控制与决策
Control and Decision

# 基于强化学习的边缘计算网络资源在线分配方法

李燕君, 蒋华同, 高美惠

引用本文:
李燕君,蒋华同,高美惠. 基于强化学习的边缘计算网络资源在线分配方法[J]. 控制与决策, 2022, 37(11): 2880–2886.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2021.0561

---

# 基于强化学习的边缘计算网络资源在线分配方法

李燕君†, 蒋华同, 高美惠

(浙江工业大学 计算机科学与技术学院,杭州 310023)

**摘 要:** 针对边缘计算应用对实时性的要求,引入软件定义网络和网络功能虚拟化技术对边缘计算网络进行重构. 基于此,考虑以最大化长期平均实时任务处理成功率为目标的计算和通信资源在线分配问题. 通过建立马尔可夫决策过程模型,提出基于$Q$学习的资源在线分配方法. $Q$学习在状态动作空间较大时内存占用大且会发生维度灾难,鉴于此,进一步提出基于DQN的资源在线分配方法. 实验结果表明,所提出算法能够较快收敛,且DQN算法相较于$Q$学习和其他基准方法能够获得更高的实时任务处理成功率.

**关键词:** 边缘计算; 资源分配; 实时任务; 马尔可夫决策过程; $Q$学习; 深度强化学习

**中图分类号:** TP273 **文献标志码:** A

**DOI:** 10.13195/j.kzyjc.2021.0561 **开放科学(资源服务)标识码(OSID):**

**引用格式:** 李燕君, 蒋华同, 高美惠. 基于强化学习的边缘计算网络资源在线分配方法 [J]. 控制与决策, 2022, 37(11): 2880-2886.

# Reinforcement learning-based online resource allocation for edge computing network

*LI Yan-jun†, JIANG Hua-tong, GAO Mei-hui*

(School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)

**Abstract:** To meet the real-time requirement of the edge computing applications, technologies of software defined network and network function virtualization are introduced to reconstruct the edge computing network. On this basis, we consider the design of online computing and communication resource allocation method, aiming at maximizing the longterm average probability of successfully processing the real-time tasks. By establishing a Markov decision process framework, an online resource allocation method based on $Q$-learning is proposed. Nevertheless, $Q$-learning occupies a lot of memory when the state action space is large, and it is prone to dimensional disasters. Therefore, a DQN-based online resource allocation method is proposed. Simulation results show that both proposed algorithms converge quickly and the average probability of successfully processing the real-time tasks achieved by the DQN algorithm is the highest among all the baseline algorithms.

**Keywords:** edge computing; resource allocation; real-time task; Markov decision process; $Q$-learning; deep reinforcement learning

## 0 引 言

随着通信和计算技术的发展,联网设备的数量正在以前所未有的速度增长. 云计算作为一种具有大规模数据存储和处理能力的技术备受关注. 全球知名IT公司均已建立云数据中心,但是云计算仍然存在未解决的技术挑战,如端到端延迟大、主干网络流量拥塞、通信成本高等问题. 边缘计算作为一种新兴计算模式,将计算、通信以及存储功能扩展到网络边缘[1],能够有效克服云计算的技术缺陷. 根据国际数据公司IDC预测,截至2023年,企业部署的新IT基础设施将有超过50%位于边缘而不是企业数据中心[2]. 边缘计算在移动网络的边缘、无线接入网络内以及紧邻移动用户的位置提供IT服务环境和计算功能,其目的是减少等待时延,确保高效的网络运营和服务交付,提供更好的用户体验[3]. 然而,多数联网设备不具备处理复杂数据的能力,不能在动态网络环境中作出决策. 软件定义网络(software defined network, SDN)提供了一种灵活的方式重构网络,其主要思想是从物理层面分离控制平面和数据平面. 因此,联网设备不需要作任务决策,而是受中央控制器的指令控

**收稿日期:** 2021-04-05; **录用日期:** 2021-07-19.

**基金项目:** 国家自然科学基金项目(61772472); 浙江省自然科学基金项目(LZ21F020005); 浙江省属高校基本科研业务费专项资金项目(RF-A2019002).

†通讯作者. E-mail: yjli@zjut.edu.cn.

制[4]. 得益于对网络信息的完全了解, SDN 控制器能够为网络边缘的联网设备制定合适的资源分配策略. 但是, 在 SDN 中基站负载过高. 网络功能虚拟化(network function virtualization, NFV)技术可以通过虚拟化层将网络功能软件实现计算、存储和网络资源分离[5]降低基站负载. 如图 1 所示, 在 NFV 基础设施(NFV infrastructure, NFVI)上可以部署硬件和软件实现特定功能, 如虚拟交换机、虚拟无线接入点和虚拟边缘服务器.

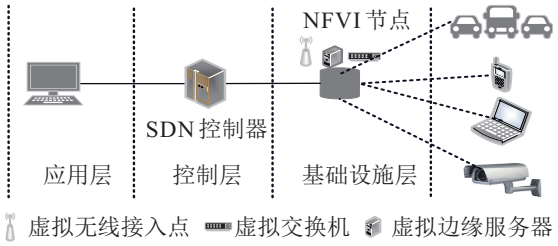



**图 1　基于 SDN 和 NFV 技术的系统结构**

边缘计算主要应用于增强现实、虚拟现实、医疗和自动驾驶[6-7]. 这些应用均存在不同程度的实时性要求, 如增强现实和虚拟现实需要在一定时延约束内将视觉和听觉信息传送至用户, 信息传递超时会造成卡顿, 降低用户体验; 自动驾驶依赖于传感器数据的实时处理, 其实时性与其他应用相比更为苛刻, 延时的指令可能会引起车祸. 目前, 大量研究针对时延进行了优化[8-12]. Lyu 等[8]提出在时延约束条件下优化边缘计算网络的能量消耗, 将该问题归纳为混合整数规划问题, 进一步转化为整数规划问题并利用动态规划算法求解, 验证了其相较于分支定界法的优越性. 但是, 在将混合整数规划问题转化为整数规划问题的过程中, 涉及对系统运行情况作出诸多前提假设, 不符合实际网络运行场景. Wang 等[9]设计了一种时延敏感型微服务部署系统, 涉及通信时延、计算时延和数据迁移时延, 提出最短路径算法和 $Q$ 学习算法分别在离线和在线条件下降低时延, 但是, 离线算法需要的先验信息一般很难获知, 而 $Q$ 学习在系统状态复杂的情况下也会出现性能下降的情况. Zhang 等[10]在车辆边缘计算网络环境下利用路边单元和远程云数据中心协同处理车辆边缘计算任务, 针对计算任务在不同位置处理时延不同的特点, 利用博弈论思想设计了一种在线学习算法, 旨在最小化车辆运行过程中收到任务结果的时延. Fu 等[11]提出李雅普诺夫随机优化方法在长期能耗和资源约束下优化由数据传输时延和本地计算时延组成的用户服务时延. Samanta 等[12]将处理时间超过时延约束的计算任务视为处理失败, 提出算法降低任务处理失败的长期平均概率. 上述研究均只考虑了边缘计算网络中的计算卸载, 没有涉及资源分配问题. 此外, 还有一些研究针对边缘计算网络的其他性能进行优化, 如能耗、公平性和吞吐量等[13-17].

本文针对边缘计算应用对实时性的要求, 引入 SDN 和 NFV 技术对边缘计算网络进行重构, 考虑动态网络环境和联网设备的动态任务数据量, 归纳以实时任务处理成功概率最大化为目标的资源分配问题, 建立马尔可夫决策过程(Markov decision process, MDP), 提出基于 $Q$ 学习($Q$-learning, QL)的资源在线分配方法. 由于 $Q$ 学习在状态动作空间较大时内存占用大且会发生维度灾难, 进一步提出基于 DQN(deep $Q$ network)的资源在线分配方法. 算法部署在 SDN 控制节点中, 为联网设备无计算能力的一类边缘计算网络的计算和通信资源在线优化分配提供解决方案.

## 1　系统模型和问题归纳

### 1.1　系统结构

如图 1 所示, 系统分为应用层、控制层和基础设施层. 应用层由管理终端构成, 通过控制层提供的北向编程接口对下层设备进行控制编程, 运营商根据接口设计网络运营规则, 为联网设备开发各种业务和应用. 控制层主要由 SDN 控制器构成, 负责向应用层设备提供接口进行网络控制, 并制定下层设备的资源分配策略, 通过南向接口协议 OpenFlow 实现[18]. 基础设施层由 NFVI 节点构成, 为联网设备虚拟化无线接入点和服务器功能, 提供数据接入和边缘计算服务, 也为上层的 SDN 控制器虚拟化交换机功能, 提供数据转发服务.

### 1.2　系统模型

以车辆边缘计算网络为例, 车辆上的各种传感器即为联网设备. 联网设备感知环境数据交付虚拟边缘服务器处理, 服务器处理后形成实时结果在终端显示器显示. 为了降低控制信息转发造成的时延影响, NFVI 节点通过有线光纤链路与 SDN 控制器关联, 管理终端通过 SDN 控制器收集到的信息管理整个网络. NFVI 节点服务 $N$ 个联网设备, 联网设备周期性产生实时计算任务, 其计算能力有限, 这些任务难以在自身的计算单元处理, 因此, 被直接迁移至虚拟边缘服务器处理. 假设时间被离散化为等长的时隙, 长度为 $\Delta t$. 在时隙 $t(t=1,2,\ldots)$, 联网设备 $i$ 产生实时计算任务 $\boldsymbol{T}_i^{(t)}=\{m_i^{(t)},z_i^{(t)}\}, i=1,2,\ldots,N$. 其中: $m_i^{(t)}$ 为完成任务需要的计算资源, 即 CPU 周期数; $z_i^{(t)}$ 为这项任务包含的数据量, 表征迁移数据所需的通信

量;$m_i^{(t)}$与$z_i^{(t)}$成线性比例关系,有$m_i^{(t)}=cz_i^{(t)}$,$c$为单位数据量所需计算资源. 联网设备通过虚拟无线接入点将自己的任务信息转发给NFVI节点,NFVI节点将任务信息、信道状态以及自身的资源信息通过虚拟交换机转发给SDN控制器. SDN控制器利用收集的网络信息制定控制决策交付给底层设备执行,NFVI节点可以同时服务多个联网设备.

### 1.2.1 通信模型

$h_i^{(t)}$为时隙$t$内联网设备$i$与NFVI节点间信道增益幅值的平方,$h_i^{(t)}$在时隙内保持不变,在时隙间独立同分布. $p_i$为联网设备$i$的发射功率,$\sigma^2$为噪声功率. 则联网设备$i$与NFVI节点间数据传输速率[19]为

$$r_i^{(t)}=\omega_i^{(t)}B\log_2\Big(1+\frac{p_ih_i^{(t)}}{\sigma^2}\Big). \tag{1}$$

其中:$B$为NFVI节点的总带宽资源,$\omega_i^{(t)}$为时隙$t$中NFVI节点分配给联网设备$i$带宽资源的比例,满足$\sum\limits_{i=1}^{N}\omega_i^{(t)}\leqslant 1,\omega_i^{(t)}\geqslant 0$.

### 1.2.2 时延模型

在初始状态,每个联网设备均会选择NFVI节点与之关联,获取数据服务和边缘计算服务. 网络中的联网设备不能自己处理计算任务,所以将计算任务全部迁移至虚拟边缘服务器处理,这时会产生传输时延,即

$$T_{\rm tr}(i,t)=\frac{z_i^{(t)}}{r_i^{(t)}}. \tag{2}$$

当计算任务迁移至虚拟边缘服务器处理时,任务处理时延取决于任务所需资源和实际分配至边缘服务器的资源. 因此在时隙$t$,联网设备$i$的计算任务处理时延可以表示为

$$T_{\rm mec}(i,t)=\frac{m_i^{(t)}}{\varphi_i^{(t)}f^E}. \tag{3}$$

其中:$\varphi_i^{(t)}$为在时隙$t$中分配给联网设备$i$的计算资源占虚拟边缘服务器计算资源的比例,满足$\sum\limits_{i=1}^{N}\varphi_i^{(t)}\leqslant 1,\varphi_i^{(t)}\geqslant 0$;$f^E$为NFVI节点中虚拟边缘服务器的计算能力. 根据通信模型和时延模型,联网设备$i$在时隙$t$处理计算任务产生的总时延为

$$T_{\rm sum}(i,t)=T_{\rm mec}(i,t)+T_{\rm tr}(i,t). \tag{4}$$

在上述任务处理过程中,网络中的控制消息较小,因此,忽略控制消息转发产生的时延.

### 1.3 问题归纳

假设任务的时间约束为一个时隙$\Delta t$,若联网设备的计算任务在$\Delta t$时间内处理完成,则任务处理成功,否则任务处理失败. 希望所有联网设备长期平均任务处理成功概率最大化,因此,归纳优化问题如下:

$$\begin{aligned}&\max\lim_{T\to+\infty}\frac{1}{TN}\sum_{t=1}^{T}\sum_{i=1}^{N}I_i^{(t)}.\\&\text{s.t. }C_1:I_i^{(t)}=\begin{cases}1,\ T_{\rm sum}(i,t)\leqslant\Delta t;\\0,\ T_{\rm sum}(i,t)>\Delta t.\end{cases}\\&\quad C_2:\sum_{i=1}^{N}\omega_i^{(t)}\leqslant 1,\ \omega_i^{(t)}\geqslant 0.\\&\quad C_3:\sum_{i=1}^{N}\varphi_i^{(t)}\leqslant 1,\ \varphi_i^{(t)}\geqslant 0.\\&\quad C_4:i=1,2,\dots,N.\end{aligned} \tag{5}$$

第1个约束条件中,$I_i^{(t)}=1$表示计算任务处理成功,$I_i^{(t)}=0$表示计算任务处理失败;第2个约束条件表示所有联网设备分配的通信资源之和不超过NFVI节点的总带宽;第3个约束条件表示所有联网设备分配的计算资源之和不超过虚拟边缘服务器计算资源. 上述问题是一个在线优化问题,在系统运行过程中状态时变,无法利用静态方法解决. 因此,将该问题建模为MDP,提出在线资源分配方法.

## 2 马尔可夫决策过程

MDP可以表示为一个三元组$\{\boldsymbol{S},\boldsymbol{A},R\}$,$\boldsymbol{S}$为状态空间,$\boldsymbol{A}$为动作空间,$R$为奖励函数,具体说明如下:

1)状态:对系统而言,每个时隙发生改变的状态量是信道增益幅值的平方$h_i^{(t)}$和联网设备产生的计算任务数据量$z_i^{(t)}$. 因此,$t$时隙系统状态可以表示为$s^{(t)}=[h_1^{(t)},h_2^{(t)},\dots,h_N^{(t)},z_1^{(t)},z_2^{(t)},...,z_N^{(t)}]$.

2)动作:影响系统性能的动作是通信资源分配比例$\omega_i^{(t)}$和计算资源分配比例$\varphi_i^{(t)}$. 因此,$t$时隙动作可以表示为$a^{(t)}=[\omega_1^{(t)},\omega_2^{(t)},\dots,\omega_N^{(t)},\varphi_1^{(t)},\varphi_2^{(t)},\dots,\varphi_N^{(t)}]$. 对动作进行离散化处理,令

$$\omega_i\in\{k\Delta\omega|k=0,1,\dots,K_\omega,\ K_\omega\Delta\omega=1\}, \tag{6}$$

$$\varphi_i\in\{k\Delta\varphi|k=0,1,\dots,K_\varphi,\ K_\varphi\Delta\varphi=1\}. \tag{7}$$

其中:$\Delta\omega$为通信资源步长,$\Delta\varphi$为计算资源步长.

3)奖励:在MDP中,智能体在状态$s^{(t)}$下选择某个动作$a^{(t)}$,得到一个奖励$R^{(t+1)}$并进入下一个状态$s^{(t+1)}$. 其中$R^{(t+1)}$为单步奖励,定义奖励为一个时隙处理成功的任务数量,即$R^{(t+1)}=\sum\limits_{i=1}^{N}I_i^{(t)}$.

在每个时隙,智能体与环境均发生了交互. $s^{(t)}\in\boldsymbol{S}$为时隙$t$智能体观察到的环境状态,在此状态下选择一个动作$a^{(t)}\in\boldsymbol{A}$. 在时隙$t+1$,智能体接收到一个数值化的奖励$R^{(t+1)}$并进入下一个状态$s^{(t+1)}$. 环境和智能体共同给出一个序列或轨迹$s^{(0)}$、$a^{(0)}$、$R^{(1)}$、$s^{(1)}$、$a^{(1)}$、$R^{(2)}$、$s^{(2)}$. 由该轨迹可见,奖励$R^{(t+1)}$与问

题优化目标存在差别,所以需要对这样的轨迹进行分幕(episodes). 首先定义终止时间$T$,若到达终止时间,则当前轨迹结束,即一幕结束. 统计一幕中所有时隙内处理成功的任务数量,在了解一幕中生成总任务数量的前提下,最终可以将马尔科夫决策过程的优化目标制定为最大化$T$时间的平均任务处理成功概率,与优化目标一致,即$\dfrac{1}{TN}\sum\limits_{t=1}^{T}\sum\limits_{i=1}^{N}I_i^{(t)}$.

## 3　基于强化学习的资源分配方法

### 3.1　基于QL的资源分配方法

QL是较常用的在线学习方法,其无法处理连续状态MDP,因此,首先需要对状态空间离散化,转化为有限状态MDP.

1) 信道状态离散化. 将信道增益幅值的平方$h_i$划分为$K_h$个等概率区间,区间阈值记作$a_k, k=0,1,\ldots,K_h$,其中$a_0=0, a_{K_h}=+\infty$,且$a_k$满足

$$\int_{a_k}^{a_{k+1}} f_h(h_i)\mathrm{d}h_i=\frac{1}{K_h},\ k=0,1,\ldots,K_h. \tag{8}$$

对于任意$h_i\in[a_k,a_{k+1})$,$h_i$处于状态$k$. 令其等于量化值$H_k,\ k=0,1,\ldots,K_h-1$,$H_k$的表达式为

$$H_k=\boldsymbol{E}[h_i|h_i\in[a_k,a_{k+1})]=\frac{\int_{a_k}^{a_{k+1}} h_i f_h(h_i)\mathrm{d}h_i}{\int_{a_k}^{a_{k+1}} f_h(h_i)\mathrm{d}h_i}. \tag{9}$$

$h_i$的状态空间可表示为$H=\{H_0,H_1,\ldots,H_{K_h-1}\}$.

2) 任务数据量离散化: 定义数据量的状态空间大小为$K_z=(z_{\max}-z_{\min})/\Delta z$. 其中: $z_{\max}$和$z_{\min}$分别为计算任务数据量的最大值和最小值,$\Delta z$为数据量离散化步长. 对于$z_i\in[z_{\min}+k\Delta z, z_{\min}+(k+1)\Delta z), k=0,1,\ldots,K_z-1$,$z_i$处于状态$k$,令其等于量化值$Z_k=\dfrac{(z_{\min}+k\Delta z)+[z_{\max}-(K-k)\Delta z]}{2}$,因此$z_i$的状态空间可表示为$Z=\{Z_0,Z_1,\ldots,Z_{K_z-1}\}$.

QL使用价值函数衡量状态-动作对的价值,即在状态$s$采取动作$a$的价值记作$Q(s,a)$. 一般情况下,在初始状态,所有状态-动作对的价值设置为0,在算法运行过程中,按下式更新每对状态-动作的价值:

$$Q(s^{(t)},a^{(t)})\leftarrow Q(s^{(t)},a^{(t)})+\alpha[R^{(t+1)}+\gamma\max_{a\in\boldsymbol{A}}Q(s^{(t+1)},a)-Q(s^{(t)},a^{(t)})]. \tag{10}$$

其中: $\alpha$为学习率,$\gamma$为折扣率. 算法在运行过程中会建立一张$Q$表存储所有状态-动作对的价值. 希望通过训练在每个时隙都能够从$Q$表中找出一个最优策略$\pi^\star$. QL的训练过程如下.

step 1: 初始化学习率$\alpha$,折扣率$\gamma$,幕长$T$,$Q(s,a)$,$\forall s\in\boldsymbol{S},a\in\boldsymbol{A}$;

step 2: 对每幕循环进行以下操作:

step 3: 对每个时隙,从环境中观测状态$s^{(t)}$;

step 4: 采用$\epsilon$-greedy策略从动作空间$\boldsymbol{A}$选择一个动作$a^{(t)}$;

step 5: 计算奖励$R^{(t+1)}$并观测至下一状态$s^{(t+1)}$;

step 6: 根据式(10)更新$Q(s^{(t)},a^{(t)})$;

step 7: 本幕训练结束.

$\epsilon$-greedy策略以$\epsilon\in(0,1]$的概率随机选择动作,以$1-\epsilon$的概率根据$Q$表选择最佳动作. 在算法的实际部署过程中,可以将训练阶段与运行阶段分离. 当$Q$表经历足够长时间的训练后,运行阶段直接按照$Q$表选择动作.

存储和计算成本分析: 在算法的训练阶段需要存储每对状态-动作的价值并不断更新,这部分的存储成本为$|\boldsymbol{S}||\boldsymbol{A}|$. 此外,算法其他参数消耗的存储成本为$O(1)$. 因此,QL的总存储成本为$|\boldsymbol{S}||\boldsymbol{A}|+O(1)$. 算法的计算开销主要是更新$Q$表,每更新一次$Q$表需要进行5次基本运算,因此,QL的计算成本为$5\epsilon MT$,$\epsilon T$为每幕训练次数,$M$为运行幕数.

### 3.2　基于DQN的资源分配方法

QL在状态和动作空间是有限且维数不高的情况下可以使用$Q$表存储每个状态-动作对的价值. 但是,实际中信道状态和计算任务数据量均是随机的,具有连续性特征,如何设置合适的状态数没有明确标准,且当状态空间维度变大时,建立有限状态空间容易发生维度灾难. 本节提出DQN算法,基于连续状态空间,利用神经网络代替$Q$表实时计算状态-动作对的价值. 为了解决使用非线性网络表示价值函数时出现不稳定的问题,DQN的神经网络由两个相同结构的人工神经网络(artificial neural network, ANN)构成,如图2所示,均包含输入层、隐藏层和输出层. 第1个神经网络称为实时ANN,用以计算当前状态-动作对的估计价值$Q(s^{(t)},a^{(t)},\theta)$,$\theta$为实时ANN的参数,该参数在每次计算当前状态的估计价值后都会更新;第2个神经网络称为延时ANN,用于计算下一状态的价值$Q(s^{(t+1)},a^{(t+1)},\theta')$,而下一状态的价值用以计算目标价值$y_w$,即

$$y_w=R^{(t+1)}+\gamma\max_{a\in\boldsymbol{A}}Q(s^{(t+1)},a,\theta'), \tag{11}$$

其中$\theta'$为延时ANN的参数. $\theta'$在计算价值的过程中保持不变,提高了算法的稳定性,经过一定的训练次数后更新为实时ANN的参数,保证其价值函数的时效. 除了通过构建2个神经网络降低状态间的相关性外,还使用经验回放方法解决相关性和非静态分

布问题. 具体做法是在状态$s^{(t)}$,采用$\epsilon$-greedy策略选取动作$a^{(t)}$,与环境交互后生成状态转移序列$\Delta^{(t)}=(s^{(t)},a^{(t)},R^{(t+1)},s^{(t+1)})$,将其存储在经验池$D$中. 当经验池中存储了一定量的状态转移序列后,神经网络开始从经验池中取出一小批数据($W$个状态转移序列)进行训练. ANN在得到状态-动作对的实时价值和延时价值后,会利用其计算损失函数. 损失函数越小,ANN的精度越高. 损失函数定义为

$$\mathrm{Loss}(\theta)=\frac{1}{W}\sum_{w\in W}(y_w-Q(s^{(t)},a^{(t)},\theta)). \tag{12}$$

在得到损失函数后,通过损失的反向传播机制可以调整神经网络的参数$\theta$. 同时,实时ANN会利用RMSprop优化器不断地降低神经网络的损失函数,提高神经网络的精度.

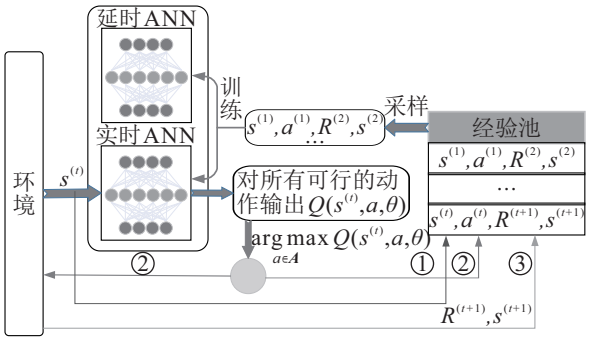



**图2 DQN结构**

DQN的算法具体如下.

step 1: 初始化学习率$\alpha$,折扣率$\gamma$、$\epsilon$,采样样本量$W$,训练间隔$t_s$,经验池大小$|D|$,幕长$T$等;

step 2: 对每幕循环进行以下操作:

step 3: 在每个时隙,观测系统状态$s^{(t)}$;

step 4: 根据$\epsilon$-greedy策略选择动作$a^{(t)}$;

step 5: 计算奖励$R^{(t+1)}$并观测至下一状态$s^{(t+1)}$;

step 6: 存储转移序列$\Delta^{(t)}=(s^{(t)},a^{(t)},R^{(t+1)},s^{(t+1)})$至经验池$D$;

step 7: 每隔$t_s$个时隙进行以下操作;

step 8: 从经验池$D$中采样$W$个转移序列;

step 9: 根据式(11)和(12)得到$y_w$和$\mathrm{Loss}(\theta)$;

step 10: 利用RMSprop优化器优化$\mathrm{Loss}(\theta)$;

step 11: 更新实时ANN的参数,每隔一定步数将延时ANN的参数更新为实时ANN的参数;

step 12: 本幕训练结束.

算法中大部分工作由神经网络完成,本文关注的是平均任务处理成功概率的变化,当其基本收敛时,表明神经网络的精度已被训练得很高.

存储和计算成本分析: DQN的存储成本为$2d(\theta)+|\boldsymbol{A}|+W+|D|+O(1)$. 其中: $d(\theta)$为ANN参数的维度,由输入的状态数决定,存储2个参数相同的ANN的存储成本为$2d(\theta)$; $|\boldsymbol{A}|$为存储所有可行动作的开销; $|D|$为经验池所占存储空间; $W$为从经验池中采样的数据量; $O(1)$为其他参数消耗的存储空间,如学习率$\alpha$、折扣率$\gamma$等. DQN的计算成本与ANN的内部计算方式相关,此处略去分析.

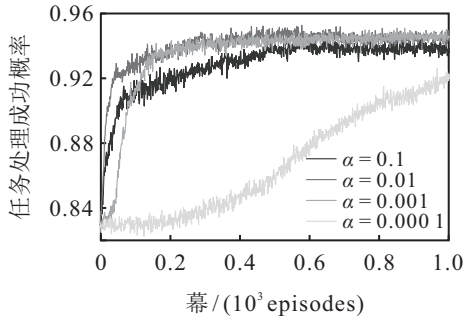
## 4 性能评估

仿真实验平台采用Python 3 + TensorFlow 1.0,硬件条件为Intel Core i5-8265 U, 8 GB内存. 系统中信道增益幅值的平方服从瑞利分布. 虚拟无线接入点将通信资源平均分配给每个联网设备,方法中所用神经网络隐藏层和输出层的神经元均为20个,在$\epsilon$的设置上采用退化$\epsilon$-greedy策略,即在算法运行过程中不断降低$\epsilon$的值(从1开始经过1 000次迭代降低至0.01),使算法能够更快收敛. 如无其他说明,参数的默认值如表1所示.

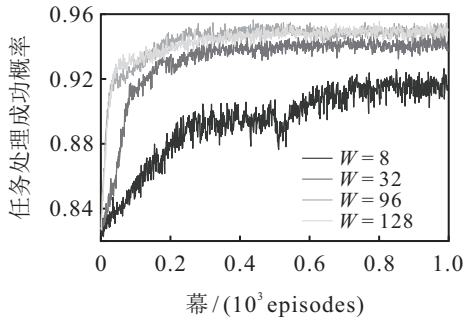
**表1 参数设置**

| 类别 | 符号 | 含义 | 值 |
|---|---|---|---|
| NFVI节点参数 | $B$ | 总带宽/MHz | 10 |
| | $f^E$ | 计算资源/GHz | 9 |
| | $\sigma^2$ | 背景噪声/dBm | −100 |
| 联网设备参数 | $N$ | 数量 | 5 |
| | $p_i$ | 发射功率/mW | 100 |
| | $z_i$ | 计算任务数据量/MB | [0.5,1] |
| | $c$ | 单位数据量所需计算资源/(megacycle/MB) | 100 |
| 算法参数 | $\gamma$ | 折扣率 | 0.9 |
| | $t_s$ | 训练间隔 | 5 |
| | $T$ | 幕长 | 2 000 |
| | $K_h$ | 信道状态离散数量 | 3 |
| | $K_z$ | 数据量状态离散数量 | 5 |
| | $\Delta\varphi$ | 计算资源步长 | 0.05 |
| | $\Delta t$ | 时隙长度/s | 0.08 |

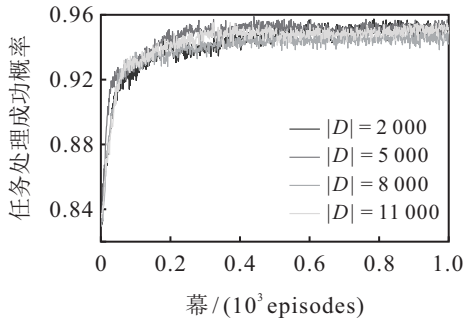
首先研究不同参数对DQN性能的影响,包括学习率、样本数量和经验池大小. 图3(a)为RMSProp优化器中学习率对DQN收敛性能的影响. 由图3(a)可见,当学习率过大(0.1)时,算法的学习速度很快,但是会使系统最终的收敛性能下降; 当学习率过小(0.000 1)时,学习速度将会急剧下降,造成算法难以收敛. 因此,在后续实验中将学习率设置为0.01. 如图3(b)所示,过小的采样数据量($W=8$)没有有效利用存储在经验池中的历史数据,造成系统性能下降. 过大的数据量($W=128$)也无法提高系统性能,反而会因为样本变多耗费更长的计算时间. 考虑时间与性能之间的权衡,在后续实验中将采样数据量设置为96. 由图3(c)可见,经验池大小对系统的性能影响比较有限,总体而言, $|D|=5\,000$时,系统性能提升最快且略高于其他参数设置下的性能. 考虑到设备需要




(a) DQN 在不同学习效率下的性能比较




(b) DQN 在不同采样数据量下的性能比较




(c) DQN 在不同经验池大小下的性能比较

**图 3　DQN 在不同参数下的收敛情况**

为经验池划分内存, 占用设备的存储空间, 在后续实验中将经验池大小设置为 5 000.

图 4 为基于 QL 和 DQN 算法的任务成功处理概率比较. 由图 4 可见, DQN 在开始的几幕内迅速将联网设备的实时计算任务处理成功率提升至 0.92 左右的较高水平, 30 幕以后提升速率降低并以近似线性的速率增长, 250 幕左右收敛至 0.956 上下浮动. 此时, 神经网络模型训练已达到较高精度, DQN 算法可以在每个状态下找到最佳策略. QL 一开始从 0.86 左右线性地提升任务处理成功概率, 100 幕左右将这个概率提升至 0.915 并保持该水平上下浮动. 因此, 所提出的两种算法均能够获得较好的收敛性能, 且 DQN 能够获得更高的成功处理概率.

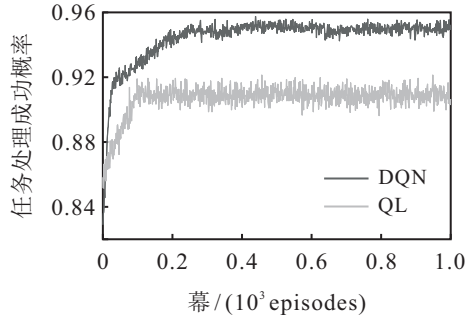



**图 4　DQN 和 QL 的在线优化效果**

除所提出的 QL 和 DQN 算法, 还给出 2 个基准方法作为对比: 1) 随机资源分配方法 (RRA): 将 $\epsilon$ 的值设置为 1, 每当系统遇到新的状态, 便在动作空间中随机选取一个动作执行; 2) 固定资源分配方法 (FRA): 事先为每个联网设备设置固定的资源分配比例, 即 $\varphi_i = 0.1 + 0.05(i-1), i = 1, 2, \ldots, N$. 图 5 为随着单位数据量所需计算资源的增大, 几种方法获得的性能变化情况. 由图 5 可见, 当 $c$ 从 70 megacycle / MB 增大至 120 megacycle / MB 时, 几种算法的性能均随之下降. 这是因为联网设备计算任务占用的计算资源越多, 其处理时延便会越大, 而此时时延约束条件不变, 任务处理成功概率便会下降. 几种算法中, DQN 获得的性能始终最优, 相比 QL、RRA 和 FRA 平均提升了 8.05 %、12.54 % 和 20.60 %, 最高提升了 16.56 %、17.91 % 和 30.60 %. 注意到, QL 算法在不同的 $c$ 下存在波动, 这可能是训练不够充分导致的, 但是由于其具有在线优化能力, 性能始终优于 2 个基准方法.

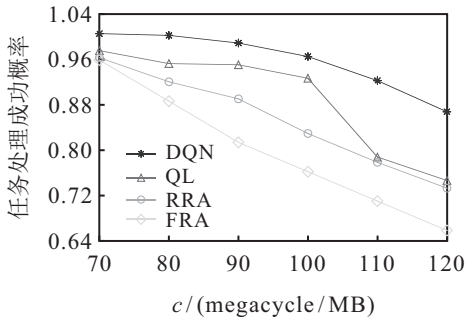



**图 5　几种方法在 $c$ 变化时的性能比较**

图 6 为几种方法在不同时隙长度下任务成功处理概率的变化情况. 由图 6 可见, 随着时隙长度的增大, 所有算法获得的任务处理成功概率均随之增大. 这是因为时延约束即为时隙长度, 当时隙长度较小时时延约束苛刻, 更多任务难以成功处理; 当时隙长度变大时时延约束变得宽松, 更多实时计算任务

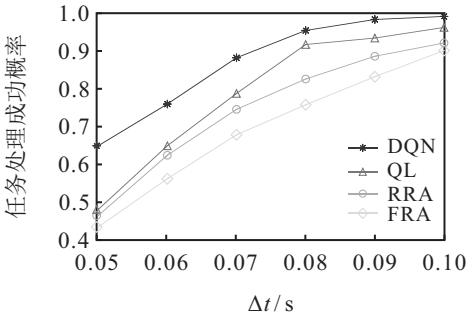



**图 6　几种方法在 $\Delta t$ 变化时的性能比较**

能够成功处理. 几种算法中, DQN获得的性能始终最优, 相比QL、RRA和FRA平均提升了8.12 %、18.83 %和27.90 %, 最高提升了16.80 %、39.32 %和48.85 %.

## 5 结 论

本文针对边缘计算应用对实时性的要求, 引入SDN和NFV技术对边缘计算网络进行重构, 考虑在动态网络环境和联网设备任务数据量动态变化的情况下, 设计计算资源和通信资源的最优在线分配方法. 以往的方法大多考虑计算卸载问题或基于先验知识用静态优化方法得到当前最优分配方案, 无法适用于动态环境下的在线资源分配. 本文以最大化实时任务处理成功率为目标, 利用MDP为问题建模, 提出了基于QL和DQN算法的资源在线分配方法. 实验结果表明, QL和DQN算法均能够较快收敛, 相较于基准算法, DQN在实时任务处理成功率性能上表现最优, 能够为联网设备无计算能力的一类边缘计算网络的计算和通信资源在线优化分配提供有效解决方案. 未来将进一步考虑联网设备具备一定处理能力的场景, 对计算卸载和资源分配进行联合优化.

## 参考文献(References)


[1] Mukherjee M, Shu L, Wang D. Survey of fog computing: Fundamental, network applications, and research challenges[J]. IEEE Communications Surveys & Tutorials, 2018, 20(3): 1826-1857.

[2] Dave McCarthy. Edge computing: Not all edges are created equal[EB/OL]. (2020-06-01)[2021-03-23]. https:// blogs.idc.com/2020/06/01/.

[3] Y C Hu, M Patel, D Sabella, et al. Mobile edge computing — A key technology towards 5 G[J]. ETSI White Paper, 2015, 11(11): 1-16.

[4] Foundation O N. Software-defined networking: The new norm for networks[EB/OL]. (2012-04-13)[2021-03-22]. https: // opennetworking.org /images /stories /downloads/ sdnresources/white-papers/new-openflow-document.pdf.

[5] Chiosi M, Clarke D, Cablelabs P W, et al. Network functions virtualisation network operator perspectives on industry progress[C]. Frankfurt-Germany: SDN and OpenFlow World Congress, 2013: 1-20.

[6] Abbas N, Zhang Y, Taherkordi A, et al. Mobile edge computing: A survey[J]. IEEE Internet of Things Journal, 2018, 5(1): 450-465.

[7] Lyu L L, Yang Z P, Zhang L. Contract theory based task offloading strategy of mobile edge computing[J]. Control and Decision, 2019, 34(11): 2366-2374.

[8] Lyu X C, Tian H, Ni W, et al. Energy-efficient admission of delay-sensitive tasks for mobile edge computing[J]. IEEE Transactions on Communications, 2018, 66(6): 2603-2616.

[9] Wang S G, Guo Y, Zhang N, et al. Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach[J]. IEEE Transactions on Mobile Computing, 2021, 20(3): 939-951.

[10] Zhang J, Guo H Z, Liu J J, et al. Task offloading in vehicular edge computing networks: A load-balancing solution[J]. IEEE Transactions on Vehicular Technology, 2020, 69(2): 2092-2104.

[11] Fu Z M, Wang J P, Si P J, et al. Research on resource management of vehicle edge computing based on Lyapunov stochastic optimization[EB/OL]. (2020-12-07)[2021-03-22]. https:// doi.org/10.13195/j.kzyjc.2020.1211.

[12] Samanta A, Li Y, Esposito F. Battle of microservices: Towards latency-optimal heuristic scheduling for edge computing[C]. IEEE Conference on Network Softwarization. Piscataway: IEEE, 2019: 223-227.

[13] Li Y Z, Wang S G. An energy-aware edge server placement algorithm in mobile edge computing[C]. IEEE International Conference on Edge Computing. Piscataway: IEEE, 2018: 66-73.

[14] Chen J F, Zhao Y S, Gao J C, et al. Resource allocation strategy of mobile edge computing system based on hybrid energy harvesting[J]. Journal of Chongqing University of Posts and Telecommunications, 2021, 33(2): 193-201.

[15] H Badri, T Bahreini, D Grosu, et al. Risk-aware application placement in mobile edge computing systems: A learning-based optimization approach[C]. IEEE International Conference on Edge Computing. Hawaii: IEEE, 2020: 83-90.

[16] Huang Y D, Song X T, Ye F, et al. Fair caching algorithms for peer data sharing in pervasive edge computing environments[C]. IEEE 37th International Conference on Distributed Computing Systems. Piscataway: IEEE, 2017: 605-614.

[17] Deng X M, Li J, Shi L, et al. Wireless powered mobile edge computing: Dynamic resource allocation and throughput maximization[J]. IEEE Transactions on Mobile Computing, DOI: 10.1109/TMC.2020.3034479.

[18] Open Netw Found. Openflow switch specification (version 1.5.1)[EB/OL]. (2015-03-26)[2021-03-22]. https://opennetworking.org/wp-content/uploads/2014/10 /openflow-switch-v1.5.1.pdf.

[19] Liu Y, Yu H M, Xie S L, et al. Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks[J]. IEEE Transactions on Vehicular Technology, 2019, 68(11): 11158-11168.


## 作者简介


李燕君 (1982−), 女, 教授, 博士生导师, 从事物联网领域等研究, E-mail: yjli@zjut.edu.cn;

蒋华同 (1996−), 男, 硕士生, 从事移动边缘计算的研究, E-mail: 248654277@qq.com;

高美惠 (1989−), 女, 讲师, 博士, 从事软件定义网络等研究, E-mail: gaomeihui@zjut.edu.cn.


(责任编辑：魏 冰)