文章编号:1674-2974(2024)06-0073-13 DOI:10.16339/j.cnki.hdxbzkb.2024268

D3DQN-CAA:一种基于DRL的自适应边缘计算任务调度方法

巨涛†,王志强,刘帅,火久元,李启南
(兰州交通大学 电子与信息工程学院,甘肃 兰州 730070)

摘 要:为解决已有基于深度强化学习的边缘计算任务调度面临的动作空间探索度固定不变、样本效率低、内存需求量大、稳定性差等问题,更好地在计算资源相对有限的边缘计算系统中进行有效的任务调度,在改进深度强化学习模型D3DQN(Dueling Double DQN)的基础上,提出了自适应边缘计算任务调度方法D3DQN-CAA.在任务卸载决策时,将任务与处理器的对应关系看作一个多维背包问题,根据当前调度任务与计算节点的状态信息,为任务选择与其匹配度最高的计算节点进行任务处理;为提高评估网络的参数更新效率,降低过估计的影响,提出一种综合性Q值计算方法;为进一步加快神经网络的收敛速度,提出了一种自适应动作空间动态探索度调整策略;为减少系统所需的存储资源,提高样本效率,提出一种自适应轻量优先级回放机制.实验结果表明,和多种基准算法相比,D3DQN-CAA方法能够有效地降低深度强化学习网络的训练步数,能充分利用边缘计算资源提升任务处理的实时性,降低系统能耗.

关键词:边缘计算;任务调度;深度Q学习;深度强化学习
中图分类号:TN391 **文献标志码**:A

D3DQN-CAA:a DRL-based Adaptive Edge Computing Task Scheduling Method

JU Tao†,WANG Zhiqiang,LIU Shuai,HUO Jiuyuan,LI Qinan
(School of Electronics and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China)

Abstract: To solve the problems faced by the existing edge computing task scheduling based on deep reinforcement learning, such as fixed action space exploration, low sample efficiency, large memory demand and poor stability and to better carry out effective task scheduling in the edge computing system with relatively limited computing resources, an adaptive edge computing task scheduling method D3DQN-CAA is proposed based on the improved deep reinforcement learning model D3DQN (Dueling Double DQN). In the task offloading decision, the corresponding relationship between the task and processor is regarded as a multidimensional knapsack problem, and the computing node with the highest matching degree is selected for task processing according to the state information

* **收稿日期**:2023-01-16
基金项目:国家自然科学基金资助项目(61862037,62262038),National Natural Science Foundation of China (61862037,62262038);甘肃省科技计划项目(23CXGA0028),Science and Technology Project of Gansu Province(23CXGA0028);甘肃省自然科学基金资助项目(22JR5RA356),Natural Science Foundation of Gansu Province(22JR5RA356).
作者简介:巨涛(1980—),男,甘肃永登人,兰州交通大学,副教授
† 通信联系人,E-mail:jutao@mail.lzjtu.cn

of the current scheduled task and the computing node; For improving the parameters updating efficiency of the evaluation network and reducing the influence of overestimation, a comprehensive Q-value calculation method is proposed; For accelerating the convergence speed of neural networks, an adaptive dynamic exploration degree of action space adjustment strategy is proposed; For reducing the storage resources required and improving the sample efficiency, an adaptive lightweight prioritized playback mechanism is proposed. Experimental results show that compared with multiple benchmark algorithms, the D3DQN-CAA algorithm can effectively reduce the number of training steps of deep reinforcement learning networks and make full use of edge computing resources to improve the real-time performance of task processing and reduce the system energy consumption.

Key words: edge computing; task scheduling; deep Q-learning; deep reinforcement learning

近年来,随着智能终端设备和各种物联网设备数目的激增,数据量呈现指数级的增长.基于云计算技术的集中数据处理方式,已经不能满足时延敏感型任务的计算需求[1-2].相较于集中式数据处理,边缘计算模式通过将计算节点部署在靠近用户的网络边缘,传输时延将大幅度减小且可以提供远高于终端设备的计算资源[2].边缘计算应用环境下的计算任务调度问题是边缘计算领域首要解决的问题.近年来出现了大量边缘计算环境下任务调度相关的研究工作和具体的解决方案.其中,机器学习算法由于其在解决复杂问题上的优越性,吸引了大批的研究者将机器学习算法与边缘计算进行结合.一方面,边缘计算下庞大的数据量,促进了机器学习算法的发展;另一方面,机器学习算法为边缘计算下的各种问题提供了强有力的解决方法,二者相辅相成[3-4].

在机器学习算法中,深度学习具有很强的表达能力,强化学习则具有很强的环境感知能力.在边缘环境下解决任务调度问题时,面对复杂的动态变化的环境,需要同时具备以上两种能力.而深度强化学习算法则同时具有很强的表达能力和很强的环境感知能力,所以深度强化学习可以有效地处理边缘计算环境下的计算任务调度问题.

DQN(Deep Q-learning)算法是解决复杂问题中应用最广泛的深度强化学习算法之一,文献[5-11]均基于DQN进行算法设计.文献[5]为解决移动边缘计算下的任务调度问题,将DQN算法中的经验回放机制替换为事后经验回放,随机抽取当前迭代后的若干个状态为回放样本进行回放学习,同时为能够更加准确地对当前环境状态进行评估,将LSTM(Long Short-Term Memory)与DQN结合,对当前的环境状态进行评估,并以传统的贪婪策略给出相应的卸载决策.文献[6]为解决在复杂环境下多机器人任务调度中出现的维度灾难问题,以神经网络代替传统强化学习中的Q值;又尝试在动作选择策略中加入探索因子以更好地进行动作空间的探索,但人为设置探索因子需要大量的实验,开销较大.文献[7]注意到原始DQN的"过估计"问题,利用Double DQN算法解决移动边缘网络下多终端多边缘节点多资源分配问题,以最小化任务的平均能耗.文献[8]致力于优化任务完成时间与能耗的加权和,以获得最大的奖励值,和传统深度Q学习不同之处是采用神经网络来存储Q值而不是Q表,以达到节省内存空间的目的,相较于Local和Random算法能够取得最优的处理性能.文献[9]为了解决边缘节点繁忙导致的长处理时延和任务被频繁丢弃问题,将LSTM与D3DQN(Dueling Double Dqn)组合,利用这一网络结构保存历史负载信息,协助调度算法做出更加准确的决策,但在动作选择策略和经验回放机制上仍为传统策略或机制,动作空间的探索效率和样本效率较低.文献[10]为解决工业作业流程中多线程的工作流调度问题,在D3DQN的基础上提出一种经验优先回放机制,从环境中获取的样本经过神经网络之后将其存储在经验池当中,再以TD(Temporal-Difference)误差作为优先级评判标准,优先选取对神经网络参数更新影响最大即误差值最大的经验样本进行优先经验回放,但这容易使神经网络远离最优解.文献[11]为了更有效地分配边缘计算系统中有限的计算机资源,在Double DQN算法的基础上,将

优化目标改为最小化长期平均完成时间和资源需求量的加权和,算法中的经验回放机制被替换为多缓存池的回放机制以便分别对不同计算资源进行记录,但调度算法中的动作选择策略与回放机制仍使用传统策略,效率较低.又因任务调度问题的动作、状态空间是连续的,而DQN适合处理离散空间问题.若使用DQN处理连续空间问题时不进行动作、状态空间的离散化,则会造成准确度的损失.因此,后续出现了以DQN为基础,但适用于连续动作和状态空间的算法,例如AC(Actor-Critic)、DDPG(Deep Deterministic Policy Gradient)、SARSA(State Action Reward State Action)等.文献[12-13]提出利用一种多线程分布式Actor-Critic算法结构A3C(Asynchronous Advantage Actor-Critic),解决边缘环境下的任务调度问题.文献[14]提出一种分布式的DDPG算法,使得临近的边缘服务器可以进行合作处理任务,同时对重复的任务进行存储以减少相应的传输时延,获得最低的长期加权开销;同时文献[15-16]分别在解决车联网中的任务调度问题和移动边缘计算下多用户任务调度问题上,均以DDPG为基础算法框架,并为动作值添加噪声,使动作值在均值周围一定距离内浮动以增加动作空间的探索度,但在任务调度问题中动作均值难以衡量.文献[17]为提高无人机避障和目标追踪算法的泛化性和抗干扰性,将DDPG与元学习算法结合设计了一个元强化学习算法.文献[18]将DDPG与记忆网络结合,设计了一个基于策略记忆的内容推荐算法,使得算法具备历史经验的记忆能力,从而提高算法对用户喜好的记忆能力,提升推荐的准确性.

以上工作中,待处理任务常被视为一个不可分割的整体,同时在进行任务卸载决策时,基本上采用贪婪策略或者相似策略,不利于计算资源的充分利用;在进行经验回放时,抽样策略基本上为均匀抽样(随机抽样),样本效率较低,在参数复制之后的一段时间内,神经网络会出现波动,稳定性较差.另外,上述采用了各种适用于连续动作、状态空间调度算法的工作中,无论采用何种算法,在神经网络的训练过程中,进行神经网络参数更新时至少需要更新两个神经网络,另外整个算法框架下包含较多的相同结构的神经网络数目.在有限计算资源的边缘服务器上,同时更新多个神经网络和存储多个神经网络,会产生较高的系统计算资源开销.同时为了增加动作空间的探索度,上述工作中多采用添加噪声,例如高斯噪声、高斯白噪声、Ornstein-Uhlenbeck噪声等,但动作值均值难以衡量,无法保证经过噪声后动作值的有效性.

本文在分析上述工作中存在的动作空间探索度固定不变、样本效率低、内存需求量大、稳定性差等问题的基础上,通过改进Dueling Double DQN提出了边缘计算任务调度方法D3DQN-CAA.为了充分利用系统中的计算资源,将一个任务表示成由多个子任务构成的有向无环图,子任务的处理顺序由它们之间的依赖关系决定;在任务卸载决策上,将任务与处理器的对应关系看作一个多维背包问题,调度决策代理根据当前调度任务与处理器的状态信息,为任务选择与其匹配度最高的处理器进行任务处理;由目标网络的输出和评估网络的输出共同决定最终神经网络的输出,以进一步降低“过估计”的影响,提高算法的稳定性、减小参数复制后的波动幅度;以损失值为基础,动态调整任务卸载时的空间探索度,同时对学习经验进行优先级排序,将中间部分的经验存储在经验池中进行回放,以提高样本效率、降低内存需求.

1　系统模型

在本文的边缘计算系统中,存在一组边缘计算节点、一组终端设备和一个决策代理.代理通过无线网络收集终端设备提交的任务信息和计算节点信息并进行任务卸载决策.若卸载任务,则将任务数据上传至边缘计算节点进行处理,并将处理结果返回至终端设备;若本地处理则,在终端设备上处理任务.具体边缘计算系统框架如图1所示.

1.1　任务模型

为了充分利用系统中的计算资源,将一个任务表示成由多个子任务构成的有向无环图DAG,如图2所示(图中为一个由10个子任务组成的计算任务).

各子任务被处理之前,必须确保不存在未被处理的前趋子任务.每个子任务都包含相同属性:所需计算资源、内存大小、存储空间大小等.例如第i个任务的第j个子任务可表示为:

$$\mathrm{task}_{ij}=\{\mathrm{cpu},\mathrm{memory},\mathrm{disk},\cdots\}$$

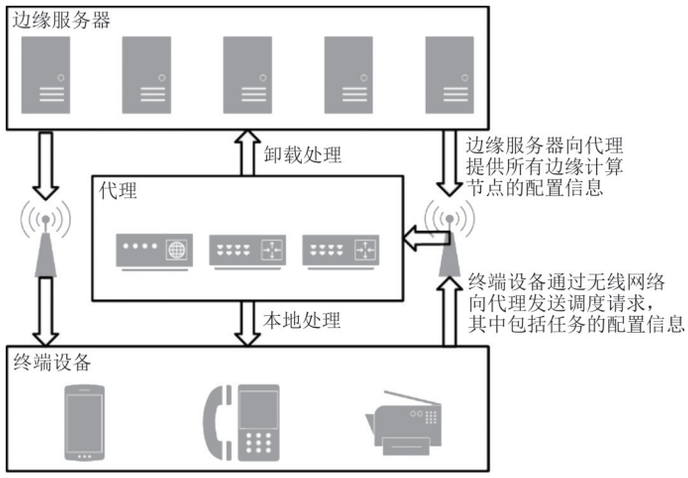


图1 边缘计算系统框架

Fig.1 The framework of edge computing system

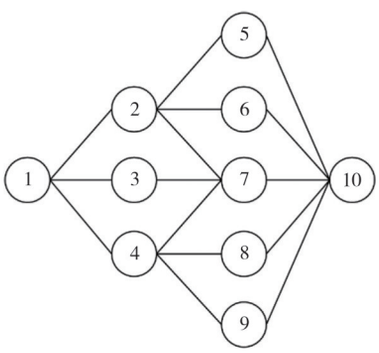


图2 任务内部依赖关系

Fig.2 Job internal dependencies

1.2 边缘计算节点和终端设备模型

设定在边缘计算系统中存在一个或多个边缘计算节点和终端设备，每个边缘计算节点的计算资源总量、储存资源总量都是相同的，每个终端设备的资源总量也是相同的，边缘计算节点和终端设备的索引分别表示为：

$$\mathrm{EN}=\{1,2,3,\cdots,N\},\ \mathrm{ED}=\{1,2,3,\cdots,D\}$$

边缘计算节点和终端设备具体属性表示为：

$$\mathrm{EN}_i/\mathrm{ED}_j=\{C,M,D,Cc,Mc,Dc\}$$

式中：C、M、D表示各项资源的可用量；Cc、Mc、Dc表示各项资源的总量.当任务在节点i或终端j上处理时，若可用量大于等于任务请求的计算资源量，则从可用量中分配任务所需的各项资源给任务，直到任务处理完成，再将所分配的资源归还计算节点；而当可用量小于任务请求的计算资源量时任务将重新进行调度决策.一种特殊情况是，当边缘计算节点或终端设备发生故障时，其可用量可以视作0.

1.3 任务调度优化模型

1.3.1 传输率

当任务被卸载到边缘计算节点进行处理时，任务相关数据将通过无线网络上传至计算节点.传输时延受任务上传数据量和信道状态影响.在实际传输过程中，上传数据量由实际任务确定，而信道状态服从衰落模型，即信道状态随时变化.根据香农定律，信道数据传输率可表示为：

$$\mathrm{Tr}=\mathrm{BD}\cdot\log_2\left[1+\frac{\lg(\mathrm{Nakagami}-m^2)}{\mathrm{noise}}\right] \tag{1}$$

式中：BD为带宽；noise为高斯白噪声；Nakagami-m^2为信道衰落模型，其分布函数如下：

$$P_z(z)=\frac{2m^m z^{2m-1}}{\Gamma(m)\mathrm{Pr}^m}\exp\left(-\frac{-mz^2}{\mathrm{Pr}^m}\right) \tag{2}$$

式中：Pr为平均功率；$\Gamma(m)$为伽马函数；m为衰落系数.m=1时，式(2)退化为Rayleigh衰落；m=∞时，代表无衰落.改变m的值，Nakagami-m^2可以转变为多种衰落模型.

1.3.2 计算时延

经代理给出任务卸载决策后，将任务调度至相应计算节点上进行处理，而任务的计算时延，由任务的计算量$\mathrm{CPU}_{\mathrm{task}}$和相应计算节点的计算能力$\mathrm{CPU}_{\mathrm{node}}$决定，计算公式如下：

$$L_{\mathrm{c}}=\mathrm{CPU}_{\mathrm{task}}/\mathrm{CPU}_{\mathrm{node}} \tag{3}$$

1.3.3 传输时延

当待处理任务被调度至边缘端进行处理时，该任务的处理时延由计算时延和传输时延组成.传输时延计算式为：

$$L_{\mathrm{t}}=\mathrm{Data}_{\mathrm{task}}/\mathrm{Tr} \tag{4}$$

式中：$\mathrm{Data}_{\mathrm{task}}$为任务卸载至边缘端时，需要上传的数

据总量;Tr表示当前信道数据传输率.

1.3.4　能耗

一个任务从开始处理到处理完成,总能耗由传输能耗和计算能耗组成.传输能耗的决定因素有单位时间的传输能耗、任务传输时延,计算公式如下:

$$NC_t = NCunit \cdot L_t \tag{5}$$

计算能耗的决定因素有单位时间的计算能耗、任务计算时延,计算公式如下:

$$NC_c = NCunit \cdot L_c \tag{6}$$

处理一个任务的总能耗可表示为:

$$NC = NC_t + NC_c \tag{7}$$

1.3.5　累计加权开销

调度策略性能的综合评价指标为长期加权开销Cost,即计算系统的计算时延、传输时延、能耗的加权和.整体加权开销表示为:

$$Cost_i = \sum_{a=1}^{A}\sum_{b=1}^{B}\alpha L_c + \sum_{a=1}^{A}\sum_{b=1}^{B}\beta L_t + \sum_{a=1}^{A}\sum_{b=1}^{B}\gamma NC \tag{8}$$

$$Cost = \sum_{i=1}^{itr} Cost_i \tag{9}$$

式中:α、β、γ分别为计算时延、传输时延和能耗的权重,值越大表示对该部分开销越重视且$\alpha+\beta+\gamma=1$;$Cost_i$表示第i次迭代的开销;A表示每次迭代的任务数目;B表示每个任务包含的子任务数目.式(9)为整个系统的整体加权开销,其中itr表示总迭代次数.

2　调度策略

在进行边缘计算下任务调度时,将任务调度问题视为一个多维背包问题来求解.每个计算节点看作一个背包,每个任务是具有价值$Cost_{taskij}$的物品,问题的最优解求解过程可转化为如何为任务选择匹配度最高的背包以获得最小累计加权开销.本文先由神经网络为任务输出其与每个计算节点的适配度,再根据输出结果按照一定的选择策略进行具体计算节点选择,从而充分利用计算节点的计算资源,实现计算任务和计算节点的较好匹配.

2.1　传统基于D3DQN的边缘计算任务调度算法框架

传统基于D3DQN的边缘计算任务调度算法基本框架如图3所示.

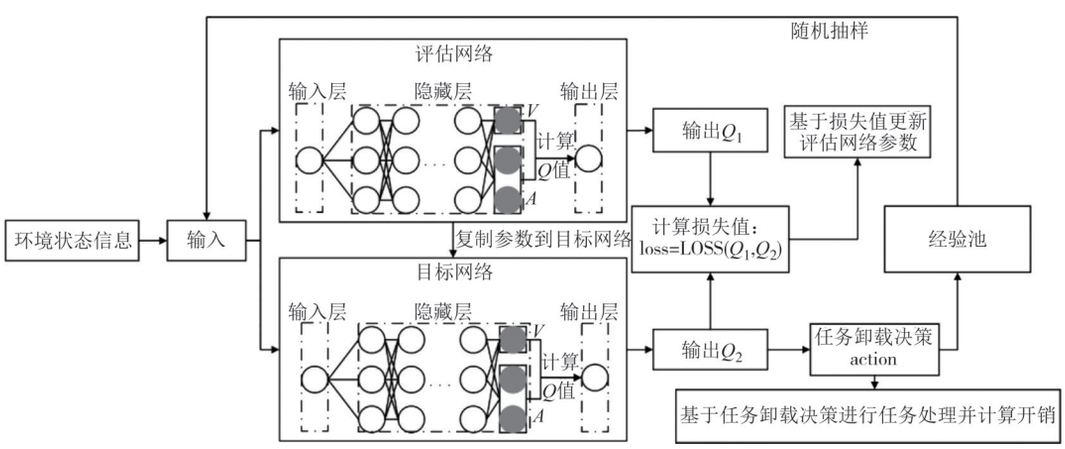


图3　基于D3DQN的边缘计算任务调度算法基本框架

Fig.3　The framework of job scheduling in edge computing algorithm based on D3DQN

图3中层的输出分为两部分.一部分为该状态的状态值(V)函数,另外一部分为每个动作的优势(A)函数.其中状态值函数关注长期动作选择,而优势函数关注当前动作选择,该结构适用于动态变化的环境.Q值由状态值函数和优势函数组成,其计算公式表示为:

$$Q = V + A \tag{10}$$

但V和A值的波动使得这两值不能唯一确定Q值,例如Q值为10时,V和A的值可能为4、6或者6、4,这不利于神经网络的更新,所以对式(10)做出如下处理:

$$Q = V + A - \mathrm{Mean}(A) \tag{11}$$

图3中目标网络根据环境状态信息获得输出后,基于该输出以ε-greedy策略进行卸载决策,再根据卸载决策将任务卸载至对应的计算节点或直接在本地进行处理,同时计算对应的加权开销,之后将本次经验存储至经验池;然后随机抽取部分经验,将抽取的经验进行回放,分别获得评估网络与目标网络

的输出,并基于两个输出计算损失值以更新评估网络参数.

2.2 D3DQN-CAA 边缘计算任务调度框架

为了降低“过估计”的影响,同时为了增加对变化环境的感知能力,本文在D3DQN算法框架的基础上进行边缘计算任务调度算法设计.在传统D3DQN算法中,损失值由目标网络和评估网络输出的差值组成,而目标网络参数的延迟更新会导致参数复制后两个网络之间输出差异过大,从而使神经网络出现波动,影响神经网络的稳定性;又采用ε-greedy策略进行卸载决策时,动作空间的探索度固定不变不利于神经网络的收敛;另外,在训练初期,利用传统经验回放机制抽取经验时可能会出现因经验数目较少而导致某些经验被重复抽取,使得神经网络学习效率低下的问题,且随机抽取的抽样策略会造成样本效率较低.为提高神经网络的收敛速度、稳定性以及进一步降低“过估计”的影响,在以上D3DQN算法框架的基础上,提出一种综合性Q值计算方法CQC,根据上次迭代损失值衡量目标网络和评估网络输出在最终输出中的占比,使得整个学习过程的输出由目标网络输出为主逐渐转变为以评估网络输出为主;为了增加动作空间探索度和进一步提高算法稳定性,设计了一种能够自适应调整动作空间探索度的动作选择策略AGP,主要基于近几次迭代的损失值均值对卸载动作进行选择,以自适应地调整动作空间的探索度;为提高样本效率和适应资源有限的边缘计算系统,设计了自适应轻量优先级回放机制ALPR,在每次迭代中基于损失值对学习经验进行优先级排序,再抽取中间部分的学习经验存储至经验池,达到指定回放周期数时进行经验回放,以确保回放经验的唯一性同时提高样本效率.将以上方法和传统D3DQN算法框架进行融合后,得到图4所示的本文D3DQN-CAA算法框架.

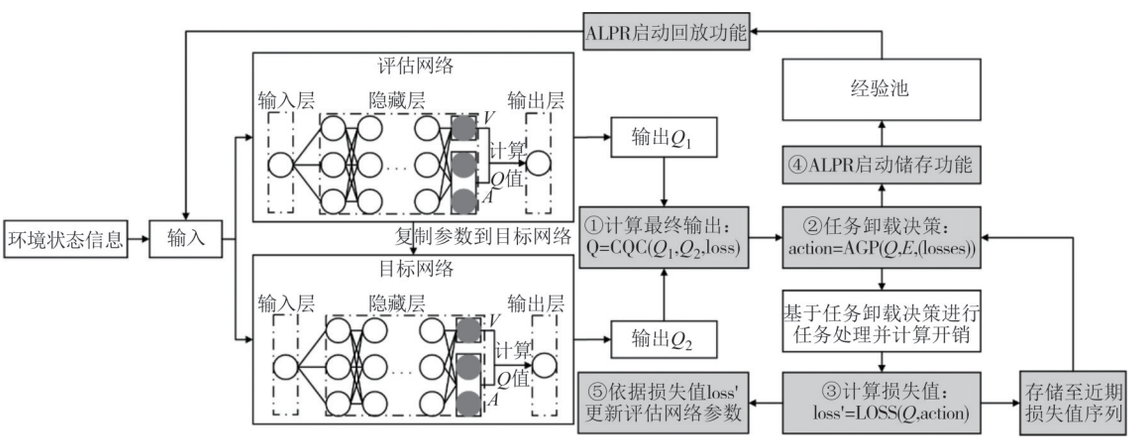


图4 D3DQN-CAA基本框架

Fig.4 The basic framework of D3DQN-CAA

图4中目标网络和评估网络根据环境状态信息分别获得输出,根据两个输出与上次迭代的损失值在CQC方法中计算获得最终输出;然后将输出值和近期损失值均值输入AGP中,获得当前任务的卸载决策,再根据卸载决策将任务卸载至对应的计算节点或直接在本地进行处理,同时计算对应的加权开销;之后计算损失值并存储到近期损失值序列中,当达到指定周期数时计算损失值均值,替换AGP中的均值并清空序列;接下来自适应轻量优先级回放机制ALPR根据损失值对学习经验进行优先级排序,并抽取中间部分的经验放入经验池中,当达到指定回放周期数时,回放经验池中全部经验并清空经验池;最后根据损失值进行参数更新.

2.3 优化策略

2.3.1 综合性Q值计算方法

在传统DQN算法中,根据环境状态信息输出所有可能卸载动作的Q值,其大小表示了卸载动作被选择的概率大小.然后选择其中Q值最大值所对应的卸载动作,作为当前待处理任务的调度决策.但在神经网络训练初期,选择最大Q值会导致神经网络在进行参数更新时实际Q值将向着比真实Q值大的方向更新,从而引起“过估计”问题.为了降低“过估计”的影响,文献[19]在DQN算法的基础上提出了Double DQN算法,基于此,可以在对评估网络和目标网络两个神经网络进行参数更新时,评估网络参数实时更新,目标网络参数延迟更新,并基于目标网络

输出进行卸载动作选择.虽然这样降低了“过估计”的影响,但这不利于评估网络的参数更新,同时将评估网络的参数复制到目标网络后会容易引起神经网络的波动.为解决上述问题,本文提出一种综合性Q值计算方法CQC,以降低“过估计”对神经网络的影响、减小参数复制后的神经网络波动幅度.具体计算公式如下:

$$\begin{cases} \mathrm{OT} = \mathrm{TNet}(\mathrm{input}) \\ \mathrm{OE} = \mathrm{ENet}(\mathrm{input}) \\ O = \mathrm{OT} \cdot (1 - \mathrm{e}^{\wedge -\mathrm{Loss}}) + \mathrm{OE} \cdot \mathrm{e}^{\wedge -\mathrm{Loss}} \end{cases} \tag{12}$$

式中:TNet、ENet分别为目标网络和评估网络;OT、OE分别为目标网络和评估网络的输出;Loss为上一次迭代的损失;O为当次迭代中神经网络的最终输出.损失值可以反映神经网络的学习程度,损失值越大说明神经网络距离收敛越远,越难以对当前环境状态做出准确评估,受“过估计”影响越大;反之距离收敛越近,受“过估计”影响越小.该方法的工作原理及工作过程如图5所示.在学习初期,神经网络的损失值较大,根据式(12)可知,整个神经网络的输出将以目标神经网络的输出为主,以降低“过估计”的影响,越接近收敛,则转变为以评估网络的输出为主,从而让目标网络和评估网络共同决定最终网络输出,以降低“过估计”的影响,保证神经网络的稳定性.

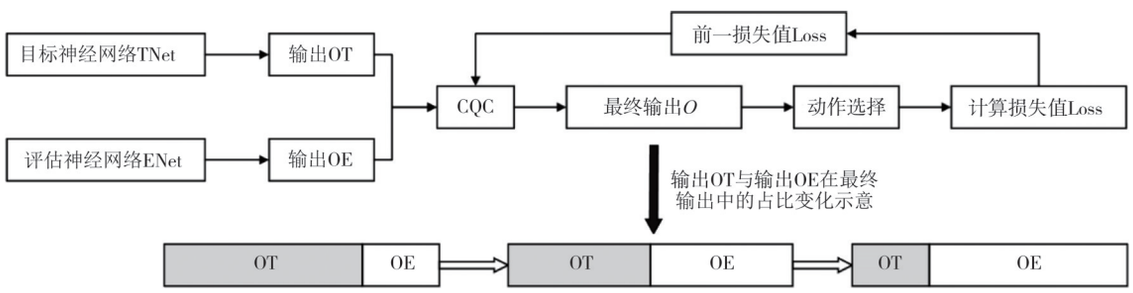


图5　CQC工作原理及工作过程

Fig.5　Working principle and process of CQC

2.3.2　自适应动作空间动态探索度调整策略

为了增加动作空间的探索度,已有的工作常常在动作选择上采用ε-greedy策略[5,7-11],以固定概率选择其他动作,反之选择最大Q值对应动作.但在神经网络的学习过程中不同阶段所需探索程度不同,在神经网络学习初期,为了尽快向最优解靠近,应当以更大的概率进行动作空间的探索即选择非最大值对应的卸载动作,随着学习的进行再以更小的概率选择非最大值对应的卸载动作.损失值可以反映神经网络的学习进度,损失值大则说明神经网络远未收敛,还需要以较大的探索度对动作空间进行探索以寻找最优解,损失值较小则相应地减小探索度.为防止损失值变化过大造成神经网络波动,本文提出一种自适应动态动作空间探索度调整策略AGP,以近几次迭代的损失值均值作为设计动作选择策略概率值的基础,实现对神经网络动作空间探索程度的动态调整,其工作原理及工作过程如图6所示.具体设计如下:

$$F = \begin{cases} \mathrm{True}, \mathrm{rd} \leqslant [1 - \left[\mathrm{e}^{E - \left(\mathrm{Loss}_{1,2,\cdots,n}\right)^2}\right] \\ \mathrm{False}, \mathrm{rd} > [1 - \left[\mathrm{e}^{E - \left(\mathrm{Loss}_{1,2,\cdots,n}\right)^2}\right] \end{cases} \tag{13}$$

式中:rd为随机数生成函数,用于生成[0,1]范围内的随机数;F值为True,则为当前待处理任务选择非最大值对应的卸载动作,为False,则选择最大值对应的卸载动作.

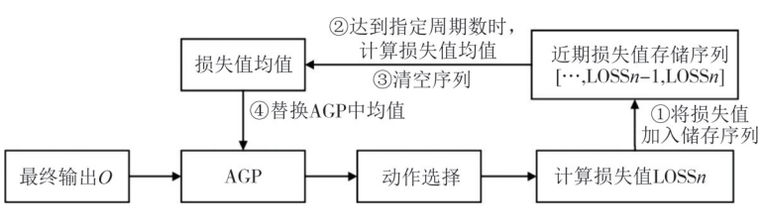


图6　AGP工作原理及工作过程

Fig.6　Working principle and process of AGP

2.3.3 自适应轻量优先级回放机制

随着神经网络状态空间维度的增加,需要更多的学习样本才能使神经网络达到满意的效果.但实际样本的数量往往是有限的,这时就需要考虑如何提高有限数量样本效率.经验回放机制不仅可以解决学习样本效率低的问题,还能打破数据关联性,常常配合DQN算法一起解决复杂高维问题.然而,在有限计算资源的边缘环境下,传统经验回放机制保存所有历史经验的做法会消耗大量的存储资源,且从历史经验中随机抽取一定数目的样本进行回放不能有效利用更高效的样本.为了提高样本效率,更好地适应资源有限的边缘计算系统下的任务调度,本文提出一种对边缘计算环境更加友好、更注重高效样本的自适应轻量优先级回放机制ALPR.

最近的学习经验才最有利于神经网络的学习,与其关联性最大,同时为确保回放经验的唯一性,本文回放机制ALPR在经验池中存储最近m次迭代中被抽取的学习经验.该机制的工作原理及工作过程如图7所示,以损失值为基础对学习经验进行排序,损失值小容易将神经网络引导向局部最优,损失值大则远离最优解,所以将中间部分x个的历史经验抽取进行存储;当迭代次数达到指定经验回放周期数时进行经验回放并清空经验池.而x值在神经网络学习的不同阶段取值也不同,学习初期神经网络需要注重新知识的学习,此时x值应取一个较小值;随着学习的深入,神经网络应当注重历史经验的回放,以稳定性能,此时x值应取一个较大值.

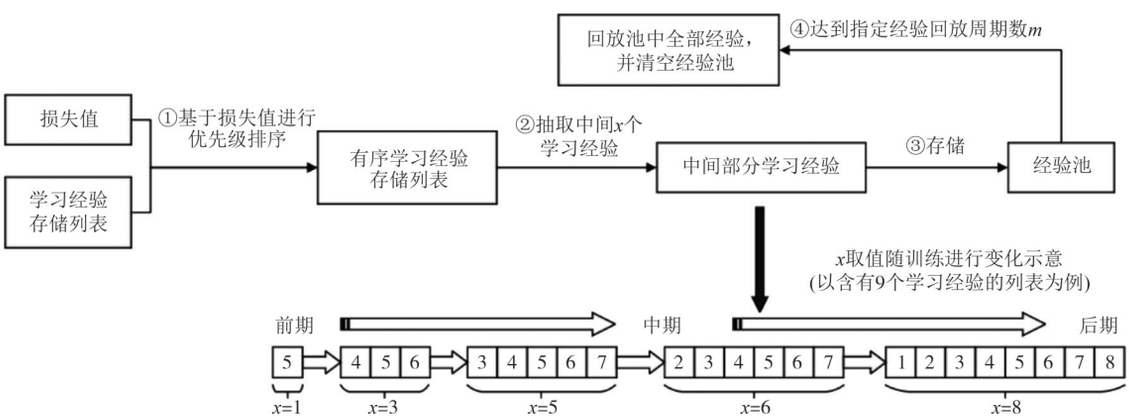


图7 ALPR工作原理及工作过程

Fig.7 Working principle and process of ALPR

3 处理过程

系统整体处理流程如图8所示.处理开始时,初始化模型各项参数,itr为1;若迭代次数满足条件1:itr mod N_1=0为参数复制周期(N_1),则复制参数到目标网络.若迭代次数满足条件2:itr mod N_2=0为经验回放周期(N_2),则回放经验池中的学习经验并清空经验池.不满足以上条件,则获取环境状态信息.获得评估网络和目标网络的输出后,利用综合性Q值计算方法CQC获得最终输出;并通过自适应动作空间动态探索度调整策略AGP生成卸载决策;根据当前任务的卸载决策将任务卸载至对应的计算节点进行处理或本地处理,并计算相应的加权开销;再计算损失值,启动自适应轻量优先级回放机制ALPR中的存储经验功能,存储优先级最高的学习经验;将损失值放入存储序列中,若满足条件3:itr mod N_3=0,为损失值均值替换周期(N_3),则计算存储序列中的损失值均值,替换AGP中均值并清空序列;基于损失值更新评估网络参数.

3.1 环境状态信息及感知

边缘计算系统的环境状态信息可以由一组与计算任务、计算节点、无线网络状况等相关的参数组成.本文的环境状态信息由计算任务数据大小、所需计算资源数、所需存储资源数和所有计算节点可用计算资源数、可用存储资源数等信息组成,即

$$\text{state}_i = (\text{task}\ d_s, t_c, t_s, \cdots) + \text{node}_i(n_c, n_s, \cdots)$$

式中:state_i表示计算任务与第i个计算节点的状态信息;d_s、t_c、t_s分别为计算任务的数据大小、所需计算资源数、所需存储资源数;n_c、n_s分别为计算节点可用计算资源数、可用存储资源数.

在处理终端设备发来的调度请求时,代理需要综合考虑当前待处理任务和所有计算节点的状态信息,以做出当前最优的调度决策.向下,代理接收终

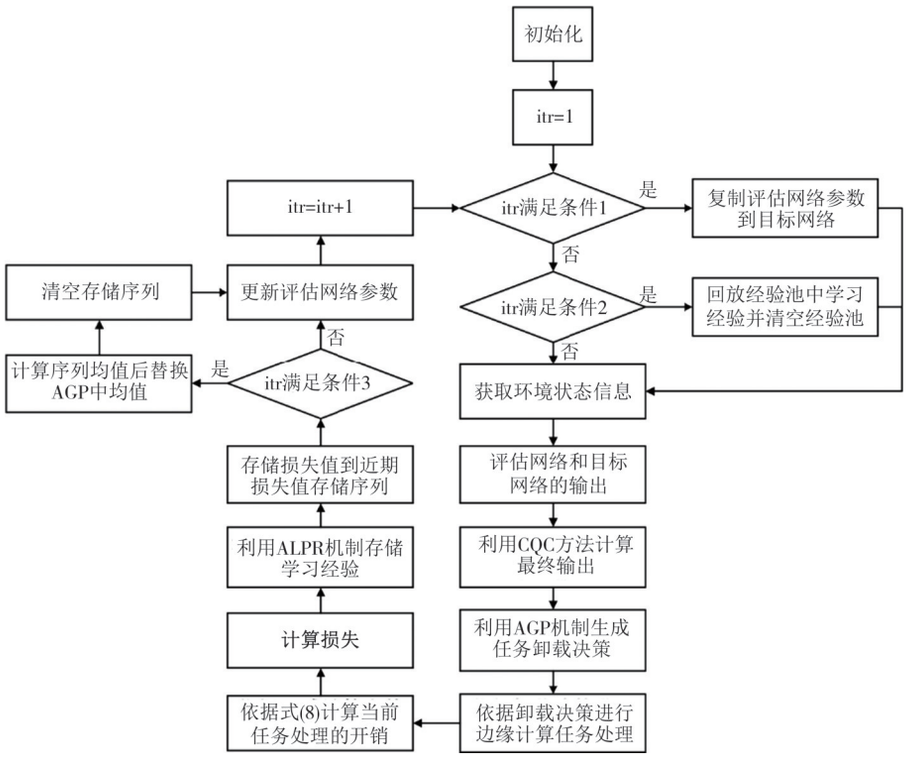


图 8　系统处理流程

Fig.8　System processing flow

端设备发送的调度请求中包含待处理任务的状态信息与终端设备的状态信息;向上,代理向边缘服务器请求所有边缘计算节点的状态信息.代理在获取以上所需的环境状态信息后即可开始调度决策.

3.2　最终输出计算

代理获得环境状态进行正规化后即可作为神经网络的输入.在CQC方法中,需要同时获得评估网络和目标网络的输出,以计算神经网络的最终输出.因此,环境状态同时作为评估网络和目标网络的输入,得到各自的输出后,通过CQC方法获得最终输出,即

$$\left.\begin{matrix}\text{Output1}\\ \text{Output2}\end{matrix}\right\} \overset{\text{CQC}}{\Rightarrow} \text{FinalOutput} \tag{14}$$

式中:Output1、Output2分别为评估网络和目标网络基于当前环境状态获得的输出.

3.3　动作与动作选择

本文调度算法将依据环境状态信息获得当前待处理任务与各个计算节点的适配度,并作为卸载决策的依据,决定将任务放置在何处进行处理,即进行动作选择.通过神经网络和CQC方法后,获得任务与每个计算节点的适配度,并按照AGP机制选择计算节点以承载计算任务.

3.4　损失函数、损失计算及存储、回放学习经验

本文将任务调度问题视为多维背包问题,每次调度时选择与任务匹配度最高的计算节点进行任务处理.因此,损失函数可使用在分类器中常用的损失函数——交叉熵损失函数,具体计算如下:

$$\text{Loss} = \text{CrossEntropy}(\text{output}, \text{action}) \tag{15}$$

式中:output为评估网络的输出;action为动作选择.ALPR机制基于损失值对当前迭代的所有学习样本进行排序,并通过ALPR机制的存储经验功能将位于中间部分的学习样本存储在经验池中.当迭代次数满足特定条件时,通过ALPR机制中的经验回放功能取出经验池中学习经验并回放,即:

$$F = \begin{cases}\text{Condition1, Sort And Store}\\ \text{Condition2, replay}\end{cases} \tag{16}$$

$$\begin{aligned}&[\text{ep}_1(\text{task}_1, \text{loss}_1), \cdots, \text{ep}_i(\text{task}_i, \text{loss}_i), \cdots] \overset{\text{sort}}{\Rightarrow}\\ &[\text{ep}_4(\text{task}_4, \text{loss}_4), \cdots, \text{ep}_n(\text{task}_n, \text{loss}_n), \cdots] \overset{\text{store}}{\Rightarrow}\\ &p[\text{ep}_a(\text{task}_a, \text{loss}_a), \cdots, \text{ep}_n(\text{task}_n, \text{loss}_n), \cdots]\end{aligned} \tag{17}$$

式中:ep_i表示学习经验;p表示经验池.通过式(16)来判断使用ALPR机制的排序存储学习经验功能还是回放功能;通过式(17)完成排序并存储学习经验功能.

3.5　参数更新

获得学习样本的损失值后,计算梯度grad:

$$grad_i = gradient(loss_i) \tag{18}$$

式中:gradient为梯度计算函数;$loss_i$为第i个学习样本的损失值.梯度值和学习率一起在特定的优化器中用于参数θ的更新,具体计算如下:

$$\theta = Optimizer(grad_i, lr|\theta') \tag{19}$$

式中:Optimizer为优化器;$grad_i$为第i个学习样本的梯度;lr为学习率;θ'为神经网络参数.

3.6 算法实现

通过以上处理过程,可以实现任务的最优调度决策,达到降低系统开销、提升任务处理实时性、降低系统能耗的目的.具体算法实现如算法1所示.

算法1:基于改进D3DQN的自适应边缘计算任务调度

输入:环境状态信息

输出:卸载决策

1:初始化经验池ALPR-P、AGP中的损失值AGP-L、CQC中的损失值CQC-L

2:for iteration=1 to Iteration do

3: if iteration/N_1==0 then

4: 复制评估网络的参数到目标网络

5: end if

6: if iteration /N_2==0 then

7: 回放ALPR-P中的学习样本

8: end if

9: 正规化输入

10:获得评估网络和目标网络的输出P_1、P_2

11:计算最终输出logits = CQC(P_1, P_2, CQC-L)

12:基于logits生成卸载动作action:

13:if random() < AGP(AGP-L) then

14: 随机卸载动作

15:else

16: 最大Q值对应卸载动作

17:end if

18:计算损失值,赋值给CQC-L后存储至AGP-L中

19:if iteration/N_3==0 then

20: 计算AGP-L的均值并替换AGP中的均值,清空AGP-L

21:end if

22:对学习经验进行优先级排序,然后存储中间部分在ALPR-P中

23:更新评估网络参数

24:end for

4 仿真实验

为了验证本文所提出D3DQN-CAA任务调度算法可行性和有效性,主要从奖励值、能耗和累计加权开销几个方面和已有较具代表性的DQN[20]和D3DQN[10]、进行实验对比,另外为了验证所提算法在相同或更多系统开销组成的情况下能够合理利用系统内计算资源以降低系统开销和能耗,再加入Random、Only Edge和Only Local方法进行实验对比.Random算法随机在所有的计算节点上为当前待处理任务选择计算节点进行任务处理;Only Edge算法则只选择边缘计算节点承载计算任务;Only Local算法仅选择终端设备进行任务处理.使用的仿真平台为EdgeCloudSimPy[21],本文仿真实验采用的数据集需要体现子任务之间依赖关系,所以具体数据集采用有向无环图生成器[22]和Alibaba Cluster Data V2017[23]模拟得到.

4.1 实验设置

在仿真实验中,每个边缘计算节点的配置为16单位CPU、16单位内存,每个终端设备的配置为4单位CPU、4单位内存.为模拟在一个时间片内,处理来自不同终端设备的计算任务,每次迭代使用5个样本,每个样本(任务)中有10个存在依赖关系的子样本(子任务),设置多个边缘计算节点和终端设备,以模拟一个多边缘节点、多终端设备且同一时间片内终端设备只生成一个计算任务的边缘计算场景.每200次迭代后将评估网络的参数复制到目标网络(N_1=200),每50次迭代进行经验回放(N_2=50),每100次迭代更新AGP机制中的损失值均值(N_3=100).

算法DQN、D3DQN和D3DQN-CAA的神经网络设置如表1所示.

表1 神经网络设置

Tab.1 Neural network setting

算法	评估网络设置	目标网络设置
DQN	隐藏层为全连接层,激活函数为Relu,输出层无激活函数	无目标网络
D3DQN	隐藏层为全连接层,激活函数为Relu,输出层无激活函数	同目标网络
D3DQN-CAA	隐藏层为全连接层,激活函数为Relu,输出层无激活函数	同目标网络

4.2 实验结果与分析

4.2.1 学习率选取分析

对于神经网络来讲,学习率的取值将在很大程度上影响模型的性能,所以在进行其他实验之前应当确定一个较为合理的学习率.

图 9 为学习率分别取值 0.001、0.005、0.01、0.02 的损失值曲线,横轴为学习步数,纵轴为损失值.可以看到,学习率为 0.001 和 0.005 时损失值曲线在 0 到 400 学习步数内下降趋势较为平缓,而学习率为 0.010 和 0.020 的损失值曲线在 2 800 学习步数处均出现了大幅度的波动;对比学习率为 0.001 和 0.005 的损失值曲线,两条曲线形状上非常相似,但学习率为 0.001 的损失值曲线在模型收敛后更加平稳,不易出现大幅度波动.所以在其他实验当中采用学习率为 0.001.

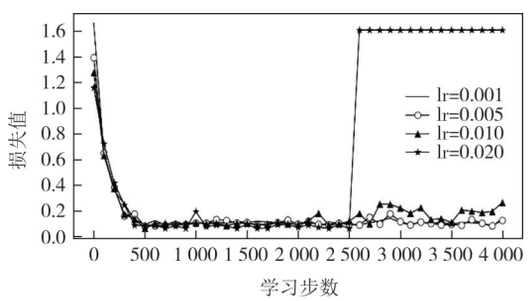


图 9　不同学习率的损失值曲线

Fig.9　Curve of loss value with different learning rates

4.2.2 损失值对比

为验证各项机制的作用和对比不同机器学习算法之间的收敛情况,下面在 D3DQN-CAA、DQN、D3DQN 之间进行损失值对比实验.

图 10 为上述算法的损失值综合对比,横轴为学习步数,纵轴为损失值,可以看出,D3DQN-CAA 的曲线最为平滑且收敛后波动幅度最小,说明 CQC 方法、ALPR 机制能够起到稳定模型的作用.由图 10 可知,D3DQN-CAA、DQN、D3DQN 损失值曲线的变化趋势相似,但 DQN 损失值曲线下降过快,又 DQN 和 D3DQN 均无法达到 D3DQN-CAA 的收敛值.同时 DQN 和 D3DQN 损失值曲线尽管已经收敛,但处在一个相对更高的收敛值且波动幅度较大,这说明 AGP 机制可以有效控制模型在动作空间上探索度,提高模型的稳定性;另从对比中可以看到 DQN、D3DQN 的损失值曲线容易在 1 200、1 400、1 600 学习步数等处出现较大幅度的波动且相比 D3DQN-CAA 达到收敛需要更多的学习步数,这说明 CQC 方法在提高神经网络收敛速度和降低参数复制后神经网络波动幅度上具有显著作用.以上实验对比结果说明,本文所提方法的损失值曲线达到收敛所需的学习步数最少,不易出现波动,性能更稳定.

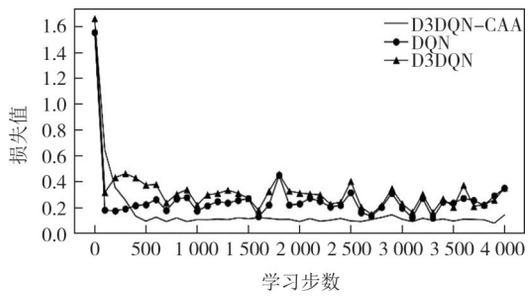


图 10　损失值曲线对比

Fig.10　Comparison of loss value curve

4.2.3 累计奖励对比

为反映算法对任务调度和处理效率,以奖励值进行衡量,本文的奖励值赋值规则为任务正常被调度和处理赋值为 0,否则赋值为-1.D3DQN-CAA 和基准算法的累计奖励值对比结果如图 11 所示,横轴为迭代周期数,纵轴为累计奖励值.仅利用了资源非常有限的终端设备的 Only Local 的累计奖励曲线远低于其他算法,这意味着该算法的任务调度和处理效率非常低,处理完全部任务往往需要冗长的时间.对比其他算法的奖励值曲线可知, D3DQN-CAA 的任务调度的处理效率最高.

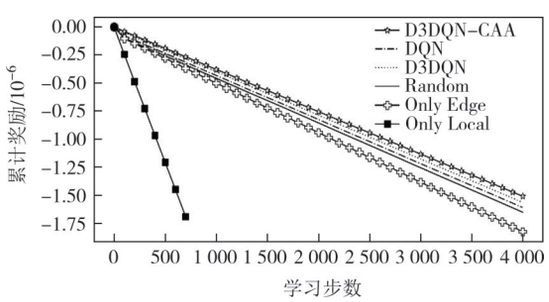


图 11　累计奖励对比

Fig.11　Comparison of cumulative reward

4.2.4 能耗及加权开销对比

运营商在提供服务时,收益的影响因素当中能耗是最关注的一个,D3DQN-CAA 和基准算法的累计能耗对比如图 12 所示,横轴为迭代周期数,纵轴为累计奖励值.Only Local 冗长的处理时间决定了非常大的能耗;相比 Only Edge 方法,尽管 Random 可以更加高效地处理任务,但算法使用到终端设备从而造成高能耗;本文方法 D3DQN-CAA 由于采用了 CQC 方法、AGP 机制和 ALPR 机制,在保证算法稳定性的同时,能够有效地降低神经网络的训练步数和

充分利用计算资源,使得本文方法的累计能耗曲线低于DQN、D3DQN和其他算法.

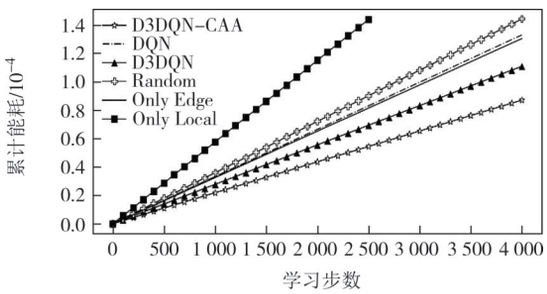


图12 累计能耗对比

Fig.12 Comparison of accumulated energy consumption

本文D3DQN-CAA算法和基准算法的综合性能对比结果如图13所示,X轴为迭代周期数,Y轴为累计奖励值且单位刻度大小为10^{-7}.从图13中可以看出,性能最差即累计加权开销曲线高于其他算法的是Only Local,性能最优的是本文调度算法D3DQN-CAA,中间部分由高到低依次为Random、DQN、Only Edge、D3DQN.

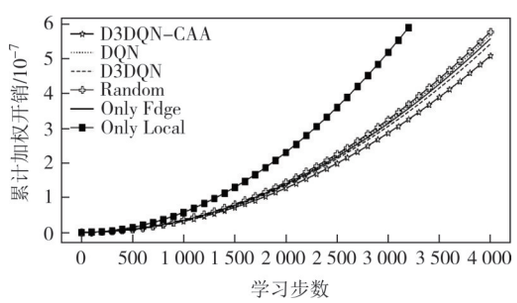


图13 累计加权开销对比

Fig.13 Comparison of cumulative weighted cost

通过以上实验分析可知,D3DQN-CAA算法能够有效地降低深度强化学习网络的训练步数,能充分利用计算资源降低系统时延和能耗的累计加权开销,提升任务处理的实时性,降低系统能耗.

5 结论

为解决已有边缘计算环境下深度强化学习算法存在的动作空间探索度固定不变、样本效率低、内存需求量大、稳定性差等问题,本文在改进深度强化学习模型Dueling Double DQN的基础上,提出了一种自适应的边缘计算任务调度算法.为了加快模型收敛速度和增加模型收敛后的稳定性,引入了综合性Q值计算方法;为了增加动作空间探索度和进一步提高算法稳定性,提出了自适应动作空间动态探索度调整策略;为了提高样本效率和适应资源有限的边缘计算系统,使用了自适应轻量优先级回放机制.结合以上优化机制,设计并实现了最终的任务调度算法.实验结果表明,本文边缘计算任务调度算法,能够有效地降低深度强化学习网络的训练步数,可以充分利用边缘计算资源,降低系统时延和能耗的累计加权开销,提升任务处理的实时性,降低系统能耗.

参考文献

[1] 周悦芝, 张迪. 近端云计算:后云计算时代的机遇与挑战[J]. 计算机学报, 2019, 42(4): 677-700.
ZHOU Y Z, ZHANG D. Near-end cloud computing: opportunities and challenges in the post-cloud computing era[J]. Chinese Journal of Computers, 2019, 42(4): 677-700. (in Chinese)

[2] WANG F X, ZHANG M, WANG X X, et al. Deep learning for edge computing applications: a state-of-the-art survey[J]. IEEE Access, 2020, 8: 58322-58336.

[3] SHI W S, CAO J, ZHANG Q, et al. Edge computing: vision and challenges[J]. IEEE Internet of Things Journal, 2016, 3(5): 637-646.

[4] WANG X F, HAN Y W, LEUNG V C M, et al. Convergence of edge computing and deep learning: a comprehensive survey[J]. IEEE Communications Surveys & Tutorials, 2020, 22(2): 869-904.

[5] 卢海峰, 顾春华, 罗飞, 等. 基于深度强化学习的移动边缘计算任务卸载研究[J]. 计算机研究与发展, 2020, 57(7): 1539-1554.
LU H F, GU C H, LUO F, et al. Research on task offloading based on deep reinforcement learning in mobile edge computing [J]. Journal of Computer Research and Development, 2020, 57(7): 1539-1554. (in Chinese)

[6] 张子迎, 陈云飞, 王宇华, 等. 基于启发式深度Q学习的多机器人任务分配算法[J]. 哈尔滨工程大学学报, 2022, 43(6): 857-864.
ZHANG Z Y, CHEN Y F, WANG Y H, et al. Multi-robot task allocation algorithm b Multirobot task allocation algorithm based on heuristically accelerated deep Q network [J]. Journal of Harbin Engineering University, 2022, 43(6): 857-864. (in Chinese)

[7] 喻鹏, 张俊也, 李文璟, 等. 移动边缘网络中基于双深度Q学习的高能效资源分配方法[J]. 通信学报, 2020, 41(12): 148-161.
YU P, ZHANG J Y, LI W J, et al. Energy-efficient resource allocation method in mobile edge network based on double deep Q-learning[J]. Journal on Communications, 2020, 41(12): 148-161. (in Chinese)

[8] ZHU A Q, GUO S T, MA M F, et al. Computation offloading for

workflow in mobile edge computing based on deep Q-learning [C]//2019 28th Wireless and Optical Communications Conference (WOCC). Beijing,China: IEEE,2019:1-5.

[9] TANG M, WONG V W S. Deep reinforcement learning for task offloading in mobile edge computing systems [J]. IEEE Transactions on Mobile Computing,2022,21(6):1985-1997.

[10] HAN B A,YANG J J. Research on adaptive job shop scheduling problems based on dueling double DQN[J]. IEEE Access,2020, 8:186474-186495.

[11] XIONG X,ZHENG K,LEI L,et al. Resource allocation based on deep reinforcement learning in IoT edge computing[J]. IEEE Journal on Selected Areas in Communications, 2020, 38(6): 1133-1146.

[12] ZOU J F, HAO T B, YU C, et al. A3C-DO: a regional resource scheduling framework based on deep reinforcement learning in edge scenario [J]. IEEE Transactions on Computers, 2021, 70(2): 228-239.

[13] QI F, ZHUO L, XIN C. Deep reinforcement learning based task scheduling in edge computing networks [C]//2020 IEEE/CIC International Conference on Communications in China (ICCC). Chongqing,China: IEEE,2020:835-840.

[14] NATH S, WU J X. Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems[J]. Intelligent and Converged Networks,2020,1(2):181-198.

[15] KE H C, WANG J, DENG L Y, et al. Deep reinforcement learning-based adaptive computation offloading for MEC in heterogeneous vehicular networks [J]. IEEE Transactions on Vehicular Technology,2020,69(7):7916-7929.

[16] NATH S, WU J X. Dynamic computation offloading and resource allocation for multi-user mobile edge computing [C]// GLOBECOM 2020—2020 IEEE Global Communications Conference. Taipei,China: IEEE,2020:1-6.

[17] 江未来,吴俊,王耀南. 基于元强化学习的无人机自主避障与目标追踪[J]. 湖南大学学报(自然科学版),2022,49(6):101-109.
JIANG W L,WU J,WANG Y N. Autonomous obstacle avoidance and target tracking of UAV based on meta-reinforcement learning [J]. Journal of Hunan University (Natural Sciences), 2022, 49(6):101-109.(in Chinese)

[18] 陈卓,姜伟豪,杜军威. 基于策略记忆的深度强化学习序列推荐算法研究[J]. 湖南大学学报(自然科学版),2022,49(8):208-216.
CHEN Z,JIANG W H,DU J W. Research on deep reinforcement learning sequential recommendation algorithm based on policy memory[J]. Journal of Hunan University (Natural Sciences), 2022,49(8):208-216.(in Chinese)

[19] VAN HASSELT H, GUEZ A, SILVER D. Deep reinforcement learning with double Q-learning[J]. Proceedings of the AAAI Conference on Artificial Intelligence,2016,30(1):129-144.

[20] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540):529-533.

[21] LI F C, HU B. DeepJS: job scheduling based on deep reinforcement learning in cloud data center[C]//Proceedings of the 2019 4th International Conference on Big Data and Computing -ICBDC 2019. May 10-12, 2019. Guangzhou, China: ACM, 2019:48-53.

[22] ARABNEJAD H, BARBOSA J G. List scheduling algorithm for heterogeneous systems by an optimistic cost table [J]. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(3): 682-694.

[23] ALIBABA.(n.d.). Alibaba/clusterdata. 2017, https://github.com/alibaba/clusterdata/tree/master/cluster-trace-v2017.