基于深度强化学习的云边协同计算迁移研究

陈思光[1,2],陈佳民[1,3],赵传信[2]
(1. 南京邮电大学江苏省通信与网络技术工程研究中心,江苏南京 210003;
2. 安徽师范大学网络与信息安全安徽省重点实验室,安徽芜湖 241002;
3. 南京邮电大学江苏省宽带无线通信和物联网重点实验室,江苏南京 210003)

摘 要: 基于单一边缘节点计算、存储资源的有限性及大数据场景对高效计算服务的需求,本文提出了一种基于深度强化学习的云边协同计算迁移机制. 具体地,基于计算资源、带宽和迁移决策的综合性考量,构建了一个最小化所有用户任务执行延迟与能耗权重和的优化问题. 基于该优化问题提出了一个异步云边协同的深度强化学习算法,该算法充分利用了云边双方的计算能力,可有效满足大数据场景对高效计算服务的需求;同时,面向边缘云中边缘节点所处环境的多样及动态变化性,该算法能自适应地调整迁移策略以实现系统总成本的最小化. 最后,大量的仿真结果表明本文所提出的算法具有收敛速度快、鲁棒性高等特点,并能够以最低的计算成本获得近似贪心算法的最优迁移决策.

关键词: 深度强化学习; 边缘计算; 计算迁移; 资源分配; 能量消耗
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2021)01-0157-10
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.12263/DZXB.20200131

Deep Reinforcement Learning Based Cloud-Edge Collaborative Computation Offloading Mechanism

CHEN Si-guang[1,2], CHEN Jia-min[1,3], ZHAO Chuan-xin[2]
(1. *Jiangsu Engineering Research Center of Communications and Network Technology, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu* 210003, *China*;
2. *Anhui Provincial Key Laboratory of Network and Information Security, Anhui Normal University, Wuhu, Anhui* 241002, *China*;
3. *Jiangsu Key Lab of Broadband Wireless Communication and Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu* 210003, *China*)

Abstract: Based on the computation and storage resources limitation of single edge node and the demand for efficient computing services in big data scenarios, this paper proposes a deep reinforcement learning based cloud-edge collaborative computation offloading mechanism. Specifically, based on a comprehensive consideration of computing resources, bandwidth and offloading policy, an optimization problem is formulated to minimize the weight sum of execution delay and energy consumption of all user tasks. An asynchronous cloud-edge collaborative deep reinforcement learning (ACEC-DRL) algorithm is proposed to solve such optimization problem. This algorithm can effectively satisfy the demand of efficient computing services in big data scenario by jointly leveraging the computation capabilities of cloud and edge nodes. Meanwhile, under the various and dynamic environments of edge nodes in the edge cloud, this algorithm can adaptively adjust offloading policy to achieve the minimization of system cost. Finally, the extensive simulation results show that the proposed ACEC-DRL algorithm has the characteristics of fast convergence rate and high robustness, and its optimal offloading policy closely approximates to the solution of greedy algorithm with the lowest computation cost.

Key words: deep reinforcement learning; edge computing; computation offloading; resource allocation; energy consumption

---

收稿日期:2020-01-20;修回日期:2020-03-19;责任编辑:覃怀银
基金项目:国家自然科学基金(No. 61971235, No. 61771258, No. 61871412);江苏省"六大人才高峰"高层次人才项目(No. XYDXXJS-044);江苏省"333 高层次人才培养工程";南京邮电大学"1311"人才计划;中国博士后科学基金(面上一等)(No. 2018M630590);南京邮电大学国家自然科学基金孵化项目(No. NY217057, No. NY218058);网络与信息安全安徽省重点实验室开放课题(No. AHNIS2020001);江苏省通信与网络技术工程研究中心开放课题重点项目(No. JSGCZX17011);赛尔网络下一代互联网技术创新项目(No. NGII20190702)

1 引言

当前,在边缘计算场景下许多性能优越的计算迁移方案被提出,例如文献[1~6].这些研究方案基本都基于精确算法或基于数学规划的近似算法来求解相应的计算迁移优化问题,在大数据场景下求解复杂组合优化问题显得力不从心;同时,上述求解算法难以基于边缘计算实际场景中动态变化的负载做出自适应的迁移决策.基于上述研究方案存在的问题,文献[7~11]等通过融合机器学习方法来实现高效自适应的网络资源分配与迁移决策.上述结合了机器学习理论的计算迁移研究都采用单一的深度学习或者强化学习理论来解决相应的优化问题.

进一步地,文献[12~16]等基于深度强化学习理论来解决边缘计算场景下的计算迁移问题,即研究通过有机结合深度学习的表征学习能力和强化学习的决策能力,使得智能体具备更强的学习能力,进而能够更好地解决复杂系统的感知决策问题.上述基于深度强化学习的计算迁移求解算法,都使用一个或者多个并行的深度神经网络来生成迁移决策,同时将生成的迁移决策存储在共享内存中,以进一步训练和改进深度神经网络.在面对任务量巨大的场景时,由于单个边缘节点资源的局限性及缺乏云边协同的考量,使得其往往难以满足大数据场景的高效数据处理需求.同时,由于单个边缘节点所能触及样本种类的局限性,这类方法在差异性较大的动态多样性环境下往往难以做出最优的迁移决策.

基于上述研究方案存在的问题,本文研究了基于深度强化学习的云边协同计算迁移机制,即首先基于计算资源、带宽和迁移决策的综合性考量,构建了一个任务执行延迟与能耗权重和最小化优化问题,用于降低任务处理的时间与能量开销.其次,基于异步优势行动者-评论家算法[17],提出一个异步云边协同的深度强化学习算法用于解决上述优化问题.该算法充分利用了云边双方的计算能力,可有效满足大数据场景的高效数据处理需求;同时,基于边缘节点所处环境的多样性及动态变化特征,能自适应快速地做出最优的迁移决策.最后,仿真结果表明,所提出的算法鲁棒性高,不受环境动态变化的影响,能快速获到近似贪心算法的最优迁移决策,且系统总成本得到极大降低.

2 系统模型

2.1 网络模型

本文构建了一个三层的边缘网络计算迁移模型,如图1所示.整个网络的体系结构联合边缘云和云模型进行数据处理,该模型由用户层,边缘云层和云层组成.图1呈现了这些层之间的确切关系,这些层的具体功能在以下部分中定义.

第一层是用户层,假设此层由 N 个用户终端(User Equipment,UE)组成,如智能手机,传感器,AR设备等.这些设备被部署在指定的区域内感知是否有计算任务产生.当终端中需要处理的任务数据超出本地的计算能力,就会向边缘云发送计算请求.

第二层是边缘云层,此层由 M 个边缘节点组成.每个边缘节点有两个角色.(1)生成迁移决策:当边缘节点接收到用户层发来的计算请求时,会结合边缘云中所有节点的带宽和计算资源的分配情况,生成最优的迁移决策,然后将迁移决策发送给对应用户层的用户终端.(2)计算迁移和数据处理:当用户层用户终端接收到对应的迁移决策,它从本地将需要计算的数据发送给对应的边缘节点,边缘节点通过计算再将对应的结果发送回对应的用户终端.

第三层是云层,云层是拥有强大计算能力的服务器,扮演着优化边缘云中迁移决策的角色.边缘云将每个边缘节点中处理过的任务信息发送给云层,云层收集每个时间段内的所有历史信息,模拟用户层与边缘云层的交互过程,生成更优的迁移决策,并将优化后的神经网络参数返回给对应的边缘节点,提升边缘节点对后续任务的处理效率.

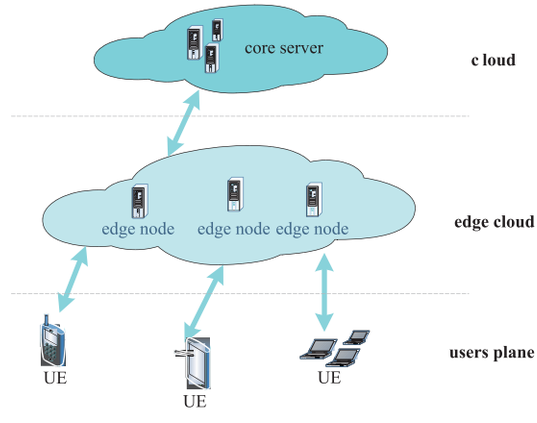

图1 网络模型

2.2 计算模型

基于上一子节的网络模型定义,假设 i 表示第 i 个用户终端,其中 $i\in\{1,2,\cdots,N\}$;j 表示第 j 个边缘节点,其中 $j\in\{0,1,2,\cdots,M\}$,当 $j=0$ 时,特指用户终端本身.每个用户可以将计算任务迁移到指定的边缘节点,边缘节点可为迁移任务分配带宽和计算资源.在共享带宽与计算资源的基础上,本文假设每个任务都是相互独立的执行单元,未考虑任务的相关性问题.当多个用户有任务需要计算时,首先确定任务的最大容许延迟,以及本地计算的时间,若本地计算的时间大于最大容

许延迟,将发送任务迁移请求,把需要计算的任务数据大小 D_i 发送给边缘节点. 边缘节点在边缘云中同步该用户的信息,生成一张如下的任务表格:

$$F_{ij}=\{x_{ij},\lambda_{ij},\beta_{ij},D_i,T_i^{\mathrm{tol}}\},\tag{1}$$

其中 x_{ij} 表示计算任务在本地执行还是被迁移;λ_{ij} 表示边缘节点 j 分配给用户 i 的带宽占比,β_{ij} 表示边缘节点 j 分配给用户 i 的计算资源占比;T_i^{tol} 表示用户终端 i 的最大容许延迟. 边缘节点通过在边缘云下同步更新和维护上述任务表格,此同步只需要在边缘节点每次做出任务迁移决策后,更新表格信息,并广播给同一边缘云下的所有边缘节点. 因此由全部任务生成的表格可以得到总任务集 F.

$$F=\{F_{ij}\mid i\in\{1,2,\cdots,N\},j\in\{0,1,2,\cdots,M\}\}\tag{2}$$

总任务集 F 将会形成于边缘云中的所有边缘节点. 基于总任务集 F,本文可以得到所有任务的迁移决策,从而计算出各个任务相应的处理成本.

(1)本地执行

当本地计算时间小于用户 i 的最大容许延迟时,任务采用本地计算执行(即 $x_{ij}=0,j=0$),任务 i 的本地执行延迟只与本地 CPU 的处理能力有关. 因此任务 i 的本地计算延迟为:

$$\boldsymbol{T}_{ij}^{l}=(1-x_{ij})\frac{D_i}{\beta_{ij}f_i^{l}}\tag{3}$$

其中 β_{ij} 表示本地用户分配给任务 i 的 CPU 占比;f_i^{l} 表示本地用户的计算能力. 因此任务 i 在本地计算时产生的能耗为:

$$E_{ij}^{l}=p_i^{l}\boldsymbol{T}_{ij}^{l}\tag{4}$$

其中 p_i^{l} 表示本地用户 i 的计算功率.

结合本地计算延迟公式(3)和相应的能量消耗公式(4),本地计算的总成本可以表示为:

$$C_{ij}^{l}=\alpha T_{ij}^{l}+(1-\alpha)E_{ij}^{l}\tag{5}$$

其中 α 和 $1-\alpha$ 分别表示任务 i 的时间和能耗成本的权重,$\alpha\in[0,1]$. 由于每个用户的任务权重可能不同,不同的任务应依据任务类型分配不同的权重,当任务为延迟敏感型任务时,适当增加权重 α 的值,当任务为能耗敏感型任务时,适当减少权重 α 的值,具体初始化值可根据大量实验择优选取.

(2)迁移执行

当用户任务 i 的本地计算时间大于其最大容许延迟时,则用户 i 选择通过计算迁移来执行任务,即当 $x_{ij}=1$ 或者 $j\geqslant 1$ 时. 整个迁移过程可以分成 5 个步骤. 首先,如果用户 i 的任务需要迁移,则向边缘云发送数据迁移请求;其次,边缘节点根据请求得出最优的迁移决策并发送给用户 i;随后,用户 i 通过无线接入网络或者蜂窝移动网络向指定的边缘节点 j 上传需要处理的数据,并且由边缘节点 j 向其分配相应的带宽;此后,边缘节点 j 分配相应的计算资源用以处理用户 i 的任务;最后边缘节点 j 将任务的执行结果返回给用户 i.

因为每个用户迁移到不同的边缘节点有不同的上下行链路速率,假设任务 i 迁移到边缘节点 j 的上行链路速率如下所示:

$$V_{ij}^{\mathrm{up}}=\lambda_{ij}B_j\log\left(1+\frac{P_i^{\mathrm{up}}\mid H_i\mid}{N_0\Gamma(g_{\mathrm{up}})d(i,j)^{\xi}}\right)\tag{6}$$

其中 B_j 表示边缘节点 j 的带宽;P_i^{up} 表示用户 i 上传数据的传输功率;H_i 表示用户 i 在无线信道中的信道增益;N_0 表示信道的噪声功率;g_{up} 表示目标误码率,$\Gamma(g_{\mathrm{up}})$ 表示以满足上行链路目标误码率而引入的信噪比余量;$d(i,j)$ 表示用户 i 和边缘节点 j 之间的距离;ξ 表示传输信道路径的损耗指数.

假设下行链路与上行链路具有相同的信道环境与噪声,因此下行链路速率可以表示为:

$$V_{ij}^{\mathrm{do}}=\lambda_{ij}B_j\log\left(1+\frac{P_i^{\mathrm{do}}\mid H_i\mid}{N_0\Gamma(g_{\mathrm{do}})d(i,j)^{\xi}}\right)\tag{7}$$

依据上述步骤,在迁移计算中用户 i 上传计算任务到边缘节点 j 的传输延迟 $T_i^{t_u}$ 可表示为:

$$T_{ij}^{t_u}=x_{ij}\frac{D_i}{V_{ij}^{\mathrm{up}}}\tag{8}$$

根据传输延迟,可以得到相应的传输能耗 $E_{ij}^{t_u}$

$$E_{ij}^{t_u}=p_i^{\mathrm{up}}T_{ij}^{t_u}\tag{9}$$

其中 p_i^{up} 为用户 i 在单位时间内上行链路的传输功率.

当数据传输到边缘节点后,利用边缘节点分配给任务的 CPU 资源来进行计算,可以得到用户 i 的任务在边缘节点 j 上的计算时间 T_{ij}^{e} 为:

$$T_{ij}^{\mathrm{e}}=x_{ij}\left(\frac{D_i}{\beta_{ij}f_j^{\mathrm{e}}}\right)\tag{10}$$

其中 f_j^{e} 表示边缘节点 j 的计算能力,β_{ij} 表示边缘节点 j 分配给用户 i 的计算资源占比.

当边缘节点 j 将用户 i 的任务计算完成后,将计算结果返回用户 i,其中,边缘节点 j 发送结果到用户 i 的传输延迟为:

$$T_{ij}^{t_d}=x_{ij}\left(\frac{D_{ij}^{\mathrm{o}}}{V_{ij}^{\mathrm{do}}}\right)\tag{11}$$

其中 D_{ij}^{o} 为边缘节点 j 返回计算结果的数据大小.

基于边缘节点 j 发送结果到用户 i 的传输延迟,本文可以得到用户 i 相应的接收能耗:

$$E_{ij}^{t_d}=p_i^{\mathrm{do}}T_{ij}^{t_d}\tag{12}$$

其中 p_i^{do} 为用户 i 在单位时间内下行链路的传输功率.

结合式(8)、(10)和(11),可以得到用户 i 的任务迁移到边缘节点 j 的执行过程总延迟为:

$$T_{ij}^{\mathrm{total}}=T_{ij}^{t_u}+T_{ij}^{\mathrm{e}}+T_{ij}^{t_d}\tag{13}$$

因此可以得到用户 i 的任务迁移到边缘节点 j 过程

中,用户 i 本地的等待能耗为:

$$E_{ij}^{\mathrm{w}} = p_i^{\mathrm{w}} T_{ij}^{\mathrm{total}} \tag{14}$$

其中 p_i^{w} 为用户 i 在等待状态过程中设备的功率.

结合式(9)、(12)和(14),可以得到,任务 i 迁移到边缘节点 j 过程中用户端消耗的总能耗为:

$$E_{ij}^{\mathrm{total}} = E_{ij}^{\mathrm{t_u}} + E_{ij}^{\mathrm{t_d}} + E_{ij}^{w} \tag{15}$$

最后结合用户 i 的任务迁移到边缘节点 j 执行过程中的总延迟和总能耗,即式(13)和(15),可以得到迁移过程的总成本为:

$$C_{ij}^{\mathrm{total}} = \alpha T_{ij}^{\mathrm{total}} + (1-\alpha) E_{ij}^{\mathrm{total}} \tag{16}$$

其中 α 和 $1-\alpha$ 分别表示边缘节点处理任务的时间和能耗成本的权重且 $\alpha \in [0,1]$. 与上述相类似,由于每个边缘节点处理任务权重可能不同,权重的具体初始化值可根据大量实验择优选取.

3 优化问题描述

时延与能耗作为度量网络性能的两个核心指标,本文的优化目标主要集中在用户层全部任务执行完成时间和能耗上,具体优化目标即为最小化所有用户的任务执行延迟和能耗的权重和,即总成本 C. 方式即为通过联合优化迁移决策、带宽分配和计算资源分配来实现,其中任务可在本地执行或者迁移执行,具体优化问题构建如下:

$$\min_{\lambda_{ij},\beta_{ij},z_{ij}} C = \min_{\lambda_{ij},\beta_{ij},z_{ij}} \sum_{j=0}^{M} \sum_{i=1}^{N} z_{ij} \left[(1-x_{ij}) C_{ij}^{l} + x_{ij} C_{ij}^{\mathrm{total}} \right] \tag{17}$$

s. t.

$$(1-x_{ij}) T_{ij}^{\mathrm{l}} + x_{ij} T_{ij}^{\mathrm{total}} \leqslant T_i^{\mathrm{tol}} \tag{17a}$$

$$\sum_{i=1}^{N} z_{ij} \lambda_{ij} \leqslant 1, \forall j \in \{0,1,\cdots,M\} \tag{17b}$$

$$\sum_{i=1}^{N} z_{ij} \beta_{ij} \leqslant 1, \forall j \in \{0,1,\cdots,M\} \tag{17c}$$

$$z_{ij} \in \{0,1\} \tag{17d}$$

上述优化问题中,目标函数(17)即为最小化全部任务完成时间与用户端能耗的权重和,用总成本 C 表示. 约束(17a)表示无论是选择本地计算所产生的延迟还是选择迁移计算产生的延迟都不能大于用户对任务执行所能容忍的最大延迟. 约束(17b)表示节点 j(j 可为本地用户或边缘节点)分配给各个任务的带宽占比和必须小于或等于 1,即迁移到边缘节点的所有用户任务占用的带宽和要小于或等于边缘节点的最大带宽,本地用户的带宽分配也是如此. 相类似,约束(17c)表示所有迁移到边缘节点(或在本地执行)的任务 CPU 占比之和小于或等于 1. 约束(17d)表示变量 z_{ij} 的取值约束,当 $z_{ij}=0$ 表示任务 i 并未选择节点 j 进行计算(j 可为本地用户或边缘节点),当 $z_{ij}=1$ 表示任务 i 选择 j 节点执行计算.

4 ACEC-DRL 算法

上述优化问题属于混合整数规划问题,用常规方法很难进行求解,因此本部分提出了一种基于异步优势行动者-评论家的异步云边协同深度强化学习算法(Asynchronous Cloud-Edge Collaborative Deep Reinforcement Learning,ACEC-DRL). 该 ACEC-DRL 算法采用了异步多线程的方法,同时将边缘云中的每个边缘节点作为一个线程来处理,不同的边缘节点和环境进行交互学习,并且每个边缘节点都把学习的梯度参数发送到云端,定期从云端接收新参数,更好地指导当前边缘节点的和后面的环境进行学习交互. ACEC-DRL 算法在不同的边缘节点上,使用不同的探索策略以保证其探索的多样性,无须采用传统的经验回放机制,并通过各个并行的边缘节点各自收集到的数据样本,进行独立的训练实验,加之与云端交互可达到有效降低样本的相关性.

4.1 ACEC-DRL 算法要素定义

ACEC-DRL 算法模型在每个边缘节点中基于观测与环境相互作用的智能体,通过不断地学习,同时与云端进行交互,从而获到最优的迁移策略. 例如:在某个时隙 t,环境处于状态 s_t,智能体执行动作 a_t,环境可以以某种可能性转移到任何可实现的后续状态 s_{t+1},并且智能体接受奖励 r_t. 智能体的长期目标是通过采取根据其观测结果调整其行动的策略 π 来最大化其获得的累计奖励,具体 ACEC-DRL 算法的三个关键要素,即状态、动作和奖励,定义如下:

状态空间定义:

$$S_t = (C_{ij}(t)) \tag{18}$$

其中 $C_{ij}(t)$ 表示时隙 t 时用户 i 的任务迁移至边缘节点 j 计算的总成本.

动作空间定义:

$$A_t = (z_{ij}(t), \lambda_{ij}(t), \beta_{ij}(t)) \tag{19}$$

其中 $z_{ij}(t)$ 表示 t 时刻用户 i 选择边缘节点 j 进行迁移;$\lambda_{ij}(t)$ 和 $\beta_{ij}(t)$ 分别表示 t 时刻边缘节点 j 分配给用户 i 的带宽和 CPU 占比.

边缘节点智能体将在执行每个可能的动作 a 后,在某个状态 s 中获得奖励值 $R(s,a)$. 因为通常奖励函数与目标函数相关,本文优化问题的目标是最小化所有用户的任务执行延迟和能耗权重和的总成本,而求解目标是获得最大的奖励函数值,因此本文的奖励函数与总成本的大小是负相关的. 据此,奖励函数定义为:

$$r_t^{ij} = \begin{cases} r_{t-1}^{ij} + v, C_{ij}(t) < C_{ij}(t-1) \\ r_{t-1}^{ij}, C_{ij}(t) = C_{ij}(t-1) \\ r_{t-1}^{ij} - v, C_{ij}(t) > C_{ij}(t-1) \end{cases} \tag{20}$$

其中 v 为具体环境决定的奖励值.

4.2 ACEC-DRL 算法

本文所提出 ACEC-DRL 算法的具体执行流程如图 2所示. 当边缘云中的边缘节点接收到所有来自用户层终端的迁移请求时,通过设计两个深度神经网络来分别近似估计边缘节点中智能体的策略函数和价值函数,其中前者用于估计一组动作的输出概率,后者用于判断某状态的好坏程度,通过边缘节点中智能体的执行,将最优的迁移策略返回给用户终端去执行. 每隔一段时间,将边缘节点中的神经网络参数梯度和缓存数据传递给云端,云端通过学习,将优化后的参数返回给每个边缘节点去优化指导智能体的执行,从而能够实现任务迁移知识的长期优化服务.

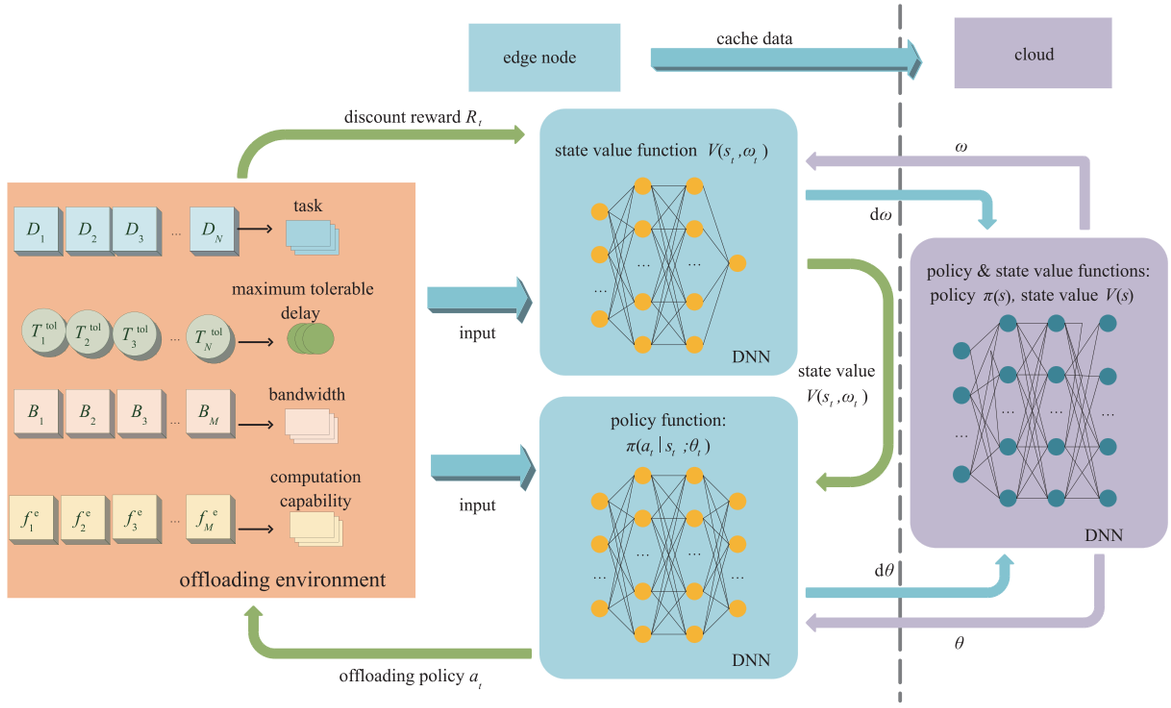

图2　ACEC-DRL算法流程

(1)迁移策略函数

在基于策略函数的深度神经网络中,深度神经网络使得智能体能够根据输入数据的不同感知到数据的本质并进行深度特征建模,为后续智能体对环境进行决策和控制提供更为坚实的基础,进而能够更好地解决复杂系统的感知决策问题. 为了将边缘节点智能体中的各种观测结果融入迁移策略制定的服务中,定义策略函数为:

$$\pi(s_t,a_t)\approx\pi(a_t|s_t;\theta_t) \tag{21}$$

其中 θ 为使用策略迭代更新网络中的权重参数. 由于策略函数的目标是最大化奖励,因此可以使用梯度上升算法计算关于奖励的期望 $\mathrm{E}[R_t]$. 策略梯度的更新公式为:

$$\nabla_\theta \mathrm{E}[R_t]=\nabla_\theta \log\pi(a_t|s_t;\theta_t)R_t \tag{22}$$

式(22)表示奖励期望越高的动作,应该提高其概率. 其中 $\pi(a_t|s_t;\theta_t)$ 表示在状态 s_t 下选择动作 a_t 的概率; $\nabla_\theta \log\pi(a_t|s_t;\theta_t)R_t$ 为对$\nabla_\theta \mathrm{E}[R_t]$的无偏估计.

实际上,假设每个动作的奖励值 R_t 均为正时(即所有的梯度值均大于或等于零),每个动作出现的概率将会随着梯度上升算法不断地被提高,上述操作很大程度上会减缓学习速率,同时使得梯度方差增大. 因此本部分对等式(22)增加标准化操作用于降低梯度的方差,则等式(22)可更新为:

$$\nabla_\theta \mathrm{E}[R_t]=\nabla_\theta \log\pi(a_t|s_t;\theta_t)(R_t-b_t(s_t)) \tag{23}$$

其中通过奖励值 R_t 减去基线函数 $b_t(s_t)$ 的方式学习策略函数,可以减小该估计的方差,保持其无偏性. 将基线函数 b_t 设为奖励值 R_t 的期望估计,通过求其梯度更新参数 θ,当总奖励超过基线动作,其概率会被提高,反之降低,同时还可以降低梯度方差.

(2)迁移策略评估目标函数

在算法流程中基于价值函数的深度神经网络中,通过边缘节点的智能体观测到任务迁移到边缘节点的映射,观测包括边缘节点的计算能力,任务的数据大小,以及任务的最大容许延迟,并且可以定义本文的动作值 Q 函数:

$$Q(s_t,a_t)\approx Q(s_t,a_t;w_t) \tag{24}$$

其中 w 为本文的权重参数.

因此,基于价值函数的深度神经网络,损失函数定义为:

$$L(w_t)=\mathrm{E}[(\text{Target } Q-Q(s_t,a_t;w_{t-1}))^2] \quad (25)$$

本部分采用多步 Q-learning 算法中的目标动作 Q 值定义方法,其优点在于一个奖励 r 可以直接影响先前 n 个 < 状态-动作 > 对,能更好地模拟迁移执行的历史经验,明显提高算法学习的有效性. 多步 Q-learning 算法中的多步是指包括计算后续 n 步的状态,因此本文定义 $Q(s,a)$ 即 Target Q 为:

$$\text{Target } Q=r_t+\gamma r_{t+1}+\cdots+\gamma^{n-1}r_{t+n-1}+\max_a\gamma^n Q(s_{t+n},a) \quad (26)$$

在 ACEC-DRL 算法中,基于策略函数的深度神经网络和基于价值函数的深度神经网络是相同的,只是同一个网络模型的不同输出流.

(3)云-边优势函数

在 ACEC-DRL 算法中,将策略函数 π 作为行动者,将基线函数 $b_t(s_t)$ 作为评论家. 云-边优势函数基于行动者-评论家算法的损失函数,结合竞争神经网络的特性[18,19],并根据迁移场景做出调整和优化,以更好地根据奖励对动作值进行估计. 在策略梯度更新的过程中,更新规则使用了折扣奖励 R_t 用于通知边缘节点哪些迁移决策是'好的',哪些迁移决策是'不好'的. 接着,进行网络更新,以确定该迁移决策的好坏程度. 定义云-边动作优势函数:

$$A(s_t,a_t)=Q(s_t,a_t)-V(s_t) \quad (27)$$

其中状态值函数 $V(s_t)$ 是在时间步 t 的状态下,所有动作值函数关于动作概率的期望;而动作值函数 $Q(s_t,a_t)$ 是单个动作所对应的价值,因此等式(27)中 $Q(s_t,a_t)-V(s_t)$ 能评价当前动作值函数相对于平均值的大小. 由于迁移策略评估目标函数中不能直接确定动作值 Q,而使用折扣奖励 R 作为动作值 Q 的估计值,最终定义云-边优势函数为:

$$A(s_t,a_t)=R(s_t,a_t)-V(s_t) \quad (28)$$

使用云-边优势函数进行估计,其好处是折扣奖励能够使边缘节点评估该迁移决策的好坏程度,并对执行该迁移决策相比于预期的结果进行评估.

将上述三个部分内容融合到 ACEC-DRL 算法框架中,可得到相应任务的最优迁移决策,返回给用户层相应的终端去执行. 除此之外,并将边缘节点上反馈的动作值函数累计梯度和策略函数累计梯度推送到云端,云端根据各个边缘节点的反馈来更新网络参数,并将新的网络参数下发到各个边缘节点,不断循环,直到学习出理想的网络参数为止. 为便于理解上述算法执行流程,将此求解过程组织成如下算法 1 的形式.

算法 1 异步云边协同的深度强化学习算法

Input:
Task size: $D_i, i\in\{1,2,\cdots,N\}$;
Maximum tolerable delay: $T_i^{\text{tol}}, i\in\{1,2,\cdots,N\}$;
Bandwidth: $B_j, j\in\{0,1,2,\cdots,M\}$;
Computation capability: $f_j^e, j\in\{0,1,2,\cdots,M\}$.
Output:
Optimal value C^* with λ_{ij}^*, β_{ij}^* and z_{ij}^*.
Begin
Initialize DNN weight parameters $\theta,\theta',\omega,\omega'$ and maximum number of iterations T in each edge node;
Set $t=1$;
For each edge node **do**
  Set $t_0=t$;
  Synchronize parameters in the edge node: $\theta'=\theta,\omega'=\omega$;
  Repeat
    Select action based on policy $\pi(a_t|s_t;\theta')$;
    Record the reward r_t and the new state s_{t+1} obtained by executing the action a_t;
    $t=t+1$;
  Until $t-t_0==T$
  For $h=t-1$ to t_0 **do**
    Optimize reward value by equation (23);
    Calculate Q value according to loss function (25) and Target Q (26);
    Combine with Q value, to calculate $R=r_h+\gamma R$;
    Update accumulate gradient $\mathrm{d}\theta$ according to:
    $\mathrm{d}\theta=\mathrm{d}\theta+\nabla_{\theta'}\log\pi(a_h|s_h;\theta')(R-V(s_h;\omega'))$;
    Update accumulate gradient $\mathrm{d}\omega$ according to:
    $\mathrm{d}\omega=\mathrm{d}\omega+\partial(R-V(s_h;\omega'))^2/\partial\omega'$;
  End for
  Update the DNN parameters of the cloud control center:
  $\theta=\theta-\rho_1\mathrm{d}\theta,\omega=\omega-\rho_2\mathrm{d}\omega$;
  Forward DNN parameters θ,ω to cloud control center;
End For
Obtain the optimal offloading policy π^* and return it to the corresponding UE;
Obtain the optimal value C^* with λ_{ij}^*, β_{ij}^* and z_{ij}^*.
End

5 仿真结果与分析

本节执行了一系列仿真,来评估本文所提出的异步云边协同深度强化学习算法性能,并且与现有的三种常用迁移方法以及深度强化学习的代表性方法即深度 Q 网络(Deep Q-Network, DQN)求解方案[12]进行对比. 在本文的仿真场景中,假设 3 个边缘节点,边缘节点的带宽分别为 100MHz、150MHz、200MHz;边缘节点的计算能力分别为 150Mb/s、100Mb/s、200Mb/s;边缘节点单位时间的计算能耗分别为 0.002J、0.003J、0.001J. 同时,假设用户终端数量为 20,表示有 20 个用

户终端有任务需要计算,每个用户终端的任务数据大小在100Mb和500Mb之间随机生成,每个用户终端与边缘节点的距离也是随机生成的. 进一步,假设用户终端的本地计算能力f_j^l为30Mb/s,用户终端本地单位时间的计算能耗为0.02J,为了方便计算,用户终端单位时间的迁移能耗都为0.01J,用户终端单位时间的等待能耗为0.001J. 假设边缘节点j分配给用户终端i的带宽占比$\lambda_{ij}(t)$和计算资源占比$\beta_{ij}(t)$都为0.01,同时,假设用户终端i本地CPU的占比$\beta_{i0}(t)$为0.6.

图3描述了ACEC-DRL云端深度神经网络损失函数的收敛性能,从图中可以发现损失函数的值在前200次迭代急剧下降,然后在600次迭代内基本达到稳定值. 这主要因为在一开始执行的动作对于奖励值影响较大,所以损失函数值会急剧下降,接着随着迭代次数的增加,逐步缓慢逼近最优值,最终会学习到最优的神经网络参数.

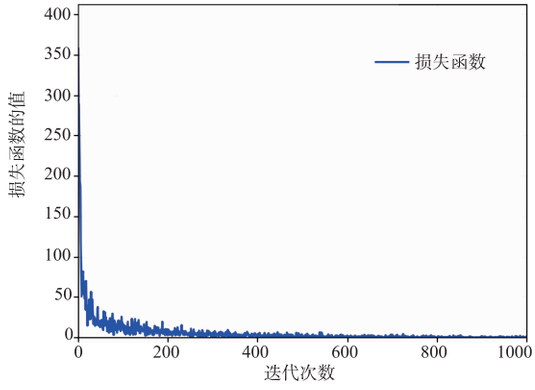

图3　损失函数的收敛过程

图4展现了云端深度神经网络在接收到边缘节点的反馈后优势函数的收敛性能,从图中可以发现每一次接收到新的边缘节点参数梯度后,优势函数在300次左右的迭代达到稳定值. 因此验证了云端深度神经网络模型每一次接收到边缘节点的参数梯度反馈后,能在有限的迭代次数内达到优势函数收敛,从而学习出理想的网络参数.

图5评估了在云端深度神经网络中不同的学习率对于奖励值的影响,从图中可以发现:(1)随着学习率的降低,奖励值的收敛逐渐缓慢,这是因为学习率过小,从而每次迭代优化的效率过低,所以云端深度神经网络中的学习率不能过低;(2)当学习率越大时,随着迭代次数的增加,可能会越过最优值,从而造成在最优值附近震荡. 因此云端深度神经网络中的学习率既不能太低,亦不能太高. 依据多次仿真的结果,本文最后选择的学习率为0.001.

图6表示了在云端深度神经网络中不同的折扣率对于奖励值的影响. 从图中可以发现折扣率越大,奖励值收敛的越快,即可更快的收敛到最大奖励值,其中折扣率表示的是深度神经网络学习过程中考虑未来的期望奖励更多还是考虑当下的期望奖励更多. 由于本文需要考虑未来更多任务的迁移情况,因此在进行了多次不同折扣率的仿真后,最后选择的云端深度神经网络折扣率为0.9.

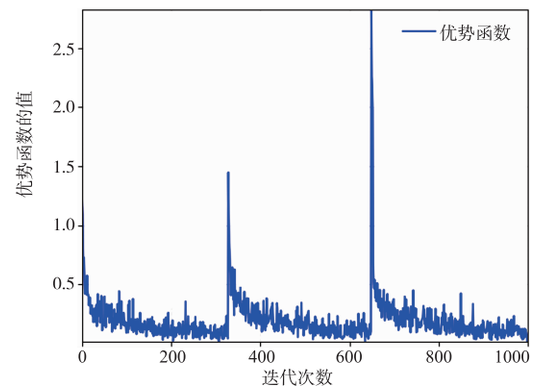

图4　优势函数的收敛过程

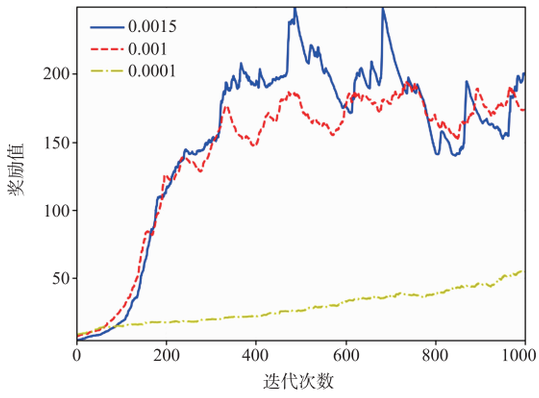

图5　不同学习率下奖励值收敛情况响

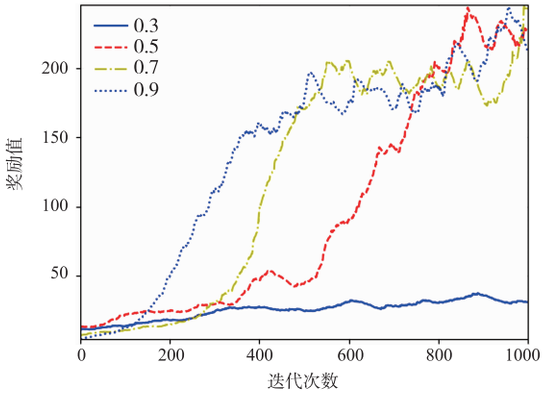

图6　不同折扣率下奖励值收敛情况

图7描述了边缘节点与云端深度神经网络相隔不同次数,任务进行同步参数对于奖励值的影响. 从图中可以发现:(1)一开始随着间隔任务次数的增大,最优

奖励值也越大;(2)当同步任务次数为 40 或 50 时,其最优奖励值反而在减小. 因此可以得到,并不是相隔任务次数越多奖励值就一直越大,而是由实际的环境所决定. 因此在本文中,经过多次仿真结果的分析,云端深度神经网络选择相隔 30 次任务再与边缘节点同步参数,可使获得的最优奖励值最大.

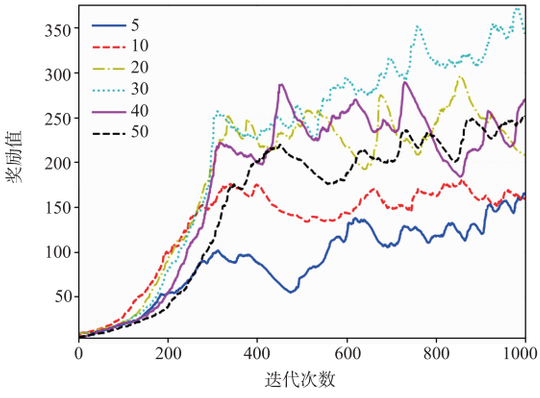

图7 不同云端同步次数下奖励值收敛情况

在全部迁移计算、本地计算、贪心算法、DQN 算法和本文提出的 ACEC-DRL 算法等五种不同算法下,图 8 呈现了总成本与计算任务数量的关系. 从图中可以发现:(1)随着计算任务数量的增加,所有方法的计算成本都不断增加;(2)一开始任务量比较小时,全部本地计算、贪心算法、DQN 算法和 ACEC-DRL 算法成本相同,这是因为当计算任务量没有超过本地计算能力时,都会选择全部本地计算;(3)整体上,全部迁移的成本最高,本地计算其次,再者是 DQN 算法,而本文提出的 ACEC-DRL 算法最逼近贪心算法的总成本. 因为贪心算法获得最优迁移决策过程的时间和能耗远比本文提出的 ACEC-DRL 算法要高的多,不符合实际应用,特别不适用于复杂大数据场景,所以本文提出的 ACEC-DRL 算法具有极大的性能优越性.

图 9 描绘了在五种不同算法下,总成本与边缘节点计算能力的关系. 从图中可以发现:(1)随着边缘节点计算能力的增加,本地计算的成本是不受影响的,因为本地计算与边缘节点的计算能力无关;(2)随着边缘节点计算能力的增加,全部迁移计算的成本逐渐减小,到达一定的计算能力时,则小于本地计算的成本,因为随着边缘节点计算能力的增长,其迁移计算的时间在减少,导致其总成本降低;(3)本文提出的 ACEC-DRL 算法更优于 DQN 算法且总成本十分逼近贪心算法的总成本,相比于贪心算法在大数据处理方面的局限性和高成本,ACEC-DRL 算法能实现低延迟和低能耗地处理大数据,因此,可以得到本文提出的 ACEC-DRL 算法优于其他四种方法.

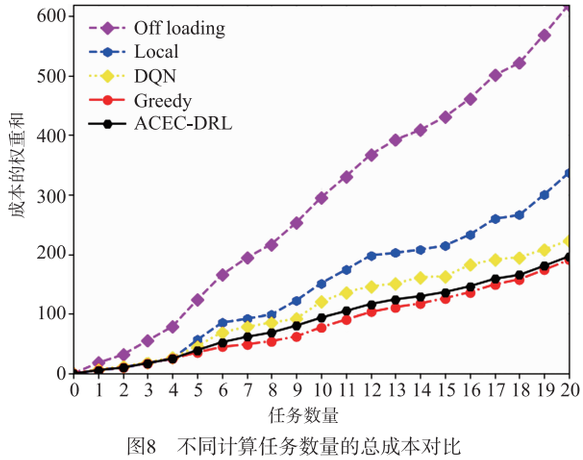

图8 不同计算任务数量的总成本对比

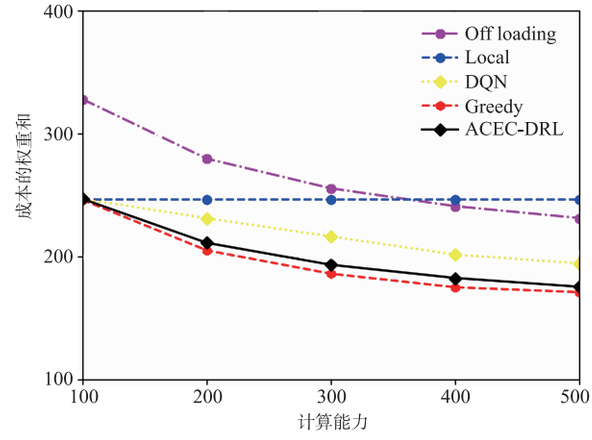

图9 边缘节点不同计算能力的总成本对比

综合上述结果,本文提出的 ACEC-DRL 算法在任务量不同以及边缘节点计算能力不同的场景中都拥有较好的鲁棒性,不受环境动态变化的影响,能依据接收到的任务,快速自适应地做出最优的迁移决策,并使得所有用户任务执行的延迟和能耗总成本最小.

6 总结

基于边缘计算迁移中计算资源、带宽和迁移决策的综合考量,本文构建了一个用户任务执行延迟和能耗权重和的最小化问题,并提出了异步云边协同的深度强化学习算法用于求解该优化问题. 通过联合云端强大的计算能力,可满足大数据场景对高效计算服务的需求;同时,在动态变化的环境中亦能快速地得到最优的迁移决策. 最后,通过仿真结果证实了所提出的 ACEC-DRL 算法的鲁棒性、有效性,同时,与本地计算、全部迁移计算和 DQN 方法相比较,该算法能够实现近似贪心算法的最优性能. 在未来的研究工作中,将把节点移动性等环境时变因素纳入考量,研究更具一般性的计算迁移方案.

参考文献

[1] Wang C, Liang C, Yu F R, et al. Computation offloading and resource allocation in wireless cellular networks with mobile edge computing[J]. IEEE Transactions on Wireless Communications, 2017, 16(8): 4924 – 4938.

[2] Haber E E, Nguyen T M, Assi C. Joint optimization of computational cost and devices energy for task offloading in multi-tier edge-clouds[J]. IEEE Transactions on Communications, 2019, 67(5): 3407 – 3421.

[3] Chen S, Zheng Y, Lu W, et al. Energy-optimal dynamic computation offloading for industrial IoT in fog computing [J]. IEEE Transactions on Green Communications and Networking, 2019, DOI: 10. 1109/TGCN. 2019. 2960767.

[4] Chen S, Zheng Y, Wang K, et al. Delay guaranteed energy-efficient computation offloading for industrial IoT in fog computing[A]. Proceedings of IEEE International Conference on Communications (ICC)[C]. Shanghai, China: IEEE, 2019. 1 – 6.

[5] Lei L, Xu H, Xiong X, et al. Joint computation offloading and multiuser scheduling using approximate dynamic programming in NB-IoT edge computing system[J]. IEEE Internet of Things Journal, 2019, 6(3): 5345 – 5362.

[6] Chen S, Zhu X, Zhang H, et al. Efficient privacy preserving data collection and computation offloading for fog-assisted IoT[J]. IEEE Transactions on Sustainable Computing, 2020, DOI: 10. 1109/TSUSC. 2020. 2968589.

[7] Yu S, Wang X, Langar R. Computation offloading for mobile edge computing: a deep learning approach[A]. Proceedings of IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)[C]. Montreal, Canada: IEEE, 2017. 1 – 6.

[8] Li L, Ota K, Dong M. Deep learning for smart industry: efficient manufacture inspection system with fog computing [J]. IEEE Transactions on Industrial Informatics, 2018, 14 (10): 4665 – 4673.

[9] Zhu X, Chen S, Chen S, et al. Energy and delay co-aware computation offloading with deep learning in fog computing networks[A]. Proceedings of IEEE International Performance Computing and Communications Conference (IPCCC)[C]. London, UK: IEEE, 2019. 1 – 6.

[10] Dab B, Aitsaadi N, Langar R. Q-learning algorithm for joint computation offloading and resource allocation in edge cloud[A]. Proceedings of IFIP/IEEE Symposium on Integrated Network and Service Management (IM) [C]. Arlington, USA: IEEE, 2019. 8 – 12.

[11] Liu X, Qin Z, Gao Y. Resource allocation for edge computing in IoT networks via reinforcement learning[A]. Proceedings of IEEE International Conference on Communications (ICC)[C]. Shanghai, China: IEEE, 2019. 20 – 24.

[12] Huang L, Feng X, Qian L, et al. Deep reinforcement learning-based task offloading and resource allocation for mobile edge computing[A]. Proceedings of International Conference on Machine Learning and Intelligent Communications (MLICOM)[C]. Nanjing, China: Springer Verlag, 2018. 33 – 42.

[13] Meng H, Chao D, Huo R, et al. Deep reinforcement learning based delay-sensitive task scheduling and resource management algorithm for multi-user mobile-edge computing systems[A]. Proceedings of International Conference on Mathematics and Artificial Intelligence (ICMAI)[C]. Chengdu, China: ACM, 2019. 66 – 70.

[14] Wei Y, Yu F R, Song M, et al. Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor-critic deep reinforcement learning[J]. IEEE Internet of Things Journal, 2019, 6(2): 2061 – 2073.

[15] Chen X, Zhang H, Wu C, et al. Performance optimization in mobile-edge computing via deep reinforcement learning [A]. Proceedings of IEEE Vehicular Technology Conference (VTC-Fall)[C]. Chicago, USA: IEEE, 2018. 27 – 30.

[16] Huang L, Bi S, Zhang Y J. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks[J]. IEEE Transactions on Mobile Computing, 2019, DOI: 10. 1109/TMC. 2019. 2928811.

[17] Mnih V, Badia A P, Mirza L, et al. Asynchronous methods for deep reinforcement learning[A]. Proceedings of the 33rd International Conference on Machine Learning (ICML)[C]. New York, USA: IMLS, 2016. 1928 – 1937.

[18] Zeinali Y, Story B A. Competitive probabilistic neural network[J]. Integrated Computer-Aided Engineering, 2017, 24(2): 105 – 118.

[19] Kaing D, Medsker L. Competitive hybrid ensemble using neural network and decision tree[J]. Advances in Intelligent Systems and Computing, 2018, 648(28): 147 – 155.

作者简介

陈思光(通信作者)　男,1984 年生于江西上饶. 现为南京邮电大学副教授、硕士生导师. 主要研究方向为雾/边缘计算、深度/强化学习、物联网通信安全、大数据处理与隐私保护和网络资源分配优化等.

E-mail: sgchen@ njupt. edu. cn

陈佳民 男,1995 年 12 月出生于江苏省南通市. 南京邮电大学在读硕士研究生. 主要研究方向为边缘计算和机器学习.
E-mail:18252010292@163.com

赵传信 男,1977 年生于安徽凤阳. 现为安徽师范大学教授、博士生导师. 主要研究方向为物联网、边缘计算及网络资源分配优化等.
E-mail:zhaocx@ahnu.edu.cn