



车联网场景下的移动边缘计算卸载策略

余 翔, 刘一勋, 石雪琴, 王 政

(重庆邮电大学 通信与信息工程学院, 重庆 400065)

摘 要: 针对移动边缘计算(MEC)车联网计算卸载系统, 考虑并发多个多优先级计算任务以及 MEC 服务器资源负载不均的情况, 提出基于遗传算法的卸载策略 GAOS。根据车辆速度、MEC 覆盖情况以及计算任务特性, 为不同优先级的计算任务设置权重。在此基础上, 对车载计算任务进行编码, 将优化问题转化为背包问题, 并通过遗传算法求解得到最佳卸载策略。仿真结果表明, 与 Random 和 ALL-MEC 策略相比, GAOS 受 MEC 服务器负载不均的影响较小, 对车载安全型计算任务的成功处理数量分别增加约 30% 和 50%。

关键词: 移动边缘计算; 车联网; 计算卸载; 遗传算法; 车载计算任务

开放科学(资源服务)标志码(OSID):



中文引用格式: 余翔, 刘一勋, 石雪琴, 等. 车联网场景下的移动边缘计算卸载策略[J]. 计算机工程, 2020, 46(11): 29-34, 41.

英文引用格式: YU Xiang, LIU Yixun, SHI Xueqin, et al. Mobile edge computing offloading strategy under Internet of vehicles scenario[J]. Computer Engineering, 2020, 46(11): 29-34, 41.

Mobile Edge Computing Offloading Strategy Under Internet of Vehicles Scenario

YU Xiang, LIU Yixun, SHI Xueqin, WANG Zheng

(School of Communication and Information Engineering,
Chongqing University of Posts and Telecommunications, Chongqing 400065, China)

[Abstract] In computing offloading systems of Mobile Edge Computing (MEC) in Internet of Vehicles (IoV), the loads of concurrent multi-priority computing tasks and MEC server resources are not balanced. To address the problem, this paper proposes a Genetic Algorithm-based Offloading Strategy (GAOS). The strategy sets weights of computing tasks with different priorities based on vehicle speed, MEC coverage and computing task features. On this basis, the computing tasks are coded. The problem of energy consumption minimization of computing offloading is transformed into a knapsack problem, and the optimal offloading strategy is obtained by using the genetic algorithm. Simulation results show that compared with the Random and all-MEC strategies, GAOS is least affected by the unbalanced loads of MEC servers, and increases the number of successfully processed on-board secure computing tasks by about 30% and 50% respectively.

[Key words] Mobile Edge Computing (MEC); Internet of Vehicles (IoV); computing offloading; genetic algorithm; on-board computing task

DOI: 10.19678/j.issn.1000-3428.0056850

0 概述

作为移动边缘计算(Mobile Edge Computing, MEC)的典型服务场景^[1], 车联网(Internet of Vehicles, IoV)为智能交通系统中的车载终端、路侧单元以及行人提供无线通信服务, 实现车对车(V2V)、车对基础设施(V2I)、车对行人(V2P)以及车对网络(V2N)的通信^[2]。

与传统移动通信中的计算任务不同, 在车联网

场景中, 车载终端会在短时间内生成大量合作感知信息^[3]与分布式环境通知消息^[4]来合作完成保障道路安全与提高交通效率的车联网安全型业务。由于车载终端的计算任务涉及人身安全, 因此对时延有严格的要求, 与传统的通信业务相比, 其计算任务数量激增且处理优先级更高。

在传统云车系统中, 移动云计算虽然极大地提高了资源利用率和计算性能, 但由于回程和骨干网络上的传输容量限制以及延迟波动, 远离移动车辆

基金项目: 国家科技重大专项(2017ZX03001004-004)。

作者简介: 余 翔(1969—), 男, 教授, 主研方向为无线通信、网络协议、信息安全; 刘一勋、石雪琴、王 政, 硕士研究生。

收稿日期: 2019-12-10 修回日期: 2020-02-24 E-mail: imliuyixun@126.com

的云服务可能使卸载效率大幅降低。配置 MEC 服务器的车联网架构被认为是一种有效缩短 V2I、V2N 应用时延的方案^[5-6]。此类方案将连接的汽车云扩展到分散的移动基站环境中,使数据和应用能够靠近车辆,实现车载计算任务的实时化处理,减少业务的时延。

计算卸载作为 MEC 的关键技术之一^[7],是 MEC 系统实现终端业务实时化处理的重要手段^[8-9]。实验证明,将任务卸载到 MEC 上,最多可减少 88% 的时延^[10]。文献[11]在 MEC 车联网场景下考虑任务执行时延、车辆移动时延以及数据传输时延,设计基于任务时延的效用函数来平衡负载和表现时延满意度,进而提出一种低复杂度的算法用于解决该整数非线性规划问题。文献[12]提出加入 MEC 服务器减少 C-V2X 中端到端信令时延的方法。文献[13]设计基于 MEC 的卸载架构,考虑 MEC 服务器的资源限制和计算任务的时延容忍,通过引入契约理论设计有效的计算卸载策略,最大限度地提高了 MEC 服务提供商的利益,增强了车辆的效用。文献[14]研究了车辆边缘网络中的多车辆计算卸载问题,提出一种基于博弈论的车载边缘网络离线算法,其考虑每辆车的卸载策略以求最小化系统总开销。文献[15]研究 5G 环境中车辆的计算任务卸载问题,从计算任务和通信两个方面对 5G 网络车辆计算卸载的能量消耗进行建模,并利用人工鱼群算法解决计算卸载能耗最小化问题,但其在边缘车辆网络的计算卸载过程中,未考虑车辆同时并发多个具有不同优先级计算任务的情况。

本文针对 MEC 车联网计算卸载系统,在 MEC 服务器计算资源负载不均的限制条件下,提出基于遗传算法的卸载策略。通过对计算任务进行编码,利用遗传算法寻找最优卸载策略,将计算任务卸载至合适的 MEC 服务器,在保障车载安全型业务优先处理的同时,减小 MEC 服务器计算资源负载不均对任务完成率的影响。

1 系统模型

本文系统模型如图 1 所示。假设 J 个配置 MEC 服务器的 RSU 在道路上均匀分布,将 MEC 服务器记为 $mec_j, j \in \{1, 2, \dots, J\}$, C 台随机分布的车辆各自携带多个计算任务,设共携带 N 个计算任务,表示为 $L_{task} = \{b, w, \omega, t^{max}, R_L\}$,其中, b 表示输入数据的大小, w 表示任务计算量, ω 是一个可变的参数,表示该计算任务的重要程度,以区分该任务为传统计算任务或车载安全型计算任务, t^{max} 表示任务截止时间,任务处理时延超出时限则表示任务处理失败, R_L 表示任务所属车载终端所在的 MEC 服务小区。

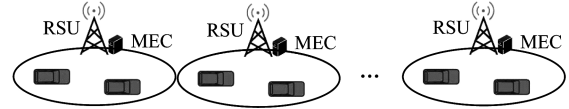


图 1 MEC 车联网系统模型

Fig. 1 System model of MEC Internet of vehicles

用 x 表示车载终端将计算任务卸载至 MEC 服务器的编号, $x = \{0, 1, \dots, J\}$, $1 \sim J$ 表示卸载至 MEC 服务器的编号, $x = 0$ 表示任务由车载终端处理完成。 N 个计算任务的卸载策略构成卸载策略向量集 $X = \{x_1, x_2, \dots, x_N\}$ 。

1.1 通信模型

车辆移动与通信模型可参考文献[16],车辆与 RSU 之间的通信是通过 LTE-V2I 直连的无线链路进行的^[17],车辆到 RSU 的上行链路设定为频率平坦型快衰落的瑞利信道^[18]。根据香农公式可以计算出上传链路的数据传输速率为:

$$R_{v2i} = B_{v2i} \log \left(1 + \frac{P_i d_i^{-\delta} h^2}{N_0} \right) \quad (1)$$

其中, B_{v2i} 表示上传信道的带宽, P_i 表示车载设备的发送功率, $d_i^{-\delta}$ 表示车辆与 RSU 之间的路径损耗, d_i 表示车辆 C_i 与 RSU 覆盖范围中心之间的距离, δ 表示路径损耗因子, h 表示上传链路的信道衰落因子, N_0 表示高斯白噪声功率。

场景中车辆以恒定单方向的速度行驶,车辆 C_i 的速度用 v_i 表示,车辆的移动性使其与 RSU 覆盖范围中心之间的距离 d_i 随时间变化,变化规律可用下式表示:

$$d_i(t) = \sqrt{e^2 + \left(\frac{s}{2} - v_i t \right)^2} \quad (2)$$

其中, e 表示车辆行驶水平线与 RSU 的距离, s 表示 RSU 的覆盖范围。

为简化问题,以平均上传速率 \bar{R}_{v2i} 表示 V2I 卸载过程中车辆将计算任务卸载到 MEC 服务器的数据传输速率,以 t_{stay} 表示车辆从初始位置到离开 RSU 覆盖范围的时间,则有:

$$\bar{R}_{v2i} = \frac{\int_0^{t_{stay}} R_{v2i}(t) dt}{t_{stay}} \quad (3)$$

1.2 计算模型

计算任务的处理分为数据传输和数据计算两部分,若任务 i 进行本地计算,则仅考虑其计算时延而不考虑传输时延。本地任务处理时延的计算公式为:

$$t_i = t_{loc} = \frac{w_i}{e^{loc}} \quad (4)$$

其中, e^{loc} 表示车载终端的计算能力。对于需要进行卸载的任务,有本地服务器卸载与其他服务器卸载 2 种情况。

1.2.1 本地服务器卸载

定义车辆直接通过无线链路进行通信的 MEC 服务器为本地服务器。车辆卸载该计算任务至当前覆盖范围的 MEC 服务器,服务器计算出结果后,立刻返回给车辆。构成计算任务的总时延主要有计算任务的上传时延和 MEC 服务器的处理时延。若卸载至 MEC 服务器,则考虑计算时延以及传输时延,以 t_{ij}^{mec} 表示将任务 i 卸载至服务器 j 处理时延, e_{ij}^{mec} 表示服务器 j 分配给任务 i 的计算资源, r_{ij} 表示计算任务所属车辆与服务器 j 之间的无线传输速率,计算公式如下:

$$t_{ij}^{\text{mec}} = \frac{w_i}{e_{ij}^{\text{mec}}} \quad (5)$$

$$t_{ij}^{\text{trans}} = \frac{b_i}{r_{ij}} \quad (6)$$

$$t_i = t_{ij}^{\text{mec}} + t_{ij}^{\text{trans}} \quad (7)$$

1.2.2 其他服务器卸载

由于当前所在范围 MEC 服务器的负载较重,车载终端决定将计算任务卸载至其他 MEC 服务器。MEC 服务器之间往往通过光纤等有线链路进行相连。假设在有线链路 l 上的计算任务平均传输等待时延为 t_w ,则此时任务处理时延的计算公式为:

$$t_{ij} = \frac{w_i}{e_{ij}^{\text{mec}}} + t_{ij}^{\text{trans}} + 2at_w \quad (8)$$

其中, a 表示计算任务卸载至其他范围 MEC 服务器的有线链路跳数。

1.2.3 计算资源分配

为保证任务在规定时间内完成且任务不中断,要求计算任务在车辆离开所属 MEC 小区前完成任务。因此,本地服务器卸载的情况应满足式(9),其他服务器卸载的情况应满足式(10):

$$\frac{w_i}{e_{ij}^{\text{mec}}} + t_{ij}^{\text{trans}} \leq \min[t_i^{\text{max}}, t_i^{\text{stay}}] \quad (9)$$

$$\frac{w_i}{e_{ij}^{\text{mec}}} + t_{ij}^{\text{trans}} + 2at_w \leq \min[t_i^{\text{max}}, t_i^{\text{stay}}] \quad (10)$$

其中, $t_i^{\text{stay}} = s_i/v_i$, s_i 表示车辆所属位置到离开 MEC 服务器覆盖范围的距离。

由此可推导出 2 种情况下完成计算任务所需计算资源分别为:

$$e_{ij}^{\text{mec}} \geq \frac{w_i}{\min[t_i^{\text{max}}, t_i^{\text{stay}}] - t_{ij}^{\text{trans}}} \quad (11)$$

$$e_{ij}^{\text{mec}} \geq \frac{w_i}{\min[t_i^{\text{max}}, t_i^{\text{stay}}] - t_{ij}^{\text{trans}} - 2at_w} \quad (12)$$

对所有卸载至 mec_j 服务器的计算任务,申请的总计算资源为:

$$e_j = \sum_{i=1}^N \sum_{x_i=j} e_{ij}^{\text{mec}} \quad (13)$$

本文研究目的是对优先级较高的计算任务优先处理,同时提升全计算任务的完成数量。因此,定义系统效用函数如下:

$$P = \sum_{i=1}^N \omega_i / \alpha \quad (14)$$

其中, α 表示计算任务总权重值。

最终计算模型建模为:

$$\begin{aligned} & \max(P) \\ & \text{s.t.} \\ & C_1: x_i \in \{0, 1, \dots, j\}, \forall i \in \mathbb{N} \\ & C_2: y_i \in \{0, 1\}, \forall i \in \mathbb{N} \\ & C_3: e_j < E_j, j = \{1, 2, \dots, J\} \end{aligned} \quad (15)$$

在式(15)中:约束条件 C_1 表示一个计算任务只能卸载至某一个 MEC 服务器,不能同时卸载至两个服务器; C_2 表示计算任务采取二元卸载,只能将整个计算任务卸载至 MEC 服务器,或者不进行卸载;约束条件 C_3 表示卸载至 MEC 服务器的计算任务所消耗的计算资源不能超过 MEC 总的计算资源。

2 基于遗传算法的任务卸载策略

2.1 问题描述

本文将优化问题转化为背包问题,并采用遗传算法^[19]来进行求解以满足计算卸载需求。不同的 MEC 服务器具有不同的计算能力,当计算资源不够的 MEC 服务器承接大量的计算任务时,不仅无法保障计算任务的完成,而且会导致 MEC 服务器负载过重影响其他业务的处理。同时随机地对计算任务进行卸载,又会导致在寻找最优计算卸载策略时迭代次数冗长。因此,本文采用基于遗传算法的任务卸载策略(Genetic Algorithm-based Offloading Strategy, GAOS)预留合适的初始染色体,结合贪婪算法寻找每个 MEC 服务器的最优卸载策略,加速迭代过程。

2.2 编码

本文采用二进制编码,每条染色体的编码为一种卸载策略 X ,用 $X_{(m)}$ 表示策略 X 中 m 位的取值, $X_{(m)}$ 取 0 或 1。为了表示所有计算任务能卸载至所有服务器,编码位数 M 应满足以下条件:

$$M \geq \text{lb } J \quad (16)$$

以 $[X_{iM} \ X_{iM+1} \ X_{iM+2} \ \dots \ X_{(i+1)M}]$ 表示计算任务 i 的卸载结果,其卸载的 MEC 服务器编号 x_i 为:

$$x_i = \sum_{m=iM}^{(i+1)M} X_{(m)} \times 2^{(i+1)M-m} \quad (17)$$

不超出 8 个 MEC 服务器的计算任务策略编码过程如图 2 所示。

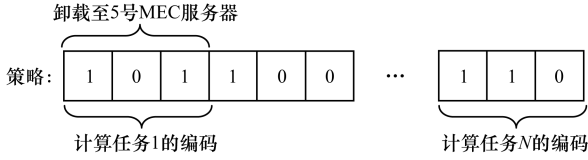


图 2 策略编码与 MEC 服务器的映射

Fig. 2 Mapping of policy coding to MEC servers

2.3 初始化

初始化遗传算法的相关参数,包括最大的迭代次数 I 、染色体长度 $N \times M$ 、变异概率 p_m 、交叉概率 p_c 、种群大小 p_s 以及预留种群 r 。多数遗传算法是随机选择初始种群,GAOS 策略则结合了预定义染色体和随机染色体算法进行种群初始化。在预定义染色体过程中,将染色体中权重最高类,即车载安全类的计算任务 ($\omega = \omega_1$) 编码设置如下:

$$\sum_{m=1M}^{(i+1)M} X_{(m)} \times 2^{(i+1)M-m} = R_i \quad (18)$$

预定义染色体编码算法描述如下:

输入 N, p_s, R_i, ω_1

输出 X

for $k = 1$ to p_s

for $i = 1$ to N

if $x_i = R_i$ and $\omega_i = \omega_1$, then

$$\sum_{m=1M}^{(i+1)M} X_{(m)} \times 2^{(i+1)M-m} = R_i$$

else

$$\sum_{m=1M}^{(i+1)M} X_{(m)} \times 2^{(i+1)M-m} = \text{radom}[0, J]$$

end for

若共 N_c 个计算任务被预定义,则剩余 $(N - N_c)$ 个计算任务的编码随机生成,剩余 $(r - p_s)$ 个种群的染色体编码随机生成,维持种群的随机性。

2.4 解集修复

由式(9)~式(12)可知,当分配给计算任务的计算资源恰好满足时限,则 2 种情况下消耗的计算资源最小,分别为:

$$e_{ij}^{\text{mec}} = \frac{w_i}{t_i^{\text{max}} - t_{ij}^{\text{trans}}} \quad (19)$$

$$e_{ij}^{\text{mec}} = \frac{w_i}{t_i^{\text{max}} - t_{ij}^{\text{trans}} - \sum_t t_w} \quad (20)$$

本文通过贪婪算法修复解集,对超出服务器 j 计算资源中的计算任务计算其价值密度 d_{ij} ,如式(21)所示:

$$d_{ij} = \omega_i / e_{ij}^{\text{mec}} \quad (21)$$

以 N_j 表示卸载到服务器 j 的任务编号,找到最小价值密度的计算任务并将其从 MEC 服务器队列移除,再重新计算消耗的计算资源,重复该步骤,直至所有计算任务消耗的计算资源不超过 MEC 服务器总的计算资源。解集修复算法描述如下:

输入 $p_s, J, e_j, E_j, N_j, d_{ij}$

输出 X

for $k = 1$ to p_s

for $j = 1$ to J

if $e_j > E_j$ then

for $i = 1$ to N_j

if $d_{ij} = \min(d_{ij})$ then

$$\sum_{m=1M}^{(i+1)M} X_{(m)} \times 2^{(i+1)M-m} = 0$$

$$e_j = e_j - e_{\min(d_{ij})}$$

until $e_j \leq E_j$

end for

end if

end for

2.5 适应度函数构造

适应度函数的构造是解决本文优化问题的关键,其目标函数如式(15)所示。因此,给出适应度函数的计算公式为:

$$F = \sum_{i=1}^N \omega_i / \alpha \quad (22)$$

2.6 选择操作

根据选择概率选择染色体,将上述个体作为第一代,采用正比于适应度的轮盘赌随机选择方式,设个体的适应度为 f_i ,则 i 被选中的概率为:

$$P_i = f_i / \sum_{i=1}^n f_i \quad (23)$$

对于初始化后的种群,计算每条染色体的适应度值及其被选择的概率进行比较,剔除概率最低的染色体,选择概率最大的染色体进行复制,代替被剔除掉的染色体。

2.7 交叉操作

本文采用一点交叉方式,交叉概率为 P_c ,具体操作是在个体串中随机设定一个交叉点,实行交叉时,该点前或后两个个体的部分结构进行互换,并生成两个新个体。

2.8 变异操作

对于本文优化问题,变异操作就是将染色体的变异位 1 变为 0,0 变为 1,其他位都保持不变,变异概率为 P_m 。因此,选择一个变异位进行变异,再计算其适应度是否大于或等于其原来的适应度,若不是则重新选择变异位进行变异。

3 仿真结果分析

本节通过仿真结果来验证 GAOS 策略的性能。模拟一个拥有 3 个 MEC 服务器的车联网计算卸载系统,采用与文献[20]相同的比较方式,将 GAOS 与以下卸载方案进行比较:

1) ALL-Local 策略:所有计算任务放在本地执行。

2) Random 策略:所有计算任务随机卸载至 MEC 服务器,根据卸载车载终端数平均分配计算资源。

3) ALL-MEC 策略:所有车载终端进行任务卸载,MEC 服务器根据剩余计算资源对计算任务平均分配计算资源。

仿真参数如表 1 所示,车载安全型计算任务参数如表 2 所示。

表 1 仿真参数

Table 1 Simulation parameters

参数	取值
小区覆盖半径/m	50 ~ 60
小区中心到马路的垂直距离/m	10 ~ 30
小区分配给单个车辆带宽/MHz	1 ~ 5
车载终端数量/辆	10 ~ 50
单个车辆并发任务数量/辆	5 ~ 10
车载终端计算能力/(cycle · s ⁻¹)	4 × 10 ⁶
车辆发射功率/W	1.3
车辆速度/(km · h ⁻¹)	30 ~ 60
高斯白噪声功率/W	3 × 10 ⁻¹³
路径损耗因子	2
上传链路信道衰落因子	4
有线链路平均等待时延/ms	5 ~ 20
MEC 计算能力/(cycle · s ⁻¹)	8 × 10 ⁷ ~ 2 × 10 ⁸
任务权重大小	1 ~ 5
任务时限/ms	200 ~ 500
任务数据大小/kb	50 ~ 100
任务计算量/(cycle · bit ⁻¹)	20 ~ 60
种群数量	60
预留种群数量	20
变异概率数量	0.8
交叉概率	0.1
最大迭代次数	30 ~ 300

表 2 车载安全型计算任务参数

Table 2 Parameters of safety on-board computing task

参数	取值
任务权重大小	10
单个车辆并发任务数量	5 ~ 10
任务时限/ms	30 ~ 100
任务数据大小/kb	100 ~ 200
任务计算量/(cycle · bit ⁻¹)	20 ~ 60

首先验证 GAOS 策略的迭代次数对任务完成率的影响,任务完成率表示为成功计算的计算任务占

总计算任务的比例。本文通过多次重复实验,研究不同车载终端数 C 下迭代次数对任务完成率的影响。由图 3 可以看出,GAOS 策略均可在有效迭代后达到平稳。

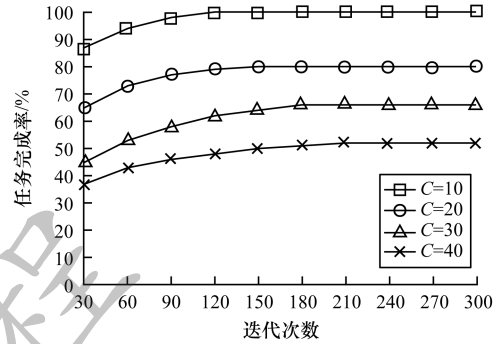


图 3 迭代次数对任务完成率的影响

Fig.3 Influence of iterations number on task completion rate

本文通过重复实验,研究用户在不同卸载方案下车载安全型计算任务($\omega = \omega_1$)的成功处理比例。由图 4 可以看出,为取得最大系统效用函数值,GAOS 策略在卸载过程中,将更多优先级高的任务卸载至 MEC 服务器。与未对优先级任务做处理的 Random 策略以及 ALL-MEC 策略相比,分别提高了约 30% 和 50% 的车载安全型计算任务成功处理数量。

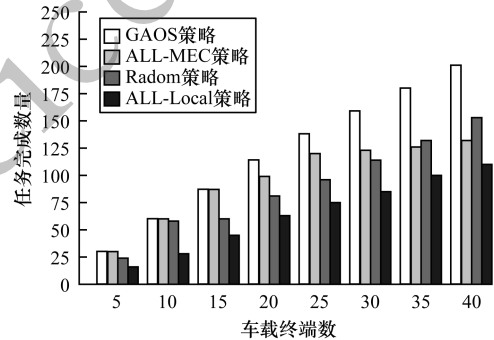


图 4 不同卸载方案下车载安全型计算任务完成数量

Fig.4 Number of completed safety on-board computing tasks of different offloading schemes

定义参数 v 为 MEC 服务器负载不均程度因子:

$$v = \lg \left(\frac{1}{j} \sum_{i=1}^j (E_i - \bar{E})^2 \right) \quad (24)$$

在固定车载终端数为 50 的情况下,研究 MEC 服务器的负载不均程度对任务完成率的影响。由图 5 可以看出,ALL-MEC 策略以及 Random 策略会因 MEC 服务器负载不均而无法保证计算任务完成率。GAOS 策略在对决策方案进行迭代时,对负载严重的 MEC 服务器减少了计算任务的卸载,对负载较轻的 MEC 服务器增加了计算任务的卸载,以达到负载

均衡的效果,相较于其他策略,其受 MEC 服务器负载不均的情况影响较小。

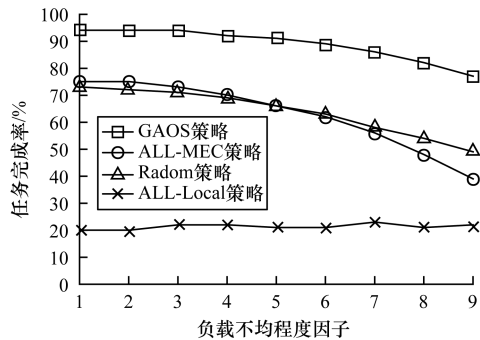


图5 负载不均程度对任务完成率的影响

Fig. 5 Influence of load unevenness degree on task completion rate

可以看出,GAOS 策略能够在有限次的迭代后收敛,得到较优的卸载策略。通过与传统计算卸载策略的实验对比可知,GAOS 策略在具有各种多优先级计算任务的车联网场景下,与可以卸载更多数量的车载安全型计算任务至边缘服务器,符合实际车联网场景下车载安全性业务的处理要求。同时其在迭代的过程中将更多的计算任务卸载至具有更多计算资源的边缘服务器,而对计算资源较少的边缘服务器减少计算任务的卸载,实现了边缘服务器负载均衡。

4 结束语

本文提出基于遗传算法的任务卸载策略 GAOS,用于在负载不均的多 MEC 服务器车联网中寻找最优卸载策略,其能够在有限次迭代后收敛,满足实际车联网场景下车载安全性业务的处理要求。仿真结果表明,在车联网环境车载计算任务不断增加的情况下,GAOS 可实现边缘服务器负载均衡,任务成功处理数量较 Random 和 ALL-MEC 策略分别增加了约 30% 和 50%。下一步将设计部分卸载策略,将车载计算任务合理拆分一部分至 MEC 服务器,一部分则留在本地,从而在结合 MEC 服务的同时充分利用本地的计算资源。

参考文献

- [1] ESTI. Mobile-edge computing—introductory technical white paper [EB/OL]. [2019-12-01]. <https://max.book118.com/html/2018/1006/8013022103001125.shtm>.
- [2] ARANITI G, CAMPOLO C, CONDOLUCI M, et al. LTE for vehicular networking: a survey [J]. IEEE Communications Magazine, 2013, 51(5): 148-157.
- [3] ESTI. Intelligent Transport Systems (ITS); vehicular communications; basic set of applications; part 2: specification of cooperative awareness basic service; ETSI TR 102 637-2—2011[S]. Sophia Antipolis, France; ETSI, 2011.

- [4] ESTI. Intelligent Transport Systems (ITS); vehicular; ETSI TS 102 637-3—2010[S]. Sophia Antipolis, France; ETSI, 2010.
- [5] ZHANG Ke, MAO Yuming, LENG Supeng, et al. Delay constrained offloading for mobile edge computing in cloud-enabled vehicular networks [C]//Proceedings of International Workshop on Resilient Networks Design and Modeling. Washington D. C., USA; IEEE Press, 2016: 288-294.
- [6] LI Liang, LI Yunzhou, HOU Ronghui. A novel mobile edge computing-based architecture for future cellular vehicular networks [C]//Proceedings of Wireless Communications and Networking Conference. Washington D. C., USA; IEEE Press, 2017: 1-6.
- [7] SONG Xiaoshi, YAN Yan, WANG Mengyuan. 5G-oriented MEC system key technologies [J]. ZTE Technology Journal, 2018, 24(1): 21-25. (in Chinese)
宋晓诗, 闫岩, 王梦源. 面向 5G 的 MEC 系统关键技术 [J]. 中兴通讯技术, 2018, 24(1): 21-25.
- [8] 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.
- [9] DOLEZAL J, BECVAR Z, ZEMAN T. Performance evaluation of computation offloading from mobile device to the edge of mobile network [C]//Proceedings of 2016 IEEE Conference on Standards for Communications and Networking. Washington D. C., USA; IEEE Press, 2016: 1-5.
- [10] MACH P, BECVAR Z. Mobile edge computing: a survey on architecture and computation offloading [J]. IEEE Communications Surveys & Tutorials, 2017, 19(3): 1628-1656.
- [11] KAMOUN M, LABIDI W, SARKISS M. Joint resource allocation and offloading strategies in cloud enabled cellular networks [C]//Proceedings of IEEE International Conference on Communications. Washington D. C., USA; IEEE Press, 2018: 5529-5535.
- [12] EMARA M, FILIPPOU M C, SABELLA D. MEC-assisted end-to-end latency evaluations for C-V2X communications [C]//Proceedings of 2018 European Conference on Networks and Communications. Washington D. C., USA; IEEE Press, 2018: 1-5.
- [13] ZHANG Ke, MAO Yuming, LENG Supeng, et al. Optimal delay constrained offloading for vehicular edge computing networks [C]//Proceedings of IEEE International Conference on Communications. Washington D. C., USA; IEEE Press, 2017: 1-6.
- [14] LIU Yujiong, WANG Shuangguang, YANG Fangchun. Poster abstract: a multi-user computation offloading algorithm based on game theory in mobile cloud computing [C]//Proceedings of 2016 IEEE/ACM Symposium on Edge Computing. Washington D. C., USA; IEEE Press, 2016: 93-94.
- [15] YANG Lichao, ZHANG Heli, LI Ming, et al. Mobile edge computing empowered energy efficient task offloading in 5G [J]. IEEE Transactions on Vehicular Technology, 2018, 67(7): 6398-6409.

(下转第 41 页)

(上接第 34 页)

- [16] WANG Hansong. Research on the offloading strategy of computing task based on MEC in telematics[D]. Beijing: Beijing University of Posts and Telecommunications, 2019. (in Chinese)
王寒松. 车联网中基于 MEC 的计算任务卸载策略研究[D]. 北京:北京邮电大学, 2019.
- [17] ZACHARY M H, ASHIQ K, KAZUAKI O, et al. V2X access technologies: regulation, research, and remaining challenges[J]. IEEE Communications Surveys & Tutorials, 2018, 20(3): 1858-1877.
- [18] KAN Zheng, FEI Liu, QIANG Zheng, et al. A graph-based cooperative scheduling scheme for vehicular networks[J]. IEEE Transactions on Vehicular Technology, 2013, 62(4): 1450-1458.
- [19] SONG Haisheng, ZHU Renyi, XU Ruisong, et al. Hybrid genetic algorithm for multi-knapsack problem [J]. Computer Engineering and Applications, 2009, 45(20): 45-48. (in Chinese)
宋海生, 傅仁毅, 徐瑞松, 等. 求解多背包问题的混合遗传算法[J]. 计算机工程与应用, 2009, 45(20): 45-48.
- [20] WANG Yan, GE Haibo, FENG Anqi. Computation offloading strategy in cloud-assisted mobile edge computing[J]. Computer Engineering, 2020, 46(8): 27-34. (in Chinese)
王妍, 葛海波, 冯安琪. 云辅助移动边缘计算中的计算卸载策略[J]. 计算机工程, 2020, 46(8): 27-34.

编辑 金胡考