这里的吞吐量指的是系统每秒钟所能处理的数据量。从图6(a)可以看出，本文提出的方案在不同终端用户数量下都有着优于其他方案的系统吞吐量，相比之下，本文方案能提升28%的吞吐量性能，是因为本文提出的任务合作卸载不仅考虑了终端用户和路边单元的资源负载情况，还借助了合作集群之间的空闲计算资源，提高了系统的资源利用率。从图6(b)可以看出，本文方案在不同路边单元数量下也同样有着较高的系统吞吐量，系统吞吐量的提升在路边单元数量40到60之间更为明显，是因为在路边单元数量较多时，仅依靠单一的路边单元进行计算卸载已经无法满足系统的计算卸载需求，更需要考虑不同路边单元之间的协同计算。



(a) 终端用户数量



(b) 路边单元数量

图 6 不同情况下系统吞吐量的变化

Fig. 6 Variation of system throughput in different situations

4 结 论

本文对车辆边缘计算构建了一个3层架构，并提出了一种车辆边缘计算中的任务合作卸载机制，通过划分路边单元合作集群和停泊车辆合作集群，来协助终端设备完成任务的计算卸载，充分利用了系统的各项资源，降低了终端用户任务的处理时延。通过实验验证了路边单元合作集群划分算法的有效性，和其他方案的对比实验发现，本文提出的方案可以降低23%的系统时延，并且能提升28%的系统吞吐量。

未来的工作包括3个方面：1)多种通信模式：本文目前的研究主要考虑的是车辆和路边单元之间的V2I通信，可以结合V2V、V2X等多种通信模式，充分利用系统的各项资源来研究车联网中的任务卸载调度过程。2)停泊车辆的激励机制：在停泊车辆协助完成任务的计算卸载时，考虑到现实中的停泊车辆不会主动贡献自己空闲的计算资源，可以结合博弈论的相关研究，设计出合理的任务卸载激励机制，激励停泊车辆来完成任务的卸载计算。3)卸载方法优化：由于车辆边缘计算中路况的复杂性以及车辆的高移动性，现实场景中系统需要实时获取车辆周围信息，可尝试将在线算法与本文方案结合以满足现实需求。

参考文献：

[1] Sengupta D J,Zhang B,Kraemer B,et al.The view of intergrating Intelligent Transportation System[J].Intelligent Transportation System,2006,93(16):496–501.

[2] Cai Lin,Pan Jianping,Zhao Lian,et al.Networked electric vehicles for green intelligent transportation[J].IEEE Communications Standards Magazine,2017,1(2):77–83.

[3] Huo Ru,Yu F R,Huang Tao,et al.Software defined networking,caching,and computing for green wireless networks[J].IEEE Communications Magazine,2016,54(11):185–193.

[4] Ndikumana A,Tran N H,Ho T M,et al.Joint communication,computation,caching,and control in big data multi-access edge computing[J].IEEE Transactions on Mobile Computing,2020,19(6):1359–1374.

[5] Paymard P,Rezvani S,Mokari N.Joint task scheduling and uplink/downlink radio resource allocation in PD-NOMA based mobile edge computing networks[J].Physical Communication,2019,32:160–171.

[6] Zhang Yutong,Di Boya,Zheng Zijie,et al.Distributed multi-cloud multi-access edge computing by multi-agent reinforcement learning[J].IEEE Transactions on Wireless Communications,2021,20(4):2565–2578.

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

10450–10464.

[8] Liu Zongkai,Dai Penglin,Xing Huanlai,et al.A distributed algorithm for task offloading in vehicular networks with hybrid fog/cloud computing[J].IEEE Transactions on Systems, Man, and Cybernetics:Systems,2022,52(7):4388–4401.

[9] Fan Wenhao,Liu Yuan'an,Tang Bihua,et al.TerminalBooster:Collaborative computation offloading and data caching via smart basestations[J].IEEE Wireless Communications Letters,2016,5(6):612–615.

[10] Zhou Yuchen,Yu F. Richard,Chen Jian,et al.Resource allocation for information-centric virtualized heterogeneous networks with In-network caching and mobile edge computing[J].IEEE Transactions on Vehicular Technology,2017,66(12):11339–11351.

[11] Zhou Huan,Jiang Kai,Liu Xuxun,et al.Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing[J].IEEE Internet of Things Journal,2022,9(2):1517–1530.

[12] Xue Jianbin,An Yaning.Joint task offloading and resource allocation for multi-task multi-server NOMA-MEC networks[J].IEEE Access,2021,9:16152–16163.

[13] Zhao Mingxiong,Yu Junjie,Li Wentao,et al.Energy-aware task offloading and resource allocation for time-sensitive services in mobile edge computing systems[J].IEEE Transactions on Vehicular Technology,2021,70(10):10925–10940.

[14] Kuang Zhufang,Ma Zhihao,Li Zhe,et al.Cooperative computation offloading and resource allocation for delay minimization in mobile edge computing[J].Journal of Systems Architecture,2021,118:102167.

[15] Dong Luobing,Ni Qiufen,Wu Weili,et al.A proactive reliable mechanism-based vehicular fog computing network[J].IEEE Internet of Things Journal,2020,7(12):11895–11907.

[16] Zhu Xiaoyu,Luo Yueyi,Liu Anfeng,et al.Multiagent deep reinforcement learning for vehicular computation offloading in IoT[J].IEEE Internet of Things Journal,2021,8(12):9763–9773.

[17] Zhu Chao,Tao Jin,Pastor G,et al.Folo:Latency and quality optimized task allocation in vehicular fog computing[J].IEEE Internet of Things Journal,2019,6(3):4150–4161.

[18] Liu Yu,Li Yong,Niu Yong,et al.Joint optimization of path planning and resource allocation in mobile edge computing [J].IEEE Transactions on Mobile Computing,2020,19(9):2129–2144.

[19] Yi Changyan,Cai Jun,Su Zhou.A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications[J].IEEE Transactions on Mobile Computing,2020,19(1):29–43.

[20] Cleuziou G.An extended version of the k-means method for overlapping clustering[C]//Proceedings of the 2008 19th International Conference on Pattern Recognition.Tampa,FL:IEEE,2009:1–4.

[21] Mao Yuyi,You Changsheng,Zhang Jun,et al.A survey on mobile edge computing:The communication perspective[J].IEEE Communications Surveys & Tutorials,2017,19(4):2322–2358.

[22] Pisinger D.Where are the hard knapsack problems?[J].Computers & Operations Research,2005,32(9):2271–2284.

[23] Hong Mingyi,Razaviyayn M,Luo Zhiquan,et al.A Unified Algorithmic Framework for Block-Structured Optimization Involving Big Data:With applications in machine learning and signal processing[J].IEEE Signal Processing Magazine,2016,33(1):57–77.

[24] Uriel Feige,Michal Feldman,Inbal Talgam-Cohen.Oblivious rounding and the integrality gap[J].Proceedings of the Leibniz International Proceedings in Informatics,2016,60:13–26.

[25] Unwired Labs.OpenCelliD[DB].(2022-08-25)[2021-04-19].https://www.opencellid.org/downloads.php.

Task Cooperative Offloading for Vehicle Edge Computing

LU Weifeng[1,2], *YIN Wenxu*[1], *WANG Jing*[1], *FEI Hanming*[3], *XU Jia*[1,2*]

(1.School of Computer Sci., Nanjing Univ. of Posts and Telecommunications, Nanjing 210023, China;
2.Jiangsu Key Lab. of Big Data Security and Intelligent Processing, Nanjing 210023, China;
3.China Railway Gecent Technol. Co., Ltd., Beijing 100081, China)

Abstract: With the rapid development of IoT technology and artificial intelligence technology, vehicle edge computing has attracted more and more attention. Effectively utilizing the various communication, computational and caching resources in the vicinity of vehicles, and employing edge computing system models to migrate computational tasks closer to the vehicles, have become a hotspot in current Internet of Vehicles research. Due to the limited computational resources of in-vehicle devices, the computational demands of vehicle users cannot be met without making full use of the computational resources available in the vicinity of vehicles. Aiming to minimize the computational latency of vehicular tasks, a

collaborative offloading mechanism for computational tasks in vehicle edge computing was investigated in this paper. Firstly, a three-layer architecture for task collaborative offloading was designed considering the computational resources of parked vehicles in the vicinity of vehicles as well as the computational resources of roadside units, which was comprised with three tiers: cloud server layer, roadside unit collaboration cluster layer, and the parked vehicle collaboration cluster layer. By means of collaborative offloading between the roadside unit collaboration cluster and the parked vehicle collaboration cluster, the free computational resources of system were fully leveraged, which further enhanced resource utilization. Then, in order to segment roadside units into collaboration clusters, a roadside unit collaboration cluster partitioning algorithm based on k-means clustering algorithm was proposed. A distributed iterative optimization approach with block-coordinate upper-bound minimization was utilized to design a task collaborative offloading algorithm for offloading the computation of terminal vehicle users' tasks. Finally, by comparing with other algorithm schemes through experiments, the algorithm proposed in this paper has better performance in terms of system latency and system throughput according to the stimulation result. Specifically, the system latency was reduced by 23% and the system throughput was increased by 28%.

Key words: edge computing; vehicle edge computing; cooperative offloading; k-means; compute offloading

（编辑　周　璇）

引用格式: Lu Weifeng,Yin Wenxu,Wang Jing,et al.Task cooperative offloading for vehicle edge computing[J].Advanced Engineering Sciences,2024,56(1):89–98.[鲁蔚锋,印文徐,王菁,等.面向车辆边缘计算的任务合作卸载[J].工程科学与技术,2024,56(1):89–98.]