Journal of System Simulation

Volume 35 | Issue 12 Article 5

12-30-2023

Task Scheduling for Internet of Vehicles Based on Deep Reinforcement Learning in Edge Computing

Xiang Ju
School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China, juxiang@sues.edu.cn

Shengchao Su
School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China, jnssc@sues.edu.cn

Chaojie Xu
School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China

Beibei He
School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China

Task Scheduling for Internet of Vehicles Based on Deep Reinforcement Learning in Edge Computing

Abstract

Abstract: Aiming at the offloading and execution of delay-constrained computing tasks for internet of vehicles in edge computing, a task scheduling method based on deep reinforcement learning is proposed. In multi-edge server scenario, a software-defined network-aided internet of vehicles task offloading system is built. On this basis, the task scheduling model of vehicle computation offloading is given. According to the characteristics of task scheduling, a scheduling method based on an improved pointer network is designed. Considering the complexity of task scheduling and computing resource allocation, the deep reinforcement learning algorithm is used to train the pointer network. The vehicle offloading tasks is scheduled by the trained pointer network. The simulation results show that with the same computing resources of edge servers, the proposed method is better than other methods in processing the number of delay-constrained computing tasks, and effectively improves the service capability of the internet of vehicles task offloading system.

Keywords

internet of vehicles, edge computing, task scheduling, pointer network, deep reinforcement learning

第 35 卷第 12 期 系统仿真学报© Vol. 35 No. 12
2023 年 12 月 Journal of System Simulation Dec. 2023

边缘计算下基于深度强化学习的车联网任务调度

琚翔，苏圣超*，徐超杰，何蓓蓓
(上海工程技术大学 电子电气工程学院，上海 201620)

摘要：针对边缘计算下车联网中时延约束型计算任务的卸载执行问题，提出一种基于深度强化学习的任务调度方法。*在多边缘服务器场景下，构建软件定义网络辅助的车联网任务卸载系统，给出车辆计算卸载的任务调度模型；根据任务调度的特点，设计一种基于改进指针网络的调度方法，综合考虑任务调度和计算资源分配的复杂性，采用深度强化学习算法对指针网络进行训练；运用训练好的指针网络对车辆卸载任务进行调度。*仿真结果表明：在边缘服务器计算资源相同的情况下，该方法在处理时延约束型计算任务的数量方面优于其他方法，有效提高了车联网任务卸载系统的服务能力。

关键词：车联网；边缘计算；任务调度；指针网络；深度强化学习
中图分类号：TP391.9 文献标志码：A 文章编号：1004-731X(2023)12-2550-10
DOI: 10.16182/j.issn1004731x.joss.22-0841
引用格式: 琚翔, 苏圣超, 徐超杰, 等. 边缘计算下基于深度强化学习的车联网任务调度[J]. 系统仿真学报, 2023, 35(12): 2550-2559.
Reference format: Ju Xiang, Su Shengchao, Xu Chaojie, et al. Task Scheduling for Internet of Vehicles Based on Deep Reinforcement Learning in Edge Computing[J]. Journal of System Simulation, 2023, 35(12): 2550-2559.

Task Scheduling for Internet of Vehicles Based on Deep Reinforcement Learning in Edge Computing

Ju Xiang, *Su Shengchao**, *Xu Chaojie*, *He Beibei*
(School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China)

Abstract: Aiming at the offloading and execution of delay-constrained computing tasks for internet of vehicles in edge computing, a task scheduling method based on deep reinforcement learning is proposed. *In multi-edge server scenario, a software-defined network-aided internet of vehicles task offloading system is built. On this basis, the task scheduling model of vehicle computation offloading is given. According to the characteristics of task scheduling, a scheduling method based on an improved pointer network is designed. Considering the complexity of task scheduling and computing resource allocation, the deep reinforcement learning algorithm is used to train the pointer network. The vehicle offloading tasks is scheduled by the trained pointer network.* The simulation results show that with the same computing resources of edge servers, the proposed method is better than other methods in processing the number of delay-constrained computing tasks, and effectively improves the service capability of the internet of vehicles task offloading system.

Keywords: internet of vehicles; edge computing; task scheduling; pointer network; deep reinforcement learning

收稿日期：2022-07-20 修回日期：2022-08-22
基金项目：国家自然科学基金(61603241)
第一作者：琚翔(1997-)，男，硕士生，研究方向为边缘计算、车联网。E-mail：juxiang@sues.edu.cn
通讯作者：苏圣超(1979-)，男，副教授，博士，研究方向为智能交通系统、物联网。E-mail：jnssc@sues.edu.cn

0 引言

车联网(internet of vehicles, IoVs)集成了通信、大数据和人工智能等技术，被认为是未来智能交通系统的重要组成部分[1-2]。当前，车联网中出现了大量的新型计算任务，如增强现实、自动驾驶等。这些计算任务要求实时性高，需要大量计算资源。因车辆受限于体积、质量等因素无法搭载功能强大的计算设备，仅凭现有车载设备的计算资源往往无法满足这些任务的计算需求[3-4]。在车辆附近部署边缘服务器，相较于云计算，边缘计算可以向用户提供近距离的计算服务。车辆将其产生的计算任务直接卸载到边缘服务器而非云端，能够减少计算任务的传输时间。因此，在IoVs中引入边缘计算可以解决车辆计算能力不足的问题，满足任务的低时延要求[5-8]。

计算卸载是边缘计算的一个技术优势，通过计算卸载可以减少计算任务的执行时延，满足用户的计算需求。Xing等[9]研究了支持设备到设备通信的多辅助移动边缘计算(mobile edge computing, MEC)系统，以减少计算任务的时延为优化目标，将问题建模为一个混合整数非线性规划的问题。考虑到任务的随机到达、服务迁移带来的额外延迟，Liu等[10]提出了一种基于参数化深度Q网络的联合服务部署和计算资源分配的新方法以最小化任务的总时间延迟。Jiang等[11]提出了一个长期MEC能源约束下的在线联合卸载与资源分配框架，通过构造能量亏损队列来引导能量消耗，从而有效使用计算资源、降低能耗。Liu等[12]为平衡系统的能耗和延迟，使用排队论对卸载过程的能耗、执行延迟和支付成本进行了深入研究。为了最小化设备的能耗和任务的执行时延，Bozorgchenani等[13]将任务卸载问题，转化为一个约束多目标优化问题，并设计了一种进化算法，求解该优化问题。由于软件定义网络(software-defined network，SDN)的控制器掌握网络的全局信息，Zhang等[14]提出了一种基于SDN的负载均衡任务卸载方案。Hou等[15]考虑到网络可能会发生故障，提出了一种可靠的任务分配和再处理机制，满足时间延迟约束下最大化计算卸载的可靠性。上述工作很好地处理了任务执行时延和能耗，但并未考虑卸载的计算任务在被调度执行前所需的等待时间。

计算任务的卸载执行可以有效降低车辆的能耗，出于延长车辆续航时间的目的，用户一般更倾向于将任务卸载执行，因此，车联网中需要卸载执行的任务数量就会进一步增加。当有大量任务卸载执行时，服务器无法同时给所有任务分配计算资源，未分配到计算资源的任务必须等待执行。此时，若等待执行的计算任务有时延要求，其等待时间不能忽视，故需要根据计算任务的执行时间与时延要求设计合理的调度策略，从而为更多卸载任务提供有效的计算服务。

考虑到SDN可以更加方便高效地管理网络资源[16]，本文将SDN集成到车联网中，在边缘计算环境下构建SDN辅助的车联网计算任务卸载系统，并给出一种车辆计算卸载的任务调度模型。然后，从任务调度和计算资源分配的复杂性方面考虑，基于深度强化学习训练了一种改进的指针网络，用于解决车联网中时延约束型计算任务在多边缘服务器中的卸载调度问题。

1 系统模型和优化问题

1.1 卸载系统架构

考虑多个车辆和多个边缘服务器的边缘环境下车联网计算任务卸载场景。定义任务超时率：卸载执行的任务中无法满足时间延迟要求的任务数占卸载任务总数的比值。本文的研究目标是尽可能为用户提供计算服务，相当于尽可能多地将计算任务卸载执行，即降低任务的超时率。假定多个边缘服务器的计算能力相同，且当服务器计算资源充足时，即计算任务无需在服务器中等待，立即能够调度执行时，任务卸载执行能够满足时

延约束。通常存储资源远比计算资源丰富，为了方便起见，采用与文献[17]类似的做法，不考虑存储资源。

本文的车联网计算任务卸载系统架构如图1所示。在一条公路旁，部署了路边单元(roadside units, RSU)、多个边缘服务器和SDN控制器。其中，RSU充当车联网的接入点，边缘服务器负责卸载任务的执行，SDN控制器负责制定任务调度方案。当车辆产生计算服务需求时，车辆会将其产生的计算任务经由RSU传输至SDN，然后由SDN控制器将该任务调度到某个服务器中进行执行。

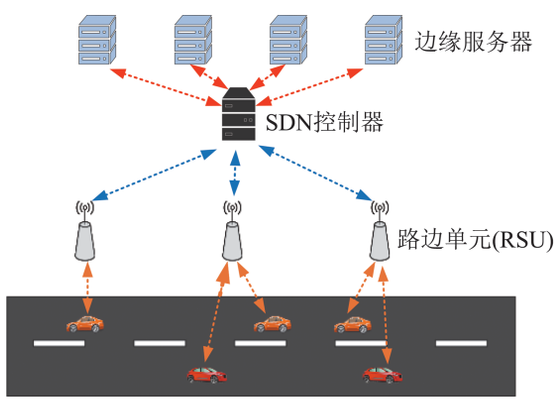

图 1 车联网计算任务卸载系统架构
Fig. 1 Architecture of internet of vehicles computing task offloading system

1.2 计算卸载模型

在图1所示的车联网任务卸载系统中，部署有P台边缘服务器，以及Q台RSU。假设某个时刻有M辆车发出了卸载请求，将车辆i，$i=1,2,\cdots,M$中产生的计算任务定义为三元组$\Psi_i=\{C_i,D_i,\tau_i\}$。其中，$C_i$表示$\Psi_i$所需的CPU cycles；$D_i$包括$D_i^{\text{in}}$和$D_i^{\text{out}}$，分别表示$\Psi_i$的输入数据量大小和任务执行完成后结果的数据量大小；τ_i表示Ψ_i的时间延迟约束，如果Ψ_i的执行总时间大于τ_i，则该任务发生超时。

当任务i卸载执行时，卸载执行总时间为数据上传时间、任务在被调度执行前所需的等待时间、任务在服务器中的执行时间、任务执行结果的返回时间的总和。任务卸载执行时，首先需要将计算任务通过无线网络上传到RSU，无线网络的上行链路传输速率R_{ij}^{up}和下行链路传输速率R_{ij}^{down}可表示为[15]

$$R_{ij}^{\text{up}}=B^{\text{up}}\ \text{lb}\left(1+\frac{p_i(s_{ij}^{-\varepsilon}|h_0|^2)}{N_0}\right) \tag{1}$$

$$R_{ij}^{\text{down}}=B^{\text{down}}\ \text{lb}\left(1+\frac{p_j(s_{ij}^{-\varepsilon}|h_0|^2)}{N_0}\right) \tag{2}$$

式中：B^{up}、B^{down}分别为分配给车辆和RSU的带宽；p_i、p_j分别为车辆i和RSU–j的发射功率；s_{ij}为车辆i和RSU–j的距离；ε为路径损耗指数；h_0为复高斯通道系数；N_0为加性高斯白噪声。那么任务i的数据传输时间为

$$t_i^{\text{up}}=D_i^{\text{in}}/R_{ij}^{\text{up}} \tag{3}$$

执行结果返回时间为

$$t_i^{\text{down}}=D_i^{\text{out}}/R_{ij}^{\text{down}} \tag{4}$$

任务i在服务器中完成执行所需时间为

$$t_i^{\text{s}}=C_i/f^{\text{s}} \tag{5}$$

式中：f^{s}为服务器的CPU频率。

任务i被调度执行前的等待时间t_i^{wait}，由SDN调度的结果确定。那么任务卸载执行的总时间为

$$t_i^{\text{off}}=t_i^{\text{up}}+t_i^{\text{down}}+t_i^{\text{s}}+t_i^{\text{wait}} \tag{6}$$

当任务卸载执行时，需要满足的时间约束为

$$t_i^{\text{off}}=t_i^{\text{up}}+t_i^{\text{down}}+t_i^{\text{s}}+t_i^{\text{wait}}\leqslant\tau_i \tag{7}$$

即，任务在服务器中的执行时间需要满足

$$t_i^{\text{s}}\leqslant\tau_i-t_i^{\text{up}}-t_i^{\text{down}}-t_i^{\text{wati}} \tag{8}$$

综合考虑任务的传输时间、结果返回时间，以及任务被调度执行前的等待时间和任务的时间延迟约束，定义剩余时间延迟约束为

$$\tau_i^{\text{r}}=\tau_i-t_i^{\text{up}}-t_i^{\text{down}}-t_i^{\text{wati}} \tag{9}$$

通过比较t_i^{s}和τ_i^{r}可以直接判断任务是否超时。

1.3 任务调度模型

为便于叙述，定义如下概念。

(1) 剩余执行时间t^{r}：当任务已经被调度到服务器中执行时，还需要t^{r}时间该任务才能完成执行。

(2) 任务执行信息 $\Psi_i=\{t_i^{\mathrm{r}},\tau_i^{\mathrm{r}}\}$：由该任务的剩余执行时间以及剩余时间延迟约束构成。

(3) 任务执行列表 execution_list：用于记录当前正在服务器中执行的所有任务的任务执行信息。

用 X 表示需要进行调度的卸载任务序列，序列中的每个任务由 $x_i=\{t_i^{\mathrm{s}},\tau_i^{\mathrm{r}}\}$ 表示。任务调度尚未开始时，所有任务的等待时间 t^{wait} 均为 0。因此，初始时所有任务的剩余时间延迟约束为 $\tau^{\mathrm{r}}=\tau-t^{\mathrm{up}}-t^{\mathrm{down}}$。

算法 1：任务调度模型

输入：timeout_queue；execution_list；Ψ 为本次调度任务的执行信息；X^t 为当前任务序列的状态

输出：更新后的 timeout_queue；更新后的 execution_list；任务序列的新状态 X^{t+1}

```
if 任务Ψ超时 then
    将该任务加入timeout_queue
else
    if 如果当前有空闲的服务器 then
        将任务Ψ调度到服务器中运行
        将Ψ加入execution_list
    else
        从execution_list中获取剩余执行时间最短的那个任务执行信息Ψ*
        等待任务Ψ*完成执行，然后将Ψ*从execution_list移除，释放占用的服务器计算资源
        将X^t中所有任务及任务Ψ的剩余时间延迟约束减去Ψ*的剩余执行时间
        将execution_list中所有任务的剩余执行时间减去Ψ*的剩余执行时间
        再次尝试调度任务Ψ，即递归调用算法1
    end if
end if
```

算法 1 模拟了 SDN 控制器对某个任务调度执行的过程。算法 1 的输入中，timeout_queue 表示超时队列，用于记录发生超时的任务。在调度第 1 个任务前，timeout_queue 和 execution_list 置为空。为便于叙述，将任务执行信息 Ψ 对应的计算任务简称为任务 Ψ。

算法 1 的执行过程：当 X 中某个任务被调度时，首先将其转换为任务执行信息的形式，即 $\Psi=\{t^{\mathrm{r}},\tau^{\mathrm{r}}\}$，初始时 $t^{\mathrm{r}}=t^{\mathrm{s}}$。通过比较 t^{r} 是否大于 τ^{r} 即可判断该任务是否超时，如果超时，则加入超时队列；如果没有超时，则进一步判断，当前是否有服务器空闲。由于服务器每次只能执行 1 个任务，因此，可以通过比较 execution_list 中的任务个数是否小于服务器总个数来判断是否有服务器空闲。如果有服务器空闲，则将该任务调度到空闲的服务器中进行执行，并将任务 Ψ 加入 execution_list 中；否则该任务需要等待，直到有服务器空闲。

可以通过 execution_list 的这些任务中最短剩余执行时间来确定至少还需要多长时间会有任务完成执行，即确定任务 Ψ 的等待时间。然后等待剩余执行时间最短的那个任务完成执行，将其从 execution_list 中移除，并释放其所占用的服务器计算资源，如算法 1 的第 8、9 行所示。此时任务 Ψ 以及那些尚未调度执行的任务进行了一定时间的等待，因此，这些任务的剩余时间延迟约束 τ^{r} 也会相应地减少，即这些任务的状态需要进行更新，如算法 1 的第 10 行所示。同时，正在运行的那些任务的剩余执行时间也应该得到更新，如算法 1 的第 11 行所示。这时，可以再次尝试调度任务 Ψ，需要再次判断该任务是否超时，可以通过递归调用算法 1 来实现。

当所有任务都完成调度后，统计出超时队列中的任务数 n，以及输入序列 X 中的任务数 m，计算任务的超时率为

$$\delta=n/m \tag{10}$$

本文的研究目标是在计算资源有限的情况下，尽可能多地使时延约束型计算任务得到有效执行，

DOI: 10.16182/j.issn1004731x.joss.22-0841

即任务调度算法实现的调度结果应使任务的超时率尽可能小。因此，将任务调度算法的优化目标定义为

$$\min: \text{reward}(Y|X)=\delta=n/m \tag{11}$$

2 任务调度算法

2.1 算法模型设计

本文任务调度问题的输入是一个任务序列，输出是输入序列的一个组合。因此，可以采用序列到序列模型(Seq2Seq)来解决该调度问题。传统的Seq2Seq模型无法解决输出序列会随着输入序列长度的改变而改变的问题，不满足调度问题的特点。Vinyals等[18]在Seq2Seq模型基础上，提出了Pointer Network模型。该模型使用注意力机制(attention mechanism)作为指针来选择输入序列的一个成员作为输出，即输出的序列长度由源序列决定，适用于解决任务调度问题。但Pointer Network结构中的编码器部分与Seq2Seq模型一样，仍然是一个循环神经网络(RNN)。考虑到待调度的任务序列应为无序集合，而非有序序列，因此，待调度任务之间的顺序关系不应作为一个特征。此外，RNN在提取输入序列特征时的计算过程是顺序执行，计算时间较并行执行长。本文将Pointer Network结构中的编码器部分RNN替换为一维卷积神经网络。相比于RNN，卷积神经网络在提取特征时不会提取输入任务之间的顺序特征，同时，该计算过程是并行的，可以缩短计算时间。由于某个任务被调度时，会影响到后续的调度结果，所以在解码器部分保留RNN。

改进的指针网络模型如图2所示，其中，虚线框中的内容为编码器，余下部分除去environment后的为解码器。解码器每解码一次，指针网络就可以得到一个输出，该输出指向输入任务序列中的某个任务，表示该任务将被调度执行。图2中编码器的输入为任务序列X的最新状态。解码器的输入为上个时刻指针网络的输出所指向的任务，即上一时刻被调度的任务。

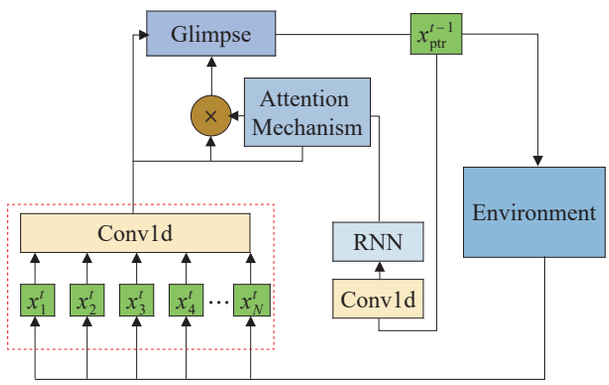

图2 指针网络

Fig. 2 Pointer network

设在t时刻，输出序列为$Y^t=\{y^i|i=1,2,\cdots,t\}$，其中，$y^i$的值为一个指针，指向输入的任务序列$X$中的某个任务。由于指针网络每次产生一个输出，因此，T在数值上等于输入序列的长度。由概率的链式法则，在给定序列X的条件下，输出结果为Y的概率为

$$P(Y|X)=P(y^1|X)P(y^2|y^1,X)\cdots P(y^T|Y^{T-1},X)=\prod_{t=1}^{T}P(y^t|Y^{t-1},X) \tag{12}$$

设z_i^t为第t个解码步骤中编码器的第i个输入，h_t为第t个解码步骤RNN的输出。与文献[19]类似，本文使用注意力机制和glimpse机制，从输入序列中提取相关信息。注意力值a^t的计算过程为

$$u_i^t=\boldsymbol{v}_1^{\mathrm{T}}\tanh(W_1[z_i^t;h_t]) \tag{13}$$

式中：z_i^t和h_t分别为注意力机制中的key和query；$[z_i^t;h_t]$为将key和query进行拼接；$\boldsymbol{v}_1$和W_1为可训练参数。

$$d^t=\text{softmax}(u^t) \tag{14}$$

$$a^t=\sum_{i=1}^{T}d_i^t z_i^t \tag{15}$$

glimpse机制由式(16)和(17)所示，将z_i^t和a^t经过一个类注意力机制的计算过程，最终可以得到一个概率分布。最后从这个概率分布中进行采样，得到指针网络的输出。由于一个计算任务只需被调度执行一次，因此，为了避免某个任务被重复

调度，在glimpse机制中添加一个屏蔽方法。该方法将已经调度过的任务所对应的计算值设置为−∞，以屏蔽那些已经被调度的任务，如式(16)所示。经过softmax函数后，该任务对应的概率值就会变为0，从而不会被采样。

$$c_i^t=\begin{cases}\boldsymbol{v}_2^{\mathrm{T}}\tanh(W_2[z_i^t;a^t]), & i\notin Y^t\\ -\infty, & i\in Y^t\end{cases} \tag{16}$$

式中：$\boldsymbol{v}_2$和W_2为可训练参数。

$$P(y^{t+1}|Y^t,X^t)=\mathrm{softmax}(c_i^t) \tag{17}$$

2.2 算法模型训练

边缘服务器的计算资源是有限的，无法为大量任务同时提供并行的计算服务，因此，对于计算任务数为N的调度问题，有近$N!$种调度方案。在这近$N!$种调度方案中寻找最优解，显然是一个NP-hard问题。目前，对于NP-hard问题的求解大多采用启发式算法，但启发式算法容易陷入局部最优解，而且算法迭代时间长，实时性差，难以满足计算任务的低时延需求。由于计算任务的到来是随机的，即边缘环境下的任务调度问题是动态的，常规算法难以对该问题进行求解。深度强化学习融合了深度神经网络的强大感知能力和强化学习的顺序决策能力，可以通过与动态环境的交互获得最优策略，而不需要动态环境的先验知识，能够很好地应对环境的变化。因而，本文采用深度强化学习的方式对问题进行求解。

本文采用actor-critic强化学习算法训练指针网络，其中，actor为指针网络，用于产生动作，该动作即为将要调度执行的计算任务。然后将该任务作为输入与环境进行互动，得到新的任务序列状态。调度任务与环境互动的过程如算法1所示。critic网络模型如图3所示，用于估计给定输入序列情况下的回报。critic网络包含4个卷积层和2个ReLU层。第1个卷积层的作用是提取输入序列的特征，中间2个卷积层作用是进行维度转换，最后1个卷积层可以将网络得到的中间结果映射为一维的估计奖励值。

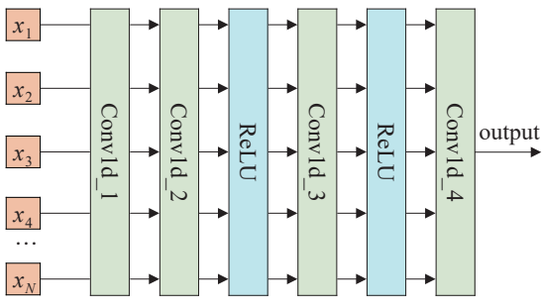

图 3 Critic网络
Fig. 3 Critic network

reward$(Y|X)$表示已知服务请求集合X的情况下，采取策略Y时的奖励值，由式(11)表示。设指针网络的参数为θ，结合式(12)，reward$(Y|X)$的期望定义如下：

$$J(\theta|X)=E_{Y\sim P_\theta(Y|X)}\mathrm{reward}(Y|X) \tag{18}$$

指针网络的优化目标为最小化reward的期望，即最小化$J(\theta|X)$。采用文献[20]提出的Reinforce算法对指针网络参数θ进行训练：

$$\nabla_\theta J(\theta|X)=E_{Y\sim P_\theta(Y|X)}$$
$$[(\mathrm{reward}(Y|X)-V(X;\theta))\nabla_\theta \ln P_\theta(Y|X)] \tag{19}$$

式中：$V(X;\theta)$为基线函数，由critic网络确定。$V(X;\theta)$只取决于问题的输入，与所采取的策略无关，用于评价当前策略的好坏。

将训练数据的batch size大小设置为B，式(20)可以近似等于

$$\nabla_\theta J(\theta|X)\approx\frac{1}{B}\sum_{i=1}^{B}[(\mathrm{reward}(Y^i|X^i)-V(X^i;\theta))\nabla_\theta \ln P_\theta(Y^i|X^i)] \tag{20}$$

critic网络的参数更新方式为

$$\mathrm{d}\theta_v\leftarrow\frac{1}{B}\sum_{i=1}^{B}\nabla_{\theta_v}(\mathrm{reward}(Y^i|X^i)-V(X^i;\theta_v))^2 \tag{21}$$

式中：reward$(Y^i|X^i)$为实际的奖励值；$V(X^i;\theta_v)$为模型的预测值。

模型的训练过程如算法2所示。带有下标i的标识符，表示与此batch中第i个样本相关的数据信息。带有上标t的标识符，表示在解码步骤t时的相关数据信息。T表示解码的步骤数，在数值上等于输入的任务序列长度。

DOI: 10.16182/j.issn1004731x.joss.22-0841

算法2：模型训练

1 以随机权重θ和θ_v初始化 actor 和 critic 网络

2 for step = 1, 2, ⋯, do

3 重置梯度：$d\theta \leftarrow 0, d\theta_v \leftarrow 0$

4 从数据集Ω中抽样B个数据

5 for i= 1, 2, ⋯, B do

6 $t \leftarrow 0$

7 Repeat

8 从概率分布$P(y_i^{t+1}|Y_i^t, X_i^t)$中采样$y_i^{t+1}$

9 调用算法1更新任务序列的状态，得到X_i^{t+1}

10 $t \leftarrow t+1$

11 Until $t == T$

12 根据式(11)，计算奖励值$r_i = r(Y_i, X_i^0)$

13 End for

14 根据式(20)和(21)更新权重θ和θ_v

15 End for

3 仿真实验及分析

3.1 仿真实验的参数设置

仿真实验场景设置：在一条长1 km的道路上，每200 m部署一个RSU，车辆随机分布在该道路上。参照文献[14-15]的参数设置，表1给出了仿真实验中设定的默认参数值，并生成仿真所需的数据集。仿真工具为PyTorch深度学习框架，版本为PyTorch=1.10.2，Python=3.9.10。将计算任务的执行时间、计算任务的数据量、计算结果的数据量，以及计算任务的时间延迟约束这4类数据作为任务的特征，将卷积神经网络的输入维度设置为4。对于网络的其他参数以及模型的训练参数的设置，通过设置不同参数对模型进行训练及测试，最终确定设置如表1所示参数时算法所实现的调度效果最好。指针网络模型中，编码器和解码器的卷积层的输入和输出通道均设为(4, 128)，卷积核设为1。解码器的RNN部分使用GRU，其输入数据特征数和隐藏层的神经元数均设置为128。critic网络中，4个卷积层的输入和输出通道数分别设为(4, 128)、(128, 20)、(20, 20)、(20, 1)，卷积核均设为1。Batch size设为16，学习率设为0.000 1，使用Adam优化器优化网络参数。

表1 默认参数设置

Table 1 Default parameters setting

参数	描述	取值
D_i^{in}/KB	Ψ_i的数据量	400~1 000
D_i^{out}/KB	Ψ_i执行结果的数据量	50~200
C_i/MHz	Ψ_i执行所需的CPU cycles	200~1 500
τ/s	Ψ_i时间延迟约束	0.5~1.0
f^s/GHz	服务器默认CPU频率	4
N	服务器默认个数	4
B^{up}/MHz	车辆的带宽	10~20
B^{down}/MHz	RSU的带宽	30~40
ε	路径损失指数	4
p_i/mW	车辆i发送功率	100
p_j/mW	RSU-j发送功率	100
h_0	复高斯通道系数	$h_0 \sim CN(0, 1)$
N_0/dBm	加性高斯白噪声	100

3.2 仿真实验的结果分析

考虑模型训练时，输入的任务序列长度可能会影响模型的性能，故本文将采用不同长度的任务序列对模型进行训练。将序列长度分别设置为10、15、20、25和30对模型进行训练，以评估输入序列长度的变化对模型训练效果的影响。对于不同的计算任务数，测试时均通过计算一个batch的数据，将所得的任务平均超时率作为测试结果，仿真结果如图4所示。

从图4可以看出，由不同的输入序列长度训练所得的模型的优化效果整体上较为一致。当任务数量较少时，由较短的输入序列训练所得模型的优化效果较好。但是随着任务数的增加，这些模型的优化效果逐渐变差。此时，由较长的输入序列训练所得模型的优化效果较好。这是因为，输入序列较短时，数据量较小，模型训练时所能

提取的特征信息有限，所以，对较长的任务序列的优化效果变得较差，而由较长序列训练所得模型观察到的数据量大，模型能够学习到长序列的特征，因而，能够更好地调度较长的任务序列。但此时由于模型注重于长序列的特征学习，故对于较短序列的优化效果较差，因此，可以根据车联网中的任务数量合理地选择序列长度对模型进行训练以得到优化效果最佳的模型。此外，从图4还可以看出，序列长度为20时训练所得模型的平均优化效果最好。

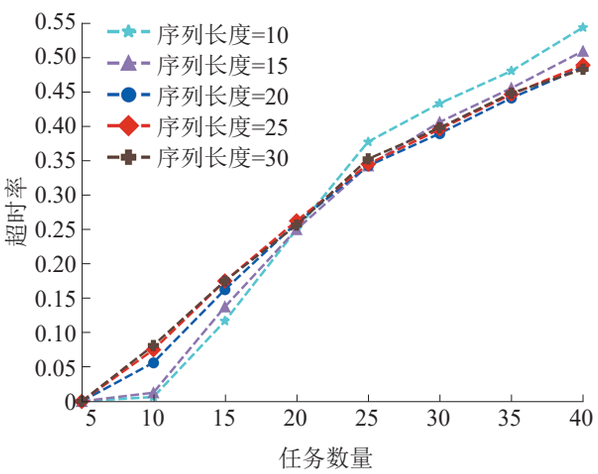

图 4　不同模型的优化结果
Fig. 4　Optimization results for different models

为验证模型的性能，将先到先服务调度算法(first come first service, FCFS)、短作业优先调度算法(shortest job first, SJF)和改进前的指针网络(pointer network, PN)作为基线算法，将本文模型的优化性能与3种基线算法进行对比实验。FCFS按照计算任务卸载到服务器的先后顺序对计算任务进行调度。SJF优先调度尚未执行的计算任务集合中执行时间最小的那个计算任务。对比实验中所采用的模型均由输入序列长度为20，其他参数默认时训练得到的。图5为不同任务数的情况下，几种算法所表现出的性能。从图5可以看出，所提算法的优化效果优于改进前的指针网络。从总体表现来看，相较于FCFS和SJF，所提算法能大幅度降低任务的超时率。

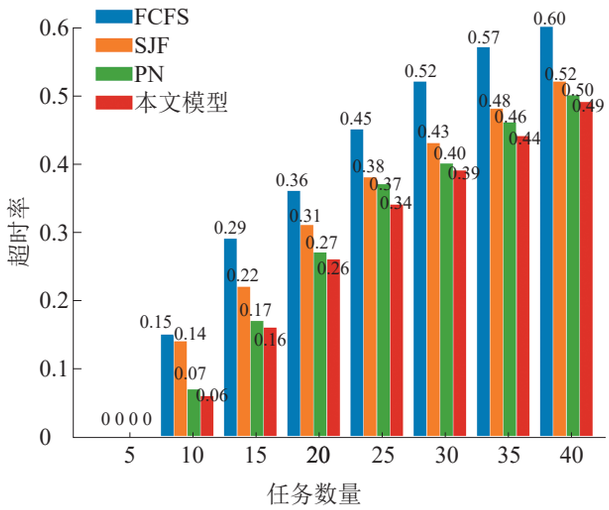

图 5　任务数量与超时率的关系
Fig. 5　Relationship between number of tasks and timeout ratio

为验证算法的泛化性能，在任务数量分别为20和40时进行了仿真实验。

图6~7展示了边缘服务器的个数为4，服务器的CPU频率从2~6 GHz变化时，不同算法对任务超时率的优化效果。从图5可以看出，当服务器的CPU频率增加时，任务超时率均有所降低。这是因为当服务器的性能提升时，每个任务执行所需的时间会减少，同时，任务在服务器中的等待时间也会进一步减少，所以任务超时率会进一步下降。

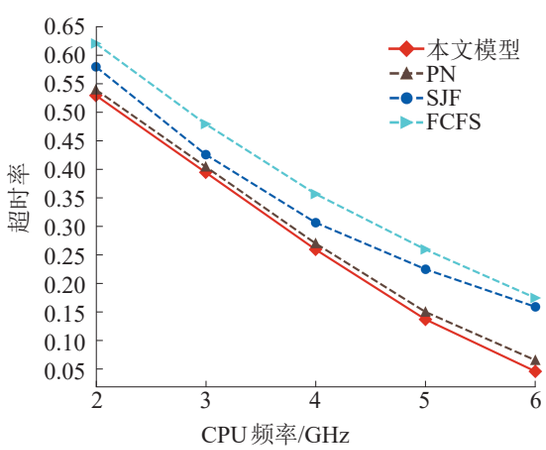

图 6　任务数为20时，CPU频率与超时率的关系
Fig. 6　Relationship between CPU frequency and timeout ratio when number of tasks is 20

DOI: 10.16182/j.issn1004731x.joss.22-0841

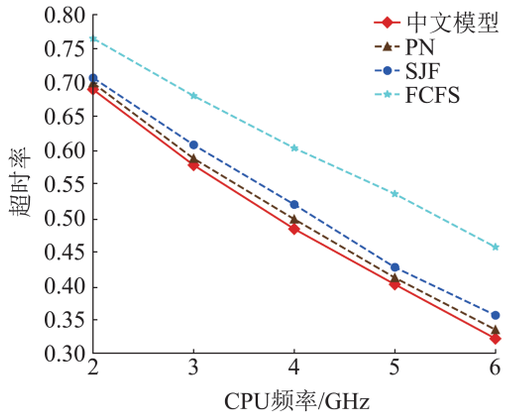

图 7　任务数为40时，CPU 频率与超时率的关系
Fig. 7　Relationship between CPU frequency and timeout ratio when number of tasks is 40

图 8 和图 9 展示了边缘服务器的 CPU 频率为 4 GHz，服务器数量从 1 增加到 6 时，不同算法对任务超时率的优化效果。从图中可以看出，服务器的数量增加时，任务超时率下降趋势较为明显。这是因为，当服务器的数量增加时，计算任务的并行数增加了，从而大大降低了那些未调度任务的等待时间。结合图 6~9 可以看出，当计算资源充足时，SJF 的优化效果有所减缓。这是因为，本文调度任务是有时间延迟约束的，当计算资源充足时，任务执行时间这一特征，对调度结果的影响减弱。仿真实验结果表明，本文所提方法能够适应于各种场景，且相较于其他算法优化效果最优。

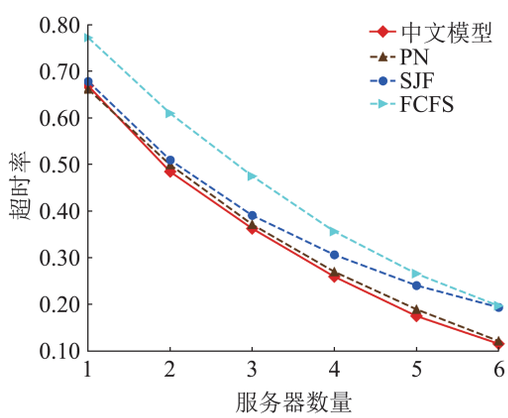

图 8　任务数为20时，服务器数量与超时率的关系
Fig. 8　Relationship between number of servers and timeout ratio when number of tasks is 20

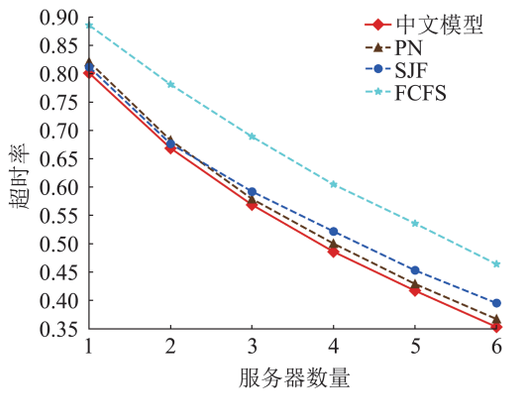

图 9　任务数为40时，服务器数量与超时率的关系
Fig. 9　Relationship between number of servers and timeout ratio when number of tasks is 40

4　结论

本文采用深度强化学习训练了一种指针网络，用于解决多边缘服务器场景下车联网中车辆产生的时延约束型计算任务的调度问题，主要贡献如下：本文提出了一种车联网任务卸载系统框架，研究了边缘服务器计算资源有限情况下的时延约束型计算任务的调度问题，同时考虑了计算任务在被调度执行前所需的等待时间对任务调度的影响。为使服务器尽可能多为计算任务提供计算服务，以减少超时的任务数为目标，提出了一种基于深度强化学习的任务调度方法。仿真结果表明，本文所提方法能够有效降低计算任务的超时率，且在调度效果方面明显优于其他算法。但本文研究的计算任务调度场景中，车辆的通信环境较为理想且计算资源的种类单一。下一步将研究在计算资源异构，车辆通信信道资源有限且信道存在干扰的计算卸载场景下的计算任务调度策略。

参考文献：

[1] Zhai Yanlong, Sun Wenxin, Wu Jianqing, et al. An Energy Aware Offloading Scheme for Interdependent Applications in Software-defined IoV with Fog Computing Architecture[J]. IEEE Transactions on Intelligent Transportation Systems, 2021, 22(6): 3813-3823.

[2] Wang Jingjing, Jiang Chunxiao, Zhang Kai, et al. Vehicular Sensing Networks in a Smart City: Principles, Technologies and Applications[J]. IEEE Wireless

Communications, 2018, 25(1): 122-132.

[3] Cao Xiaowen, Wang Feng, Xu Jie, et al. Joint Computation and Communication Cooperation for Energy-efficient Mobile Edge Computing[J]. IEEE Internet of Things Journal, 2019, 6(3): 4188-4200.

[4] Zhu Tongxin, Shi Tuo, Li Jianzhong, et al. Task Scheduling in Deadline-aware Mobile Edge Computing Systems[J]. IEEE Internet of Things Journal, 2019, 6(3): 4854-4866.

[5] 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[J]. 通信学报, 2021, 42(1): 18-26.
Liu Lei, Chen Chen, Feng Jie, et al. Joint Intelligent Optimization of Task Offloading and Service Caching for Vehicular Edge Computing[J]. Journal on Communications, 2021, 42(1): 18-26.

[6] Li Qiuping, Zhao Junhui, Gong Yi, et al. Energy-efficient Computation Offloading and Resource Allocation in Fog Computing for Internet of Everything[J]. China Communications, 2019, 16(3): 32-41.

[7] 谢人超, 廉晓飞, 贾庆民, 等. 移动边缘计算卸载技术综述[J]. 通信学报, 2018, 39(11): 138-155.
Xie Renchao, Lian Xiaofei, Jia Qingmin, et al. Survey on Computation Offloading in Mobile Edge Computing[J]. Journal on Communications, 2018, 39(11): 138-155.

[8] Feng Jie, F Richard Yu, Pei Qingqi, et al. Cooperative Computation Offloading and Resource Allocation for Blockchain-enabled Mobile-edge Computing: A Deep Reinforcement Learning Approach[J]. IEEE Internet of Things Journal, 2020, 7(7): 6214-6228.

[9] Xing Hong, Liu Liang, Xu Jie, et al. Joint Task Assignment and Resource Allocation for D2D-enabled Mobile-edge Computing[J]. IEEE Transactions on Communications, 2019, 67(6): 4193-4207.

[10] Liu Tong, Ni Shenggang, Li Xiaoqiang, et al. Deep Reinforcement Learning Based Approach for Online Service Placement and Computation Resource Allocation in Edge Computing[J]. IEEE Transactions on Mobile Computing, 2023, 22(7): 3870-3881.

[11] Jiang Hongbo, Dai Xingxia, Xiao Zhu, et al. Joint Task Offloading and Resource Allocation for Energy-constrained Mobile Edge Computing[J]. IEEE Transactions on Mobile Computing, 2023, 22(7): 4000-4015.

[12] Liu Liqing, Chang Zheng, Guo Xijuan, et al. Multiobjective Optimization for Computation Offloading in Fog Computing[J]. IEEE Internet of Things Journal, 2018, 5(1): 283-294.

[13] Bozorgchenani A, Mashhadi F, Tarchi D, et al. Multi-objective Computation Sharing in Energy and Delay Constrained Mobile Edge Computing Environments[J]. IEEE Transactions on Mobile Computing, 2021, 20(10): 2992-3005.

[14] Zhang Jie, Guo Hongzhi, Liu Jiajia, et al. Task Offloading in Vehicular Edge Computing Networks: A Load-balancing Solution[J]. IEEE Transactions on Vehicular Technology, 2020, 69(2): 2092-2104.

[15] Hou Xiangwang, Ren Zhiyuan, Wang Jingjing, et al. Reliable Computation Offloading for Edge-computing-enabled Software-defined IoV[J]. IEEE Internet of Things Journal, 2020, 7(8): 7097-7111.

[16] Kaur K, Garg S, Aujla G S, et al. Edge Computing in the Industrial Internet of Things Environment: Software-defined-networks-based Edge-cloud Interplay[J]. IEEE Communications Magazine, 2018, 56(2): 44-51.

[17] Ma Yu, Liang Weifa, Li Jing, et al. Mobility-aware and Delay-sensitive Service Provisioning in Mobile Edge-cloud Networks[J]. IEEE Transactions on Mobile Computing, 2022, 21(1): 196-210.

[18] Vinyals O, Fortunato M, Jaitly N. Pointer Networks[C]// Proceedings of the 28th International Conference on Neural Information Processing Systems. Cambridge, MA, USA: MIT Press, 2015: 2692-2700.

[19] Bello I, Pham H, Le Q V, et al. Neural Combinatorial Optimization with Reinforcement Learning[EB/OL]. (2017-01-12) [2021-12-10]. https://arxiv. org/abs/1611.09940.

[20] Williams R J. Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning[J]. Machine Learning, 1992, 8(3): 229-256.

DOI: 10.16182/j.issn1004731x.joss.22-0841