文章编号:1007-5321(2023)02-0043-07 **DOI**:10.13190/j.jbupt.2022-051

面向高移动性车联网场景的预测卸载决策算法

彭维平, 王 戈, 宋 成, 阎俊豪
(河南理工大学 计算机科学与技术学院, 焦作 454000)

摘要: 针对车联网场景下高移动性车辆在不同边缘服务器之间频繁切换导致任务卸载失败率高的问题,提出了一种新的预测卸载决策算法。首先,构建了本地、边缘服务器和云服务器的计算模型,并基于计算任务量大小、最大容忍时延、服务器资源等约束条件,预测任务的卸载方式;其次,针对边缘服务器的卸载方式,利用长短期记忆网络构建车辆位置预测模型,生成可用于卸载的边缘服务器集合;最后,采用改进的蚁群算法在多个边缘服务器之间实现最优的任务分配。仿真实验结果表明,所提算法可提高任务的完成率和资源利用率。

关 键 词: 车联网; 边缘计算卸载; 位置预测; 长短期记忆网络; 蚁群算法
中图分类号: TN929.5 **文献标志码**: A

Predictive Offloading Decision Algorithm for High Mobility Vehicular Network Scenarios

PENG Weiping, WANG Ge, SONG Cheng, YAN Junhao
(College of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454000, China)

Abstract: To solve the issue of the high failure rate of task offloading caused by frequent switching between different edge servers of highly mobile vehicles in the scenario of the Internet of vehicles, a novel predictive offloading decision algorithm is proposed. First, the local server edge server and cloud server computing models are constructed, and the offloading mode of tasks is predetermined based on the constraints of the size of computing tasks, maximum latency tolerance, and server resources. Then, a vehicle location prediction model is constructed by generating edge servers based on a long short-term memory network, which can be used for offloading. Finally, the improved ant colony algorithm is employed to optimize the offloading task allocation among multi-edge servers. Simulation results show that the proposed algorithm improves task completion rate and resource utilization rate.

Key words: vehicular network; edge computing offloading; location prediction; long short-term memory; ant colony optimization

随着第 5 代移动通信系统和无人驾驶技术的发展,移动车辆对高算力的需求呈现指数级增长。车载边缘计算的发展缓解了智能汽车资源受限的问题,但车辆的高移动性易导致边缘节点无法在有效时间内返回计算结果。目前,学者们主要通过设计合理的卸载策略和资源分配方案来解决这个问题。在卸载策略方面, Luo 等[1]在博弈论模型基础上提出了一种自学习的分布式卸载算法,其总开销相比于集中式卸载方案降低了 12%。Ning 等[2]设计了一种支持模拟学习的分枝定界算法,将人工智能与传统的目标优化算法相结合,通过少量的训练样本获得最优决策,在真实数据集上展现了高效的性能。

收稿日期: 2022-03-09
基金项目: 国家重点研发计划项目(2018YFC0604502); 河南省青年骨干教师计划项目(2019GGJS061)
作者简介: 彭维平(1979—), 男, 教授, 硕士生导师, 邮箱: pwp9999@hpu.edu.cn。

在资源分配方面,Jiang 等[3]构建了基于长短期记忆(LSTM, long short term memory)的深度学习算法来预测小型基站的流量,并根据预测结果,提出了一种基于交叉熵的离线移动数据卸载策略。Tu 等[4]提出了基于深度强化学习和 LSTM 的在线预测卸载算法,利用 LSTM 实时预测边缘服务器的负载。实验结果表明,该算法有效提高了卸载收敛精度和速度。然而,LSTM 较适用于解决时间关联性强的问题,且服务器特征的维度较少,易导致预测结果出现欠拟合。上述方案的性能在不同的应用场景下均有一定的提升,但学者们仅考虑了车辆在接收任务时所在位置周边的计算资源,忽略了在任务最大容忍时延内车辆行驶路径中覆盖的其他计算资源。因此,笔者提出了一种预测快速收敛蚁群(PRE-RCACO, predicted-rapid convergence ant colony optimization)算法,以弥补该缺陷。

1 系统模型

1.1 场景定义

在图 1 所示的车联网场景中,具有通信、存储和计算能力的移动车辆会接收各种任务,这些任务对资源需求不同,且时间限制严格。由于车辆的移动性强且服务器资源有限,并非所有的任务都可在最大容忍时延内完成,车辆需根据任务的属性将其卸载至合适的位置。

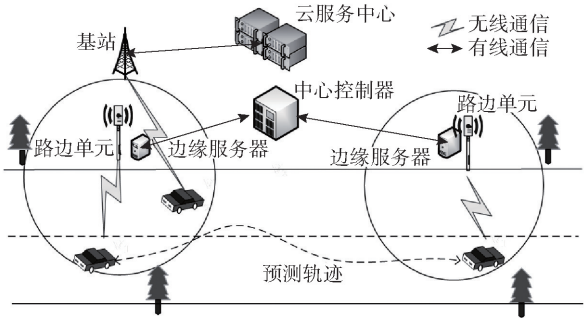

图 1 车联网场景

定义车辆集合为 V,边缘服务器(ES, edge server)集合为 M,云服务器的信息为 Z。其中 $V=\{v_i \mid i=1,2,\cdots,n\}$,$n$ 为车辆的数量,$v_i=((x_i,y_i),\boldsymbol{X}_i,S_i,D_i,\theta_i,\gamma_i,f_{\mathrm{lo},i})$,其元素分别表示车辆接收任务时的位置、历史路径、接收任务时的速度、车辆任务传输数据大小、计算任务大小、最大容忍时延和车辆计算能力;$M=\{m_j \mid j=1,2,\cdots,q\}$,$q$ 为 ES 的数量,$m_j=((a_j,b_j),B_j,C_j,R_j,f_{\mathrm{mec},j})$,其元素分别表示 ES 的位置、带宽、缓存容量、覆盖半径和计算能力;$Z=(B_Z,\rho_Z,\tau_{Z,\mathrm{BS}},f_Z,R_{\mathrm{BS},v_i})$,其元素分别表示总频谱带宽、信道数、云服务器与基站(BS, base station)之间的传输速率、云服务器计算能力、BS 与车辆 v_i 间的数据传输速率。

1.2 通信模型

车辆 v_i 与边缘服务器 m_j 之间的数据传输速率为

$$E_{v_i,m_j}=B_j\mathrm{lb}\left(1+\frac{P}{N_0 d_{v_i,m_j}^{-\sigma}}\right) \tag{1}$$

其中:v_i 到 m_j 的距离表示为

$$d_{v_i,m_j}=\sqrt{(x_i-a_j)^2-(y_i-b_j)^2} \tag{2}$$

P 为 ES 的发射功率,N_0 为高斯白噪声功率,σ 为路径损耗指数。数据传输的时间表示为

$$T_{v_i,m_j}^{\mathrm{tr}}=\frac{D_i}{E_{v_i,m_j}} \tag{3}$$

1.3 计算模型

1.3.1 本地计算与边缘服务器计算

本地计算的最大时长与最大容忍时延相同,可得本地最大计算量为

$$\phi_{\mathrm{lo}}=\gamma_i\ f_{\mathrm{lo},i} \tag{4}$$

车辆路径与 ES 覆盖范围的关系如图 2 所示。若任务卸载至 ES,根据初始位置和终止位置,在以下 3 种情况下,m_j 可为 v_i 提供最大计算量。

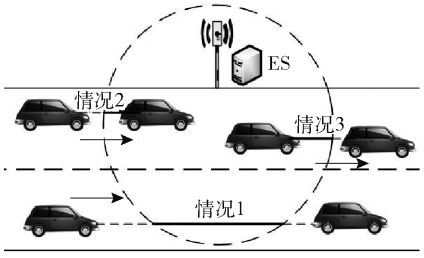

图 2 车辆路径与 ES 覆盖范围的关系

情况 1 v_i 的初始位置(x_i,y_i)和预测终止位置$(x_i^{\mathrm{end}},y_i^{\mathrm{end}})$都在 m_j 覆盖范围外,v_i 的停留时间为

$$T_{\mathrm{st},1}=\frac{d_{\mathrm{st},1}}{S_i} \tag{5}$$

其中:d_{st}为 v_i 在 m_j 覆盖范围内的行驶距离,即

$$d_{\mathrm{st},1}=2\sqrt{R_j^2-d_{\mathrm{pa},i,m_j}^2} \tag{6}$$

d_{pa,i,m_j}为 m_j 到预测路径的最短距离,即

$$d_{\mathrm{pa},i,m_j}=\left|\frac{(y_i-y_i^{\mathrm{end}})a_j+(x_i^{\mathrm{end}}-x_i)b_j+y_i^{\mathrm{end}}x_i-y_ix_i^{\mathrm{end}}}{\sqrt{(y_i-y_i^{\mathrm{end}})^2+(x_i^{\mathrm{end}}-x_i)^2}}\right| \tag{7}$$

在此情况下,m_j 可为 v_i 提供的最大计算量为

$$\phi_{\mathrm{mec},1}=f_{\mathrm{mec},j}(T_{\mathrm{st},1}-T_{v_i,m_j}^{\mathrm{tr}}) \tag{8}$$

情况 2　v_i 的初始位置在 m_j 覆盖范围外，终止位置在 m_j 覆盖范围内，车辆的停留时间为

$$T_{\mathrm{st},2}=\frac{d_{\mathrm{st},2}}{S_i} \tag{9}$$

v_i 在 m_j 覆盖范围内的行驶距离为

$$d_{\mathrm{st},2}=\sqrt{R_j^2-d_{\mathrm{pa},i,m_j}^2}-\sqrt{(a_j-x_i^{\mathrm{end}})^2+(b_j-y_i^{\mathrm{end}})^2-d_{\mathrm{pa},i,m_j}^2} \tag{10}$$

在此情况下，m_j 可为 v_i 提供的最大计算量为

$$\phi_{\mathrm{mec},2}=f_{\mathrm{mec},j}(T_{\mathrm{st},2}-T_{v_i,m_j}^{\mathrm{tr}}) \tag{11}$$

情况 3　车辆 v_i 的初始位置在 m_j 覆盖范围内，终止位置在 m_j 覆盖范围外，车辆的停留时间为

$$T_{\mathrm{st},3}=\frac{d_{\mathrm{st},3}}{S_i} \tag{12}$$

车辆在 m_j 覆盖范围内的行驶距离为

$$d_{\mathrm{st},3}=\sqrt{R_j^2-d_{\mathrm{pa},i,m_j}^2}-\sqrt{(a_j-x_i)^2+(b_j-y_i)^2-d_{\mathrm{pa},i,m_j}^2} \tag{13}$$

在此情况下，m_j 可为 v_i 提供的最大计算量为

$$\phi_{\mathrm{mec},3}=f_{\mathrm{mec},j}(T_{\mathrm{st},3}-T_{v_i,m_j}^{\mathrm{tr}}) \tag{14}$$

由此，可根据车辆路径与边缘服务器覆盖范围的关系，得到 ES 可为车辆提供的最大计算量。

1.3.2　云服务器计算

若将车辆任务卸载至云服务器进行计算，需满足的前提条件是云服务器处理该任务的时间不大于任务的最大容忍时延。云服务器处理的总时间为

$$T_{v_i,Z}=T_{v_i,Z}^{\mathrm{tr}}+T_{v_i,Z}^{\mathrm{ba}}+T_{v_i,Z}^{\mathrm{com}} \tag{15}$$

其中：$T_{v_i,Z}^{\mathrm{tr}}$为任务卸载至云服务器的时间，$T_{v_i,Z}^{\mathrm{ba}}$为云服务器返回计算结果的时间，$T_{v_i,Z}^{\mathrm{com}}$为云服务器计算的时间，分别为

$$T_{v_i,Z}^{\mathrm{tr}}=\frac{D_i}{R_{\mathrm{BS},v_i}}+\frac{D_i}{\tau_{Z,BS}} \tag{16}$$

$$T_{v_i,Z}^{\mathrm{ba}}=\frac{\theta_i}{R_{\mathrm{BS},v_i}}+\frac{\theta_i}{\tau_{Z,\mathrm{BS}}} \tag{17}$$

$$T_{v_i,Z}^{\mathrm{com}}=\frac{\theta_i}{f_z} \tag{18}$$

$$R_{\mathrm{BS},v_i}=\frac{B_Z}{\rho_z} \tag{19}$$

1.4　预测模型

1.4.1　车辆位置预测模型（VLPM，vehicle location prediction model）

为了避免临近车辆对运动轨迹的影响，提出了 VLPM。基于 LSTM 预测每辆车的行驶路径，同时以车的位置为中心建立网格，将格内每个时间步长车辆的行为都作为预测该车辆的依据，即共享相邻车辆的 LSTM。

1.4.2　社交池化层

为了更有效地表示并利用网格内车辆的 LSTM 信息，引入社交池化层。在社交池化层的每个时间步长记录邻居车辆的 LSTM 隐藏状态信息。定义车辆 v_i 轨迹 δ_i 的隐藏状态张量为

$$\boldsymbol{H}_i^t(m,n,:)=\sum_{j\in N_i}F_{mn}[x_j^t-x_i^t,y_j^t-y_i^t]\boldsymbol{h}_j^{t-1} \tag{20}$$

其中：$\boldsymbol{h}_j^{t-1}$ 为车辆 v_j 在 $t-1$ 时刻 LSTM 的隐藏状态；F_{mn}为指标函数，用于判断车辆 v_j 是否在以车辆 v_i 为中心的 $m\times n$ 网格中；N_i 为与车辆 v_i 对应的相邻对象的集合。将合并的隐藏状态张量 $\boldsymbol{H}_i^t$ 嵌入向量 $\boldsymbol{g}_i^t$，将车辆 v_i 的坐标嵌入向量 $\boldsymbol{e}_i^t$，将 $\boldsymbol{g}_i^t$，$\boldsymbol{e}_i^t$ 连接起来，并用作 t 时刻对应 LSTM 单元的输入，表示为

$$\boldsymbol{e}_i^t=\varphi(\boldsymbol{x}_i^t,\boldsymbol{y}_i^t;\boldsymbol{W}_e) \tag{21}$$

$$\boldsymbol{g}_i^t=\varphi(\boldsymbol{H}_i^t;\boldsymbol{W}_g) \tag{22}$$

$$\boldsymbol{h}_i^t=\mathrm{LSTM}(\boldsymbol{h}_i^{t-1},\boldsymbol{e}_i^t,\boldsymbol{g}_i^t;\boldsymbol{W}_{\mathrm{lo}}) \tag{23}$$

其中：$\varphi(\cdot)$为具有 ReLU 非线性的嵌入函数；$\boldsymbol{W}_e$ 和 $\boldsymbol{W}_g$ 为嵌入权重；$\boldsymbol{W}_{\mathrm{lo}}$为 LSTM 嵌入权重。LSTM$(\cdot)$表示长短期神经网络层，其细胞单元由 3 个 sigmoid 函数和一个 tanh 函数构成。

1.4.3　输入和输出

模型的输入序列主要是车辆与该网格内所有车辆的历史轨迹 $\boldsymbol{X}=[\boldsymbol{X}_0,\boldsymbol{X}_1,\cdots,\boldsymbol{X}_n]$，其中 $\boldsymbol{X}_i=[\boldsymbol{x}_i^1,\boldsymbol{x}_i^2,\cdots,\boldsymbol{x}_i^o]$，其中 $\boldsymbol{x}_i^t$ 是车辆 v_i 在 $t(1\leqslant t\leqslant o)$时刻的真实坐标，其中 o 表示观测长度。模型的输出为对应车辆的预测位置概率分布，即

$$\boldsymbol{\Theta}=[\boldsymbol{\Theta}_0,\boldsymbol{\Theta}_1,\cdots,\boldsymbol{\Theta}_n],\boldsymbol{\Theta}_i=[\boldsymbol{\Theta}_i^{o+1},\boldsymbol{\Theta}_i^{o+2},\cdots,\boldsymbol{\Theta}_i^{2o}],$$

$$\boldsymbol{\Theta}_i^t=[\mu_x^t,\mu_y^t,\sigma_x^t,\sigma_y^t,\partial]$$

其中：μ_x^t，μ_y^t 分别为车辆 v_i 在 t 时刻 x 轴和 y 轴预测位置的均值，对应的方差分别为 σ_x，σ_y；∂为其相关系数。根据二元高斯分布，预测位置的坐标范围为

$$\hat{\boldsymbol{y}}_i^t\sim\mathrm{CN}(\boldsymbol{\mu}^t,\boldsymbol{\Omega}^t) \tag{24}$$

其中：$\boldsymbol{\mu}^t=(\mu_x^t,\mu_y^t)^{\mathrm{T}}$，$\boldsymbol{\Omega}^t=\begin{pmatrix}(\sigma_x^t)^2 & \sigma_x^t\sigma_y^t\\ \sigma_x^t\sigma_y^t & (\sigma_y^t)^2\end{pmatrix}$。

1.4.4　损失函数

在 VLPM 中通过最小负对数似然函数进行训练，损失函数为

$$G_i=-\sum_{o\leqslant t\leqslant 2o}\ln(P(\boldsymbol{y}_i^t|\hat{\boldsymbol{y}}_i^t)) \tag{25}$$

其中：$\boldsymbol{y}_i^t$ 为 v_i 在时刻 $t(o+1\leqslant t\leqslant 2o)$的真实坐标，

P 为对于不同预测位置概率分布中出现真实位置概率分布的概率。

1.5 问题模型

任务完成率(TCR, task completion rate) υ 和服务器缓存资源利用率 ϑ 的计算公式分别为

$$\upsilon = \frac{\sum_{1\leqslant i\leqslant n} l_i}{n} \tag{26}$$

$$l_i = \begin{cases} 1, & v_i \subseteq U_{\text{lo}} \text{或} U_{\text{ES}} \text{或} U_Z \\ 0, & \text{其他} \end{cases} \tag{27}$$

$$\vartheta = \frac{D_i \sum_{1\leqslant i\leqslant n} l_i}{\sum_{1\leqslant j\leqslant q} C_j} \times 100\% \tag{28}$$

其中:$l_i = 1$ 时,表示可以完成车辆 v_i 的任务,$l_i = 0$ 时,表示无法完成车辆 v_i 的任务;U_{lo} 为卸载到本地的任务集合;U_{ES} 为卸载到 ES 的任务集合;U_Z 为卸载到云服务器的任务集合;C_j 为容量。解决任务卸载失败率高的问题可转化为使 TCR 最大化的问题,即

$$\mathrm{J}_1: \max\ \upsilon$$

$$T_{\text{st},k} > T^{\text{tra}}_{v_i,m_j}, \quad k \in \{1,2,3\} \tag{29}$$

$$\phi_{\text{mec},k} \geqslant \theta_i, \quad k \in \{1,2,3\} \tag{30}$$

$$\sum_{i\in Q_j} D_i \leqslant C_j \tag{31}$$

其中 Q_j 为 m_j 要处理的车辆任务集合。

1.6 快收敛蚁群卸载模型

问题 J_1 为 NP 难问题,而采用启发式算法并行计算和正反馈的机制可以解决此类问题。因此,可采用启发式算法中具有较强全局寻优能力的蚁群算法来寻找最优解。车辆 v_i 的任务在 t 轮迭代中卸载至 m_j 的概率为

$$p_{ij}(t) = \begin{cases} \dfrac{[\alpha_{ij}(t)]^{\psi}[\eta_{ij}(t)]^{\zeta}}{\sum_{m_j\in a_i}[\alpha_{ij}(t)]^{\psi}[\eta_{ij}(t)]^{\zeta}}, & m_j \in \beta_i \\ 0, & m_j \notin \beta_i \end{cases} \tag{32}$$

其中:ψ 为信息素启发因子;ζ 为期望启发因子;β_i 为车辆 v_i 的任务可卸载服务器的集合;信息素浓度 $\alpha_{ij}(t)$ 表示第 t 轮迭代中车辆 v_i 的任务卸载至 m_j 的权重,α 的更新方法为

$$\left.\begin{aligned} \alpha_{ij}(t+1) &= \omega\alpha_{ij}(t) + \Delta\alpha_{ij}(t) \\ \Delta\alpha_{ij}(t) &= \sum_{m_j\in\beta_i} \alpha_{ij}(t) \end{aligned}\right\} \tag{33}$$

$\Delta\alpha_{ij}(t)$ 为第 t 轮迭代中车辆 v_i 的任务卸载至可卸载服务器的权重之和;ω 为随迭代次数递增且映射在(0,1)之间的函数,以增强全局搜索能力,即

$$\omega = \frac{1 - \mathrm{e}^{-\lambda}}{1 + \mathrm{e}^{-\lambda}} \tag{34}$$

λ 为迭代次数;启发式信息 $\eta_{ij}(t)$ 为车辆 v_i 的任务卸载至 m_j 的期望,即

$$\eta_{ij}(t) = C_j^- \tag{35}$$

C_j^- 为 m_j 的剩余容量。

2 预测快速收敛蚁群算法

2.1 系统初始化

设定道路的长度为 L,宽度为 A。初始化所有车辆节点 $\{v_i \mid i = 1,2,\cdots,n\}$ 的计算能力 $f_{\text{lo},i}$ 和所有 ES $\{m_j \mid j = 1,2,\cdots,q\}$ 的位置 (a_j, b_j)、带宽 B_j、覆盖半径 R_j 和计算能力 $f_{\text{mec},j}$。

2.2 任务预处理

步骤 1 当 v_i 接收到计算任务的请求时,先根据式(4)计算出本地最大计算量 ϕ_{lo},比较 ϕ_{lo} 与计算任务 θ_i 的大小。若 $\phi_{\text{lo}} \geqslant \theta_i$,则在本地即可完成计算任务,跳转至步骤 9;否则,执行步骤 2。

步骤 2 若车辆任务的最大容忍时延 γ_i 小于驶出当前 ES 覆盖范围所需时间,则判断该 ES 是否可以同时满足约束[式(29)~式(31)],若满足约束,则将任务卸载至该 ES;否则,判定无法完成该任务,跳转至步骤 9。若车辆任务的最大容忍时延 γ_i 大于驶出当前 ES 覆盖范围所需时间,则执行步骤 3。

步骤 3 若车辆任务的最大容忍时延 γ_i 大于在云服务器处理的总时长($\gamma_i > T_{v_i,Z}$),则卸载至云服务器,跳转至步骤 9;否则执行步骤 4。

步骤 4 选择车辆任务起始 ES。遍历所有 ES,当某 ES 的位置与 v_i 当前的位置满足

$$x_i \in (a_j - \sqrt{R_j^2 - (y_i - b_j)^2}, a_j + \sqrt{R_j^2 - (y_i - b_j)^2})$$

时,将其加入 v_i 的备选服务器集合 m'。当 ES 遍历完时,若 $m' = \{\phi\}$,则系统延迟 0.5 s 后再次发起遍历;若 3 次延迟后,仍无备选 ES,则判定该任务无法完成,跳转到步骤 9;若 $|m'| = 1$,则选择该 ES 为 v_i 的起始 ES;若 $|m'| > 1$,则分别计算 v_i 与集合中每个备选服务器的距离,选择其中距离最近的服务器作为 v_i 的起始 ES,定义为 $m_{j,\text{st}}$,执行步骤 5。

步骤 5 预测车辆终止位置。车辆将信息通过 $m_{j,\text{st}}$ 上传至中心控制器,由 VLPM 预测车辆在经过

最大容忍时延 γ_i 后的位置(x_i^{end},y_i^{end})。

步骤 6　选择车辆任务终止 ES。与步骤 4 的区别仅是将目标由起始 ES 替换为终止 ES,不再赘述。

步骤 7　判定车辆可卸载服务器集合。若起始服务器满足约束:$\phi_{\text{mec},3} \geqslant \theta_i$,则将其加入 β_i;其次,判断终止 ES 是否满足可卸载条件,若终止 ES 满足约束:$\phi_{\text{mec},2} \geqslant \theta_i$,则将终止 ES 加入 β_i;最后,遍历除起始服务器和终止服务器的服务器节点。当某 ES 满足条件:横坐标 $a_j \in [x_i, x_{\text{end}}]$,且 ES 与车的最短距离小于等于该 ES 覆盖范围 R_j,同时满足约束:$\phi_{\text{mec},1} \geqslant \theta_i$,则将其加入 β_i。ES 遍历结束后,执行步骤 8。

步骤 8　中心控制器将计算任务量 θ_i 和 β_i 交付卸载模块进行卸载决策(见 2.3 节)。

步骤 9　任务预处理结束,退出系统。

2.3　任务卸载决策

卸载模块收到预测模块预处理得到的计算任务量 θ_i 和 a_i 后进行卸载决策处理,包括以下 7 个步骤。

步骤 1　初始化内部循环次数 I,迭代次数 λ,信息素浓度,信息素启发因子,期望启发因子,启发式信息。

步骤 2　将任务按 θ_i 进行升序排序。

步骤 3　遍历排序后的任务集合,车辆 v_i 的任务根据式(31)选择要卸载的节点 k_i,若任务在卸载时不满足约束[式(30)],记 k_i 为 -1,表示该任务无可卸载服务器。遍历任务结束后,记录每辆车的卸载节点集合为 1 个可行解 $K = \{k_1, k_2, \cdots, k_n\}$。

步骤 4　若已循环 I 轮,跳转至步骤 5;否则,跳转至步骤 3。

步骤 5　记录在当前迭代次数时全部可行解 K 的集合 K',得到最优解 $K^* = \arg\max_{K \in K'} U_{\text{TCR}}$。若当前迭代次数的最优解大于上一轮的最优解,则根据式(33)对信息素浓度进行更新。

步骤 6　重复步骤 3 ~ 5,直至得到最大迭代次数时记录的最优解并退出循环,执行步骤 7。

步骤 7　将任务按照最优方案分配至各 ES。

3　仿真实验与分析

3.1　实验环境

设定路段长 L 为 1 000 m,宽 A 为 20 m,每辆车的计算能力分布在[10^6,3×10^6]Hz,指令集分布在[8×10^6,10^7]条,行驶速度分布在[20,30] m/s,ES 的计算能力分布在[10^7,3×10^7] Hz,覆盖范围为 100 m,带宽为 1 MHz,发射功率为 1.3 W,高斯白噪声的功率为 3×10^{-13},路径损耗指数为 2,信息素的初始浓度设置为 100,ψ 和 ξ 分别为 1.5 和 2,迭代次数为 50,内部循环次数为 100。仿真实验所用编程语言为 Python3.8,操作系统为 Windows10,中央处理器为 Inteli5 9300HF。

3.2　实验结果分析

下面将所提算法与基于 Q 学习的卸载(QLOF,Q-learning based offloading)[5]算法、权衡重定位次数和最小化响应时间(TradeRC,trade off relocations and response time)[6]算法、自适应任务卸载算法(ATOA,adaptive task offloading algorithm)[7]和移动感知部分(MAP,mobility-aware partial)[8]算法进行对比。

3.2.1　车辆数对 TCR 和资源利用率的影响

假设任务大小和服务器缓存容量恒定,仅改变车辆的数量,车辆数量对 TCR 和资源利用率的影响分别如图 3、图 4 所示。

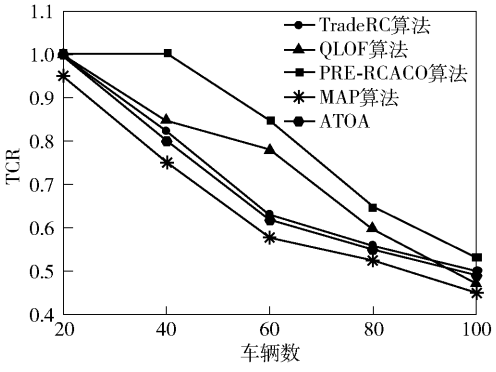

图 3　车辆数量与 TCR 的关系

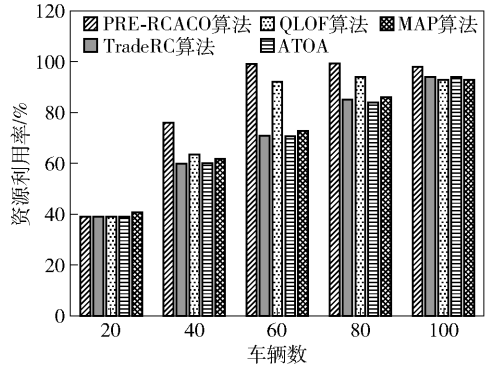

图 4　车辆数与资源利用率的关系

由图 3 可见,5 种算法的 TCR 都随着车辆数的增多而降低。大部分情况下所提 PRE-RCACO 算法

的 TCR 均高于其他算法。由图 4 可见,在车辆数为 60 时,资源利用率达到了 98%,而使用 TradeRC 算法的资源利用率仅为 71%。实验结果表明,在不同的道路环境中,用所提算法可以保持更高的 TCR 和资源利用率。

3.2.2 服务器缓存容量的影响

假定场景中的车辆数和任务计算大小恒定,仅改变服务器缓存容量,服务器的缓存容量对 TCR 和资源利用率的影响分别如图 5、图 6 所示。

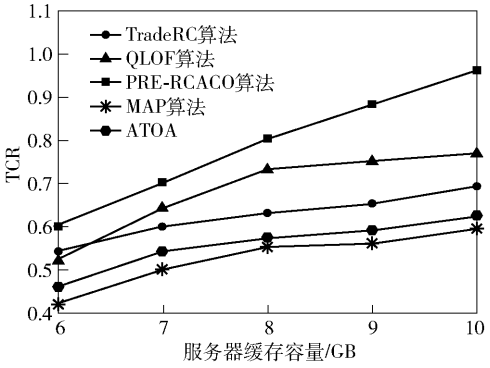

图 5 服务器缓存容量与 TCR 的关系

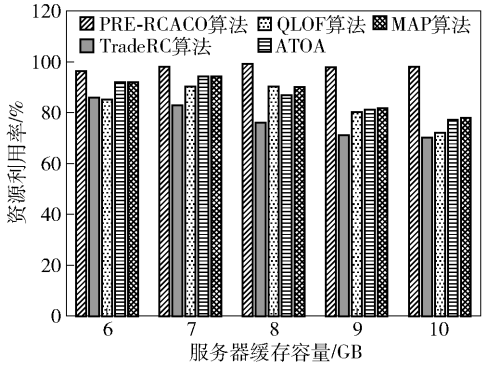

图 6 服务器缓存容量与资源利用率的关系

由图 5 可知,在不同的服务器缓存容量下,所提 PRE-RCACO 算法的 TCR 均高于其他 4 种算法。由图 6 可知,在不同的服务器缓存容量下,所提算法的资源利用率始终在 95% 以上,而其他算法的资源利用率却呈下降趋势。实验结果表明,服务器缓存处于不同的容量时,所提算法的 TCR 和资源利用率始终高于其他算法。

3.2.3 不同算法的收敛情况

将所提算法与通过迭代方法求得最优值的 QLOF 和 TradeRC 算法的收敛性进行了对比。设定场景任务车辆为 95 辆,任务数据大小为 0 ~ 40 MB,ES 缓存容量为 2 GB,迭代次数为 50。为排除收敛曲线的偶然性,对 3 种算法分别进行 20 次仿真实验,并对每次迭代的结果取均值,作为最终的 TCR。3 种算法的 TCR 与迭代次数的关系如图 7 所示。

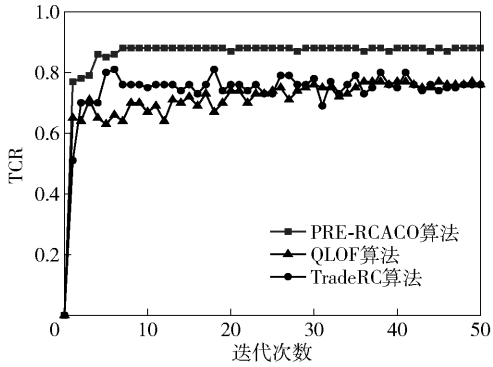

图 7 TCR 与迭代次数的关系

由图 7 可见,用所提 PRE-RCACO 算法在第 7 次迭代时已搜寻到近似最优解并趋于收敛。而 QLOF 和 TradeRC 算法在迭代 40 次后才有收敛趋势。这表明所提算法对收敛参数的改进扩大了寻找最优解的范围并加快了收敛速度。

4 结束语

通过对车辆轨迹的预测,基于计算任务量大小、最大容忍时延、服务器资源等约束条件预判任务卸载方式,同时采用改进的蚁群算法实现在多边缘服务器间卸载任务的最优分配,提升了车联网场景的任务卸载效率,为移动性任务卸载提供了一定的参考。在以后的工作中,将考虑将联邦学习方法应用到卸载决策,进一步提升场景中车辆任务完成率的精度。

参考文献:

[1] LUO Q Y, LI C L, LUAN T H, et al. Self-learning based computation offloading for Internet of vehicles: model and algorithm[J]. IEEE Transactions on Wireless Communications, 2021, 20(9): 5913-5925.

[2] NING Z L, ZHANG K Y, WANG X J, et al. Intelligent edge computing in Internet of vehicles: a joint computation offloading and caching solution[J]. IEEE Transactions on Intelligent Transportation Systems, 2021, 22(4): 2212-2225.

[3] JIANG H, PENG D, YANG K X, et al. Predicted mobile data offloading for mobile edge computing systems [C] // Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018: 153-162.

[4] TU Y P, CHEN H M, YAN L J, et al. Task offloading

based on LSTM prediction and deep reinforcement learning for efficient edge computing in IoT[J]. Future Internet, 2022, 14(2): 30-41.

[5] LIAO Y Z, QIAO X H, YU Q, et al. Intelligent dynamic service pricing strategy for multi-user vehicle-aided MEC networks [J]. Future Generation Computer Systems, 2021, 114: 15-22.

[6] ROY P, TAHSIN A, SARKER S, et al. User mobility and quality-of-experience aware placement of virtual network functions in 5G [J]. Computer Communications, 2020, 150: 367-377.

[7] LIU C H, LIU K, GUO S T, et al. Adaptive offloading for time-critical tasks in heterogeneous Internet of vehicles[J]. IEEE Internet of Things Journal, 2020, 7(9): 7999-8011.

[8] RAZA S, LIU W, AHMED M, et al. An efficient task offloading scheme in vehicular edge computing[J]. Journal of Cloud Computing, 2020, 9(1): 1-14.

(上接第 42 页)

参考文献：

[1] LI J, SHEN X M, CHEN L, et al. Service migration in fog computing enabled cellular networks to support real-time vehicular communications[J]. IEEE Access, 2019, 7: 13704-13714.

[2] MARTINEZ I, HAFID A S, JARRAY A. Design, resource management, and evaluation of fog computing systems: a survey[J]. IEEE Internet of Things Journal, 2021, 8(4): 2494-2516.

[3] ZHAO T C, ZHOU S, GUO X Y, et al. A cooperative scheduling scheme of local cloud and Internet cloud for delay-aware mobile cloud computing[C] // 2015 IEEE Globecom Workshops. Piscataway, NJ: IEEE Press, 2015: 1-6.

[4] DENG R L, LU R X, LAI C Z, et al. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption [J]. IEEE Internet of Things Journal, 2016, 3(6): 1171-1181.

[5] XIAO Y, KRUNZ M. QoE and power efficiency tradeoff for fog computing networks with fog node cooperation[C]// 2017 IEEE Conference on Computer Communications. Piscataway, NJ: IEEE Press, 2017: 1-9.

[6] DONG Y F, HAN C, GUO S T. Joint optimization of energy and QoE with fairness in cooperative fog computing system[C]//2018 IEEE International Conference on Networking, Architecture and Storage (NAS). Piscataway, NJ: IEEE Press, 2018: 1-4.

[7] DONG Y F, GUO S T, LIU J D, et al. Energy-efficient fair cooperation fog computing in mobile edge networks for smart city[J]. IEEE Internet of Things Journal, 2019, 6(5): 7543-7554.

[8] 葛欣炜, 段聪颖, 陈思光. 基于雾计算的能耗最小化公平计算迁移研究[J]. 计算机技术与发展, 2022, 32(3): 107-113.
GE X W, DUAN C Y, CHEN S G. Fog computing based energy minimization and fair computation offloading mechanism [J]. Computer Technology and Development, 2022, 32(3): 107-113.

[9] SUN Y, LIN F H, XU H T. Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-Ⅱ [J]. Wireless Personal Communications, 2018, 102(1): 1-17.