# 基于移动路径预测的车载边缘计算卸载切换策略研究

李 波 牛 力 黄 鑫 丁洪伟*

(云南大学信息学院 昆明 650500)

**摘 要**：车载云计算环境中的计算卸载存在回程网络延迟高、远程云端负载大等问题，车载边缘计算利用边缘服务器靠近车载终端，就近提供云计算服务的特点，在一定程度上解决了上述问题。但由于汽车运动造成的通信环境动态变化进而导致任务完成时间增加，为此该文提出一种基于移动路径可预测的计算卸载切换策略MPOHS，即在车辆移动路径可预测情况下，引入基于最小完成时间的计算切换策略，以降低车辆移动性对计算卸载的影响。实验结果表明，相对于现有研究，该文所提算法能够在减少平均任务完成时间的同时，减少切换次数和切换时间开销，有效降低汽车运动对计算卸载的影响。

**关键词**：车载边缘计算；计算卸载；计算切换；计算切换决策

**中图分类号**：TN919.3; TP39 **文献标识码**：A **文章编号**：1009-5896(2020)11-2664-07

**DOI**: 10.11999/JEIT190483

# Mobility Prediction Based Computation Offloading Handoff Strategy for Vehicular Edge Computing

LI Bo NIU Li HUANG Xin DING Hongwei

(*School of Information Science and Engineering*, *Yunnan University*, *Kunming* 650500, *China*)

**Abstract**: In the vehicular cloud computing environments, computation offloading faces the problems such as high network delay and large load of the remote cloud. The vehicular edge computing takes advantage of the edge servers to be close to the vehicular terminals, and provides the cloud computing service to solve the problem mentioned above. However, due to the dynamic change of communication environment caused by vehicle movement, the task completion time will increase. For this reason, this paper proposes a Mobility Prediction-based computation Offloading Handoff Strategy (MPOHS), which tries to minimize the average completion time of offloaded tasks by migrating tasks among edge servers according to the prediction of vehicle movement. The experimental results show that, compared with the existing research, the proposed strategy can reduce the average task completion time, cut down the handoff times and handoff time overhead, and effectively reduce the impact of vehicle movement on the performance of computation offloading.

**Key words**: Vehicular edge computing; Computation offloading; Computation handoff; Computation handoff strategy

## 1 引言

随着车联网技术的不断成熟，如何有效处理日益增长的车载数据成为制约智慧交通发展的关键问题。由于汽车终端自身的计算能力有限，传统的解决方案是将任务上传至远程云端借助其丰富的计算资源以满足用户对计算服务的需求，这种将任务从计算资源受限设备迁移到其他计算资源充沛的代理设备的过程称为计算卸载[1]。然而随着物联网设备不断地融入日常生活，即时性计算任务的数量呈现井喷式增长，若将全部计算密集型任务卸载至云服务器必会造成云端负荷过载。为此，ETSI组织于2014年提出了移动边缘计算(Mobile Edge Computing, MEC)的概念和技术[2]，MEC技术通过将核心网络的计算和存储等功能下沉至靠近本地设备的网络边缘，以实现云计算服务本地化的目的[3]。车载边缘计算(Vehicular Edge Computing, VEC)是MEC技术与车联网技术的融合，VEC的提出为解决车载网络环境中的计算卸载问题提供了新思路[4]。在VEC环境中，汽车作为主要的研究对象，其高速

收稿日期：2019-06-28；改回日期：2020-03-25；网络出版：2020-08-31
*通信作者： 丁洪伟 dhw1964@163.com
基金项目：国家自然科学基金(61562092)，云南大学信息学院研究生科研创新项目(Y2000211)

Foundation Items: The National Natural Science Foundation of China (61562092), The Graduate Research and Innovation Project of Yunnan University (Y2000211)

移动性会导致计算卸载环境中通信链路频繁地发生动态变化，从而影响任务的卸载效果，如何有效地降低汽车高速移动性对计算卸载性能的影响是VEC环境中进行计算卸载亟待解决的一个重要问题。

近年来，工业界和学术界对此问题进行了深入研究，并提出了一些代表性解决方案。文献[5]针对汽车节点的移动轨迹和卸载任务所需执行时间开销提出了基于路径可预测场景下的计算卸载策略，通过选择满足计算卸载需求的上传代理节点，以实现提高计算卸载效率的目的，但该算法仅通过预测性上传的方式进行计算卸载，并未解决车辆移动性对计算卸载性能的影响，且该算法仅适用于对时延不敏感的计算卸载任务。文献[6]深入研究用户移动性对计算卸载性能的影响，提出了一种将Follow Me Cloud和MEC相结合的移动性管理框架，即Follow Me edge-Cloud框架，该框架适用于网络拓扑结构动态变换的卸载场景，通过实现边缘服务器(Edge Server, ES)之间的服务迁移，以保证卸载任务随着用户移动而进行实时的代理节点更新选择。然而，上述解决方案尽管提出了服务随着车辆的移动而在ES之间实时迁移的思路，但没有给出适用于VEC环境的计算卸载切换决策流程，也没有给出在切换决策过程中如何降低切换开销、提升卸载性能的优化切换策略，导致该解决方案的实际性能受车辆移动速度的影响，可能会频繁触发服务迁移，从而增加计算卸载开销，降低用户体验质量。

针对现有研究的不足，本文首先提出了适用于VEC环境的计算卸载切换框架。然后，提出了一种基于移动路径可预测的计算卸载切换策略MPOHS，该策略使得卸载任务在计算卸载执行过程中，随着车辆的不断移动，当任务所在代理节点的计算资源或当前卸载环境的通信链路无法满足用户对时延的要求时，基于车辆移动轨迹的可预测性和基于最小完成时间的计算切换策略，以进行实时有效的服务迁移，从而在满足用户对服务延迟需求的同时，进一步降低计算切换开销，为车载用户提供更好的用户体验。最后，本文通过仿真对所提出的计算卸载切换流程和切换策略的性能进行了分析。

## 2　车载边缘计算环境下的计算切换决策

VEC的卸载场景分为3层，包括云计算层、边缘计算层和车载计算层[7]，如图1所示。在该场景中部署有1个可与云数据中心共享站址的宏基站BS，该BS可为其覆盖范围内的$M$个路侧单元节点(Road Side Unit, RSU)和$N$辆汽车节点提供通信服务，云数据中心提供计算和存储服务。在该场景中假设$K$个ES部署在$K$个不同的位置[8]，其中$K$是常

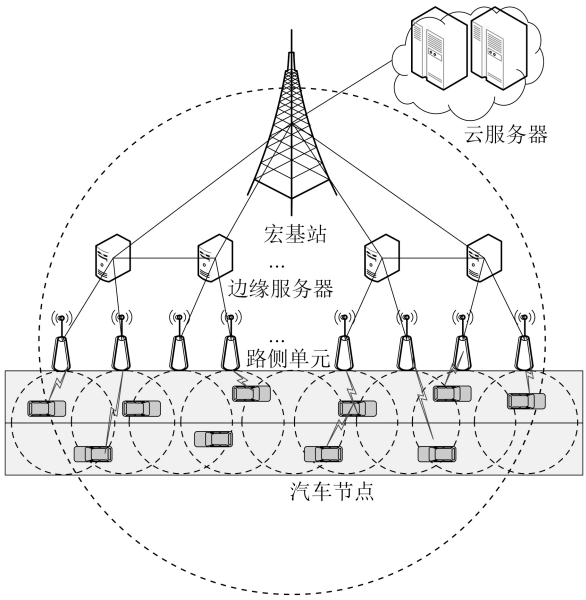



图 1 车载边缘计算场景

量。每个ES配备有限的计算资源，可为一定范围内的RSU提供计算和存储服务。ES之间以回程网络的方式进行相互通信，汽车节点可通过RSU向ES发送服务请求，其中每个RSU的无线通信范围是有限的[9]。在该场景中的$N$辆汽车组成汽车集合$V=\{V_1,V_2,\cdots,V_N\}$，本文将产生卸载任务的汽车节点视为源节点。

假设源节点产生的任务类型为串行任务流，其中每个子任务的任务类型均属于可中断类型[10]，任务流可以表示为$Q=\left\{D_i^{\text{in}},D_i^{\text{out}},C_i,T_i^{\text{d}}\right\}$，其中$D_i^{\text{in}}$表示子任务$i$的上传数据量，$D_i^{\text{out}}$表示子任务$i$的结果数据量，$C_i$表示子任务$i$的计算量，$T_i^{\text{d}}$表示子任务$i$的截止完成时间。假设串行任务流中共有$q$个子任务，上个子任务的输出结果会作为下个子任务的输入，即子任务$i$的输入数据量$D_i^{\text{in}}$等于子任务$i-1$的输出数据量$D_{i-1}^{\text{out}}$，其中$1\le i\le q$。

当源节点$V_i$产生任务后，计算卸载判决模块根据当前时刻的任务属性、计算资源状态以及节点特性等诸多因素进行计算卸载判决。计算卸载过程包括上传阶段、执行阶段和下载阶段。在VEC场景中无线数据传输采用瑞利信道模型[11]，无线传输模式主要有V2R, V2V和V2B 3种。在计算卸载过程中，选择合适的任务调度算法可增强计算卸载效果，提升边缘层的计算资源利用率[12]。本文以任务最小完成时间为优化目标，始终保证任务卸载至完成时间开销最小的代理节点。

然而由于汽车节点的不断运动，其位置不断发生改变，影响了通信链路的稳定性，进而增加任务的完成时间。为了缩短由于移动性造成的任务完成

时间增加，以降低汽车移动性对计算卸载的影响，文献[13]提出了在计算卸载过程中引入计算切换策略，即综合考虑通信开销和计算开销对计算卸载的影响，在汽车运动过程中根据不同的计算切换准则进行代理节点的更新选择，以满足用户对服务时延和用户体验的需求，实现任务在不同ES节点上的实时迁移。

计算切换过程[14]主要包括信息收集、切换决策和切换执行3个阶段，其中切换决策阶段是计算切换的核心，该阶段的切换判决结果将直接决定切换触发条件的选择、切换到哪里以及如何切换等关键问题，计算切换的框架如图2所示。

在切换决策阶段，计算切换准则是执行计算切换决策的前提，选择合适的计算切换准则是保证任务完成时间的关键。本文采用基于最小完成时间的计算切换准则，即

$$t_1 > t_2 + t_\phi \tag{1}$$

其中，$t_1$表示切换代理节点前的任务完成时间，$t_2$表示切换代理节点后的任务完成时间，$t_\phi$表示切换开销。

在VEC环境中，随着源节点的运动，计算资源监测模块发现新的可用代理资源，通过将剩余任务映射至代理资源得到相应的剩余任务完成时间，计算切换决策模块基于最小完成时间的计算切换准则判断并执行计算切换过程。

本文采用基于Docker的容器迁移技术[15]，其中切换开销$t_\phi$主要包括打包时间$t_{\mathrm{p}}$，传输时间$t_{\mathrm{h}}$和重启时间$t_{\mathrm{r}}$。

打包时间$t_{\mathrm{p}}$可表示为

$$t_{\mathrm{p}} = \frac{D_{\mathrm{p}}}{D_{\mathrm{p}}^{\mathrm{s}}} \times T_{\mathrm{p}}^{\mathrm{s}} \tag{2}$$

其中，$D_{\mathrm{p}}$表示需要打包的镜像文件数据量大小，$T_{\mathrm{p}}^{\mathrm{s}}$指的是在切换前的代理节点需要打包镜像文件数据量大小为$D_{\mathrm{p}}^{\mathrm{s}}$的任务所消耗的总时间。

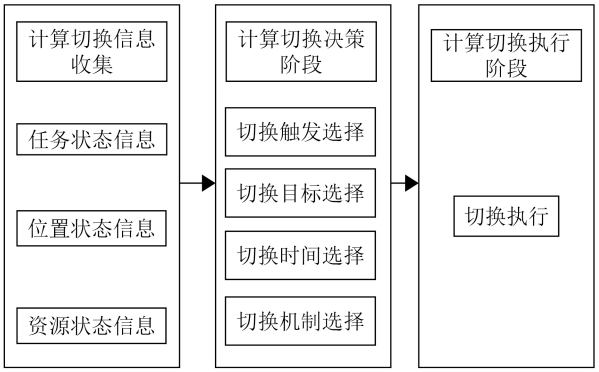



图 2 计算切换框架

传输时间$t_{\mathrm{h}}$可表示为

$$t_{\mathrm{h}} = \frac{D_{\mathrm{h}} \times (1 + r_1)}{D_{\mathrm{h}}^{\mathrm{s}}} \times T_{\mathrm{h}}^{\mathrm{s}} \tag{3}$$

其中，$D_{\mathrm{h}}$表示镜像文件初始时需要迭代传输的数据量，$T_{\mathrm{h}}^{\mathrm{s}}$指对数据量为$D_{\mathrm{h}}^{\mathrm{s}}$的任务进行迭代传输时所耗费的总时间，$r_1$表示由于切换迭代传输过程中的信号干扰等因素导致的该过程中需要重新迭代传输的数据比例。

重启时间$t_{\mathrm{r}}$可表示为

$$t_{\mathrm{r}} = \frac{D_{\mathrm{r}}}{D_{\mathrm{r}}^{\mathrm{s}}} \times T_{\mathrm{r}}^{\mathrm{s}} \tag{4}$$

其中，$D_{\mathrm{r}}$表示传输到新的代理节点的压缩包的数据量大小；$T_{\mathrm{r}}^{\mathrm{s}}$指的是在新的代理节点上对$D_{\mathrm{r}}^{\mathrm{s}}$大小的压缩包进行重启所消耗的总时间。

计算切换流程图如图3所示。

## 3 基于移动路径可预测的计算卸载切换策略

### 3.1 算法模型

随着源节点的不断运动，在任务执行过程中各个代理节点的剩余任务完成时间不同，因而可能频繁发生计算切换。随着切换次数的不断增加，任务

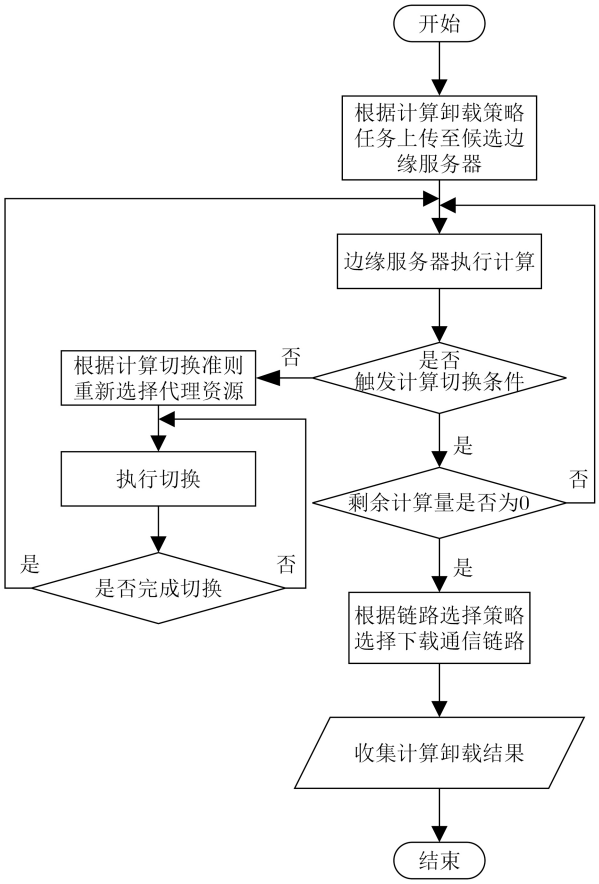



图 3 计算切换过程

的完成时间相应增加。在路径可预测场景下，在计算切换决策过程中考虑汽车运动轨迹因素不仅可缩短任务完成时间也可减少计算切换开销，为用户提供更好的用户服务体验。为此，本文提出了一种基于移动路径可预测的计算卸载切换策略(Mobility Prediction-based computation Offloading Handoff Strategy, MPOHS)，该卸载切换策略过程包括上传阶段、执行阶段和下载阶段。

(1) 上传阶段

假设任务从零时刻开始进行，当源节点产生上传数据量为$D_i^{\text{in}}$的任务时，获取源节点位置从而判断其归属的RSU和ES，通过V2R的方式将任务上传至与其归属的ES上，即上传开销$t^{\text{up}}$为

$$t^{\text{up}} = \frac{D_i^{\text{in}}}{R_{ij}} + t_{\text{del}}^{\text{up}} \tag{5}$$

其中，$R_{ij}$表示任务上传阶段的数据传输速率，$t_{\text{del}}^{\text{up}}$表示任务上传过程中的端到端延时的总和。

假如在$t_j$这一时间段内的任务上传速率$R_j$保持稳定，则可以统计当前任务的上传数据量为$D_j^{\text{in}}$

$$D_j^{\text{in}} = t_j \times R_j \tag{6}$$

假设任务从0时刻开始上传，当到$t_{\text{a}}$时刻任务的累积上传量等于任务所需的上传量时

$$\left[\sum_{j=0}^{j=t_{\text{a}}} D_j^{\text{in}}\right] = D_i^{\text{in}} \tag{7}$$

即满足式(7)时，任务上传结束，统计任务开始上传至任务上传结束的时间跨度为任务上传时间$\sum_{j=0}^{j=t_{\text{a}}} t_j^{\text{up}}$。

(2) 执行阶段

当任务开始执行计算后，假设当前任务总计算量为$C_i$，任务已完成的计算量为$C_i'$，则任务在当前时刻的已完成的执行时间$t_{\text{ex}}'$为

$$t_{\text{ex}}' = \frac{C_i'}{F_k} \tag{8}$$

剩余执行时间为$t_{\text{ex}}^{\text{rk}}$

$$t_{\text{ex}}^{\text{rk}} = \frac{C_i - C_i'}{F_k} \tag{9}$$

其中，$F_k$为当前任务所在服务器的计算能力，在不发生切换的情况下任务预期下载开始时刻$t_{\text{dn}}^{\text{bk}}$为

$$t_{\text{dn}}^{\text{bk}} = \sum_{j=0}^{j=t_{\text{a}}} t_j^{\text{up}} + t_{\text{ex}}' + t_{\text{ex}}^{\text{rk}} \tag{10}$$

在路径可预测的情况下，可将当前任务所在代理节点的直连RSU设为目标下载RSU，通过上述(同任务上传阶段)方法求出当前服务器位置下的任务下载时间$t^{\text{dn}}$，则任务仍然在该服务器上执行的预计完成时间$T_k$为

$$T_k = \sum_{j=0}^{j=t_{\text{a}}} t_j^{\text{up}} + t_{\text{ex}}' + t_{\text{ex}}^{\text{rk}} + t^{\text{dn}} \tag{11}$$

当存在其他可用计算资源时，任务可以切换至其他的空闲资源上，由式(2)至式(4)就可以得到切换开销$t_\phi$，假设切换后的边缘服务器计算能力为$F_j$，任务切换后的剩余任务执行时间为$t_{\text{ex}}^{\text{rj}}$

$$t_{\text{ex}}^{\text{rj}} = \frac{C_i - C_i'}{F_j} \tag{12}$$

则发生切换后的任务预期下载开始时刻$t_{\text{dn}}^{\text{bj}}$为

$$t_{\text{dn}}^{\text{bj}} = \sum_{j=0}^{j=t_{\text{a}}} t_j^{\text{up}} + t_{\text{ex}}' + t_{\text{ex}}^{\text{rj}} + t_\phi \tag{13}$$

可求得切换后的任务下载时间$t_{\text{dn}}'$，得出切换后的预计完成时间$T_j$为

$$T_j = \sum_{j=0}^{j=t_{\text{a}}} t_j^{\text{up}} + t_{\text{ex}}' + t_{\text{ex}}^{\text{rj}} + t_\phi + t_{\text{dn}}' \tag{14}$$

如有可提供切换服务的资源共$z$个，可将所有的预计完成时间$T$表示为

$$T = \{T_k, T_{k+1}, \cdots, T_{k+z}\} \tag{15}$$

其中预计最快的完成时间表示$T_y$为

$$T_y = \min(T) \tag{16}$$

(3) 下载阶段

在路径可预测的情况下，车辆的运动轨迹作为切换判决的一部分，所以当任务执行结束后，只需要将当前结果所在的路程单元作为目标下载RSU即可，结果下载过程类似于任务上传阶段。

### 3.2 车辆运动模型

在VEC环境中，常见的车辆运动模型根据地理环境可分为地理受限运动模型和地理不受限运动模型。本文采用公路运动模型，其运动过程可分为匀速、匀加速和匀减速直线运动，属于地理受限运动模型[16]，在一定程度上其运动轨迹是可预测的。

在公路运动模型中，当源节点在距离站点$d_u$时，以初始速度为$V_u(t)$开始匀减速运动，且到达站点时速度$V_d(t)$为零，则在该过程的耗时$t_u$为

$$t_u = \frac{d_u}{V_u(t)} \tag{17}$$

用$\overline{V_u(t)}$表示该减速过程中的平均速度值

$$\overline{V_u(t)} = \frac{V_d(t) + V_u(t)}{2} \tag{18}$$

该过程中的加速度$a_u(t)$的值为

$$a_u(t)=\frac{\Delta V}{\Delta t} \tag{19}$$

其中汽车加速过程中加速度符号取正，减速过程中符号取负，触及仿真区域的边界时移动方向取反。

## 4 实验仿真与分析

本文重点研究VEC环境下计算切换策略解决车辆移动性对计算卸载性能影响的问题，以任务完成时间开销为主要优化目标，选择平均任务完成时间、平均任务切换次数以及平均任务切换时间开销作为判定计算卸载性能的重要指标，将本文所提算法与其他3种算法进行性能比较：

(1) MCT(Minimum Complete Time)算法[10]：即计算任务卸载至当前完成时间最短的代理节点上。该算法能保证静态环境中的卸载效率，然而当代理节点的计算资源状态发生变化时，其卸载性能有可能会降低。

(2) FMeC(Follow Me edge-Cloud)算法[6]：即随着车辆的移动，卸载任务通过服务迁移的方式始终选择通信开销最小的代理节点。该算法可保证卸载任务的通信开销最小，但并未充分考虑计算开销对计算卸载性能的影响，且受车辆移动速度的影响，可能会频繁触发计算切换。

(3) MCTH(Handoff based on MCT)算法[13]：即随着车辆的移动，综合考虑通信开销和计算开销对计算卸载的影响，卸载任务通过计算切换的方式始终选择当前完成时间开销最小的代理节点。

### 4.1 环境设定

实验根据Geek2MIPS准则，设定计算资源代理节点的计算能力，根据节点类型不同设定不同的计算能力；RSU有固定的服务覆盖范围，设定其服务半径为100～150 m均匀分布，具体参数设置如表1所示。

实验仿真过程为：(1)参数设定与环境配置；(2)任务属性设置；(3)源节点位置更新与归属判决；(4)根据计算卸载决策执行计算卸载；(5)设置计算切换决策模块；(6)基于车辆移动轨迹的计算切换决策执行计算切换过程；(7)结果统计收集分析。

实验将城区道路作为实验场景，并提出以下3点假设：

(1) 仿真区域内的任何一辆车在行车状态下仅考虑仿真区域内的其他车辆对该车的行驶影响，不考虑其他任何设施、行人或者是周边建筑对该车行驶状态的影响。

(2) 当任何通信设备彼此进行通信时，不考虑例如信号干扰等对通信传输速率的影响，通信传输速率只和需要进行通信的两节点的链路选择策略及链路本身有关。

(3) 任务的执行仅限于产生任务的汽车自身、边缘服务器和云服务器，不考虑通过V2V将任务由一辆车卸载至另一辆车的卸载模式。

### 4.2 性能参数

本次实验以平均任务完成时间、平均任务切换次数以及平均任务切换时间开销等性能指标对计算卸载性能进行评价。

(1) 平均任务完成时间：指完成计算卸载的时间开销，包括上传、执行和下载等过程。

(2) 平均任务切换次数：指在计算卸载执行过程中执行计算切换过程的次数。

(3) 平均任务切换时间开销：指执行计算切换过程所需的时间开销。

### 4.3 结果分析

本次实验的统计结果基于1000次仿真实验的性能参数的平均值作为算法性能的度量标准，对比分析在动态环境下引入计算切换对计算卸载效果的影响以及不同计算切换算法对切换开销的影响，实验结果保留两位小数。

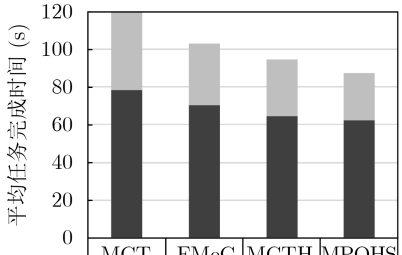
图4给出了动态VEC环境下不考虑切换的MCT算法和3种考虑切换的算法(即FMeC算法、MCTH

**表 1 仿真参数表**

| 参数名 | 参数值 | 参数名 | 参数值 |
|---|---|---|---|
| 基站覆盖范围 | 100% | V计算能力(MIPS) | 82335×0.75 |
| RSU半径(m) | U[100 150] | ES计算能力(MIPS) | 82335×1.5 |
| Cloud数目(个) | 1 | Cloud计算能力(MIPS) | 82335×2 |
| BS数目(个) | 1 | V2R单跳带宽(Mbps) | 15 |
| RSU数目(个) | 64 | V2B单跳带宽(Mbps) | 2 |
| ES数目(个) | 16 | E2E单跳带宽(Mbps) | U[15 20] |
| V数目(个) | 10 | V2V端到端延时(ms) | U[5 15] |
| 任务计算量(MI) | U[7 9]×$10^6$ | V2R端到端延时(ms) | U[20 30] |
| 任务数据量(M) | U[30 50] | V2B端到端延时(ms) | U[450 550] |

算法和MPOHS算法)所对应的的平均任务完成时间结果。从中可知，就平均任务完成时间而言，相较于MCT算法，FMeC算法、MCTH算法以及MPOHS算法分别提升了14.97%, 21.78%和27.74%。这是因为在计算卸载执行过程中引入计算切换策略，随着车辆的移动，基于不同的计算切换触发准则执行相应的计算迁移过程，满足用户对不同服务需求的目的，降低服务延迟，从而证明了计算切换策略的有效性。从图4中可进一步观测到不同的计算切换准则对应不同的平均任务完成时间，其中MPOHS算法的平均任务完成时间最短，为87.36 s，相较于FMeC算法和MCTH算法，分别提升了15.05%和7.64%。这是因为MPOHS算法在基于移动路径可预测的前提下，引入以最小完成时间为优化目标的计算切换策略，在动态卸载场景下，充分考虑计算开销和通信开销对计算卸载的影响，从而降低了任务的完成时间，而FMeC算法仅以通信开销为计算切换准则，虽然减少了计算卸载的通信开销，但没有计算开销对卸载的影响，从而增加了任务完成时间。MCTH算法基于局部贪心的思想，通过局部最优逼近全局最优解，而MPOHS算法通过得到一组近似最优的切换策略，从而进一步减少任务完成时间。

为了进一步比较FMeC, MCTH和MPOHS 3种切换算法在切换过程中所引发的切换开销，图5和图6依次给出了这3种切换算法对应的平均任务切换次数和平均任务切换时间开销结果。




| | MCT | FMeC | MCTH | MPOHS |
|---|---|---|---|---|
| 平均任务通信时间 (s) | 42.79 | 32.58 | 30.23 | 24.94 |
| 平均任务执行时间 (s) | 78.16 | 70.26 | 64.36 | 63.42 |

图 4 不同计算卸载策略下的任务平均完成时间对比

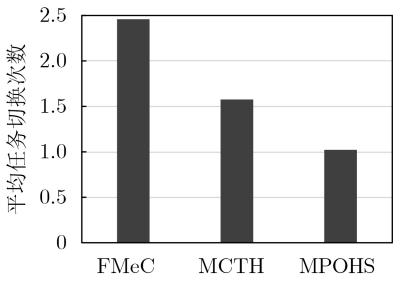



图 5 不同切换策略的平均任务切换次数对比

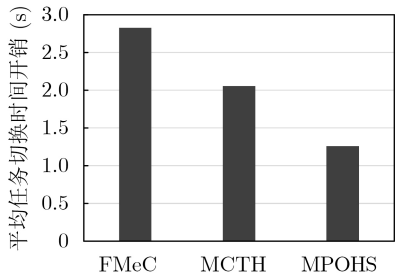



图 6 不同切换策略的平均切换时间开销对比

从图5中可知，就平均任务切换次数，MPOHS算法相较于FMeC算法、MCTH算法分别提升了58.7%和35.84%。从图6可知，就平均任务切换时间开销，MPOHS算法相较于FMeC算法、MCTH算法分别提升了57.14%和39.7%。这是因为MPOHS算法在移动路径可预测的情况下，根据节点的移动路径，预计任务的剩余计算量，保证任务在完成时刻节点处于当前代理节点的覆盖范围，从而降低任务的下载时间开销，保证任务的完成时间最短。通过对比可以得知，减少计算切换次数，可以降低计算切换开销，从而缩短任务的完成时间，保证用户的服务质量。

## 5　结束语

本文总结分析了在VEC环境下由于车辆移动性而导致卸载场景动态变化从而影响计算卸载性能的问题，构建了适用于VEC动态环境下的计算卸载模型，充分考虑计算卸载的通信开销、计算开销以及车辆移动性对计算卸载的影响，在计算卸载执行过程中引入计算切换策略并提出了考虑卸载场景状态动态变化的计算切换框架，提出了基于移动路径可预测的计算卸载切换策略MPOHS，该策略联合考虑车辆移动轨迹的可预测性和基于最小完成时间的计算切换策略。仿真结果表明所提出的算法在满足任务完成时间延迟的同时，降低了计算切换开销，缓解了车辆移动性对计算卸载的消极影响，进一步满足了用户对服务及时性的需求。

在未来研究中，将重点研究VEC环境下的不同计算切换触发条件对计算卸载的影响，以及针对运动模型的预测机制和各种环境下的运动模型进行深入研究，由此推广本文提出的算法的适用场景。

## 参 考 文 献

[1] MACH P and BECVAR Z. Mobile edge computing: A survey on architecture and computation offloading[J]. *IEEE Communications Surveys & Tutorials*, 2017, 19(3): 1628–1656. doi: 10.1109/COMST.2017.2682318.

[2] TRAN T X, HAJISAMI A, PANDEY P *et al*. Collaborative mobile edge computing in 5G networks: New paradigms,

scenarios, and challenges[J]. *IEEE Communications Magazine*, 2017, 55(4): 54–61. doi: 10.1109/MCOM.2017.1600863.

[3] 张海波, 李虎, 陈善学, 等. 超密集网络中基于移动边缘计算的任务卸载和资源优化[J]. 电子与信息学报, 2019, 41(5): 1194–1201. doi: 10.11999/JEIT180592.
ZHANG Haibo, LI Hu, CHEN Shanxue, *et al*. Computing offloading and resource optimization in ultra-dense networks with mobile edge computation[J]. *Journal of Electronics & Information Technology*, 2019, 41(5): 1194–1201. doi: 10.11999/JEIT180592.

[4] 张海波, 栾秋季, 朱江, 等. 基于移动边缘计算的V2X任务卸载方案[J]. 电子与信息学报, 2018, 40(11): 2736–2743. doi: 10.11999/JEIT180027.
ZHANG Haibo, LUAN Qiuji, ZHU Jiang, *et al*. V2X task offloading scheme based on mobile edge computing[J]. *Journal of Electronics & Information Technology*, 2018, 40(11): 2736–2743. doi: 10.11999/JEIT180027.

[5] ZHANG Ke, MAO Yuming, LENG Supeng, *et al*. Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading[J]. *IEEE Vehicular Technology Magazine*, 2017, 12(2): 36–44. doi: 10.1109/MVT.2017.2668838.

[6] AISSIOUI A, KSENTINI A, GUEROUI A M *et al*. On enabling 5G automotive systems using follow me edge-cloud concept[J]. *IEEE Transactions on Vehicular Technology*, 2018, 67(6): 5302–5316. doi: 10.1109/TVT.2018.2805369.

[7] NING Zhaolong, WANG Xiaojie, and HUANG Jun. Mobile edge computing-enabled 5G vehicular networks: Toward the integration of communication and computing[J]. *IEEE Vehicular Technology Magazine*, 2019, 14(1): 54–61. doi: 10.1109/MVT.2018.2882873.

[8] CHEN Hongyang, GAO Feifei, MARTINS M, *et al*. Accurate and efficient node localization for mobile sensor networks[J]. *Mobile Networks and Applications*, 2013, 18(1): 141–147. doi: 10.1007/s11036-012-0361-7.

[9] KHAN Z, FAN Pingzhi, ABBAS F, *et al*. Two-level cluster based routing scheme for 5G V2X communication[J]. *IEEE Access*, 2019, 7: 16194–16205. doi: 10.1109/ACCESS.2019.2892180.

[10] MATHEW T, SEKARAN K C, and JOSE J. Study and analysis of various task scheduling algorithms in the cloud computing environment[C]. 2014 International Conference on Advances in Computing, Communications and Informatics, New Delhi, India, 2014: 658–664.

[11] CHEN Hongyang, WU Jianming, and SHIMOMURA T. New reference signal design for URLLC and eMBB multiplexing in new radio wireless communications[C]. The 29th IEEE Annual International Symposium on Personal, Indoor and Mobile Radio Communications, Bologna, Italy, 2018: 1220–1225.

[12] LI Bo, PEI Yijian, WU Hao, *et al*. Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds[J]. *The Journal of Supercomputing*, 2015, 71(8): 3009–3036. doi: 10.1007/s11227-015-1425-9.

[13] 李波, 黄鑫, 牛力, 等. 车载边缘计算环境中的任务卸载决策和优化[J]. 微电子学与计算机, 2019, 36(2): 78–82.
LI Bo, HUANG Xin, NIU Li, *et al*. Task offloading decision in vehicle edge computing environment[J]. *Microelectronics & Computer*, 2019, 36(2): 78–82.

[14] XIAO Kaiyi and LI Changgen. Vertical handoff decision algorithm for heterogeneous wireless networks based on entropy and improved TOPSIS[C]. The 18th IEEE International Conference on Communication Technology, Chongqing, China, 2018: 706–710.

[15] MA Lele, YI Shanhe, and LI Qun. Efficient service handoff across edge servers via docker container migration[C]. The 2nd ACM/IEEE Symposium on Edge Computing, San Jose, USA, 2017: 1–13.

[16] 郭丽芳, 李鸿燕, 李艳萍, 等. 无线Ad Hoc网络移动模型大全[M]. 北京: 人民邮电出版社, 2014.
GUO Lifang, LI Hongyan, LI Yanping, *et al*. The Encyclopedia of Wireless Ad Hoc Network Mobility Model[M]. Beijing: The People's Posts and Telecommunications Press, 2014.

李 波：男，1976年生，教授，研究方向为移动计算中的计算资源共享.

牛 力：男，1994年生，硕士生，研究方向为车载边缘计算中的计算卸载.

黄 鑫：男，1994年生，硕士生，研究方向为车载边缘计算.

丁洪伟：男，1964年生，教授，研究方向为网络通信技术.

责任编辑：陈 倩