doi:10.3969/j.issn.1003-3114.2020.06.001
引用格式:郑莹,段庆洋,林利祥,等.深度强化学习在典型网络系统中的应用综述[J].无线电通信技术,2020,46(6):603-623.[ZHENG Ying,DUAN Qingyang,LIN Lixiang,et al. A Survey on the Applications of Deep Reinforcement Learning in Classical Networking Systems[J]. Radio Communications Technology,2020,46(6):603-623.]

深度强化学习在典型网络系统中的应用综述

郑　莹[1,3],段庆洋[2],林利祥[2],游新宇[2],徐跃东[2],王　新[1,3*]
(1.复旦大学 计算机科学技术学院,上海 200433;
2.复旦大学 信息科学与工程学院,上海 200433;
3.上海市智能信息处理重点实验室,上海 200433)

摘　要:近几年来,以深度强化学习(Deep Reinforcement Learning,DRL)为代表的人工智能技术被引入计算机网络系统设计中,促使网络领域走向数据驱动和智能化,并在典型的网络系统中不断取得新的突破。计算机网络应用的难点是难以对多变的网络环境进行复杂准确的建模,借助深度神经网络出色的特征提取能力,深度强化学习能够更好地以试错的方式探索更优的决策,并具有端到端的设计优势。首先阐述深度强化学习技术的原理,包括多种典型的深度学习中使用的神经网络结构、基于值函数和基于策略梯度的深度强化学习训练算法;之后详细分析了深度强化学习技术在计算机网络领域中解决资源调度问题的研究现状,包括任务调度、视频传输、路由选择、TCP 拥塞控制以及网络缓存;最后给出了在计算机网络应用中使用深度强化学习仍存在的挑战。

关键词:深度强化学习;计算机网络;任务调度;视频传输;路由选择;TCP 拥塞控制;网络缓存

中图分类号:TP393,TP181　　**文献标志码**:A　　**开放科学(资源服务)标识码(OSID)**:

文章编号:1003-3114(2020)06-0603-21

A Survey on the Applications of Deep Reinforcement Learning in Classical Networking Systems

ZHENG Ying[1,3],DUAN Qingyang[2],LIN Lixiang[2],YOU Xinyu[2],XU Yuedong[2],WANG Xin[1,3*]
(1.School of Computer Science,Fudan University,Shanghai 200433,China;
2.School of Information Science and Engineering,Fudan University,Shanghai 200433,China;
3.Shanghai Key Laboratory of Intelligent Information Processing,Shanghai 200433,China)

Abstract:In the past few years,Artificial Intelligence technology represented by deep reinforcement learning(DRL) has been introduced into the design of computer network systems,propelling the networking research towards a data-driven and intelligent paradigm. New breakthroughs have been continuously made in typical networking systems. The common challenge of versatile networking applications is that they are difficult to be modelled in complex network environments. With the excellent feature extraction capability of deep neural network,deep reinforcement learning is able to perform better decision-makings through trial-and-errors. Meanwhile,it demonstrates the privilege of an end-to-end design rationale. This article first explains the principles of deep reinforcement learning,including a variety of typical neural network structures,training algorithms of deep reinforcement learning based on value functions and policy gradients. We then describe in details the deep reinforcement learning applications in networking systems consisting of job scheduling,video streaming,routing,TCP congestion control and content caching. Finally,this article describes the challenges of using deep reinforcement learning in computer networking applications.

Key words:deep reinforcement learning;computer network;job scheduling;video streaming;routing;TCP congestion control;content caching

收稿日期:2020-09-03
基金项目:国家自然科学基金项目(61772139,61971145);上海市港澳台科技合作项目(18510760900);广东省重点领域研发计划项目(2020B010166003)
Foundation Item:National Natural Science Foundation of China(61772139,61971145);Shanghai-Hong Kong Collaborative Project(18510760900);Key-Area Research and Development Program of Guangdong Province(2020B010166003)

0 引言

近几年来，深度学习(Deep Learning)在机器视觉、语音识别及自然语言处理等领域取得了巨大的成功。通过利用多层神经网络对于观测数据的分层特征表示及其非线性变换，抽取特征信息，以发现数据的特征规律[1]。强化学习(Reinforcement Learning)作为机器学习的另一个热点，其基本思想是通过模仿人类或动物学习中的“尝试与失败”机制，运用智能体在与环境的交互过程中获得的奖励来学习最佳决策行为。然而传统的基于表格的强化学习训练方法存在严重的维度灾难(Curse of Dimension)问题，因而使用函数近似器的方法成为主流。随着AlphaGo的出现，深度神经网络与强化学习相互融合，形成既具有强大感知能力，又具有决策能力的深度强化学习(Deep Reinforcement Learning，DRL)。

DRL目前已经被运用至计算机网络的各个层次，例如网络层的路由协议、传输层的拥塞控制协议、应用层的视频流媒体和云计算资源分配等。在这些应用中，DRL呈现出两大显著的优势。首先，DRL方法在原理上能够有效地提升网络系统的性能。网络系统中的许多研究问题归结到资源的优化分配上，而网络系统的动态性和复杂性导致在具体的研究问题上往往难以建立精确的数学模型和设计高性能的算法。其次，DRL方法提供了一种端到端的设计模式，使得不同网络系统中资源分配算法设计的边界模糊化。现有的基于DRL的网络系统，无论在所处层次和问题模型上差异多大，基本都采用通用的结构。网络方面的任务主要包括输入特征选择、奖励函数设置和输出动作设计，人工智能方面的任务是采用标准的或者定制化的神经网络结构和强化学习算法。因而，在这种状态输入至决策动作输出的端到端设计模式下，网络资源分配算法设计被极大地简化，并且设计的经验可以迁移至原本分野清晰的网络系统中。

本文首先对DRL技术进行介绍，然后针对5个典型的计算机网络应用，包括任务调度、视频传输、路由选择、TCP拥塞控制和缓存，详细描述了这些应用使用DRL技术的动机、建模方法和测试效果，并介绍这些基于DRL的网络系统面临的挑战。

1 深度强化学习方法

1.1 深度学习

深度学习在多个领域均取得了显著的进步，如计算机视觉、机器翻译等。深度学习的基础模型为人工神经网络(Artificial Neural Network，ANN)。全连接神经网络(Fully Connected Neural Network，FCNN)是最简单的神经网络模型，通过对输入数据的多次非线性变换，抽取特征进而产生输出结果。用于处理图像数据的卷积神经网络(Convolutional Neural Network，CNN)，通过权重共享机制可显著降低参数数量，缓解图片数据引起的参数过多问题。FCNN和CNN均为前馈神经网络，即一个神经元的输出只取决于当前时刻的输入以及与该输入相对应的权重参数，并不适用于处理时序数据。为解决这个问题，循环神经网络(Recurrent Neural Network，RNN)应运而生，通过具有自反馈特征的神经元处理可变长的时间序列数据；长短期记忆网络(Long Short Term Memory Network，LSTM)可用于解决简单循环神经网络中的长程依赖问题；此外，还有专门用于处理图结构数据的图神经网络(Graph Neural Network，GNN)等。更多关于深度学习和神经网络的介绍可见参考文献[2]。

1.2 强化学习

强化学习是智能体(Agent)和环境(Environment)的交互过程，该序贯决策过程可使用马尔可夫决策过程(MDP)进行建模，一般使用五元组 $<S,A,P,R,\gamma>$ 描述，其中 S 代表状态空间，A 代表动作空间，P 为状态转移概率，$p(s'\mid s,a)$ 表示在状态 s 下采取动作 a 随后环境状态转移到 s' 的概率，$R:S\times A\rightarrow\mathbb{R}$ 为奖励函数，$\gamma\in[0,1]$ 为折扣因子，反映了智能体对未来回报的重视程度。强化学习智能体的策略使用 $\pi:S\times A\rightarrow[0,1]$ 表示，智能体根据策略 π 选择动作。强化学习原理框图如图1所示。

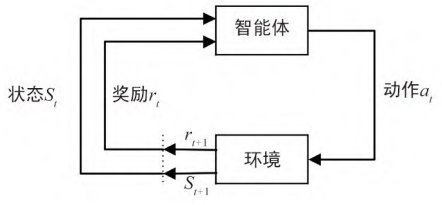

图1 强化学习原理框图

Fig. 1 Framework of reinforcement learning

如图1所示，一个典型的交互过程包括以下几个过程：在每个时刻 t，智能体首先观测环境的状态 $s_t\in S$，随后基于策略 $\pi(s_t)$ 选择动作 a_t，动作 a_t 作用于环境导致环境状态转变到 s_{t+1}，同时，智能体收到环境返还给的奖励 r_t，智能体依据新的环境状

态和策略做出新的决策,该交互过程不断重复。强化学习智能体的目标即为最大化累计奖励的期望 $\mathbb{E}[\sum_{t=0}^{\infty}\gamma^t r_t]$,当 $T=\infty$ 时,$\gamma<1$ 可防止目标函数需要计算无穷多的瞬时奖励之和。

有 3 类标准的方法用于解决上述问题,包括动态规划方法、蒙特卡洛(Monte Carlo)方法和时间差分(Temporal-Difference)方法[3]。

1.2.1 基于动态规划求解强化学习问题

令 $V_\pi(s)$ 表示从状态 s 开始,执行策略 π 能获得的累计奖励,根据贝尔曼方程:

$$V_\pi(s)=\mathbb{E}_\pi[r_{t+1}+\gamma V_\pi(s_{t+1})\mid s_t=s]=\sum_a\pi(a\mid s)\sum_{s',r}p(s',r\mid s,a)[r+\gamma v_\pi(s')],\tag{1}$$

可以使用动态规划的方法求解最优策略,其中关键的 2 个步骤是策略评估(Policy Evaluation)和策略改进(Policy Improvement)。策略评估基于贝尔曼方程计算当前策略下各个状态的值函数,用于评价当前策略 π 的好坏,而策略改进则指明如何修改策略从而获得更高的奖励。令 $Q_\pi(s,a)$ 代表在状态 s 选择 a,随后基于策略 π 进行动作选择所能获得的累计奖励,策略改进的规则为选择使累计奖励最大的动作,即:

$$\begin{aligned}\pi'(s)&=\operatorname{argmax}_a Q_\pi(s,a)=\\&\operatorname{argmax}_a\mathbb{E}_\pi[r_{t+1}+\gamma V_\pi(s_{t+1})\mid s_t=s,a_t=a]=\\&\operatorname{argmax}_a\sum_{s',r}p(s',r\mid s,a)[r+\gamma V_\pi(s')]。\end{aligned}\tag{2}$$

策略迭代即不断重复策略评估和策略改进过程,从而收敛到最优策略。基于动态规划的方法虽然数学原理性较强,但是需要已知完整且精确的环境模型,因此称之为基于模型(Model-based)的算法。由于实际的环境模型大多未知,基于动态规划的方法适用范围较窄。

1.2.2 基于蒙特卡洛方法求解强化学习问题

蒙特卡洛(Monte Carlo)方法不依赖于环境模型,只通过智能体和环境交互生成的轨迹数据以采样平均的方式进行学习,其中轨迹数据包括环境状态、选择动作以及获得奖励。为了计算估计的值函数,蒙特卡洛方法要求智能体和环境的交互过程必须以回合(Episode)的形式进行。一个 Episode 是指从智能体开始和环境交互,直到停止此次交互的完整中间过程,在此过程中产生的轨迹数据可以表示为 $S_1,A_1,R_1,S_2,A_2,R_2,\cdots S_T,A_T,R_T$,$T$ 为本次交互的截止时间。估计状态 S_t 的累计奖励可通过下式进行计算:

$$G_t=R_t+\gamma R_{t+1}+\gamma^2R_{t+2}+\cdots+\gamma^{T-t}R_T。\tag{3}$$

针对参数的更新,蒙特卡洛方法仍然采取策略迭代的思想,当一个或多个 Episode 结束时,对值函数进行更新,同时对策略进行改进,随后根据新的策略继续生成轨迹数据。值函数的更新是基于采样平均的方法,针对蒙特卡洛采样的多条轨迹数据,只有出现状态 S_t 的轨迹数据才会被计算累计奖励,使用多条轨迹数据求出的均值去更新值函数。增量式更新公式为:

$$V(s_t)\leftarrow V(s_t)+\alpha(G_t-V(s_t))。\tag{4}$$

当轨迹数据较多时,采样平均的方法是对值函数的无偏估计。在实际应用中,由于动作值函数可以更确切地表示出策略,因此估计的是动作值函数,其计算方法和计算值函数的方法类似。

1.2.3 基于时间差分方法求解强化学习问题

蒙特卡洛方法虽然不需要精确的环境模型,并且实现方法简单,但是对状态值函数的更新必须要等待一个 Episode 结束。时间差分方法可以直接在每一步对值函数进行更新:

$$V(s_t)\leftarrow V(s_t)+\alpha[R_{t+1}+\gamma V(s_{t+1})-V(s_t)],\tag{5}$$

以上式进行更新的方法称为 TD(0),也称为单步时间差分(One-step TD)。单步时间差分的 2 个代表性算法为 SARSA 和 Q-learning 算法。SARSA 算法为同策略(On-policy)算法,其动作值函数的更新公式为:

$$Q(S_t,A_t)\leftarrow Q(S_t,A_t)+\alpha[R_{t+1}+\gamma Q(S_{t+1},A_{t+1})-Q(S_t,A_t)]。\tag{6}$$

Q-learning 算法为离策略算法(Off-policy),其动作值函数的更新公式为:

$$Q(S_t,A_t)\leftarrow Q(S_t,A_t)+\alpha[R_{t+1}+\gamma\max_a Q(S_t,a)-Q(S_t,A_t)],\tag{7}$$

式中,On-policy 是指 SARSA 算法在状态 S_{t+1} 通过当前策略采取动作 A_t;而 Q-learning 算法在更新时采用状态 S_t 下对应的最大值函数作为目标,为 Off-policy。

1.3 深度强化学习

传统的强化学习使用表格的方法表示策略 π。然而当状态-动作空间较大时,无法继续使用基于表格的方法,此时可使用函数近似器去近似策略 π,如图 2 所示。当使用参数为 θ 的神经网络去近似策略 π 时,称之为 DRL,并使用 π_θ 表示策略。

由于大部分情况下外界环境的模型均是未知

(Model-free)的,本节只介绍模型未知情况下如何使用 DRL 方法训练智能体。

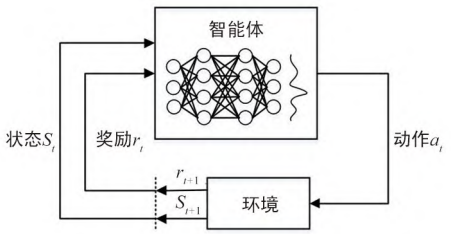

图 2　深度强化学习原理框图

Fig. 2　Framework of deep reinforcement learning

1.3.1　基于值函数的深度强化学习方法

(1) 深度 Q 网络

深度 Q 网络(Deep Q-network, DQN)[4] 是 Q-learning 算法的扩展,使用神经网络去近似动作值函数,优化目标即最小化损失函数:

$$L_i(\theta_i) = \mathbb{E}_{\pi_{\theta_i}}[(y_i - Q(s,a;\theta_i))^2], \tag{8}$$

式中,i 表示第 i 次迭代,$y_i = r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})$。

当使用神经网络近似动作值函数时,强化学习易出现不稳定的现象,原因有以下几个方面:① 智能体采集的轨迹数据前后关联性较大;② Q 网络的更新对策略的影响较大,导致采集的轨迹数据呈现不同的数据分布;③ Q 值和目标 y_i 的相关性也增加了训练的难度。为解决这些问题,DQN 做出了 3 个调整:① 采用经验回放(Experience Replay)机制打破数据之间的关联,经验回放是指将智能体和环境交互产生的轨迹数据存储到经验池中,并随机采样(状态,动作,奖励)数据对进行训练;② 降低奖励值和误差项,保证 Q 值和梯度取值在合理的范围;③ 单独设置另一个 Q 网络专门用于产生目标 Q 值,称为目标 Q 网络,新的目标值为 $y_i = r + \gamma \max_{a'} Q(s', a'; \theta_i^-)$,其中 θ_i^- 为新设置的 Q 网络的参数。计算梯度的公式为:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{\pi_{\theta_i}}[(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s,a;\theta_i)) \cdot \nabla_{\theta_i} Q(s,a;\theta_i)]。 \tag{9}$$

Q 网络参数 θ 的更新可使用标准的随机梯度下降。

(2) 深度双 Q 网络

由于 DQN 使用相同的神经网络去产生目标 Q 值 y_i,即目标 Q 值的动作选择和目标 Q 值的计算均通过一个相同的目标 Q 网络 θ_i^-,导致过高估计 Q 值的问题。为解决这个问题,Van 等人[5] 提出使用深度双 Q 网络(Deep Double Q-network, DDQN)。DDQN 基于当前 Q-网络进行目标 Q 值的动作选择,使用目标 Q-网络计算该动作对应的 Q 值。目标值 y_i 的计算公式为:

$$y_i = r + \gamma Q(s', \operatorname{argmax}_a Q(s', a; \theta_i); \theta_i^-)。 \tag{10}$$

(3) 基于优先采样的深度双 Q 网络

DQN 使用经验回放机制打破数据之间的关联性,然而随机采样轨迹数据进行训练的方式并不能反映不同样本的重要性。此外,在回放池容量有限的情况下,存在样本还未被采样就丢弃的现象。为解决这个问题,Schaul 等人[6] 提出了基于优先采样的深度双 Q 网络(Double Deep Q-network with Proportional Prioritization),使用样本的时间差分偏差(TD-error)作为判断优先级的标准,对于偏差为 δ_i 的样本,其采样概率的计算公式为:

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}, \tag{11}$$

式中,p_i 为通过样本偏差计算出的等级。

此外,还有其他针对 DQN 进行改进的工作,Wang 等人[7] 提出了 Dueling Nerwork,通过将动作值函数拆分为值函数和优势函数相加的方式,解决在某些状态下选择不同动作影响较小的问题,从而提升训练效率。Hausknecht 等人[8] 基于 RNN 提出 DRQN 来处理历史状态信息。

1.3.2　基于策略梯度的深度强化学习方法

上节中基于值函数的方法通过 Q 网络输出在给定输入状态下针对各个动作的估计的动作值函数,只适用于解决离散动作空间的问题。在本节中介绍的基于策略梯度的方法不仅可以解决离散动作的问题,还可以解决连续动作空间的问题。

基于策略的方法使用参数为 θ 的神经网络直接表示策略,称之为策略网络。针对离散动作的情况,神经网络直接输出概率分布,指明了选择各个动作的概率;针对连续动作的情况,神经网络可输出选择动作的正态分布均值和方差,并依据该分布去采样连续的动作值。在 Model-free 情况下,使用蒙特卡洛(Monte Carlo)采样方法进行数据的采集,智能体和环境的交互必须是以 Episode 的方式进行的,这种方法是为了更好地计算某个(状态,动作)数据对的累计奖励。直接对目标函数计算梯度得到:

$$\nabla_{\theta_i} \mathbb{E}_{\pi_{\theta_i}}[\sum_{t=0}^{\infty} \gamma^t r_t] = \mathbb{E}_{\pi_{\theta_i}}[\nabla_{\theta_i} \ln \pi_{\theta_i}(s,a) Q^{\pi_{\theta_i}}(s,a)]。 \tag{12}$$

典型的基于策略梯度的算法 REINFORCE[9] 通

过下式进行神经网络参数更新：

$$\theta_i \leftarrow \theta_i + \alpha \sum_t \nabla_{\theta_i} \ln(\pi_{\theta_i}(s_t, a_t)) v_t, \tag{13}$$

式中，v_t 可直接选取为 $Q^{\pi_{\theta_i}}(s,a)$。然而，由于参数的更新是依赖于蒙特卡洛采样的结果，因此方差较大，可使用优势函数（Advantage Function）$A^{\pi_{\theta_i}}(s,a)$ 作为 v_t 从而降低方差，$A^{\pi_{\theta_i}}(s,a) = Q^{\pi_{\theta_i}}(s,a) - b$，$b$ 是一个基线（Baseline），有多种选取的方法，简单的方法即为选取多次采样的均值，此时可直接认为 b 是一个常数。

1.3.3 混合值函数和策略梯度的方法

（1）行动者-评论家算法及其改进

上文提到基于策略梯度的方法在进行参数更新时，为降低训练的方差，需要选择一个合适的基线，当使用迭代过程中动态改变的值函数作为基线，这就是行动者-评论家（Actor-critic）算法[10]。Actor-Critic 算法是基于值函数的方法和基于策略梯度方法的融合，共包括两个神经网络，其中 Actor 网络用于动作决策，Critic 作为附加模块专门用于计算基线的取值；与此同时，Critic 网络也随着迭代次数的增加不断进行自我更新。令 Actor 网络的神经网络参数为 θ_{a_i}，Critic 网络的神经网络参数为 θ_{v_i}。Actor 网络参数的更新公式为：

$$\theta_{a_i} \leftarrow \theta_{a_i} + \alpha \sum_t \nabla_{\theta_{a_i}} \ln(\pi_{\theta_{a_i}}(s_t, a_t)) v_t, \tag{14}$$

Critic 网络输出对当前状态的值函数的估计值，参数更新公式为：

$$\theta_{v_i} \leftarrow \theta_{v_i} - \alpha' \sum_t \nabla_{\theta_{v_i}} (r_t + \gamma V^{\pi_{\theta_{i-1}}}(s_{t+1}; \theta_{v_i}) - V^{\pi_{\theta_{v_i}}}(s_t; \theta_{v_i}))^2。 \tag{15}$$

注意 Actor 网络和 Critic 网络除输出层外均采用相同的神经网络结构。

Mnih 等人提出的 A3C 算法[11]是轻量级的异步 Actor-Critic 算法，利用多个智能体在多个线程和各自的环境进行交互，从而产生轨迹数据；每个线程的智能体基于自身存储的轨迹数据进行参数更新，一个中心智能体负责收集多个线程并生成全局的网络参数，随后分发给各个智能体。这种方式成功打破了数据相关性，并且解决了经验回放机制只能使用离策略（Off-policy）的局限性。

（2）信赖域策略优化算法及其改进

策略梯度方法存在高方差的问题，并且算法的性能对超参的设置较敏感。信赖域策略优化算法（Trust Region Policy Gradient，TRPO）[12]解决的问题是如何合理地选择步长从而使得回报函数单调递增。它将优化的损失函数修改为：

$$J(\theta) = \mathbb{E}_{\pi_{\theta_{\text{old}}}} \left[\sum_t \gamma^{t-1} \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} A^{\theta_{\text{old}}}(a_t, s_t) \right] \tag{16}$$

$$\text{s.t. } KL[\pi_{\theta_{\text{old}}} \mid \pi_\theta] < \delta,$$

式中，$A_\theta(s_t, a_t) = \mathbb{E}_\theta[R_t \mid s_t, a_t] - V^\theta(s_t)$。近端策略优化（Proximal Policy Optimization，PPO）算法[13]可看作是 TRPO 算法的近似版本，它将 TRPO 的约束项作为正则项添加到损失函数中，降低了 TRPO 算法的实现复杂性，应用性更广。此外，Heess 等人[14]提出了适用于大规模训练的分布式近端策略优化算法。

（3）深度确定性策略梯度算法及其改进

尽管 PPO 算法的学习效率很高，但是 PPO 算法使用的随机策略在动作空间维度较大时，需要采集大量的样本，才能对当前的策略进行合理的评估。其中随机策略只针对某一个输入状态，策略网络输出的是选择动作的概率分布。为解决这个问题，Lillicrap 等人[15]提出深度确定性策略梯度算法（Deep Deterministic Policy Gradient，DDPG），DDPG 为确定性策略，即策略网络直接输出具体的选定动作。

DDPG 仍然采用 Actor 网络用于输出动作，Critic 网络用于输出对值函数的估计。为了保证算法一定程度的探索过程，DDPG 给输出动作添加了额外的噪声 N_t：

$$a_t = \pi(s_t; \theta) + N_t。 \tag{17}$$

此外，为了打破数据之间的相关性，DDPG 也采用经验回放机制和额外设立的目标网络。DDPG 实现连续控制需要训练 2 个网络，而 Gu 等人提出的 NAF[16]，将动作值函数分解为状态值函数和优势值函数，只需要训练一个模型。更多关于 DRL 的技术介绍见文献[17-18]。

2 基于深度强化学习的网络应用

本文主要介绍 DRL 在 5 个典型的网络系统中的应用，包括任务及数据流调度、视频传输、路由选择、TCP 拥塞控制以及缓存问题。

2.1 任务及数据流调度

任务调度是计算机网络中的一个典型应用，如图 3 所示。互联网上的用户源源不断地产生计算任务，从普通的网页浏览到对计算量需求较高的大数据分析以及深度学习计算任务等，这些任务会上传到云数据中心由计算机集群进行处理。任务调度算

法在这个过程中起着至关重要的作用,一个高效的任务调度算法可以合理利用计算资源,根据用户的不同需求进行优化,从而在保证用户满意度的同时降低数据中心的开销。最优的调度策略与任务负载和环境条件(如任务到达的数据分布)有关。尽管具体的应用场景以及假设有所不同,但是目前的解决方法基本上是针对某种特定的任务负载以及任务到达的分布,建立理论模型并设计特定的任务调度算法,且为了达到较好的性能常常需要在实际环境下进行详尽的测试与修正。当这些环境条件发生改变时,可能需要重复上述流程重新设计算法。近年来出现了许多使用 DRL 技术实现自动化的任务调度算法设计。

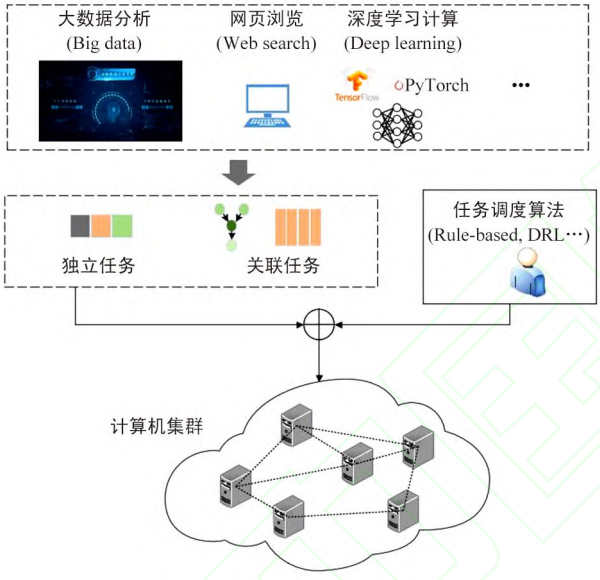

图 3　云数据中心任务调度原理图

Fig. 3　Framework of job scheduling in cloud data center

2.1.1　面向独立任务的调度

MIT 的工作 DeepRM[19] 最早将 DRL 应用到数据中心任务调度中,DeepRM 是一个简化的集群多资源任务调度器,称之为 DRL 智能体。在离散的时间片,任务以设定好的分布在线到达从而形成任务队列,其中每个任务请求一定时间长度的 CPU 和内存资源。在每个时间片,DeepRM 观测系统状态,包括 CPU 和内存的占用情况以及任务队列中任务的资源请求状况,并根据系统状态和策略网络进行决策,策略网络为使用同全连接神经网络近似的强化学习策略,决策内容为调度当前任务队列内的一个或多个任务到相应的 CPU 以及内存的待执行队列中。DeepRM 每做出一个决策,将收到系统返还的奖励信号,由于最终的优化目标设定为最小化平均每个任务的完成时间与请求时间的比值,因此将单步奖励信号设置为:

$$\sum_{j\in J}\frac{-1}{T_j}, \tag{18}$$

式中,J 代表当前决策时间点和上一个决策时间点之间的系统内所有任务,T_j 为任务 j 的请求时间。DeepRM 使用带有基线的 REINFORCE 算法进行训练。实验结果显示 DeepRM 的性能超过目前基于领域知识的传统算法,且在重负载的情况下效果更明显。

DeepRM 假设所有资源均在一个统一的资源池中,然而在实际情况下,常常需要不同物理位置的数据中心进行跨区域的任务调度,因此 Chen 的工作[20]将 DeepRM 扩展到更复杂的情况,即多资源池的情况。针对智能体的输入状态,除了考虑任务队列中任务的请求资源状况外,它将单个机器的 CPU 和内存占用状况拓展到两个机器的 CPU 和内存的占用状况,动作空间扩大一倍。实验结果显示,经过拓展后的 DeepRM 仍然取得了比传统任务调度算法(如最短任务优先算法)更好的性能。

Chen 的工作[20]存在的问题是,由于输入状态要考虑所有资源池(机器)的资源占用情况,而神经网络的输入必须是固定维度的,限制了它只能应用到资源池数量较少的场景。为解决这个问题,Li 等人提出了 DeepJS[21],DeepJS 仍然是一个基于 DRL 的任务调度系统,不同之处在于 DeepJS 以装箱问题为原型,优化目标是最小化 Makespan,其中 Makespan 定义为最后一个任务的完成时间。DeepJS 基于神经网络计算不同的(任务,机器)数据对的适应度(Fitness),在分配任务时,将任务分配到适应度最高的机器,由于此时神经网络的输入状态只需要考虑某一特定的机器的资源占用状况和新到达的任务的请求资源状况,因而可以固定输入的维度。这种方法的可扩展性较高,不再受限于输入维度的限制。

2.1.2　面向关联任务的调度

上一节的讨论并没有考虑任务之间的关联性,然而在实际的大规模并行化任务处理平台,如 Spark,经常需要处理有关联性的任务,即多个任务之间的执行顺序具有关联关系,当关联任务对资源的需求不同时,使得求解最优策略变得更加困难,是公认的 NP-难问题[22-23]。

Spear[24]结合蒙特卡洛树搜索(MCTS)和 DRL 技术实现对关联任务的调度,任务之间的关联关系

被建模为有向无环图(DAG),一个完整的任务对应一个 DAG,一个 DAG 内有多个节点,称之为子任务,子任务之间的关联关系通过有向边表示。在普通的 MCTS 中,扩展和模拟的过程会随机选择动作,为更有效地探索状态空间,Spear 在扩展和模拟的步骤中使用 DRL 智能体协助探索。DRL 智能体的输入状态包括系统资源的占用状况、队列中有限任务个数对资源的请求状况、每个节点的孩子节点个数、有关运行时间以及任务负载的关键路径信息,输出动作为选择一个队列中的任务去调度或者空动作引发时间前进一步。为最小化 Makespan,Spear 在只有选择空动作引发时间前进时才会获得一个奖励,其数值为-1,由此将累计奖励和优化目标进行关联。实验测试时将 Spear 与多个传统算法进行对比,包括 Graphene,Tetris,SJF,结果显示 Spear 取得了更低的 Makespan 时间,主要原因是 Spear 考虑了更多的特征。

Mao 等人提出的 Decima[25] 是另一个基于 DRL 实现关联任务调度的系统。Decima 仿照 Spark 平台的设置,考虑了更为复杂的情况,一个完整的任务对应一个 DAG。一个 DAG 内有多个节点(子任务),每个节点内有多个可并行处理的任务。为了提升系统的可扩展性,Decima 使用了图神经网络来应对不同类型的 DAG。具体来讲,Decima 首先对 DAG 中的每个节点进行编码,使用的信息为所有孩子节点的编码向量和当前节点的特征向量。其次,Decima 为每个 DAG 新增了一个全局节点,该全局节点将该 DAG 内所有节点视为孩子节点,对该全局节点进行编码得到 DAG 级别的全局节点向量。除此之外,Decima 将所有 DAG 的全局节点视为子节点得到最终的编码向量。注意以上所有的编码过程都是通过图神经网络完成的。在智能体需要做出决策时,Decima 针对当前所有可调度的节点,基于上述的各类编码向量计算调度分数,并依此得到调度决策和分配的处理器数量。为了最小化平均任务完成时间,Decima 将奖励信号设置为前后两次相邻决策时间点内系统存在的任务数量与该时间差的乘积。基于 Spark 平台的测试结果显示,在一个具有 25 个节点的集群上,即使针对性能最优的加权公平(Weighted Fair)传统算法,Decima 仍降低了约 21%的平均任务完成时间。

除了任务执行先后顺序的依赖关系外,大规模并行任务的调度也是需要解决的重要问题。由于动作空间的大小会随着任务数量和执行任务的服务器的数量呈现指数型增长,文献[27]在异构服务器的场景下提出了基于多任务深度强化学习(Multi-task Learning)的调度算法进行高效调度并行任务,其中异构服务器指不同服务器具有不同的 CPU 核数量以及主频。每个在线到达的任务包含多个可并行执行的子任务。训练算法使用 A3C,多任务学习的思想体现在针对每个任务的子任务均设置一个 Actor 网络,多个子任务共享一个 Critic 网络,其中各个子任务的 Actor 网络的前 3 个全连接层共享模型参数,与此同时每个子任务独占随后的一个全连接层以及一个输出层,这种架构成功将动作空间的大小由 N^M 降为 MN,增加了算法的可扩展性。

2.1.3 面向计算和节能的联合调度

随着深度学习的发展,数据中心也出现了更多运行机器学习、大数据分析等对计算量需求较大的应用,导致数据中心消耗的电量逐年增长。研究表明,由于计算而消耗的数据中心电量约占比 56%,冷却系统消耗的电量占比达到 30%[28]。因此如何同时实现合理的任务调度,从而高效利用计算资源以及降低冷却系统的电量消耗是一类重要问题。目前,联合考虑上述两条因素进行优化的方法大多依赖于静态的或不恰当的动态模型,不能很好地解决上述问题。

DeepEE[29] 利用 DRL 技术处理动态多变的系统状态,如动态变化的工作负载和环境温度。由于优化计算资源(任务调度)为离散动作,而优化冷却策略需要做出调整气流速率的连续动作,因此 DeepEE 将策略梯度和 DQN 进行结合,策略网络输出调整气流速率的连续动作,该连续动作也将作为 DQN 中 Q 网络(Q-network)输入状态的一部分,依据 Q-network 的输出可确定将任务调度到哪台服务器执行的连续动作。与此同时,DeepEE 也采用了两级时间的控制方法,即增加一个时间标志变量去解决优化计算和冷却策略需要不同时间粒度的问题。

DeepEE 假设机架内所有的服务器都是同构的,Deep-EAS[30] 则是针对异构服务器利用 DRL 技术进行高效计算和节约能量的联合优化。除此之外,文献[31]提出使用分层架构去解决任务调度和能量管理的问题,包括全局层面和本地层面。全局层面使用 DRL 技术进行任务调度,并使用自编码器和权重共享机制去加速训练;本地层面使用 LSTM 神经网络进行负载预测,并使用无模型的强化学习进行能量管理。针对 Google 的真实数据进行测试,结果显示该分层的架构可以保证在相同延迟的基础上

降低能量消耗。

针对一些精心设计的数据中心,电量有效利用率(PUE)已经降到了较低的水平,此时不必再考虑冷却系统等附加设备对电量的消耗,而如何合理地进行任务调度从而高效利用计算资源成为关键问题。目前已有方法的逻辑多是迁移虚拟机以抢先执行计算量小的任务,然而当前数据中心出现了大量强计算需求的任务时,如大规模数据分析、深度学习等,导致前人的调度算法仍有一定的提升空间。文献[32]专门针对强计算需求的任务,利用 DRL 技术进行自动化的任务调度。仿真数据显示,针对具有 1 152 个处理器的新加坡国立超算中心(NSCC),训练一个任务调度的 DRL 智能体需要消耗高达 40 天的时间。为解决这一问题,文献[32]提出了离线训练 DRL 智能体的方法。它首先使用从真实数据中心收集的任务到达信息和服务器的计算信息预训练了一个基于 LSTM 神经网络的计算模型,利用该计算模型输出服务器的电量消耗、温度等近似信息,DRL 智能体则基于这些近似信息进行离线训练。针对 NSCC 的真实任务到达数据的测试结果显示,DRL 智能体节约了近 10% 的能源消耗,并降低了 3℃的处理器温度。

2.1.4 面向数据中心网络流的调度

数据中心网络流调度技术是指合理调度由数据中心应用产生的数据流,从而有效改善数据中心的网络流传输和优化用户体验[33]。一条数据流是指发送端和接收端之间跨过网络传输的一个应用数据单元[34]。数据中心的数据流可分为两种:一种为单独的数据流,另一种是一组有关联关系的网络流组(Coflow)。针对数据中心网络流调度问题,通常需要专家花费较长的时间去设计依赖于负载和环境条件的启发式算法。然而当环境条件改变时,如流大小的分布发生变化,启发式算法性能会出现明显下降,需要重新花费时间和人力成本设计新的启发式算法。

针对单独的数据流,Auto[35] 利用 DRL 技术进行自动化的数据中心网络流调度。数据中心的数据流呈现明显的长尾分布,大部分数据流均为短流,但传输的数据量主要由小部分的长流决定。针对上述数据流的统计特征,Auto 采用了具有外围系统和中心系统的二级架构,其中外围系统直接部署在终端服务器,采用多级队列的方式进行短流本地快速决策。每一个到达的数据流首先进入最高级别的队列,当转发的数据量超过一定阈值时,则自动被降低到更低级别的队列中,而判断数据流优先级的阈值则每隔一段时间由中心系统的 DRL 智能体(sRLA)决定。除此之外,中心系统还有另一个智能体(lRLA)负责只针对长流的决策,包括确定优先级、速率和路径。sRLA 智能体接收固定数量的已完成的短流信息作为输入状态,输出动作为区分各个队列的判断阈值,奖励信号设置为前后两个时间段内的数据流完成时间。lRLA 智能体接受固定数量活跃的和已完成的长流信息作为输入状态,奖励信号设置为前后 2 次决策时间段内吞吐量的比值。在一个具有 32 个服务器的小型数据中心对 Auto 的测试结果显示,相比于目前已有算法,Auto 降低了 48.14%的平均流完成时间。

针对 Coflow 的调度问题,DeepWeave[36] 使用 DRL 技术产生 Coflow 的调度策略,将一个任务产生的 Coflow 建模为一个 DAG。DeepWeave 的输入状态为 Coflow 中每个数据流信息和 DAG 的形状。智能体主要包括 3 个部分,首先使用图神经网络模块提取 DAG 中的特征信息,然后策略网络模块接收这些特征信息并输出一个优先级列表,最后策略转化模块负责将优先级列表转化为具体的调度决策。实验结果显示 DeepWeave 在任务完成时间指标上实现了 1.7 倍的加速。

2.2 视频传输

目前视频传输所消耗的流量已占据互联网流量的绝大部分。流媒体系统分为服务器端和客户端,服务器端主要存储媒体描述文件(.mpd 文件)和视频块(Chunk)文件。媒体描述文件包含视频的基本信息,包括各个视频块文件请求的路径模板、可请求码率等;视频块文件是服务器端将一段完整的视频切成若干秒(一般为 2~10 s)后保存的不同清晰度的视频文件版本。当客户端需要观看视频时,会先向服务器请求并解析媒体描述文件,随后开始不断地向服务器请求不同码率的视频块文件。当服务器端收到客户端的请求后开始向客户端传输视频块文件,客户端在播放器上进行播放。为了保证在复杂的网络环境中尽可能地提升用户的观看体验,客户端的自适应比特率选择算法(ABR)需要实时地进行视频块码率的选择,以保证用户的观看体验。ABR 算法需要考虑避免视频卡顿、码率波动过于频繁等因素。如何根据网络带宽的变化而实时地进行恰当的码率决策是非常复杂和值得研究的问题。

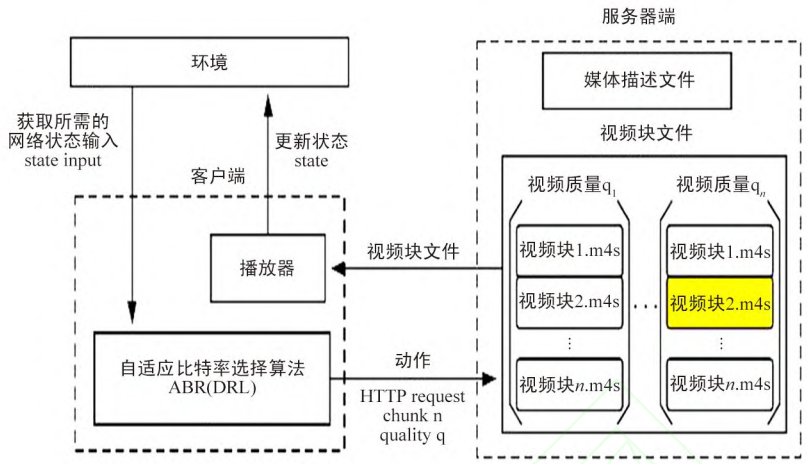

图 4　基于 HTTP 的视频传输原理图

Fig. 4　Framework of HTTP adaptive video streaming

在流媒体系统中,码率决策的传统经典算法有很多,例如 Huang 等人提出的 BBA(Buffer Based Approach)[37]便是客户端在进行码率决策时,根据当前的缓存长度来选择下一个视频块的码率大小。除了基于当前缓存长度做码率决策外,Liu 等[38]人提出可以根据前几个时刻的网络吞吐量来预测下一个时刻的吞吐量,从而进行码率决策。传统算法虽然相对容易实现,也在大多数网络情况下可行,但是固定的参数与固定的决策方式并不适应于所有的网络环境。BBA 算法对缓存上下界的不恰当设置可能导致有些网络环境中的卡顿及码率频繁波动,影响用户观看体验。因此,结合 DRL 的视频码率决策方法应运而生,通过对 DRL 的模型进行各种网络带宽环境的训练,使训练出的模型在视频播放过程中做出更加正确与恰当的码率决策。

2.2.1　视频点播

视频点播(Video on Demand,VoD)是流媒体应用中最为常见的应用,服务器端将用户可以点播的视频制成不同码率的视频块并存储,客户端不断请求已经存储好的视频块,在播放器进行拼装与播放。

2013 年,Claeys 等人[39]提出了一种基于 Q-learning 的 HAS(Http Adaptive Streaming)客户端,与现有的传统启发式算法相比,Q-learning 的方法训练出的决策模型可以让客户端根据当前网络环境动态地做出相应的码率决策。实验结果表明,结合 Q-learning 的决策算法比传统算法的观看体验(Quality of Experience,QoE)提升 9.7%。但基于 Q-Learning 进行模型训练,存在对环境的变化反应较慢的问题,针对一个给定状态,当一个动作对应的 Q 值相比于其他动作较低时,该动作被选择的概率较低,导致该动作的信息获取变慢,造成该动作对应的 Q 值修改速度慢。在低学习率的情况下,上述问题尤为明显。针对这个问题,Claeys 等人[40]又提出了一种频率调整 Q-Learning 算法(Frequency Adjusted Q-Learning Approach),对模型的更新方式进行了修改,这样的改进有效避免了模型中 Q 值低的动作不易被选择的问题,加快模型的更新,进一步提升模型的性能。Martin 等人[41-42]于 2016 年提出基于 Q-Learing 自适应方法 DASH-QL 来提升视频播放的 QoE。该系统考虑的网络状态包括当前缓存长度、当前预测带宽和上一个视频块质量,并依据这些状态信息决策下一时刻的视频码率。实验结果表明,DASH-QL 取得了更高的 QoE,且收敛速度更快。

Q-Learning 算法在处理连续状态空间时,只能将状态变量的连续取值进行大致的切分。当进行相对细致的切分时,会导致维度爆炸问题,并且在实际测试环境中很难把每个区间都不断地遍历,增加了训练 DRL 智能体的难度。为解决这个问题,Gadaleta 等人[43]提出用基于 DQN 的机器学习框架,通过神经网络近似动作值函数,并使用经验回放机制加速模型收敛。系统状态包括前一个视频质量 q_{t-1}、当前缓存长度 B_t 以及下一个可选取的视频块特征等,结合当前视频质量 q_t 来选择下一时刻请求的视频块码率。Gadaleta 等人在仿真和真实环境都进行了系统部署,实验结果显示 D-Dash 在各种带宽条件下对 QoE 都有提升,并且 D-Dash 框架收敛速度更快。

Mao 等人于 2017 年提出 Pensieve[44],该模型通

过客户端视频播放器获取过去 8 个历史时刻信息，据此生成系统状态，包括下载每个视频块的平均吞吐量、下载每个视频块所花的时间、当前缓存长度、上一个请求片段的码率、剩余未下载的视频块数量以及可供下载的视频码率。Pensieve 基于系统状态选择下一个视频块的请求码率，实验结果显示 Pensieve 比传统算法实现了更高的 QoE。

为了将 Pensieve 在实时运行过程中的计算开销降低，并进一步提升 Pensieve 模型的性能，2020 年 Huang 等人将 DRL 与传统算法 BBA 进行结合，提出了 Stick 网络[45]。Stick 将原本 Pensieve 的离散动作修改为连续动作，即 BBA 决策的上下边界。客户端通过修改后 BBA 算法的上下边界和当前缓存长度去选择请求的下一个视频块码率。Huang 等人同时又提出一个轻量级的 Trigger 网络，用来决策当前情况是否需要调用 Stick 网络来对 BBA 的上下边界进行调整，以减少频繁使用 Stick 算法带来的计算开销。Huang 等人先在仿真系统上进行了验证，并在以 Dash. js 为基础的真实系统上进行了部署。实验结果表明，相比于原本的 Pensieve 模型，Stick+Trigger 网络模型可以提升 9. 41%的性能，并减少 88%的计算开销。

受 Pensieve[44] 模型的启发，Guan 等人[46] 于 2020 年提出 Perm 模型，运用 DRL 技术优化视频点播在多径传输情况下的用户体验。Perm 在 Pensieve 原有的架构中加入了一个新的 Actor 网络负责流量在各条路径上的分配，原有的 Actor 网络依旧负责下一个视频块的码率选择，Critic 网络负责协助优化两个 Actor 网络的参数。因为流量分配的动作为连续取值，Perm 使用 DDPG 方法进行训练。他们在仿真系统和真实系统中都进行了系统的部署，两条传输路径分别为 4G 与 WiFi。与传统多径算法和单条路径的流媒体系统相比，QoE 与 QoS(Quality of Service)提升了 10%~15%。

2.2.2 360 全景视频传输

随着全景摄像机和头戴式设备的发展，360 全景视频形式逐渐流行。而完整的 360 全景视频的传输通常需要极大的带宽支持，可通过视角分块(Tile)的方法解决。为了保证用户的 QoE，需要对用户的视角进行预测，并在视角区域内为用户分配尽可能多的带宽去下载更清晰的视频。

Zhang 等人[47] 于 2019 年提出结合 LSTM 神经网络的 Actor-Critic 模型的 DRL360 流媒体框架。客户端将之前各个时间点上网络的带宽和视角范围作为 LSTM 网络的输入，LSTM 网络预测下一时刻的带宽和视角位置。随后，将 LSTM 网络预测的带宽和视角位置，结合当前视频缓存长度以及保存的下一时刻可获取的各个视频块大小，作为 Actor-Critic 模型的输入，让其决策下一时刻在预测视角范围内所有 Tile 的码率。他们将 DRL360 部署在真实系统上并与已有的算法进行比较。实验结果表明，在多种带宽环境下，DRL360 平均提升了系统 20%~30%的 QoE。

由于 360 视频的决策是指数级的决策空间，因此直接同时对 360 视频的每个 Tile 都做决策是非常困难的。假设有 N 个 Tile，每个 Tile 都有 k 个码率，则动作空间共有 N^k 个决策。DRL360 模型为了避免这个问题，码率决策只是做了预测视角范围内的视频块码率决策，并没有考虑所有 Tile 的视频块决策。Fu 等学者[48] 为解决这个问题，提出基于序列化的 ABR 360 视频决策模型，将其命名为 360SRL。假设 360 视频被分为 N 个 Tile，每个 Tile 处都有 k 种码率可以选择。他们的模型通过连续做 N 次视频块码率决策，将 N^k 的动作空间降低到每次动作空间只有 k，但连续做 N 次决策，从而可以对 360 度视频的所有 Tile 做出码率决策。在决策过程中，360SRL 根据当前网络的状态去轮番选择每个 Tile 下一时刻的码率，系统状态包括过去 m 个时间点下载视频块的平均吞吐量、此处 Tile 可选择的视频块大小、上一时刻该位置 Tile 在视角范围的预测概率、播放缓存长度、此处 Tile 出现在视角范围内的预测概率，以及上一时刻选择的所有 Tile 的视频块总大小。360SRL 使用 DQN 网络来训练模型。仿真系统实验表明，360SRL 相较于传统 360 视频算法，对平均 QoE 的提升达到了约 12%。

2.2.3 直播

实时的视频流传输(Live Streaming)也是如今流媒体应用中一个重要研究领域。由于观看直播视频的用户一般期望客户端播放视频的时延较小，因此直播系统的码率选择问题比点播更加困难，视频的卡顿、码率的切换和缓存长度都会导致用户体验的下降。

Hooft 等人于 2015 年[49] 提出引入强化学习的方法，在直播系统中对两个传统启发式方法(MSS 和 QoE-RAHAS)做参数优化，以提高这两个方法的性能。他们通过强化学习感知网络的带宽特性，提高客户端对网络带宽的预测能力。在直播系统中，奖励函数也被重新定义，缓存的长度作为一项线性

的惩罚。Hooft 等人使用 Q-Leaning 模型训练,模型考虑的状态包括当前可用带宽和当前带宽变化剧烈程度。随后使用两个启发式模型去决策下一时刻选择的视频码率,再通过奖励函数去修改两个启发式算法中的参数。NS-3 仿真平台下的实验表明,使用强化学习方法优化传统启发式方法的参数后,可以使 MSS 算法与 QoE-RAHAS 算法得到一定程度的改进,在保证视频质量相对较好的情况下,时延分别减少 2.5%与 8.3%。

Huang 等人[50]提出,视频的码率并不能总是精准地反映视频质量。视频质量和编码方式与图像各种特征都息息相关,应该用更科学准确的方法为视频质量打分,然后据此进行直播码率的选择。为解决这个问题,Huang 等人[50]提出视频质量感知码率控制(Video Quality Aware Rate Control,QARC)系统。该系统中视频质量预测网络(Video Quality Prediction Network,VQPN)设计结合了 Netflix 提出的 VMAF(Video Multi-method Assessment Fusion)评价标准,可以根据之前时刻及当前时刻的网络状态和视频帧等特征预测下一时刻的视频质量。系统中的视频质量强化学习选择网络(Video Quality Reinforcement Learning,VQRL)为 A3C 架构,通过综合考虑下一时刻的视频质量、当前的网络状态等各个特征,决定下一时刻选取的直播码率。在仿真环境下将 QARC 和目前多个现有多个算法的对比结果显示,QARC 对平均视频质量提升最高可达 18%~25%,并且减少了 23%~45%的平均延时。

2.3 路由

随着互联网的高速发展,网络规模不断扩大,传输数据量也不断增加。传统的路由算法路由策略较为固定,且很难感知实时的网络状况,易造成较大的传输时延和网络资源的浪费。DRL 提供了一种解决路由问题的新手段,无需对网络环境进行建模,通过智能体与网络环境的不断交互,即可实现端到端的智能路由算法。图 5 为基于 DRL 的路由算法框架图。路由决策控制器为强化学习智能体,通过观测实时网络环境获取相应状态信息,经过神经网络处理后得到路由决策,进而执行数据包的传输。DRL 在路由问题中的应用主要包含数据包路由和域内流量规划两类问题:在数据包路由问题中,状态信息一般为数据包目的节点,路由决策为数据包下一跳节点,优化目标为降低数据包平均传输时延;在域内路由规划问题中,状态信息一般为流量需求矩阵,路由决策为数据流链路分配比,优化目标为平衡链路负载。

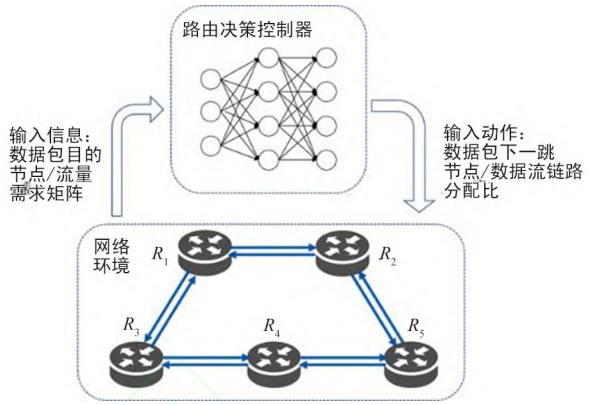

图 5 基于 DRL 的路由算法框架图

Fig. 5 Framework of DRL-based routing algorithm

2.3.1 数据包路由

Boyan 等人[51]首次将强化学习应用于数据包路由问题中,提出了动态的路由转发策略 Q-routing 算法,以最小化数据包的平均传输时延。各路由节点均视为独立的智能体,并拥有独立的 Q 值表用于存储各状态动作值函数,以此作为节点间传输时间的估计。Q-routing 算法中的状态空间为数据包的目的节点编号,动作空间为各路由器的相邻节点,奖励为数据包的链路传输时延和排队时延。各路由器基于 Q-learning 算法进行训练,当数据包从当前节点传输至所选动作相应节点后,当前节点将收到包含对应奖励的反馈信息,进而完成 Q 值表的更新。在与最短路径算法的对比实验中,Q-routing 算法能实现较低的数据包传输时延,并能适应动态的网络环境。然而,Q-routing 算法在网络负载较大时无法及时调整路由转发策略,且对于网络负载变化适应较缓慢。为解决上述问题,Choi 等人[52]提出了 PQ-routing 算法,利用历史的路由信息来预测链路流量;Kumar 等人[53]提出了 DRQ-routing 算法,利用前向和后向探索信息来学习更优的路由决策。

Mukhutdinov 等人[56]基于多智能体 DQN 算法,设计了动态的数据包路由算法 DQN-routing,以降低数据包的传输时延。在多智能体强化学习中,各路由器被视为独立的智能体,且拥有独立的神经网络用于路由决策。各智能体仅可观测到局部状态信息,其中包含数据包的目的节点编号、当前路由器编号及其相邻节点编号。此外,动作空间和奖励函数与 Q-routing 算法[51]一致。各智能体的神经网络参数全局共享,且其更新方式保持全局同步性,但在决策过程中保持独立性,即各智能体的路由决策无需

其他智能体的参与。在仿真实验中,相比于基于链路状态的路由算法和 Q-routing 算法,DQN-routing 算法在动态变化的网络负载下均能达到最低的数据包传输时延。

由于在集中式强化学习中,参数聚合和参数分发过程会造成巨大的带宽消耗,进而影响网络中数据包的传输。为了解决该问题,You 等人[57]将完全分布式多智能体强化学习应用于数据包路由问题中,以最小化数据包平均传输时延为目标,设计了端到端的动态数据包路由算法 DQRC。各路由器视为独立的智能体,均拥有独特的神经网络用于参数更新和动作选择,且其学习过程和决策过程均为分布式。DQRC 使用 LSTM 神经网络处理时序相关的输入信息。与 Q-routing 算法[51]输入信息仅包含数据包目的节点编号不同,DQRC 算法加入了历史决策、队列中未来数据包目的节点以及相邻节点队列长度等状态信息,以便于智能体学习到自适应的数据包路由转发策略。在北美 AT&T 网络[58]拓扑的仿真环境中,DQRC 算法在优越性和稳定性上均优于 Q-routing, Backpressure[59] 等对比算法。此外,DQRC 算法对于神经网络结构和相邻节点信息交互间隔具有较强的鲁棒性。

不同于以上数据包路由算法以降低数据包传输时延为优化目标,Ali 等人[60]基于 DDQN 强化学习算法[61],设计了分层数据包路由算法 DDQN-routing,以平衡网络链路负载。该算法以节点间链路传输时延为分类标准,将网络中的节点分为多个类别,且各类别均拥有一个“队长节点”以辅助源节点的路由决策。“队长节点”仅可观测到局部信息,其中包括链路利用率、链路传输时延和队列排队时延。此外,奖励函数包含全局奖励和局部奖励两部分,其中前者与源节点对于已选链路的满意度有关,后者与已选链路的传输时延、排队时延和丢包率有关。为了解决 DQN 算法中的过估计问题,DDQN-routing 算法构建了两个神经网络,分别用于选择贪婪策略和评估动作优劣,并引入了优先经验回放算法,以最大化利用经验回放池中的采样数据。在仿真实验中,DDQN-routing 算法能适应动态的数据包请求和较大的网络拓扑,并能有效地利用链路资源和平衡链路负载。

2.3.2 域内流量规划

Xu 等人[62]将 DRL 应用于域内流量工程(Traffic Engineering,TE)问题中,并提出了经验驱动的域内流量工程方案 DRL-TE。考虑到直接应用 DDPG 算法无法达到良好的性能表现,作者提出了两种改进措施。一是 TE-aware 探索机制,即以 ϵ 的概率选择基准 TE 方案(如最短路径算法等),以 $1-\epsilon$ 的概率选择神经网络的输出方案,以达到加速智能体学习的效果;二是优先经验回放算法,即以时序差分数值作为样本选取概率的基准,进而高效利用经验回放池中的环境转移样本。强化学习状态空间包含吞吐量和时延两部分,动作空间为数据链路分配比,奖励设置为系统效用函数。在 NS-3 的仿真实验平台上[63],DRL-TE 相比于最短路径、负载均衡等算法能达到更短的时延和更大的网络吞吐量。

Stampa 等人[54]将 DRL 应用于软件定义网络(Software-Defined Network,SDN)中并提出了 SDN-routing 算法,以优化数据流传输时延。基于单智能体强化学习算法 DDPG[15],SDN-routing 算法将 SDN 控制器视为集中式智能体,该智能体能够观测网络的全局信息,并控制网络中各路由器的数据流分配决策。其中,强化学习状态空间为网络流量需求矩阵,代表各源点-终点对间的带宽需求;动作空间为数据流的链路分配比;奖励设置为数据流传输时延。智能体的训练过程和决策过程均为集中式,即允许各路由器间交互环境转变信息和神经网络参数信息。在仿真实验中,SDN-routing 算法在各流量等级下性能表现均优于比较算法。

Valadarsky 等人[55]考虑了与 SDN-routing[54]相似的路由问题,考虑到监督学习无法准确估计未来的网络流量需求,基于 TRPO 算法训练端到端学习的智能体,以平衡网络链路负载。由于原始问题中动作空间过大,强化学习算法难以收敛。Valadarsky 等人将输出神经元个数由 $|E||V|^2$ 减少为 $|E|$(其中,$|V|$ 代表网络拓扑中路由节点数目,$|E|$ 代表链路数目),并采用 Softmin 算法计算数据流链路分配比。智能体以历史流量需求矩阵作为状态信息,并以历史链路利用率与最优链路利用率的比值作为奖励,基于 TRPO 算法进行强化学习训练。在仿真实验中,Valadarsky 等人基于稀疏/非稀疏的重力/双峰模型生成不同类型的流量需求矩阵来验证算法性能。相比于比较算法,作者提出的算法能取得最高的网络链路利用率,并在性能表现上接近最优路由规划。

Yu 等人[64]基于 DDPG 算法设计了通用的 SDN 路由优化算法 DROM。强化学习状态空间为当前网络负载的流量需求矩阵,动作空间为链路的权重,奖励函数为预设的系统效用函数(如时延、吞吐量

等)。Yu 等人采用 Sprint network 骨干网络[65]作为网络拓扑,并基于重力模型生成多个流量需求矩阵。在不同的网络负载下,DROM 收敛效果稳定,并相比于 OSPF 算法能实现更低的传输时延。

Sun 等人[66]结合 RNN 和 DDPG 算法,设计 SDN 路由优化算法 TIDE,以处理具有时间序列特性的网络状态。算法架构分为三层,由下到上分别是数据层实现数据流的传输、控制层收集网络状态和传递路由决策、AI 决策层实现路由策略的生成更新。算法循环由三部分组成,分别为状态与奖励的收集、策略执行和策略提升。强化学习状态空间为网络状态矩阵,动作空间为链路权重矩阵,奖励函数为可自定义的 QoS 函数(即为时延、吞吐量、丢包率的加权和)。智能体以 RNN 作为决策神经网络,以提取时序相关的网络状态中的特征信息。Sun 等人在网络拓扑中部署真实的终端、服务器和路由器,以验证算法的有效性。在不同的网络负载下,TIDE 相比于最短路径算法能实现更低的传输时延,且算法复杂度低、计算速度快。

2.4 TCP 拥塞控制

TCP 是有连接的可靠数据传输协议。拥塞控制是 TCP 的核心部分,直接影响着 TCP 的传输性能。如图 6 所示,在传统方法中,部署于 TCP 数据发送端的拥塞控制算法通过发送数据包而获得关于网络状况的反馈信息。基于这些反馈信息,拥塞控制算法调节拥塞控制窗口或者数据发送速率,从而避免网络拥塞,提高数据传输效率。拥塞控制算法需要在保证高带宽利用率、低时延的同时,也能够保证算法的公平性和稳定性。目前互联网中使用比较广泛的传统拥塞控制算法是 Cubic[67]和 BBR[68]。

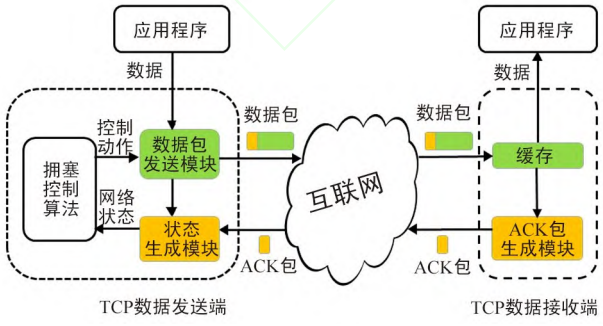

图 6 TCP 拥塞控制原理框图

Fig. 6 Framework of TCP congestion control

强化学习应用于 TCP 拥塞控制的相关研究工作很早就已经开始了,例如文献[69-70]。随着深度神经网络的引入,基于 DRL 的拥塞控制得到了学术界更多的关注。本节把基于 DRL 的拥塞控制研究工作粗略地分为如下三类:面向 TCP 整体性能的拥塞控制、针对特定传输问题的拥塞控制和与传统算法相结合的拥塞控制。

2.4.1 面向 TCP 整体性能的拥塞控制

这类研究工作的特点是,抛弃传统拥塞控制决策机制,让 DRL 的决策网络来选择拥塞控制的调节动作。这些工作的目的是提高传统互联网架构中 TCP 数据传输的整体性能,例如高吞吐量和低时延,并尽可能保证一定的公平性和鲁棒性。因此这些较早期的工作更加注重于如何选择强化学习的状态、动作、奖励函数及训练算法等。

Li 等人[71]在 2018 年提出了基于 Q-learning 的自适应拥塞控制算法 QTCP,其强化学习框架包含连续的状态空间。这些状态分别是发送包的平均间隔时间、接收到 ACK 包的平均间隔时间和平均 RTT。动作空间则为离散的,分别为对拥塞窗口增加 10 Byte、减小 1 Byte 和保持不变。QTCP 的效用函数定义为:

$$U = a \cdot \log(throughput) - b \cdot \log(RTT)。 \quad (19)$$

之后根据当前时间段的效用函数相对于前一时间段是否升高而选择一个正值或负值常数作为当前时间步的奖励。在多种网络场景下,QTCP 表现出优于 TCP NewReno 的性能。

Yan 等人[72]在 2018 年提出了基于模仿学习的拥塞控制算法 Indigo。Indigo 的离线训练在可重复实验的仿真器 Pantheon[72]中开展,因此可以获取网络场景信息作为专家知识,从而使用模仿学习作为其训练算法。Indigo 使用单层的 LSTM 网络作为决策网络,并且采用连续的状态空间,包括排队延迟的指数加权移动平均值、发送速率的指数加权移动平均值、接收速率的指数加权移动平均值、当前拥塞窗口的大小以及前一步采取的动作。其动作为离散的 5 个动作,它们分别是对拥塞窗口进行“/2”,“-10”,“+0”,“+10”,“×2”操作。Indigo 在训练过的场景中表现出优越的性能。但是由于需要针对每个训练场景精心选择专家知识,Indigo 只能在有限的网络场景中训练。

Jay 等人[73]在 2019 年提出了基于 DRL 的拥塞控制算法 Aurora。Aurora 的输入状态包括延迟梯度、延迟比和发送比率。Aurora 把神经网络的输入状态设置为当前时刻之前 10 个时间片的状态组合,从而获取更充分的历史信息。此外,Aurora 也改变了之前基于离散动作的速率调节方式,而是用如下

的公式去连续地调节发送速率。

$$x_t = \begin{cases} x_{t-1}(1+\alpha a_t), & a_t \geqslant 0 \\ \dfrac{x_{t-1}}{1+\alpha a_t}, & a_t < 0 \end{cases} \tag{20}$$

式中，x_t 为数据包的发送速率，a_t 为神经网络的输出，α 为常数。Aurora 的决策网络为两层全连接神经网络。Aurora 使用 PPO 算法进行训练。将 Aurora 与 TCP Cubic 和 TCP BBR 等算法进行对比，发现 Aurora 在固定带宽和动态带宽的单链路网络场景下，都表现出超过传统算法的性能。

2.4.2 针对特定传输问题的拥塞控制

本节的关注点为，在设计 DRL 算法时，如何有效结合互联网中 TCP 数据传输的网络特征或流量特征，或者如何解决强化学习过程与拥塞控制过程的不匹配问题。这部分所要介绍的工作包括：针对短流的 DRL 算法、融合多流 TCP 的 DRL 算法，以及两个关于解决延迟问题的工作。

Nie 等人[74]对百度移动搜索服务的数据流进行测量，发现超过 80%的 TCP 数据流都在算法的慢启动阶段便已经传输完成。因此对于这些数据流来说，决定其带宽利用率等性能的关键要素是初始拥塞窗口，而不是拥塞控制算法的速率调节机制。基于这样的网络流特征，他们在 2019 年提出 TCP-RL[74]来动态地选择初始拥塞窗口。TCP-RL 把初始拥塞窗口的选择看做一个多臂老虎机问题，并用折扣 UCB 算法对其进行训练。而在速率调节阶段，TCP-RL 并未使用 DRL 的神经网络来直接产生调节动作，而是使用神经网络的输出来选择现存的 14 个拥塞控制算法之一，作为当前时间段的调节算法。研究人员把 TCP-RL 的初始拥塞窗口选择机制部署在百度移动搜索服务的一个数据中心。超过一年的测量显示，TCP-RL 可以提高 23%的平均 TCP 响应时间。

多路径 TCP 允许数据发送端连接多个路径来最大化带宽资源利用率。Xu 等人[75]于 2019 年提出了基于 DRL 的多路径 TCP 拥塞控制算法——DRL-CC。这一算法选择 LSTM 网络作为 DRL 的决策网络，并使用 Actor-critic 算法进行训练。每个 TCP 子流计算一个时间片的发送速率、有效吞吐量、平均 RTT、平均 RTT 偏差和拥塞窗口作为当前时间片的状态。所有 TCP 子流的过去 N 个时间片的所有状态组合在一起作为 DRL 的输入状态。在每个决策的时间点，DRL-CC 只对一个子流产生一个连续的调节动作。Xu 等人将 DRL-CC 部署在 Linux 内核的多流 TCP 协议中，并在简单的链路设置中测试其与其他常见的多路径 TCP 拥塞控制算法。实验表明，DRL-CC 在不牺牲公平性的情况下取得了更高的吞吐量。类似的基于 DRL 的多路径 TCP 拥塞控制算法还有 Li 等人在 2019 年提出的基于 Q-learning 的 SmartCC[76]。

Xiao 等人[77]认为，对于拥塞控制问题来说，数据发送端执行动作后，需要经过特定的时间延迟之后才能得到对应的奖励和下一个状态。而数据发送端强化学习时间步长的长度并不与这个时间延迟一致。为了尽快适应网络变化情况，时间步长被设置得更小。因此这并不是一个标准的强化学习过程。Xiao 等人提出的 Drinc[77]使用特殊的缓存回放机制来解决这个延迟问题，即每个时间步得到的(状态，动作，奖励)数据对都会保存在缓存中，并设定当前时间步测量的 RTT 作为前面所述的时间延迟的一个估计，然后根据这个延迟估计在缓存中重新组合成合理的数据对(状态，动作，奖励，下一个状态)来用于训练。除此之外，Drinc 选择使用连续的 64 个时间片的状态组合作为神经网络的输入状态，并且使用节点更多、结构更复杂(包括全连接层、LSTM 层、卷积层、池化层等)的神经网络作为决策网络。有研究人员提出基于 DRL 的拥塞控制算法无法部署的主要障碍是，神经网络做出决策需要一定的时间，而在这个时间内不能处理的网络拥塞控制请求。为解决这个问题，Sivakumar 等人提出 MVFST-RL[78]，并在 QUIC[79]中实现并训练。MVFST-RL 使用异步的强化学习训练方法，并用 Off-policy 的方法来修正。

2.4.3 与传统算法相结合的拥塞控制

纯数据驱动的 DRL 拥塞控制算法面临低稳定性、低适应性的问题。传统拥塞控制算法经过数年的实际部署和运行，已经证明了它们的稳定性。近期的研究工作已经开始在 DRL 拥塞控制算法的设计中融入传统拥塞控制算法(Cubic，BBR 等)的领域知识。例如，借助传统拥塞控制算法来加速训练 DRL 智能体，或是用 DRL 直接对传统拥塞控制算法的决策机制进行改进和优化。前文提到的 TCP-RL 调节机制设计已经用到了类似的思想。

传统基于拥塞的算法，例如 Cubic 等只有在路由器的缓存过大时才会降低拥塞控制窗口，即它们调节的拥塞窗口与最优的相比通常较大。因此，Abbasloo 等人提出 DeepCC[80]，用 DRL 给传统拥塞算法的调解结果制定一个最大值，来优化传统算法。

仿真结果显示,DeepCC 可以在不牺牲带宽利用率的情况下,提高传统算法的平均 RTT。

Emara 等人于 2020 年提出的 Eagle[81],借助 BBR 算法的专家知识来优化训练的 DRL 拥塞控制算法。首先 Eagle 参照 BBR 的状态设计,为基于 DRL 的调节机制也设计了类似的指数启动、排空队列和带宽探测三个不同的状态阶段。其次,在神经网络的训练过程中,Eagle 使用一个同步 BBR 算法来周期性地产生一些调节动作投入训练中,从而加速训练过程,优化动作选择策略。实验结果显示,Eagle 的性能优于其他经典算法,但并没有明显优于 BBR。

Abbasloo 等人提出了基于 Cubic 和 DRL 的混合算法[82]。他们首先测试了单纯基于 DRL 的拥塞控制算法和 Cubic 在动态链路的性能表现,发现这些基于 DRL 的算法存在收敛速度慢等问题。同时考虑计算量等问题,Abbasloo 等人认为 Cubic 用低计算量便实现了在时间域中精细的调节。因此在这一混合算法中,DRL 在每个固定时间周期中收集信息,然后计算出一个基础拥塞控制窗口,之后在下一个时间周期中,Cubic 基于这个拥塞控制窗口数值根据自身的控制逻辑进行调节。本文的这个时间周期被设定为 20 ms,当然也存在更优的动态调节时间周期的机制。实验表明,DRL 在这个混合算法中可以帮助 Cubic 更快地适应网络带宽变化。

2.5 网络缓存

在互联网内容分发服务中,很小一部分网页、文件或视频被大量用户访问,而绝大部分互联网内容却极少被重复访问,即内容的热度呈现“扭曲”特性。热度的差异导致移动互联网的数据流量呈现很大的冗余性,特别是当网络带宽成为瓶颈时,互联网的内容分发服务质量显著降低。随着存储技术的进步,磁盘存储系统容量越来越大,单位存储容量的价格显著降低,这为网络缓存(Cache)技术提供了重要前提。网络内缓存技术将用户访问过的部分视频段存储在网络边缘,使得这些视频段的后续请求本地化,降低了服务器端的流量和内容传输的时间延迟,提高了网络服务性能。缓存技术的最重要性能指标是用户请求的命中率。命中率越高,能够节约的带宽需求越大,相应的内容传输延迟可能会越小。

缓存研究的核心问题是“存储什么内容”。广义来说,内容缓存的策略分为“被动式(Reactive)”和“主动式(Proactive)”两类。被动式内容缓存是被动地根据用户产生的请求做出相应的决策,即依据预先规定的缓存机制,决定文件的存入与移除,其关注的核心是缓存内容的替换策略(Cache Replacement Policy)。缓存中的数据如何更新需要考虑以下要素:文件最近一次被请求的时间、文件的大小以及文件被请求的频次。常用的被动式缓存替换策略有“最近最少使用”(Least Recently Used,LRU)、“先入先出”(First In First Out,FIFO)、“后入先出”(Last In First Out,LIFO)、“随机替换”(Random Replacement,RR)、“最低使用频率”(Least Frequently Used,LFU)以及“生存时间”(Time-to-Live,TTL)等。主动式缓存策略是在对用户需求进行预测的基础上,主动进行缓存内容的推送与更新。本节聚焦于介绍互联网中常用的被动式缓存替换策略的 DRL 算法设计。

DeepCache[84]的模型包括两部分:基于 LSTM 编码器模型的目标特征预测器和缓存替换管理模块。其目标在于提前预测出到达请求的目标特征,自动学习到达流量模式的变化(特别是爆发式和非稳态的流量),预测流行度变化等。DeepCache 采用 LSTM 神经网络,将视频流行度预测视作一个 Seq2seq 问题。输入输出均为 3D 张量,即输入中的元素为所有个体目标在一个时间窗口内的概率,输入维度是个体目标的数量,输出为未来若干个时间单位内所有个体目标的概率分布。在合成的访问请求数据实验上,DeepCache 表现出比 LRU 和 LFU 显著优越的性能。

RL-Cache[85]研究采用 DRL 设计缓存接纳策略,即当一个没有被缓存的请求到达时,是否要将该请求内容接纳至缓存中,缓存的剔除策略依旧采用 LRU。本文将缓存接纳问题建模成无模型的强化学习问题,采用前馈神经网络(FNN)结合蒙特卡洛采样及随机优化来求解。对于每一个输入目标,状态特征为该目标的大小、出现频率、最近一次请求时间间隔、指数平滑后的请求时间间隔、自上次访问该目标以来的其他目标访问次数、该访问次数的指数平滑、该目标的访问频次与目标大小的比值、访问频次和目标大小的乘积。在训练过程中,为了使训练的开销保持在较低水平,RL-Cache 从第一个请求开始,然后每次向后滑动 K 个请求。

Kirilin 等人[86]提出使用前馈神经网络预测内容流行度,并运用最小堆去实现 LFU 的缓存替换策略。输入特征向量为一个内容在过去 K 个时间片内的被请求概率。实验结果表明,基于 FNN 的 Cache 性能优于 LRU 和 LFU,并且 FNN 的性能优于

两种 LSTM 流行度预测算法。然后,Kirilin 的进一步研究表明,当使用更为简单的线性估计方法时,缓存性能下降很小,这使得深度学习用于缓存算法设计受到一定程度的质疑。

Wang 等人[87]提出采用纯强化学习设计缓存算法。输入状态为一个四元组:到达请求的大小、上次该内容被请求的时间间隔、该目标迄今为止的访问次数和剩余的缓存大小比例。所采用的训练方法为标准的 A2C 强化学习算法。强化学习的输出动作为是否接纳一个未在缓存的新请求。奖励为自上次决策以来总的命中字节数。其研究重点在于利用欠采样的方法将强化学习缓存策略应用至大型缓存系统中。具体方法是按照 $1/K$ 的比例从数据集采样数据,欠采样的方法是对目标的 ID 进行随机的哈希,同时将缓存容量缩小 K 倍。数据驱动的实验结果表明,欠采样的强化学习缓存策略与原来的强化学习缓存策略性能接近。

Zhong 等人[87]设计了基于 DRL 的缓存替换策略。该 DRL 模型的神经网络为双侧全连接结构,强化学习模型基于 Wolpertinger 策略,包括 3 个部分:Actor 网络、K 最近邻算法(KNN)和 Critic 网络,训练的方法为 DDPG。采用 Wolpertinger 架构的原因是,作为在线算法,该架构能够适应数据的变化规律;Actor 网络能够避免考虑一个非常巨大的动作空间,而 Critic 网络能够纠正演员网络(actor network)的决策,并且 KNN 方法能够拓展动作的选择范围。Zhong 等人假设文件数量是固定的,并且每个文件的大小相同。其状态空间包含缓存内容在短期、中期和长期时间内总的请求数量,奖励设置为缓存短期命中率和长期命中率的加权和,动作空间是使用当前请求内容替换某个已缓存内容。

3 挑战

尽管这些基于 DRL 的方法在计算机网络的各个领域均取得了较好的性能,但仍没有替换传统算法实现大规模的实际部署。针对基于专家知识的“白盒”算法,其内部的工作原理和判定逻辑清晰透明,一旦算法出错网络工程师可以立即进行修正。针对基于 DRL 的“黑盒”算法,在智能体和环境的交互过程中,给定一个输入状态,通过策略网络或 Q 网络的输出来确定决策动作,其中策略网络和 Q 网络的本质都是神经网络,而一个神经网络通常由成千上万的参数经过非线性激活函数组成。这种给定输入即生成输出的黑盒行为以及复杂的神经网络结构带来了新的挑战,包括算法的可解释性、算法的鲁棒性以及同领域知识的有效融合问题,下面将分别对这几点进行叙述。

3.1 可解释性

针对基于神经网络的计算机网络系统的可解释性,Zheng 等人[88]于 2018 年使用基于隐含层神经元最大激活的方式去分析神经网络在执行任务调度时对任务请求时长的敏感程度。但是,计算机网络的工程师更倾向于直接使用简单的逻辑规则去表示神经网络,而不是去分析其复杂的激活规律。为了让网络工程师可以更直接地对这些基于神经网络的系统进行调试、修正和部署,Meng 等人于 2019 年提出了 PiTree[89],将基于 DRL 的视频传输系统 Pensieve 转换为决策树。Meng 等人于 2020 年又提出了通用性更强的 Metis[90],对于搭建在客户端的小型系统,如 Pensieve[44]和 Aurora[73],Metis 通过将神经网络转化成决策树的方式提供可解释性;对于进行规模控制的大型系统,如 Decima[25],Meitis 可将其转化为超图(Hypergraph)。实验结果显示,Metis 可以在不降低系统性能的条件下提供更高程度的可解释性。然而,Metis 目前还不能解决对于 RNN 这种复杂神经网络结构,而且决策树的构建仍需要消耗大量的时间,如何简单高效地提供通用性神经网络的可解释性仍然是一个待解决的关键问题。

3.2 鲁棒性问题

除了缺乏可解释性外,基于 DRL 的计算机网络系统也存在鲁棒性的问题。由于计算机网络环境的多变性,环境条件经常会发生变化。如在任务调度过程中任务的到达分布发生改变,文献[91]指出此时基于 DRL 的负载均衡(Load Balance)系统的性能出现了大幅度下降,该解决方法是使用一个性能更稳定的传统算法作为后盾,当发现 DRL 智能体的性能出现下降时,则使用传统算法替代智能体进行决策。但处在输入驱动的计算机网络环境中,如何准确判断性能是否真的出现下降,以及能否直接训练一个可以容忍环境变化和通用的智能体仍是待解决的问题。

3.3 DRL 和领域知识的融合

对于网络系统来说,领域知识是许多年的研究积累,是非常充沛的。它们可以帮助 DRL 算法变得更加可靠,降低模型推断的复杂度而不损害其性能。然而,目前的“黑盒模式”在将网络系统状态和决策编码成输入和输出后,DRL 的训练和决策与领域知识往往无关。

上一节中提到的使用传统算法作为 DRL 智能体的后盾,本质上是希望能将领域知识和智能体进行融合。这些领域知识虽然可通过智能体无尽地探索与试错得到,但是直接高效地融合领域知识可以使智能体获取更快的训练速度和更高的性能表现。针对这个研究领域目前已有了一些初步的成果,Eagle[81]借助 BBR 算法的领域知识来优化训练的 DRL 拥塞控制算法,它使用一个同步 BBR 算法来周期性地产生一些调节动作投入训练中,从而加速训练过程,优化动作选择策略。然而,从诸多文章的实验测试结果可以看出,DRL 智能体的性能是超过传统算法的,在训练的起始阶段,融入传统算法的建议可以加速训练速度,但是过多融入传统算法的建议势必会对智能体的性能带来负面影响,如何合理地掌握传统算法的参与程度也是需要解决的问题。

4 结束语

由于 DRL 可以避免对网络环境进行不准确的建模,且智能体与环境的自主交互过程可以节省网络专家对基于规则算法的频繁参数配置,近年来 DRL 在计算机网络中的应用不胜枚举。本文首先对 DRL 技术原理进行介绍,详细阐述了多种典型的神经网络结构、传统的以表格形式求解强化学习问题的方法以及广泛应用的 DRL 算法;其次,针对目前使用 DRL 技术的典型网络系统进行了全面的综述,包括任务和数据流调度、视频传输、路由选择、TCP 拥塞控制和网络缓存;最后,指出了目前这些使用 DRL 的计算机网络系统面临的新挑战,包括可解释性问题、鲁棒性问题以及同领域知识进行融合的问题,并给出了在这些问题上已有的工作进展。

参 考 文 献

[1] 刘全,翟建伟,章宗长,等. 深度强化学习综述[J]. 计算机学报,2018,41(1):1-27.

[2] 邱锡鹏. 神经网络与深度学习[M]. 北京:机械工业出版社,2020:78-150.

[3] SUTTON R S, BARTO A G. Reinforcement learning: An Introduction[M]. Cambrige, MA: MIT Press, 2018:23-191.

[4] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-Level Control through Deep Reinforcement Learning[J]. Nature, 2015, 518(7540):529-533.

[5] VAN HASSELT H, GUEZ A, SILVER D. Deep Reinforcement Learning with Double Q-learning [EB/OL]. https://arxiv.org/pdf/1509.06461.pdf.

[6] SCHAUL T, QUAN J, ANTONOGLOU I, et al. Prioritized Experience Replay [EB/OL]. https://arxiv.org/pdf/1511.05952.pdf.

[7] WANG Z, SCHAUL T, HESSEL M, et al. Dueling Network Architectures for Deep Reinforcement Learning[C]//International Conference on Machine Learning. New York: ACM, 2016:1995-2003.

[8] HAUSKNECHT M, STONE P. Deep Recurrent Q-learning for Partially Observable MDPs[EB/OL]. https://arxiv.org/pdf/1507.06527.pdf.

[9] SUTTON R S, MCALLESTER D A, SINGH S P, et al. Policy Gradient Methods for Reinforcement Learning with Function Approximation[C]//Advances in Neural Information Processing Systems Cambrige, MA: MIT Press, 2000:1057-1063.

[10] KONDA V R, TSITSIKLIS J N. Actor-critic Algorithms [C]//Advances in Neural Information Processing Systems Cambrige, MA: MIT Press, 2000:1008-1014.

[11] MNIH V, BADIA A P, MIRZA M, et al. Asynchronous Methods for Deep Reinforcement Learning[C]//International Conference on Machine Learning. New York: ACM, 2016:1928-1937.

[12] SCHULMAN J, LEVINE S, ABBEEL P, et al. Trust Region Policy Optimization[C]//International Conference on Machine Learning. Lille: ACM, 2015:1889-1897.

[13] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal Policy Optimization Algorithms [EB/OL]. https://arxiv.org/pdf/1707.06347.pdf.

[14] HEESS N, TB D, SRIRAM S, et al. Emergence of Locomotion Behaviours in Rich Environments[EB/OL]. https://arxiv.org/pdf/1707.02286.pdf.

[15] LILLICRAP T P, HUNT JJ, PRITZEL A, et al. Continuous Control with Deep Reinforcement Learning[EB/OL]. https://arxiv.org/pdf/1509.02971.pdf.

[16] GU S, LILLICRAP T, SUTSKEVER I, et al. Continuous Deep Q-learning with Model-based Acceleration[C]// International Conference on Machine Learning. New York: ACM, 2016:2829-2838.

[17] ARULKUMARAN K, DEISENROTH M P, BRUNDAGE M, et al. Deep Reinforcement Learning: A Brief Survey [J]. IEEE Signal Processing Magazine, 2017, 34(6):26-38.

[18] 刘建伟,高峰,罗雄麟. 基于值函数和策略梯度的深度强化学习综述[J]. 计算机学报, 2019, 42(6):1406-1438.

[19] MAO H, ALIZADEH M, MENACHE I, et al. Resource Management with Deep Reinforcement Learning[C]// Proceedings of the 15th ACM Workshop on Hot Topics in Networks. GA: ACM, 2016:50-56.

[20] CHEN W,XU Y,WU X. Deep Reinforcement Learning for Multi-resource Multi-machine Job Scheduling[EB/OL]. https://arxiv.org/ftp/arxiv/papers/1711/1711.07440.pdf.

[21] LI F,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. Shenzhen:ACM,2019:48-53.

[22] MASTROLILLI M,SVENSSON O. (Acyclic) Job Shops Are Hard to Approximate[C]//2008 49th Annual IEEE Symposium on Foundations of Computer Science. Philadelphia:IEEE,2008:583-592.

[23] MASTROLILLI M,SVENSSON O. Improved Bounds for Flow Shop Scheduling[C]//International Colloquium on Automata,Languages,and Programming. Berlin:Springer,2009:677-688.

[24] HU Z,TU J,LI B. Spear:Optimized Dependency-Aware Task Scheduling with Deep Reinforcement Learning[C]//2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). Texas:IEEE,2019:2037-2046.

[25] MAO H,SCHWARZKOPF M,VENKATAKRISHNAN S B,et al. Learning Scheduling Algorithms for Data Processing Clusters[C]//Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM). Beijing:ACM,2019:270-288.

[26] WU Q,WU Z,ZHUANG Y,et al. Adaptive Dag Tasks Scheduling with Deep Reinforcement Learning[C]//International Conference on Algorithms and Architectures for Parallel Processing. Guangzhou:Springer,2018:477-490.

[27] ZHANG L,QI Q,WANG J,et al. Multi-task Deep Reinforcement Learning for Scalable Parallel Task Scheduling[C]//2019 IEEE International Conference on Big Data. California:IEEE,2019:2992-3001.

[28] ZHANG W,WEN Y,WONG Y W,et al. Towards Joint Optimization over ICT and Cooling Systems in Data Center:A survey[J]. IEEE Communications Surveys & Tutorials,2016,18(3):1596-1616.

[29] RAN Y,HU H,ZHOU X,et al. DeepEE:Joint Optimization of Job Scheduling and Cooling Control for Data Center Energy Efficiency Using Deep Reinforcement Learning[C]//2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). Texas:IEEE,2019:645-655.

[30] ESMAILI A,PEDRAM M. Energy-aware Scheduling of Jobs in Heterogeneous Cluster Systems Using Deep Reinforcement Learning[C]//2020 21st International Symposium on Quality Electronic Design (ISQED). California:IEEE,2020:426-431.

[31] LIU N,LI Z,XU J,et al. A Hierarchical Framework of Cloud Resource Allocation and Power Management Using Deep Reinforcement Learning[C]//2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). GA:IEEE,2017:372-382.

[32] YI D,ZHOU X,WEN Y,et al. Toward Efficient Compute-Intensive Job Allocation for Green Data Centers:A Deep Reinforcement Learning Approach[C]//2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). Texas:IEEE,2019:634-644.

[33] 胡智尧,李东升,李紫阳. 数据中心网络流调度技术前沿进展[J]. 计算机研究与发展,2018,55(9):1920-1930.

[34] SINGH A,ONG J,AGARWAL A,et al. Jupiter Rising:A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network[J]. ACM SIGCOMM Computer Communication Review,2015,45(4):183-197.

[35] CHEN L,LINGYS J,CHEN K,et al. Auto:Scaling Deep Reinforcement Learning for Datacenter-scale Automatic Traffic Optimization[C]//Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication(SIGCOMM). Budapest:ACM,2018:191-205.

[36] SUN P,GUO Z,WANG J,et al. DeepWeave:Accelerating Job Completion Time with Deep Reinforcement Learning-based Coflow Scheduling[C]//Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. Yokohama:Morgan Kaufmann,2020:3314-3320.

[37] HUANG T Y,JOHARI R,MCKEOWN N,et al. A buffer-Based Approach to Rate Adaptation:Evidence from a Large Video Streaming Service[C]//Proceedings of the 2014 ACM conference on SIGCOMM. Chicago:ACM,2014:187-198.

[38] LIU C,BOUAZIZI I,GABBOUJ M. Rate Adaptation for Adaptive HTTP Streaming[C]//Proceedings of the Second Annual ACM Conference on Multimedia Systems. California:ACM,2011:169-174.

[39] CLAEYS M,LATRÉ S,FAMAEY J,et al. Design of a Q-learning-based Client Quality Selection Algorithm for HTTP Adaptive Video Streaming[C]//Adaptive and Learning Agents Workshop,Part of AAMAS2013 (ALA-2013). Paul:Spring,2013:30-37.

[40] CLAEYS M,LATRÉ S,FAMAEY J,et al. Design and Optimisation of a (FA) Q-learning-based HTTP Adaptive Streaming Client[J]. Connection Science,2014,26(1):25-43.

[41] MARTÍN V,CABRERA J,GARCÍA N. Evaluation of Q-learning Approach for HTTP Adaptive Streaming[C]//2016 IEEE International Conference on Consumer Electronics (ICCE). Nantou:IEEE,2016:293-294.

[42] MARTÍN V, CABRERA J, GARCÍA N. Design, Optimization and Evaluation of a Q-Learning HTTP Adaptive Streaming Client[J]. IEEE Transactions on Consumer Electronics, 2016, 62(4): 380-388.

[43] GADALETA M, CHIARIOTTI F, ROSSI M, et al. D-DASH: A Deep Q-learning Framework for DASH Video Streaming[J]. IEEE Transactions on Cognitive Communications and Networking, 2017, 3(4): 703-718.

[44] MAO H, NETRAVALI R, ALIZADEH M. Neural adaptive Video Streaming with Pensieve[C]//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. California: ACM, 2017: 197-210.

[45] HUANG T, ZHOU C, ZHANG R X, et al. Stick: A Harmonious Fusion of Buffer-based and Learning-based Approach for Adaptive Streaming[C]//IEEE INFOCOM 2020-IEEE Conference on Computer Communications. Beijing: IEEE, 2020: 1967-1976.

[46] GUAN Y, ZHANG Y, WANG B, et al. PERM: Neural Adaptive Video Streaming with Multi-path Transmission [C]//IEEE INFOCOM 2020-IEEE Conference on Computer Communications. Beijing: IEEE, 2020: 1103-1112.

[47] ZHANG Y, ZHAO P, BIAN K, et al. DRL360: 360-degree Video Streaming with Deep Reinforcement Learning[C]// IEEE INFOCOM 2019-IEEE Conference on Computer Communications. Paris: IEEE, 2019: 1252-1260.

[48] FU J, CHEN X, ZHANG Z, et al. 360srl: A Sequential Reinforcement Learning Approach for Abr Tile-Based 360 Video Streaming[C]//2019 IEEE International Conference on Multimedia and Expo (ICME). Shanghai: IEEE, 2019: 290-295.

[49] HOOFT J, PETRANGELI S, CLAEYS M, et al. A Learning-Based Algorithm for Improved Bandwidth-Awareness of Adaptive Streaming Clients[C]//2015 IFIP/IEEE International Symposium on Integrated Network Management (IM). Washington: IEEE, 2015: 131-138.

[50] HUANG T, ZHANG R X, ZHOU C, et al. Qarc: Video Quality Aware Rate Control for Real-Time Video Streaming Based on Deep Reinforcement Learning[C]//Proceedings of the 26th ACM International Conference on Multimedia. Seoul: ACM, 2018: 1208-1216.

[51] BOYAN J A, LITTMAN M L. Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach[C]//Advances in Neural Information Processing Systems. Colorado: MIT Press, 1994: 671-678.

[52] CHOI S P M, YEUNG D Y. Predictive Q-routing: A Memory-Based Reinforcement Learning Approach to Adaptive Traffic Control[C]// Advances in Neural Information Processing Systems. Denver: MIT Press, 1996: 945-951.

[53] KUMAR S, MIIKKULAINEN R. Dual Reinforcement Q-Routing: An On-Line Adaptive Routing Algorithm[C]//Proceedings of the Artificial Neural Networks in Engineering Conference. Lausanne: Spring, 1997: 231-238.

[54] STAMPA G, ARIAS M, SÁNCHEZ-CHARLES D, et al. A Deep-Reinforcement Learning Approach for Software-Defined Networking Routing Optimization [EB/OL]. https://arxiv.org/pdf/1709.07080.pdf.

[55] VALADARSKY A, SCHAPIRA M, SHAHAF D, et al. Learning to Route[C]//Proceedings of the 16th ACM Workshop on Hot Topics in Networks. California: ACM, 2017: 185-191.

[56] MUKHUTDINOV D, FILCHENKOV A, SHALYTO A, et al. Multi-agent Deep Learning for Simultaneous Optimization for Time and Energy in Distributed Routing System[J]. Future Generation Computer Systems, 2019, 94: 587-600.

[57] YOU X Y, LI X J, XU Y D, et al. Toward Packet Routing with Fully Distributed Multiagent Deep Reinforcement Learning[J/OL]. [2019-11-14]. http://arxiv.org/abs/1905.03494.

[58] KNIGHT S, NGUYEN H X, FALKNER N, et al. The Internet Topology Zoo[J]. IEEE Journal on Selected Areas in Communications, 2011, 29(9): 1765-1775.

[59] TASSIULAS L, EPHREMIDES A. Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks[C]// 29th IEEE Conference on Decision and Control. Honolulu: Hawaii: IEEE, 1990: 2130-2132.

[60] ALI R E, ERMAN B, BAŞTU E, et al. Hierarchical Deep Double Q-Routing[C]//ICC 2020-2020 IEEE International Conference on Communications (ICC). Dublin: IEEE, 2020: 1-7.

[61] VAN HASSELT H, GUEZ A, SILVER D. Deep Reinforcement Learning with Double Q-Learning [EB/OL]. https://arxiv.org/pdf/1509.06461.pdf.

[62] XU Z, TANG J, MENG J, et al. Experience-driven Networking: A Deep Reinforcement Learning Based Approach [C]//IEEE INFOCOM 2018-IEEE Conference on Computer Communications. Honolulu: IEEE, 2018: 1871-1879.

[63] HENDERSON T R, LACAGE M, RILEY G F, et al. Network Simulations with the Ns-3 Simulator[J]. SIGCOMM Demonstration, 2008, 14(14): 527.

[64] YU C, LAN J, GUO Z, et al. DROM: Optimizing the Routing in Software-Defined Networks with Deep Reinforcement Learning[J]. IEEE Access, 2018, 6: 64533-64539.

[65] FRALEIGH C, MOON S, LYLES B, et al. Packet-level Traffic Measurements from the Sprint IP Backbone[J]. IEEE Network, 2003, 17(6): 6-16.

[66] SUN P, HU Y, LAN J, et al. TIDE: Time-Relevant Deep Reinforcement Learning for Routing Optimization [J]. Future Generation Computer Systems, 2019, 99: 401-409.

[67] HA S, RHEE I, XU L. CUBIC: A New TCP-Friendly High-Speed TCP Variant [J]. ACM SIGOPS Operating Systems Review, 2008, 42(5): 64-74.

[68] CARDWELL N, CHENG Y, GUNN C S, et al. BBR: Congestion-Based Congestion Control [J]. Queue, 2016, 14(5): 20-53.

[69] TARRAF AA, HABIB I W, SAADAWI T N. Reinforcement Learning-Based Neural Network Congestion Controller for ATM Networks [C]//Proceedings of MILCOM'95. California: IEEE, 1995, 2: 668-672.

[70] HWANG K S, WU C S, SU H K. Reinforcement Learning Cooperative Congestion Control for Multimedia Networks [C]//2005 IEEE International Conference on Information Acquisition. Hong Kong: IEEE, 2005: 221-226.

[71] LI W, ZHOU F, CHOWDHURY K R, et al. QTCP: Adaptive Congestion Control with Reinforcement Learning [J]. IEEE Transactions on Network Science and Engineering, 2018, 6(3): 445-458.

[72] YAN F Y, MA J, HILL G D, et al. Pantheon: the Training Ground for Internet Congestion-Control Research [C]// 2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18). Boston: USENIX, 2018: 731-743.

[73] JAY N, ROTMAN N, GODFREY B, et al. A Deep Reinforcement Learning Perspective on Internet Congestion Control [C]//International Conference on Machine Learning. California: ACM, 2019: 3050-3059.

[74] NIE X, ZHAO Y, LI Z, et al. Dynamic TCP Initial Windows and Congestion Control Schemes Through Reinforcement Learning [J]. IEEE Journal on Selected Areas in Communications, 2019, 37(6): 1231-1247.

[75] XU Z, TANG J, YIN C, et al. Experience-Driven Congestion Control: When Multi-Path TCP Meets Deep Reinforcement Learning [J]. IEEE Journal on Selected Areas in Communications, 2019, 37(6): 1325-1336.

[76] LI W, ZHANG H, GAO S, et al. SmartCC: A Reinforcement Learning Approach for Multipath TCP Congestion Control in Heterogeneous Networks [J]. IEEE Journal on Selected Areas in Communications, 2019, 37(11): 2621-2633.

[77] XIAO K, MAO S, TUGNAIT J K. TCP-Drinc: Smart Congestion Control Based on Deep Reinforcement Learning [J]. IEEE Access, 2019, 7: 11892-11904.

[78] SIVAKUMAR V, ROCKTÄSCHEL T, MILLER A H, et al. MVFST-RL: An Asynchronous RL Framework for Congestion Control with Delayed Actions [EB/OL]. https://arxiv.org/pdf/1910.04054.pdf.

[79] LANGLEY A, RIDDOCH A, WILK A, et al. The QUIC TRANSPORT PROTOCOL: DESIGN and Internet-Scale Deployment [C]//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. California: ACM, 2017: 183-196.

[80] ABBASLOO S, YEN C Y, CHAO H J. Make Tcp Great (Again?!) in cellular networks: A Deep Reinforcement Learning Approach [EB/OJ]. https://arxiv.org/pdf/1912.11735.pdf.

[81] EMARA S, LI B, CHEN Y. Eagle: Refining Congestion Control by Learning from the Experts [C]//IEEE INFOCOM 2020-IEEE Conference on Computer Communications. Beijing: IEEE, 2020: 676-685.

[82] ABBASLOO S, YEN C Y, CHAO H J. Classic Meets Modern: A Pragmatic Learning-Based Congestion Control for the Internet [C]//Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. New York: ACM, 2020: 632-647.

[83] ZHONG C, GURSOY M C, VELIPASALAR S. A Deep Reinforcement Learning-Based Framework for Content Caching [C]//2018 52nd Annual Conference on Information Sciences and Systems (CISS). Princeton: IEEE, 2018: 1-6.

[84] KIRILIN V, SUNDARRAJAN A, GORINSKY S, et al. Rl-Cache: Learning-Based Cache Admission for Content Delivery [C]//Proceedings of the 2019 Wokshop on Network Meets AI&ML. Beijing, ACM, 2019: 57-63.

[85] FEDCHENKO V, NEGLIA G, RIBEIRO B. Feedforward Neural Networks for Caching: N Enough or Too Much? [J]. ACM SIGMETRICS Performance Evaluation Review, 2019, 46(3): 139-142.

[86] WANG H, HE H, ALIZADEH M, et al. Learning Caching Policies with Subsampling [C]//Workshop of 33rd Conference on Neural Information Processing Systems. Vancouver: MIT Press, 2019.

[87] ZHONG C, GURSOY M C, VELIPASALAR S. A Deep Reinforcement Learning-Based Framework for Content Caching [C]//2018 52nd Annual Conference on Information Sciences and Systems (CISS). Princeton: IEEE, 2018: 1-6.

[88] ZHENG Y, LIU Z, YOU X, et al. Demystifying Deep Learning in Networking [C]//Proceedings of the 2nd Asia-Pacific Workshop on Networking. Beijing: ACM, 2018: 1-7.

[89] MENG Z, CHEN J, GUO Y, et al. PiTree: Practical Implementation of ABR Algorithms Using Decision Trees [C]// Proceedings of the 27th ACM International Conference on Multimedia. Nice France: ACM, 2019: 2431-2439.

[90] MENG Z, WANG M, BAI J, et al. Interpreting Deep Learning-Based Networking Systems[C]//Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. New York: SIGCOMM'20, 2020: 154-171.

[91] MAO H, SCHWARZKOPF M, HE H, et al. Towards Safe Online Reinforcement Learning in Computer Systems[C]//Workshop of Conference on Neural Information Processing Systems. Vancouver: NIPS, 2019.

作者简介：

郑　莹　复旦大学计算机科学技术学院博士研究生。主要研究方向：分布式机器学习、深度强化学习。

段庆洋　复旦大学信息科学与工程学院硕士研究生。主要研究方向：深度强化学习、传输层协议。

林利祥　复旦大学信息科学与工程学院硕士研究生。主要研究方向：深度强化学习、流媒体视频传输。

游新宇　复旦大学信息科学与工程学院硕士研究生。主要研究方向：深度强化学习、计算机网络。

徐跃东　复旦大学信息科学与工程学院副研究员。主要研究方向：计算机网络系统、分布式机器学习系统、数据驱动的网络设计、机器学习理论。2001 年获安徽大学自动化系学士学位，于 2004 年获华中科技大学控制科学与工程系硕士，于 2009 年获香港中文大学计算机科学与工程系博士，2009 年至 2012 年期间于法国国家自动化与计算机研究院（INRIA）及 Avignon 大学进行博士后研究。

（*通信作者）**王　新**　复旦大学计算机科学技术学院教授，博士生导师。主要研究方向：新一代互联网体系结构、无线与移动网络、数据中心网络、网络存储系统等。分别于 1994 年和 1997 年获西安电子科技大学学士学位和硕士学位，2002 年获日本静冈大学计算机系博士学位。复旦-谷歌 Android 实验室负责人，复旦-Xilinx SDN 实验室负责人。曾任中国计算机学会理事、互联网专委会副主任委员、信息存储技术专委会常务委员，中国电子学会信息论专委会委员。