一种面向车载边缘计算基于服务缓存的任务协同卸载算法

唐朝刚[1),2)] 李 召[1)] 肖 硕[1),2)] 吴华明[3)]

[1)](中国矿业大学计算机科学与技术学院 江苏 徐州 221116)

[2)](矿山数字化教育部工程研究中心 江苏 徐州 221116)

[3)](天津大学应用数学中心 天津 300072)

摘 要 为充分利用边缘服务器的有限资源,提高应用服务的缓存效益,本文提出了以应用服务缓存为基础的协同卸载的车载边缘计算模型。在此基础上,以卸载任务的时延和能耗最小化为优化目标,展开对应用服务缓存和计算卸载问题的研究。将服务缓存、任务卸载以及计算资源分配的联合优化建模为非线性整数规划问题。为降低求解难度,将原问题分解为服务缓存和计算卸载联合决策子问题以及边缘服务器计算资源分配子问题。其中,将服务缓存和计算卸载子问题建模为马尔科夫决策过程,并提出了一种基于深度强化学习的缓存卸载方案。仿真结果表明,相较于其它基准方法,本文提出的方案能够将优化目标值降低约7%,响应时延减少约12%,同时将缓存命中率提升约9%。

关键词 车载边缘计算;任务卸载;应用缓存;协作卸载;深度强化学习

中图法分类号 TN92 **DOI号** 10.11897/SP.J.1016.2025.00864

Service Caching Based Collaborative Task Offloading Algorithm in Vehicular Edge Computing

TANG Chao-Gang[1),2)] LI Zhao[1)] XIAO Shuo[1),2)] WU Hua-Ming[3)]

[1)](School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, Jiangsu 221116)

[2)](Mine Digitization Engineering Research Center of the Ministry of Education, Xuzhou, Jiangsu 221116)

[3)](The Center for Applied Mathematics, Tianjin University, Tianjin 300072)

Abstract Vehicular Edge Computing (VEC) extends computing resources from the cloud to the network edge, such as Roadside Units (RSUs), providing nearby vehicles with computational support and enabling data processing and analysis at the edge. Consequently, offloading tasks to the network edge, such as roadside units (RSUs), for execution rather than processing them in the cloud or on vehicles themselves, can better meet the complex requirements of vehicular tasks. Particularly, this approach not only alleviates pressure on the backhaul network but also reduces the network latency. However, several objective factors still constrain the rapid development and performance enhancement of VEC systems. For instance, the computing power and storage resources at Edge Servers (ESs) are significantly lower than those in remote cloud centers. The mobility of vehicles results in a dynamic topology for vehicular ad-hoc networks (VANETs). The real-time requirements of vehicular tasks increase the complexity of allocating network bandwidth and computing resources at the network edge. To further enhance the performance of VEC

收稿日期:2024-04-09;在线发布日期:2025-02-08。本课题得到国家自然科学基金面上项目(62476276, 62271486)资助。**唐朝刚**,博士,讲师,主要研究领域为车载边缘计算、智慧交通和物联网等。E-mail:cgtang@cumt.edu.cn。**李 召**,硕士研究生,主要研究领域为车载边缘计算以及物联网等。**肖 硕**(通信作者),博士,教授,主要研究领域为物联网,自然语言处理等。E-mail:sxiao@cumt.edu.cn。**吴华明**,博士,教授,主要研究领域为边缘计算、物联网和DNA存储等。

systems, it is essential to consider the aforementioned factors and develop more efficient algorithms for task offloading and resource allocation in VEC. In this paper, to make full use of the limited resources of edge servers and improve the caching benefits of application services, a service caching based collaborative offloading model for vehicular edge computing was proposed. On this basis, the research on service caching and task offloading was conducted, with the purpose of minimizing the weighted sum of delay and energy consumption. Considering the reusability of application services, the application service can be cached at the edge server such as roadside units to serve the offloading requests. On the other hand, if the application service is not cached at the network edge, the vehicles can choose to offload the blocks of the required application services to the network edge in a collaborative way, on the assumption that the application service can be divided into smaller parts, i. e. , blocks. The proposed joint optimization of service caching, task offloading and computing resource allocation in vehicular edge computing was actually a nonlinear integer programming problem. To simplify this problem, the original optimization problem was divided into two subproblems. One is the joint optimization problem of service caching and task offloading in vehicular edge computing and the other is the optimization problem of computing resource allocation in vehicular edge computing. Specifically, the optimization problem of service caching and computing offloading was modelled as a Markov Decision Process (MDP), and a deep reinforcement learning-based caching and offloading algorithm was proposed. The optimization problem of computing resource allocation was proven to be convex, which can be solved by existing technologies such as interior point method. Extensive simulation was con-ducted to evaluate the efficiency and effectiveness of the proposed service caching and task offloading problem. Simulation results demonstrate that, in comparison with other baseline approaches, the proposed scheme exhibits superior performance, specifically by reducing the optimal values and response latency by 7% and 12%, respectively, and enhancing the cache hit ratio by 9%.

Keywords　vehicular edge computing; task offloading; service caching; collaborative offloading; deep reinforcement learning

1　引　言

近年来，智慧交通取得了显著进展，各类新能源汽车企业不断涌现，极大地推动了高阶自动驾驶、人机交互，以及车载应用的快速发展。近期，小米SU7的问世不仅激发了国民对中国新能源汽车的热情，也成为中国新能源汽车发展的重要里程碑。智能化已成为当前新能源汽车的显著特征，以燃料电池驱动的智慧车辆实现了软硬件的高度融合，其提供的车机服务由基础的导航指引、辅助驾驶、车辆监测等功能迅速向更高阶的自动驾驶、人车对话、车内VR/AR游戏甚至车载元宇宙等方向发展[1]。在这一演变过程中，车载应用数量以及数据规模都迎来了爆炸式增长。车辆感知的数据不仅包含与娱乐服务相关的延迟不敏感数据，还包含与安全预警相关的延迟敏感数据。这些数据能够及时处理和分析与否不仅关系到能否为用户提供低时延、高可靠的服务质量(Quality of Service，QoS)，更是确保驾乘安全的关键。

车载云计算通过将车载任务卸载到云端处理，显著缓解了车辆自身算力不足的问题。然而，这一模式不仅耗费大量的回程网络带宽资源，还难以满足各类车载应用的实时性需求。相比之下，车载边缘计算(Vehicular Edge Computing，VEC)作为实现智慧交通的关键技术之一，近年来受到了广泛关注。VEC通过将计算资源从云端迁移至网络边缘，如路侧单元(Roadside Unit，RSU)，为车辆提供近距离的算力支持，实现了数据处理和分析在边缘节点的本地化执行，不仅有效减轻了回程网络压力，还大幅度缩短了数据传输导致的网络延迟[2-4]。尽管如此，如下客观因素仍制约着VEC系统的性能提升。例如，边缘服务器(Edge Server，ES)算力和存储等资源远不如远程云中心丰富，因此需要更为高

效的资源分配策略和算法；车辆的移动特性导致车辆自组织网络的拓扑结构频繁变化[5]；车载任务的高实时性要求进一步加大了网络带宽与算力资源分配的复杂度。

另一方面，车载应用的迅猛发展不仅增加了卸载请求的数量，还提高了同一服务被不同用户请求的可能性[6]。因此，服务缓存能在一定程度上有效应对日益增长的应用服务需求，减轻前传网络的压力，缩短车载任务的响应时间，并提升VEC系统的整体性能。例如，当请求已缓存在边缘端的应用服务时，请求者仅需传输相关参数，而无需卸载与任务关联的源码以及数据库资源。针对上述问题，研究者们从任务调度和资源分配的角度进行了广泛研究，并取得了一定的成果[7-8]。

论文作者围绕车载边缘计算中的服务缓存已开展了部分工作[2,9-11]，例如，通过任务缓存策略，以任务卸载及计算资源分配为切入点，旨在最大化VEC系统的整体效能。然而，上述工作仅假设车载任务可以缓存，或者是车载任务与可缓存的服务间存在简单的映射关系；此外，本文尚未深入探讨车辆间的协同任务卸载机制；同时，忽略了服务缓存与计算卸载之间的紧耦合关系，导致缓存和卸载决策过程相对独立，主要依赖贪心算法，这限制了VEC系统在响应延迟和系统能耗等方面的优化潜力。

针对上述问题，本文不仅考虑了服务缓存与任务卸载之间的紧耦合关系，还扩展了两者间的映射关系，并基于车辆之间协同卸载的方式，提出了一个以缓存、卸载及资源分配为决策变量，以车载任务响应时延及能量消耗的加权平均为优化目标的联合优化问题。本文的具体贡献陈述如下：

(1)构建车辆任务与应用服务的映射关系，并将可缓存的服务组件化，当车载任务所关联的服务未被缓存时，车辆可通过协同卸载的方式完成任务的卸载和执行。

(2)对任务卸载、组件上传、服务缓存以及计算资源分配进行联合决策，并提出了一个以任务响应时延及能量消耗的加权平均为目标函数的优化问题。该优化问题本质上是一个非线性整数规划问题，进一步将其分解为服务缓存和计算卸载联合决策子问题以及边缘服务器计算资源分配子问题。

(3)将服务缓存和计算卸载决策子问题建模为马尔科夫决策过程，并提出了一种基于近端策略优化(Proximal Policy Optimization，PPO)的联合服务缓存和任务卸载方案(Joint Application Caching and Task Offloading Based on PPO，JACTOP)。

(4)设计了仿真实验平台，细化了评价指标，从多个角度对所提策略进行评估；仿真结果显示，与传统缓存置换算法相比，本文方法在优化目标、执行时延和缓存命中率等方面表现出更优的性能。

2 相关工作

随着智慧车辆的兴起，车载计算设备如车载单元(On Board Unit，OBU)以及路侧单元RSU受到了广泛关注并取得了快速发展。VEC通过将计算资源部署在OBU和RSU上，实现了在严格响应时间要求下运行多种应用服务的能力。近年来，VEC的任务卸载和服务外包因其独特优势而日益受到重视[12]。

Zeng等[13]提出了一种由志愿者辅助的VEC模型，志愿者车辆参与协助VEC服务器执行任务，并从中获取收益。他们分别定义了车辆的成本函数以及VEC服务器的效用函数，利用博弈论分析车辆的最优卸载决策，并通过遗传算法确定VEC服务器资源的最优定价策略。Raza等[14]针对5G通信环境下的VEC场景提出了一个计算效率(即单位能耗可处理的任务数据量)的优化问题。为最大化计算效率，提出了一种基于博弈论的分布式卸载策略，并通过凸优化方法解决了计算资源的分配问题。文献[15]将VEC环境中的任务到达过程建模为M/M/1排队模型，并依据任务等待时间与平均等待时间的差值调整计算资源的分配。

近年来，深度强化学习(Deep Reinforcement Learning，DRL)凭借着其卓越的感知决策能力在边缘计算领域得到了广泛应用[16]。DRL无需依赖任何先验知识，通过环境状态感知数据间的关系，采用迭代交互训练方式，实现行动策略的智能化探索[17-18]。文献[19]针对服务缓存感知的VEC场景，利用ES间的协作，探索了多种可分割的任务卸载情况，提出了一种基于DRL的多动作混合策略，旨在联合优化服务缓存和计算卸载，从而最小化平均任务处理延迟。然而，该工作未充分考虑到服务缓存与任务卸载之间的紧耦合关系。文献[20]研究了三层移动边缘计算系统中的任务卸载、服务缓存以及资源分配的联合优化问题，并将该问题建模为马尔科夫决策过程，提出了一种基于DDQN的方案来解决该优化问题。该工作主要聚焦于边缘服务器和云服务器之间的协同合作，以支持移动设备获取

所需资源，但忽略了移动设备间的合作。文献[21]综合考虑了网络带宽、延迟、边缘节点负载、延迟敏感度等因素，提出了一种融合长短期记忆网络(LSTM)和DRL的混合算法，用于解决边缘端工作负载问题。具体来说，LSTM网络负责预测未来的边缘节点负载，并在决策阶段使用双深度Q网络确定任务卸载位置。张宇等[22]结合数据命名网络与边缘计算，设计了一种智能化的资源管理框架。该框架利用矩阵分解技术预测内容流行度；通过从高维度向低维度的映射提高稀疏矩阵的处理能力；最后利用DRL算法对计算和缓存资源的分配以及缓存放置策略进行联合优化，以实现系统收益的最大化。

Li等[23]针对VEC场景中服务缓存替换以及RSU间的协同合作进行了研究，旨在最小化计算成本和处理时延，并提出了一种结合Gibbs采样和DRL的迭代算法以解决上述最小化问题。Hao等[24]探索了任务缓存和卸载的联合优化问题，提出一种基于交替优化的迭代算法，以实现用户能耗的最小化。Zhou等[25]基于边缘计算的智能电网场景，提出了一种协同计算卸载和资源分配方法。首先基于博弈论求解计算卸载问题，然后将计算资源切分成单元块，为每个任务初始化资源，并根据梯度下降进行资源分配，得到近似最优解。

文献[10]针对车载应用对VEC资源的需求，提出了一种任务卸载模型；该模型不仅考虑了应用服务缓存，还通过将车载应用外包至服务器端执行，有效降低了车载任务的平均响应时延。与之不同的是，文献[26]中的任务执行依赖多个应用程序的集合，而非单一应用。因此，问题被定义为应用缓存和计算卸载的联合优化，旨在最大化服务器利润。考虑到问题求解的复杂性，采用了交叉熵方法来寻找近似的最优策略。

上述研究主要集中在服务缓存和计算卸载的联合优化问题上，但较少涉及本地化的服务向ES上传的情况。文献[10]虽然将应用外包到ES上执行，但未充分考虑同一应用程序可能被多个车辆同时卸载的实际情形。这种重复的应用卸载不仅会消耗额外的链路资源，还会导致任务传输时延的增加。

3　系统模型

考虑一种典型的VEC系统，如图1所示。该模型由一个集成ES的RSU、远程云中心，以及多个智慧车辆构成。ES通过有线方式与云服务器相连，可以通过回程链路从云端下载相应的车载应用服务程序，并将其缓存在边缘端。车辆采用V2I(Vehicle-to-Infrastructure)通信技术与基础设施RSU进行无线通信。假设RSU高度为H，通信覆盖范围受限，通信半径为R，ES的计算能力为F，缓存能力为C。假设N个有任务卸载需求的车辆在此RSU覆盖的路段上自由行驶。该系统存在K个可缓存的车载应用程序，$\mathcal{A}=\{A_1,\cdots,A_K\}$，其中$A_k=\{S_k, W_k\}$，$S_k$表示车载应用程序的数据规模，$W_k$表示工作负载，即执行该应用所需的CPU周期数。此外，假设应用程序可分，即每个应用程序A_k都可划分为M个固定的应用组件$A_k=\{a_{k,1},\cdots,a_{k,M}\}$，组件$a_{k,m}$的数据规模为$s_{k,m}$，则有如下约束成立，即$S_k=\sum_{m=1}^{M}s_{k,m}$。任务车辆集合为$\mathcal{N}=\{1,\cdots,N\}$，随机分布在该RSU的通信范围内，车辆$n$的速度为$sp_n$，车辆在本地已经缓存了部分常用的应用程序；每个时隙都有车辆进入或驶离RSU覆盖范围，并假设RSU覆盖范围内的任务车辆数始终保持不变。假设任务车辆n在每个时隙会产生一个与车载应用关联的任务卸载请求$\Omega_n=\{D_n, A_k, T_n^{\max}\}$，$D_n$表示用户参数大小，$A_k\in\mathcal{A}$表示一个车载应用程序，$T_n^{\max}$表示任务响应时间约束。

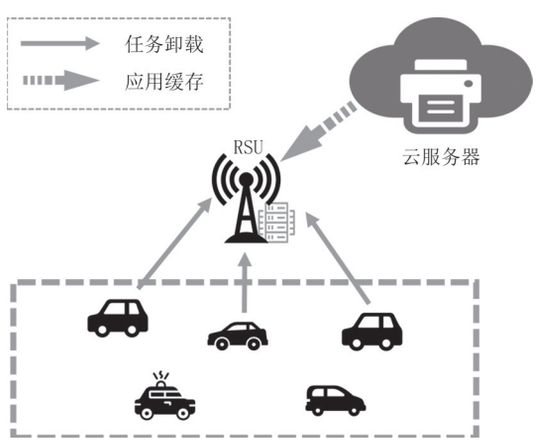

图1　VEC系统模型

通常，车辆可以选择利用自身算力资源在本地执行任务，也可选择将任务卸载到网络边缘RSU处执行，以此缓解自身算力不足的缺点，减少任务的响应时延，提升用户体验。考虑到应用程序可在ES处缓存，与应用程序关联的任务卸载可具体描述如下：如果与任务关联的应用程序已被ES缓存，那么车辆只需要上传相应的用户参数D_n；否则，车辆需要将完整应用程序如程序源码卸载至ES处执行。定义两个二进制变量$\varphi_{n,k}$和ϕ_k，其中$\varphi_{n,k}$表示车辆n

的任务是否与应用程序A_k相关联，如果任务关联A_k，则$\psi_{n,k}=1$；否则，$\psi_{n,k}=0$。假设一个车载任务同时只能与一个应用程序相关，即存在约束$\sum_{k=1}^{K}\psi_{n,k}=1,\forall n\in\mathcal{N}$。$\phi_k$表示应用程序$A_k$是否被请求卸载的任务所关联，如果存在某个需要卸载的任务关联A_k，则$\phi_k=1$；否则，$\phi_k=0$。需要指出，多个车载任务可能同时与应用程序A_k关联。此外，为了便于表述，文中形式化描述所需的主要参数已在表1中列出。

表1 系统参数

参数	含义
K	系统中车载应用程序的数量
N	车辆用户的数目
M	应用程序的组件数目
C	ES的缓存能力
F	ES的计算能力
B	ES的带宽
S_k	第k个应用程序的数据大小
W_k	执行第k个应用程序所需的CPU周期数
$s_{k,m}$	应用程序A_k的第m个组件的数据大小
Ω_n	车辆n所产生的任务
D_n	任务n所需的用户参数
$T_n^{\max}$	任务n的时延约束
p_n	车辆n的传输功率
h_n	车辆n与RSU之间的信道增益
σ^2	传输信道的高斯噪声
$\psi_{n,k}$	任务n的执行是否需要应用程序A_k参与
ϕ_k	应用程序A_k是否被请求参与执行任务
α_k	应用程序A_k是否缓存在ES中
β_n	任务n是否卸载到ES

3.1 缓存模型

如果应用程序已经缓存在ES，那么车辆只需要上传关联的用户参数，因此可以显著减少任务的响应时间。定义缓存变量α_k，如果A_k被缓存，则$\alpha_k=1$；否则，$\alpha_k=0$。受限于ES的缓存能力，车载应用服务程序的缓存数量需要满足以下约束

$$\sum_{k=1}^{K}\alpha_k S_k\leqslant C \tag{1}$$

其中，C表示ES的缓存能力。

3.2 通信模型

车辆通过V2I无线通信将任务卸载至RSU执行，定义二进制变量β_n为任务卸载的决策变量。当车辆n产生的任务需要卸载到RSU处执行时，$\beta_n=1$；否则$\beta_n=0$。p_n表示车辆n的传输功率，h_n表示车辆n与RSU之间的信道增益，根据香农定理可得，车辆n向RSU传输数据的速率为

$$R_n=B_n\log\left(1+\frac{p_n h_n}{\sigma^2}\right) \tag{2}$$

其中，$B_n=B/\sum_{n=1}^{N}\beta_n$是RSU分配给车辆$n$的带宽资源，$B$是RSU的总带宽，$\sigma^2$表示高斯噪声。

假设车载任务Ω_n关联应用程序A_k，如果ES已经缓存A_k，那么车辆n只需上传用户参数即可；否则，车辆n需将完整的A_k卸载至ES，然后再传输相关参数。

需要指出，在当前时隙，可能存在多个车载任务与应用程序A_k关联。变量N_k表示卸载到RSU处执行并关联应用程序A_k的任务数量，则$N_k=\sum_{n=1}^{N}\psi_{n,k}\beta_n$；定义$V_k$为发起上述任务的车辆集合，$V_k\in\mathcal{N}$。如果$A_k$已在ES中缓存，则$V_k$中的车辆只需上传任务所需的参数；如果$A_k$未缓存，$V_k$中的车辆可协作上传$A_k$的部分组件，以减轻前传网络的工作负载，同时降低传输时延。具体来说，在任务卸载请求阶段，RSU通过与车辆交互感知车辆任务信息，进而制定组件分配及上传策略，然后通知V_k中的车辆上传各自负责的A_k组件以及用户参数。定义二进制变量$\chi_{n,k,m}$表示应用组件$a_{k,m}$是否由车辆$n\in V_k$负责上传，若$a_{k,m}$由车辆n上传，则$\chi_{n,k,m}=1$；否则，$\chi_{n,k,m}=0$。因此，$\chi_{n,k,m}$满足如下约束：

$$\sum_{n\in V_k}\chi_{n,k,m}=1,\ \forall k\in\{1,\cdots,K\},m\in\{1,\cdots,M\} \tag{3}$$

因此，车辆n卸载应用组件所需的传输时延为

$$T_n^{up,comp}=\beta_n\sum_{k=1}^{K}\psi_{n,k}(1-\alpha_k)\sum_{m=1}^{M}\chi_{n,k,m}\frac{s_{k,m}}{R_n} \tag{4}$$

车辆n上传用户参数的传输时延为

$$T_n^{up,para}=\beta_n\sum_{k=1}^{K}\psi_{n,k}(1-\alpha_k)\frac{D_n}{R_n} \tag{5}$$

因此，任务n的传输时延可表示为

$$T_n^{up}=T_n^{up,comp}+T_n^{up,para} \tag{6}$$

车辆n的传输链路能耗为

$$E_n^{up}=p_n T_n^{up} \tag{7}$$

3.3 等待模型

当任务被卸载至ES后，ES立即为任务分配计算资源，如创建虚拟机（Virtual Machine，VM）执行卸载的任务。尽管时延和能耗是评估VEC系统性能的两个关键指标，但现有文献往往忽略了虚拟机创建过程中的时间和能耗成本。创建VM不仅涉及显著的时间开销，其能耗也远超维持VM运行所需的能耗。因此，为了完善任务卸载模型，本文将这两项指标纳入所提出的系统模型中。

假设在VEC系统中只有车辆n卸载与应用程序A_k关联的车载任务Ω_n。此时，如果A_k已在ES缓存，任务Ω_n则无需等待ES创建VM；否则，任务Ω_n需要等待ES为A_k创建VM并分配必要的算力资源。结合上述两种情况，任务Ω_n的等待时间可表示为

$$T_n^{wait}=\sum_{k=1}^{K}\psi_{n,k}(1-\alpha_k)T_k^{init} \tag{8}$$

其中，T_k^{init}表示ES为A_k初始化VM所需的时间。

假设当前有多个车辆卸载与A_k关联的车载任务，即车辆集合为V_k，且该集合大小为$N_k=\sum_{n=1}^{N}\psi_{n,k}\beta_n$。此时，任务$\Omega_n$的等待时间可按如下情况分别讨论：

（1）如果A_k已缓存，ES需要复制N_k-1份应用程序，接着再创建N_k-1个VM；应用程序的复制和VM的创建过程可并行执行，考虑到ES具有较强的算力资源，此处忽略了应用程序的复制时间。因此，Ω_n的等待时间可表示为

$$T_n^{wait}=\sum_{k=1}^{K}\psi_{n,k}T_k^{init} \tag{9}$$

（2）如果A_k未被缓存，ES需要等待A_k的所有组件都到达后，将其重组为完整的应用程序；接着复制N_k-1份应用程序，并创建N_k个VM。定义A_k的整合时间为$T_k^{inte}=S_k\eta$，其中η表示整合单位比特数据所需时间。注意，当车辆用户参数传输结束时刻晚于应用组件整合结束时刻，则Ω_n无需等待可直接执行。因此，任务Ω_n的等待时间可表示

$$T_n^{wait}=\sum_{k=1}^{K}\psi_{n,k}\max\left(\max_{u\in V_k}T_u^{up,comp}+T_k^{inte}-T_n^{up},0\right) \tag{10}$$

在上述过程中，创建以及维持VM所需的能耗依赖于特定的应用程序，E_k表示应用A_k的能耗开销，则E_k可定义为

$$E_k=\phi_k\left((1-\alpha_k)E_k^{init}+\alpha_kE_k^{sus}+(N_k-1)E_k^{init}\right)=\phi_k\left((N_k-\alpha_k)E_k^{init}+\alpha_kE_k^{sus}\right) \tag{11}$$

3.4　计算模型

计算任务Ω_n既可以在本地执行也可以卸载到RSU执行，根据卸载决策，任务Ω_n的计算时延如下

$$T_n^{com}=\sum_{k=1}^{K}\psi_{n,k}\left((1-\beta_n)\frac{W_k}{F_n}+\beta_n\frac{W_k}{f_n}\right) \tag{12}$$

其中，f_n表示ES分配给任务n的计算资源。

车辆n的任务对应的计算能耗可表示为

$$E_n^{com}=\sum_{k=1}^{K}\psi_{n,k}\left((1-\beta_n)\varepsilon_nW_kF_n^2+\beta_n\varepsilon_0W_kf_n^2\right) \tag{13}$$

其中，ε_n和ε_0分别表示车辆n和ES的能耗系数。

因此，VEC系统中所有车辆任务的响应时延可以表示为

$$T^{total}=\sum_{n=1}^{N}\left(T_n^{up}+T_n^{wait}+T_n^{com}\right) \tag{14}$$

VEC系统的能量消耗可表示为

$$E^{total}=\sum_{n=1}^{N}\left(E_n^{up}+E_n^{com}\right)+\sum_{k=1}^{K}E_k \tag{15}$$

3.5　问题建模

针对动态变化的VEC环境以及RSU有限的算力资源，本文通过联合优化应用缓存、计算卸载、组件和计算资源分配，旨在提升VEC系统的整体性能。具体而言，在满足各项约束条件的前提下，最小化系统中所有任务的处理时延和能耗。为此，目标函数设计如下

$$U(\alpha,\beta,\chi,f)=\omega T^{total}+(1-\omega)E^{total} \tag{16}$$

其中，$\omega\in(0,1)$表示加权因子，$\alpha=\{\alpha_k|1\leqslant k\leqslant K\}$，$\beta=\{\beta_n|1\leqslant n\leqslant N\}$，$\chi=\{\chi_{n,k,m}|1\leqslant n\leqslant N,1\leqslant k\leqslant K,1\leqslant m\leqslant M\}$，$f=\{f_n|1\leqslant n\leqslant N\}$。

至此，VEC中应用缓存、计算卸载、组件及计算资源分配的联合优化问题可建模如下：

$$\begin{aligned}
&P1:\min_{\alpha,\beta,\chi,f}U\\
&\text{s.t.}C1:\sum_{k=1}^{K}S_k\alpha_k\leqslant C\\
&C2:\sum_{n=1}^{N}\beta_nB_n\leqslant B\\
&C3:\sum_{n=1}^{N}\beta_nf_n\leqslant F\\
&C4:\sum_{n\in V_k}\chi_{n,k,m}=1,\forall m\in\{1,\cdots,M\}\\
&C5:\alpha_k\in\{0,1\},\forall k\in\{1,\cdots,K\}\\
&C6:\beta_n\in\{0,1\},\forall n\in\mathcal{N}\\
&C7:\chi_{n,k,m}\in\{0,1\},\forall n\in\mathcal{N},k\in\{1,\cdots,K\},m\in\{1,\cdots,M\}
\end{aligned} \tag{17}$$

约束$C1$表示应用程序缓存所需存储空间不得超过RSU的缓存容量；$C2$表示在任务卸载过程中，对带宽资源的分配不得超出设定的阈值；$C3$表示ES分配的计算资源不能超出其最大算力；$C4$表明应用程序的各组件同一时刻只能由一个车辆负责卸载；$C5$、$C6$、$C7$是对二进制决策变量的约束。

显然，上述问题既包含α,β,χ三个离散变量也包括计算资源分配的连续变量f，因此，P1本质上是一个非凸的混合整数规划问题，这无疑增加了问题的求解难度。然而，仔细分析发现，当确定α,β,χ

后，问题P1就转化为一个只与连续变量f相关的优化问题，为叙述方便，将此只与f相关的优化子问题定义为P2。

假设当前有$J=\sum_{n=1}^{N}\beta_n$个车辆任务卸载到RSU执行，并且VM也已分配完毕。此时，J个任务的计算时延和计算能耗的加权和可表示为

$$G(f_1,\cdots,f_J)=\sum_{j=1}^{J}\left(T_j^{com}+E_j^{com}\right)=\sum_{j=1}^{J}\left(\sum_{k=1}^{K}\psi_{j,k}\left(\omega\frac{W_k}{f_j}+(1-\omega)\varepsilon_0 W_k f_j^2\right)\right) \tag{18}$$

此处，$\psi_{j,k}$在任务请求阶段已经确定。因此，问题P2可表述为

$$\begin{aligned}&\min_{f} G\\&\text{s.t.}\ C3\end{aligned} \tag{19}$$

而且，P2中的目标函数G关于f是凸的，证明如下：首先，计算目标函数G关于f的Hessian矩阵

$$H(G)=\begin{pmatrix}\frac{\partial^2 G}{\partial f_1^2} & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \frac{\partial^2 G}{\partial f_J^2}\end{pmatrix} \tag{20}$$

显然，$H(G)$是一个对角矩阵，对于任意的f_j，$j\in\{1,\cdots,J\}$，其二阶混合偏导为

$$\frac{\partial^2 G}{\partial f_j^2}=2\sum_{k=1}^{K}\psi_{j,k}\left(\omega\frac{W_k}{f_j^3}+(1-\omega)\varepsilon_0 W_k\right)>0 \tag{21}$$

因此，$H(G)$是一个正定矩阵，目标函数G关于f是一个凸函数，可以通过现有技术如内点法进行求解。另外，对于缓存、卸载以及组件分配的决策即α,β,χ的求解将在接下来的章节中着重阐述。

4 算法设计

4.1 基于PPO的应用缓存和任务卸载联合优化

近年来，DRL算法因其卓越的感知与决策能力，在多个领域显露头角。为应对车联网的动态变化，本文提出一种基于DRL的联合应用缓存与任务卸载的决策方法。该方法依赖历史卸载和缓存数据来制定缓存决策。因此，本文采用具有经验回放机制的PPO算法解决缓存和卸载的优化问题。对比深度Q网络(Deep Q Network，DQN)算法，PPO根据策略梯度进行更新，不仅训练速度更快，而且具有更好的收敛性能。PPO算法的流程图如图2所示。将应用缓存和计算卸载问题建模为马尔科夫决策过程，并基于PPO算法对这两种策略进行联合优化。

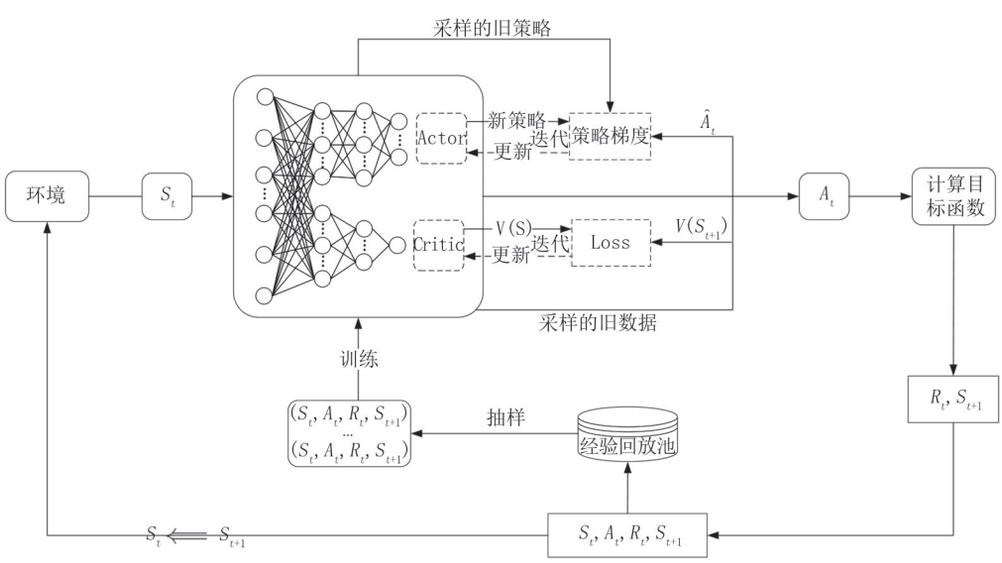

图2 PPO算法流程图

首先，确定状态空间，动作空间以及奖励函数如下：

(1) 状态空间。环境状态包含了深度强化学习所需的全部信息，如应用程序的历史请求信息、车载任务信息和RSU(ES)相关信息如缓存能力以及算力资源等。因此，状态空间可定义为$S_t=\{Q_{t-L},\cdots,Q_{t-1},V_1,\cdots,V_N,f\}$，其中$Q_t=\{c_{t1},\cdots,c_{tk}\}$，$c_{tk}$表示应用程序$A_k$在时隙$t$参与任务卸载请求的频次；$V_n=\{loc_n,sp_n,dir_n,p_n,\varepsilon_n,task_n\}$，分别表示车辆的位置、速度、方向、传输功率，能耗系数以及任务信息；f

表示ES的剩余计算资源。

(2) 动作空间。智能体根据网络的状态输入，预测并输出优化的动作。动作空间主要包含缓存动作和卸载动作两部分，即，$A_t=\{\alpha_1,\cdots,\alpha_k,\beta_1,\cdots,\beta_n\}$，其中$\alpha_k$表示应用程序$A_k$的缓存决策，$\beta_n$表示任务$n$的卸载决策。

(3) 奖励函数。本文旨在最小化时延和能耗的加权和，而DRL以最大化未来累计奖励为目标，因此，奖励函数可设为目标函数的相反数，即$R_t=-U$。

PPO的网络结构包含一个策略网络Actor和一个评估网络Critic，前者用于优化策略梯度$\pi(a_t|s_t;\theta)$，后者评估当前状态价值$V(s_t;\theta')$。PPO利用优势函数指示当前状态下的输出动作所具有的优势，并采用k步广义优势估计(Generalized Advantage Estimation，GAE)进行计算，如下所示

$$\hat{A}_t=\sum_{i=0}^{k-1}(\gamma\lambda)^i\delta_{t+i} \tag{22}$$

$$\delta_t=r_t+\gamma V(s_{t+1})-V(s_t) \tag{23}$$

δ_t是时间差分(TD，Temporal Difference)，表示单个时隙内，真实值与预测值之间的差距，γ、λ是折扣因子。

PPO算法引入重要性采样技术，从经验回放中采样旧数据，用以评估新策略的期望。

$$J^{\bar{\theta}}(\theta)=E_{(s_t,a_t)\sim\pi_{\bar{\theta}}}\left[\frac{p_\theta(s_t,a_t)}{p_{\bar{\theta}}(s_t,a_t)}\hat{A}_t\right] \tag{24}$$

这里，$\bar{\theta}$是旧的策略网络参数，θ是要优化的策略网络参数，算法训练目标是使得新旧策略的分布逐步接近，即$ratio=\frac{p_\theta(s_t,a_t)}{p_{\bar{\theta}}(s_t,a_t)}\to 1$，其中ratio表示重要性权重。在训练过程中，新旧策略可能因分布差异过大造成网络梯度更新的效果欠佳。为了降低这种影响，引入裁剪函数，如下式所示

$$clip(ratio,1-\epsilon,1+\epsilon) \tag{25}$$

如果$ratio<1-\epsilon$，输出$1-\epsilon$；如果$ratio>1+\epsilon$，输出$1+\epsilon$。PPO策略网络的损失函数表达式如下

$$loss_\theta=E_{(s_t,a_t)\sim\pi_{\bar{\theta}}}\left[\min\left(ratio,clip(ratio,1-\epsilon,1+\epsilon)\right)\hat{A}_t\right] \tag{26}$$

评价网络的损失函数表示为

$$loss_{\theta'}=\left(r_t+\gamma V(s_{t+1};\theta')-V(s_t;\theta')\right)^2 \tag{27}$$

算法1.　基于PPO的应用缓存和任务卸载联合优化

输入：应用的历史卸载请求频次，车辆任务的相关信息以及ES的可用资源

输出：应用的缓存决策，车辆任务的卸载决策

1. 初始化PPO的网络参数，定义θ,θ'分别是PPO的Actor和Critic网络权重，经验回放池为D，每次学习次数eps；设置记数器$t=0$，最大训练迭代次数MT。
2. WHILE $t<MT$：
3. 　获取系统状态空间S_t，由PPO策略网络给出动作A_t；
4. 　对缓存进行约束，初始化变量TS，记录应用缓存所占空间大小；
5. 　FOR $i=0$ TO K：
6. 　　$TS+=S_i*A_t[i]$；
7. 　　IF $TS>C$：
8. 　　$A_t[i]=0$；
9. 　环境执行动作A_t，得到VEC环境的反馈奖励R_t，下一缓存状态空间S_{t+1}；
10. 　将$<S_t,A_t,R_t,S_{t+1}>$存储到D中；
11. 　更新PPO网络参数：
12. 　从D中抽样一批经验数据，根据式(22)计算优势函数$\hat{A}_t$；
13. 　FOR $kt=0$ TO eps：
14. 　　根据式(23)和(27)执行梯度下降更新θ'，根据式(26)执行梯度下降更新θ；
15. 　$t=t+1$；

4.2　基于贪心选择的组件分配决策

当多个卸载任务都需要关联同一个应用程序，如A_k，且A_k未被RSU缓存时，车辆可与RSU通信，使得后者获知A_k的相关信息，并可确定集合V_k，然后RSU负责制定相应的组件分配决策。

合理的组件分配策略可以有效缩短任务的响应时延，而分配不当则会增加响应时延。例如，若应用的组件仅由单一车辆负责卸载，尽管其他车辆不会因组件卸载产生额外的传输时延，但它们可能需要等待更长时间以完成组件的卸载与整合，从而导致整体任务响应时延增加。基于此分析，组件分配的目标可形式化表述为最小化组件传输的最大时延，即$\min(\max_{n\in V_k}T_n^{up,comp})$。为此，设计基于贪心规则的组件分配算法，将组件按照传输时延的大小，贪心地分配给传输时延最小的车辆，如算法2所示。

算法2.　基于贪心选择的组件分配策略

输入：应用A_k，卸载决策中关联A_k的任务车辆集合V_k

输出：车辆集合V_k关于A_k的组件分配决策

1. 将A_k的应用组件按照数据大小，从大到小排序；初始化列表CQ并保存组件分配结果。
2. FOR $m=1$ TO M：　# 假设当前的组件为$a_{k,m}$；
3. 　FOR $n\in V_k$：

4. 当前车辆n的组件集合$CQ[n]$,计算车辆n上传组件集合$CQ[n]\cup a_{k,m}$的传输时延$T_n^{up,comp}$;

5. 将组件$a_{k,m}$分配给车辆$n=\underset{i\in V_k}{\operatorname{argmin}} T_i^{up,comp}$,并将$a_{k,m}$添加到$CQ[n]$;

6. RETURN CQ。

4.3 计算复杂度分析

算法2的时间复杂度简要分析如下:从系统整体角度分析,算法2不仅与卸载决策β相关,还与应用程序的组件数目M和请求该应用的车辆集合相关,最好情况下,卸载决策中关联应用程序A_k的车辆集合V_k中的元素个数小于1,则算法2无需执行;最坏情况下,卸载决策中的任务车辆都参与了算法2的执行,因此,时间复杂度为$O\left(M\sum_{n=1}^{N}\beta_n\right)$。

5 仿真实验

本节使用Python进行仿真实验。为了验证本文方法的有效性,分别与以下三种基准方案进行对比分析。

(1)PPO缓存与全部卸载(PCOC,PPO caching and offloading completely):由PPO进行缓存决策,之后将车载任务全部卸载到ES去执行。

(2)最近最少使用缓存与PPO卸载(LFUCPO,least frequently used caching and PPO offloading):根据历史请求数据,置换最近最少请求的应用,将最近经常请求的应用缓存在ES,并利用PPO进行卸载决策。

(3)随机缓存与PPO卸载(RCPO,random caching and PPO offloading):在满足ES缓存空间限制的情况下,随机缓存一定数目的应用,并利用PPO进行卸载决策。

5.1 仿真参数设计

在仿真实验中,车辆的数目为10~30,随机分布在RSU所覆盖的道路段内,RSU通信半径为100 m,带宽200 MHz,ES的缓存空间500 M bits,计算资源20 GHz。车载应用程序的数目为20个,每个应用程序由20个组件构成,每个组件的数据大小为1 M bits~5 M bits,应用程序执行一个任务所需的CPU周期数200 MHz~400 MHz。车辆任务的用户参数大小为1 M bits~5 M bits,任务最大时延约束0.6 s~1.2 s,车辆行驶速度5 m/s~10 m/s,传输功率0.3 W~0.6 W,本地计算资源600 MHz~1000 MHz。仿真参数如表2所示。

表2 仿真参数

参数	值
车辆数目/(辆)	10~30
RSU通信半径/(m)	100
RSU带宽/(MHz)	200
ES缓存空间/(M bits)	500
ES计算能力/(Megacycle)	20 000
ES能耗系数ε	10^{-26}
应用程序数目/(个)	20
应用组件数目/(个)	20
应用组件数据大小/(M bits)	1~5
应用执行任务所需CPU/(Megacycle)	200~400
任务的参数大小/(M bits)	1~5
最大时延约束/(s)	0.6~1.2
车辆行驶速度/(m/s)	5~10
车辆计算能力/(Megacycle)	600~1000
车辆的能耗系数	10^{-26}~10^{-25}

5.2 仿真结果和分析

图3展示了不同方案下的策略训练过程。如图所示,随着训练迭代次数的增加,奖励逐渐增加,目标函数逐渐减小,最终趋于收敛。这表明智能体通过不断与环境交互,优化更新自身的动作策略,从而变得更加智能。与其他基准方案相比,联合优化的JACTOP算法获得了更大的奖励回报,表现出更优的决策能力。在1000次迭代之前,LFUCPO和JACTOP比较接近,但受LFU缓存决策能力的限制,LFUCPO的卸载决策难以在后续训练过程中进一步优化。全部卸载对PCOC产生了较大的限制,因为ES的带宽资源和计算资源都是有限的,得益于其缓存策略,最终优于RCPO方法。

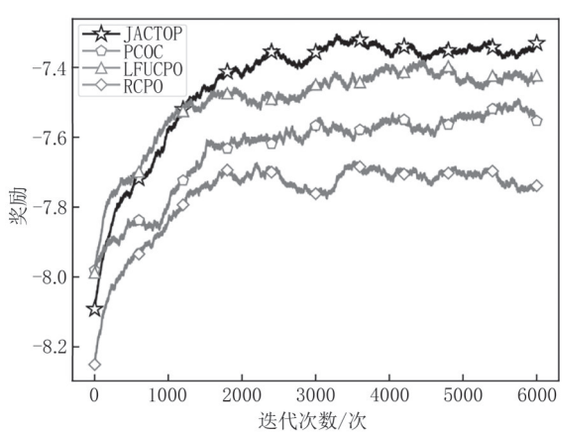

图3 训练过程

图4展示了时延和能耗随权重因子变化的趋势。目标函数的权重因子用于量化时延和能耗的影

响力度，权重因子越大，目标函数越倾向于优化时延；反之，则更侧重于能耗的优化。为了确定合适的权重值，在20个车辆任务的条件下，通过调整权重因子，获得了如图4的实验结果。在时延和能耗曲线的交点附近（约0.6），权重因子对两者的影响相当，因此，在后续的实验中，将其设置为0.6。

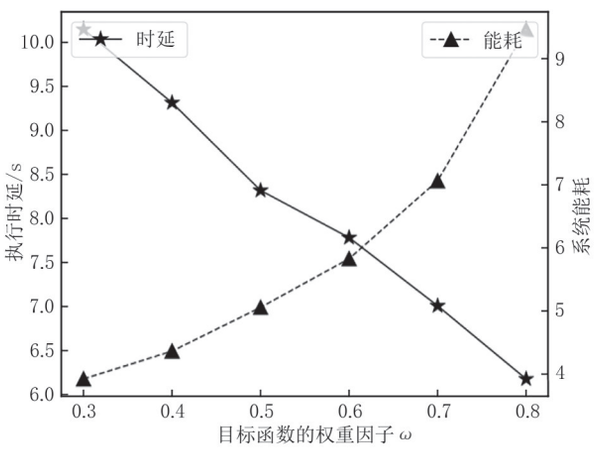

图4　权重因子对时延和能耗的影响

图5和图6分别对比了不同方案下，目标函数值和任务平均执行时延随车辆任务数目的变化趋势。如图所示，本文提出的JACTOP方法表现出色，因为PPO能够有效感知历史信息与环境状态之间的动态关系，从而做出最优的缓存和卸载决策。特别是当多个车辆请求了同一个应用服务时，缓存能够最大程度地减少时延和能耗。当任务数目较少时，PCOC同样能够最大限度地优化目标函数，这得益于PPO缓存决策的优势以及ES服务器充足的算力资源。然而，随着任务数量的增加，该方案的性能逐渐下降。一方面，由于带宽资源有限，每个任务分配到的带宽减少，导致传输时延增加，任务的紧急程度相对提升，为了满足时延约束，需要更多的计算资源；另一方面，有限的计算资源难以满足大量任务的需求，从而导致更高的计算时延。从LFUCPO的表现来看，基于历史请求频次进行缓存能够发挥一定作用，而随机缓存方案则难以适应任务对应用的随机需求，因此表现较差。

图7展示了任务执行时延随ES计算能力变化的趋势。随着计算资源的增加，各方案的计算时延均呈现先逐步下降，随后趋于稳定的特征。由于本文旨在最小化时延和能耗的加权和，时延的减小往往伴随着能耗的增加，因此需要在两者间寻求最优平衡点，使得执行时延随着ES计算资源的增加逐步进入一个稳定区间。此外，当ES计算能力为10 GHz时，PCOC方案表现出较高的执行时延，这归因于有限的计算资源无法有效应对大量的任务需求，从而导致较高的任务执行时延。LFUCPO方法的整体性能优于RCPO，主要得益于其更为有效的缓存策略，经过PPO的卸载优化后，依然保持着这一优势。

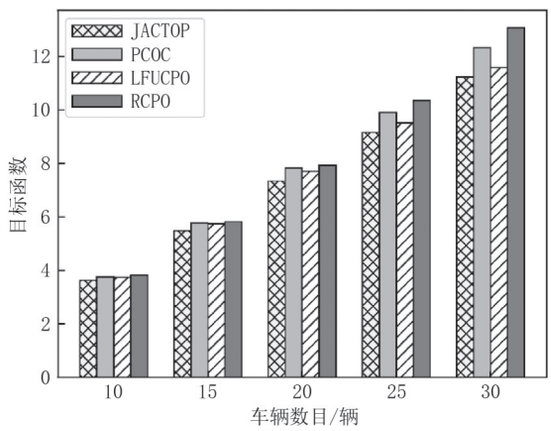

图5　目标函数随车辆数目的变化

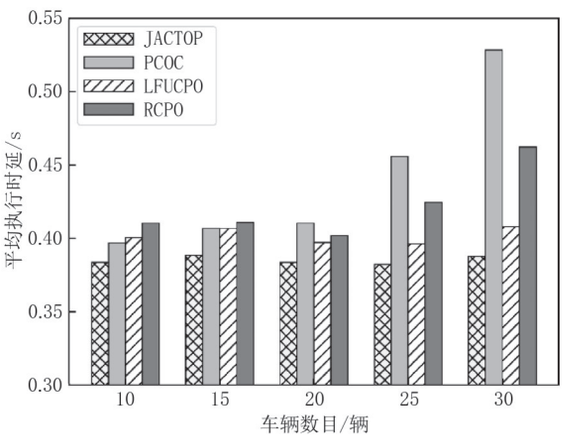

图6　不同车辆数目下任务平均执行时延的比较

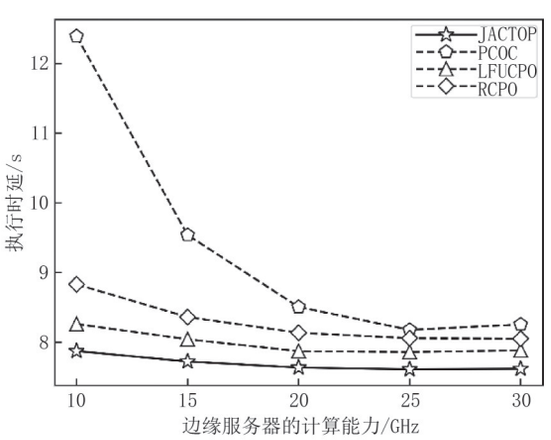

图7　任务执行时延随ES计算能力的变化

图8展示了平均缓存命中率和ES缓存能力之间的变化关系。随着ES缓存能力的提升，缓存命中率也随之增加，这表明有效的缓存能够显著提高系统效率。从图中可以看到，JACTOP算法表现最优，其能够自适应地调整缓存与卸载的比例。尽管PCOC和LFUCPO性能相近，但不能简单地认为LFUCPO策略优于PCOC，因为PCOC将所有任务都卸载到ES，而LFUCPO仅卸载部分任务。根据RCPO的表现，随机缓存策略最差。

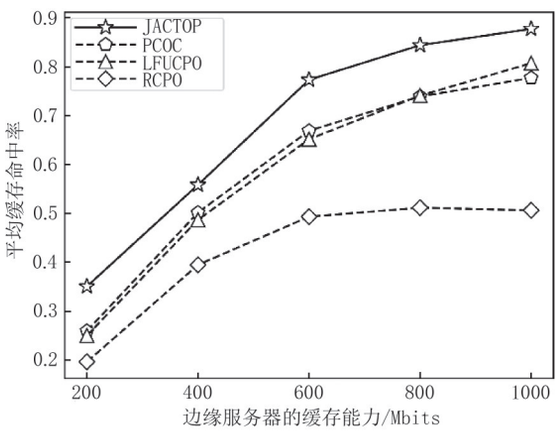

图8 平均缓存命中率和ES缓存能力的关系

图9展示了任务执行时延随ES缓存空间的变化趋势，得益于平均缓存命中率的提高，ES缓存空间的增大能够有效地减少任务的执行时延。因为较大的缓存空间能够缓存更多的应用程序，并且一个应用缓存可以服务多个请求该应用的车辆任务，因此可以极大减少应用数据的传输，避免不必要的传输时延，从而留有更多的时间去执行任务计算。JACTOP能够从历史数据中感知缓存和卸载之间的联系，通过训练优化策略梯度给出最优动作。从图中可以看出，PCOC从中受益最多，因为它将所有任务卸载到ES上执行，应用程序能够服务更多的任务，但是受ES计算资源的限制，该方法的优化空间有限。RCPO算法受限于缓存策略的随机性，导致卸载策略受到随机缓存状态的影响，难以给出良好的卸载决策，进而影响整体的表现。

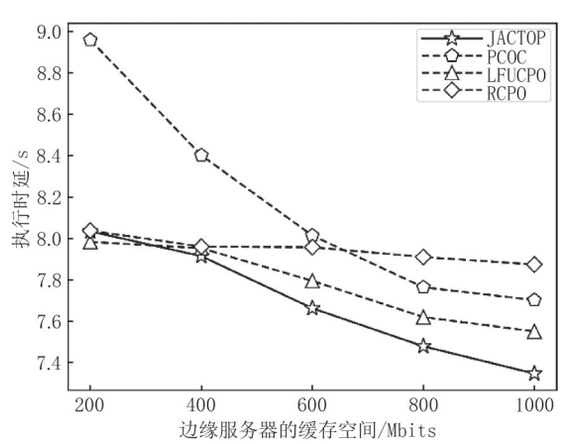

图9 任务执行时延和ES缓存空间的关系

5.3 应用前景

目前已有汽车制造商将神经网络模型应用于车载信息系统，如小鹏汽车的AI天玑系统，通过引入大型神经网络模型，实现了多场景智慧出行。部分车型配备了高通骁龙8155乃至8295车机芯片，算力已超过500 TOPS，为车辆间任务协同卸载提供了强大的硬件支撑；同时，V2X车机通信技术的快速发展，进一步为云-边-端以及车-车任务协同卸载提供了坚实的技术基础。因此，本文工作具备较为广阔的应用前景和潜在价值。

6 结束语

本文针对VEC中的应用缓存和任务卸载问题进行了深入研究。首先，考虑到车辆任务对于车载应用程序的需求，以及应用服务缓存的效益，车载应用被划分为多个应用组件，构建了协作卸载和应用缓存的边缘计算模型；然后，以最小化任务执行时延和系统能耗的加权和作为优化目标，提出了基于PPO的联合应用缓存和任务卸载的算法，并将过去多个时隙的历史环境信息作为智能体的网络输入，以感知缓存和卸载之间的联系，从而做出更优的决策；提出一种基于贪心规则的组件分配算法提升任务响应时延。最后，仿真实验结果表明，本文的JACTOP策略与其他方法相比，在目标函数优化、时延缩减和命中率提升方面具备了更大的优势。

参 考 文 献

[1] Wang F Y. Metavehicles in the metaverse: Moving to a new phase for intelligent vehicles and smart mobility. IEEE Transactions on Intelligent Vehicles, 2022, 7(1): 1-5

[2] Tang C, Chen W, Zhu C, et al. When cache meets vehicular edge computing: architecture, key issues, and challenges. IEEE Wireless Communications, 2022, 29(4): 56-62

[3] Li Y, Gao Q, Yao Z, et al. Joint optimization method of intelligent service arrangement and computing-networking resource allocation for MEC. Journal on Communications. 2023, 44(7): 51-63 (in Chinese)
(李云，高倩，姚枝秀，等. 移动边缘计算中智能服务编排和算网资源分配联合优化方法. 通信学报, 2023, 44(7): 51-63)

[4] Zhang D, Wang S, Zhang J, et al. A content distribution method of internet of vehicles based on edge cache and immune cloning strategy. Ad Hoc Networks, 2023, 138: 103012

[5] Yang K, Sun P, Yang D, et al. A novel hierarchical distributed vehicular edge computing framework for supporting intelligent driving. Ad Hoc Networks, 2024, 153: 103343

[6] Sonmez C, Tunca C, Ozgovde A, Ersoy C. Machine learning-based workload orchestrator for vehicular edge computing. IEEE Transactions on Intelligent Transportation Systems, 2021, 22(4): 2239-2251

[7] Liu Y, Yang C, Chen X, Wu F. Joint hybrid caching and replacement scheme for UAV-assisted vehicular edge computing networks. IEEE Transactions on Intelligent Vehicles, 2024, 9(1): 866-878

[8] Liu L, Chen Z. Joint optimization of multiuser computation offloading and wireless-caching resource allocation with linearly related requests in vehicular edge computing system. IEEE Internet Things Journal, 2024, 11(1): 1534-1547

[9] Tang C, Zhao Y, Wu H. Lyapunov-guided optimal service placement in vehicular edge computing. China Communications, 2023, 20(3): 201-217

[10] Tang C, Zhu C, Wu H, et al. Toward response time minimization considering energy consumption in caching-assisted vehicular edge computing. IEEE Internet of Things Journal, 2021, 9(7): 5051-5064

[11] Tang C, Wu H. Joint optimization of task caching and computation offloading in vehicular edge computing. Peer-to-Peer Networking and Applications, 2022, 15(2): 854-869

[12] Liu L, Chen C, Feng J, et al. Joint intelligent optimization of task offloading and service caching for vehicular edge computing. Journal on Communications, 2021, 42(1): 18-26 (in Chinese)
(刘雷，陈晨，冯杰，等. 车载边缘计算中任务卸载和服务缓存的联合智能优化. 通信学报，2021, 42(1): 18-26)

[13] Zeng F, Chen Q, Meng L, et al. Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing. IEEE Transactions on Intelligent Transportation Systems, 2020, 22(6): 3247-3257

[14] Raza S, Wang S, Ahmed M, et al. Task offloading and resource allocation for IoV using 5G NR-V2X communication. IEEE Internet of Things Journal, 2021, 9(13): 10397-10410

[15] Wu X, Huang X, Yuan S, et al. A resource allocation method based on weighted DRF algorithm in mobile edge computing// Proceedings of the 2021 IEEE 5th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). Xi'an, China, 2021: 86-90

[16] Zhou H, Jiang K, Liu X, et al. Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing. IEEE Internet of Things Journal, 2021, 9(2): 1517-1530

[17] Zhou H, Wu T, Zhang H, et al. Incentive-driven deep reinforcement learning for content caching and D2D offloading. IEEE Journal on Selected Areas in Communications, 2021, 39(8): 2445-2460

[18] Zhou H, Wang Z, Zheng H, et al. Cost minimization-oriented computation offloading and service caching in mobile cloud-edge computing: an A3C-based approach. IEEE Transactions on Network Science and Engineering, 2023, 10(3): 1326-1338

[19] Xue Z, Liu C, Liao C, et al. Joint service caching and computation offloading scheme based on deep reinforcement learning in vehicular edge computing systems. IEEE Transactions on Vehicular Technology, 2023, 72(5): 6709-6722

[20] Wang L, Zhang G. Joint service caching, resource allocation and computation offloading in three-tier cooperative mobile edge computing system. IEEE Transactions on Network Science and Engineering, 2023, 10(6): 3343-3353

[21] Silva E, Silva F. Deep reinforcement learning edge workload orchestrator for vehicular edge computing//Proceedings of the 2023 IEEE 9th International Conference on Network Softwarization (NetSoft). Madrid, Spain, 2023: 44-52

[22] Zhang Y, Cheng M. Joint optimization of edge computing and caching in NDN. Journal on Communications, 2022, 43 (8): 164-175 (in Chinese)
(张宇，程旻. NDN 中边缘计算与缓存的联合优化. 通信学报，2022, 43 (8): 164-175)

[23] Li Z, Yang C, Huang X, et al. CoOR: collaborative task offloading and service caching replacement for vehicular edge computing networks. IEEE Transactions on Vehicular Technology, 2023, 72(7): 9676-9681

[24] Hao Y, Chen M, Hu L, et al. Energy efficient task caching and offloading for mobile edge computing. IEEE Access, 2018, 6: 11365-11373

[25] Zhou H, Zhang Z, Li D, et al. Joint optimization of computing offloading and service caching in edge computing-based smart grid. IEEE Transactions on Cloud Computing, 2023, 11(2): 1122-1132

[26] Zhou W, Xia J, Zhou F, et al. Profit maximization for cache-enabled vehicular mobile edge computing networks. IEEE Transactions on Vehicular Technology, 2023, 72(10): 13793-13798

TANG Chao-Gang, Ph. D., lecturer. His main research interests include vehicular edge computing, smart transportation, and Internet of Things.

LI Zhao, M. S. candidate. His main research interests include vehicular edge computing and internet of things.

XIAO Shuo, Ph. D., professor. His main research interests include Internet of Things and natural language processing.

WU Hua-Ming, Ph. D., professor. His main research interests include edge computing, Internet of Things and DNA storage.

Background

Intelligent transportation has been greatly developed in recent years, and various new energy vehicle companies have sprung up, providing a strong driving force for promoting the rapid development of high-level automatic driving, human-computer interaction, and on-board applications. The vehicular services are rapidly developing and transiting from basic navigation guidance, assisted driving, vehicle monitoring, etc., to higher-level autonomous driving, human-vehicle dialogue, in-vehicle VR/AR games, and even in-vehicle meta-universe. Vehicle-perceived data include both delay-non-sensitive data related to entertainment services and delay-sensitive data related to safety warnings. The timely processing and analysis of data is not only related to the ability to provide users with low latency and high reliable Quality of Service (QoS), but also an important guarantee for driving safety.

Vehicular Edge Computing (VEC) extends the computing resources from the cloud to the edge of the network, such as Roadside Unit (RSU), provides computing power support for vehicles nearby, and implements data processing and analysis at the edge of the network, which not only reduces the pressure on the backhaul network, but also reduces the network delay. However, the following objective factors still restrict the rapid development of VEC and the improvement of the performance of VEC systems. For example, computing power and storage resources at Edge Server (ES) are much less than the remote cloud center, so more efficient resource allocation strategies and algorithms are required. The mobility of vehicles makes the topology of VANET dynamic. The real-time requirement of vehicular tasks increases the difficulty of allocating network bandwidth and computing resources at the edge. On the other hand, the development of in-car applications increases the probability of the same service being requested by different vehicular tasks. Therefore, service caching can alleviate the increasing demand for application services to some extent, reduce the pressure on the forward network, shorten the response delay of vehicular tasks, and improve the performance of VEC systems.

In this paper, based on the collaborative task offloading among vehicles, a joint optimization problem is proposed, which takes service caching, task offloading and resource allocation as decision variables, and takes the weighted sum of response latency and energy consumption for tasks as the optimization objective. In particular, the subproblem of service caching and computing offload is modeled as a Markov decision process, and a new approach based on Proximal Policy Optimization is proposed. The simulation results show that the proposed approach can achieve better performance, compared to other baseline approaches in terms of optimal values, response latency and cache hit ratio.

This work was supported in part by the National Natural Science Foundation of China under Grants 62476276 and 62271486.