基于深度强化学习的转发效能感知流量调度算法

沙宗轩[1]，霍如[1,2]，孙闯[3]，汪硕[2,4]，黄韬[2,4]

（1. 北京工业大学信息学部，北京 100124；2. 网络通信与安全紫金山实验室，江苏 南京 211111；
3. 清华大学自动化系，北京 100084；4. 北京邮电大学网络与交换国家重点实验室，北京 100876）

摘 要：软件定义网络（SDN）通过将控制平面与数据平面分离，可实现灵活的流量调度，更有效地利用网络资源。但是，流表项数量、设备负载率及连接主机数量增加等因素的共同作用会导致 SDN 交换机的转发效能降低，进而影响端到端的数据传输时延。为了解决上述问题，提出了基于深度强化学习的转发效能感知流量调度算法。首先，将交换机状态信息统一到感知模型中，通过神经网络建立交换机状态信息和转发效能的映射关系。然后，结合网络状态和流量信息，通过深度强化学习产生流量调度策略。最后，通过由最短路径和负载均衡算法产生的专家样本引导模型训练，不仅使模型学习到专家样本的知识以提升性能，同时提升模型训练效率。实验结果表明，与其他算法相比，所提算法不仅使端到端的平均传输时延降低了 15.31%，而且保证了网络整体的负载均衡。

关键词：软件定义网络；深度强化学习；流量调度；转发效能感知；负载均衡

中图分类号：TN911.5

文献标志码：A

DOI: 10.11959/j.issn.1000−436x.2022148

Forwarding efficiency aware traffic scheduling algorithm based on deep reinforcement learning

SHA Zongxuan[1], HUO Ru[1,2], SUN Chuang[3], WANG Shuo[2,4], HUANG Tao[2,4]

1. Information Department, Beijing University of Technology, Beijing 100124, China
2. Purple Mountain Laboratories, Nanjing 211111, China
3. Department of Automation, Tsinghua University, Beijing 100084, China
4. State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract: The software defined network separates the control plane from the data plane to achieve flexible traffic scheduling, which can use network resources more efficiently. However, with the increase of the number of flow entries, load rate, the number of connected hosts, and other factors, the forwarding efficiency of the SDN switch will be reduced, which will affect the end-to-end transmission delay. To solve the above problems, the forwarding efficiency aware traffic scheduling algorithm based on deep reinforcement learning was proposed. First, the switch state was integrated into the perception model, and the mapping relationship between switch state information and forwarding efficiency was established based on neural network. Then, combined with network state and traffic information, traffic scheduling policy was generated through deep reinforcement learning. Finally, the expert samples generated by the shortest path and load balance algorithms could guide the model training, which enabled the model to learn knowledge from expert samples to improve performance and accelerated the training process. The experimental results show that the proposed algorithm not only reduces the average end-to-end transmission delay by 15.31%, but also ensures the overall load balance of the network, compared with other algorithms.

Keywords: software defined network, deep reinforcement learning, traffic scheduling, forwarding efficiency aware, load balance

收稿日期：2022−04−26；**修回日期：**2022−07−19

通信作者：霍如，huoru@bjut.edu.cn

基金项目：2020 年工业互联网创新发展工程基金资助项目（工业互联网标识资源搜索系统）；中国通信学会青年人才托举计划−托举培养协议基金资助项目（No.YESS20200287）

Foundation Items: The MIIT of China 2020 (Identification Resources Search System for Industrial Internet of Things), Young Elite Scientist Sponsorship Program by CAST and China-CIC (No.YESS20200287)

0　引言

互联网日益成为人们生产生活中不可或缺的基础设施，随着网络应用和服务的进一步发展，对网络数据流实施有效的监测管理愈发重要，研究人员提出了多种类型的流量工程（TE, traffic engineering）技术以完成网络优化任务。流量工程，或称流量管理，是指针对网络中数据流的行为进行动态的分析预测和有目的的管理[1]。在 20 世纪 80 年代的异步传输模式（ATM, asynchronous transfer mode）网络中，拥塞最小化是 TE 最常见的目标之一[2]。到了 20 世纪 90 年代，IP 网络逐渐成为主流，TE 常用于路由优化；有研究结合 ATM 和 IP 的优点提出了多协议标签交换（MPLS, multiprotocol label switching）技术。与传统 IP 路由方式相比，MPLS 不需要在每一跳都分析 IP 报文头，节约了处理时间。但随着网络需求的不断变化，以 IP 为网络层的体系架构存在扩展性差、缺乏安全机制、不具备虚拟化特征等问题使其难以持续发展[3]。

软件定义网络（SDN, software defined network）将数据平面与控制平面分离，为有效解决上述问题提供了思路。一方面，SDN 控制器具有全局视野，可实现灵活控制流量、降低网络运营成本以及促进网络创新[4-5]。该架构目前已广泛部署在 Google、微软、Facebook 等公司的内网建设中。例如，Google 希望通过 SDN 使资源利用率提升 20%~30%，同时优化网络传输性能。另一方面，不同厂商的设备使用统一的编程接口，可以提供充分的开放性。

近年来，SDN 与深度强化学习（DRL, deep reinforcement learning）算法的结合引起了人们的广泛关注。DRL 是由深度学习（DL, deep learning）[6]与强化学习（RL, reinforcement learning）融合产生的智能决策工具[7]。凭借深度模型强大的表示能力，与传统 RL 算法相比，DRL 在处理具有连续状态空间及动作空间的复杂问题时具有更好的性能，已被证明是一种可行有效的复杂系统自主控制解决方案。利用 DRL 的优势解决动态网络下的流量调度问题正成为领域内的热点[8-9]。

提高网络资源利用率和改善数据传输性能对保障网络服务质量具有重要意义。Hartman 等[10]指出在资源受限的网络环境中输出具有最大流量的链路集是 NP 完全问题。目前解决此问题广泛使用两类算法：开路最短路径优先（OSPF, open shortest path first）算法和负载均衡（LB, load balance）算法，将流量分配到最短路径或考虑负载均衡的传输路径上。流量调度算法通常会面临以下挑战。

1) 根据网络环境变化快速调整策略。由于大量用户请求的网络服务种类繁多，使网络环境快速变化。在复杂动态的网络环境中，传统基于规则的启发式算法适应性较差。各种约束条件也增加了启发式流量调度算法的求解效率。这就要求流量调度算法必须高效，能够快速根据网络环境及时调整输出策略[11-12]。

2) 调度算法需考虑多维因素的影响。基于 RL 的流量调度算法可以适应动态变化的网络环境。在目前的研究中，相关算法常常根据跳数、链路时延及拥塞情况等因素产生流量调度方案。而 SDN 架构中数据包端到端的完整传输时间还包含流量调度策略生成时间和数据包在交换机的转发时延等，传统方法考虑因素不足，也导致在复杂网络环境中流量调度的性能不是最优[13-14]。

3) 模型收敛慢且优化目标单一。在一些利用 RL 算法解决网络流量调度的研究中，由于训练初期缺乏知识，智能体在接近随机策略的控制下输出动作，从环境获得的正反馈低，导致模型训练速度慢。且由于优化目标单一，使模型在某一性能方面表现较为突出，而整体网络的其他指标不稳定[15]。

为此，本文针对 SDN 架构下的流量调度问题，综合考虑了交换机状态，如流表项数量、设备负载率及连接主机数量等因素对其转发效能的影响，进而影响端到端数据传输时延，提出了一种转发效能感知流量调度（FEATS, forwarding efficiency aware traffic scheduling）算法。FEATS 算法通过神经网络对设备的转发效能进行估计，并利用 DRL 结合该估计值和网络状态信息产生流量调度策略。具体来说，本文的主要贡献介绍如下。

1) 针对考虑交换机转发效能的 SDN 流量调度问题，设计了基于 DL 的转发效能估计模块，通过采集交换机状态数据，对其转发效能进行准确估计，为控制器输出传输时延更低的流量调度方案提供依据。

2) 提出了 FEATS 算法，该算法在考虑 SDN 架构中交换机转发效能对数据传输时延影响的基础上，进一步结合流量需求及当前网络状态，

实现对数据流更合理的自动化控制与调度。

3) 设计了专家样本产生模块，利用在同属性同参数的平行虚拟网络环境中运行具有不同优化目标的流量调度算法，如 OSPF 和 LB 算法，产生专家样本注入经验池。从经验池中提取样本引导模型训练，一方面可以加速模型初期的训练速度，另一方面可以使模型吸取不同优化目标的专家样本中包含的经验，优化模型多方面的性能。

1 相关工作

传统基于规则的流量调度算法需要对网络环境和流量请求进行建模，这种方式在网络环境动态变化时暴露出适应性差的缺点[16]。因此，基于机器学习的算法成为流量调度领域的主要工具，其中利用 DRL 的自主学习机制，通过构建智能体与环境交互采集状态数据，在观察到的网络状态的基础上，根据策略产生一系列行动，即可实现智能化的流量调度[17]。目前，对于讨论如何在 SDN 中进行流量调度相关工作的主要目标集中在降低传输时延和平衡链路负载两方面，对这两方面的研究概括如下。

1.1 降低传输时延

网络用户追求更低的数据传输时延，以获得更高的体验质量。Huang 等[13]认为对数据流的控制是优化用户体验质量（QoE, quality of experience）的关键问题，其首先利用 DRL 动态分配网络资源来实现 QoE 驱动的无模型流量调度，利用 SDN 控制器具有网络的全局视图以获取环境状态，输出数据流路径和明确的带宽分配。其次，利用 QoE 作为奖励参数，但频繁地与用户进行交互是不现实的，这会使来自用户的真实 QoE 数据反馈周期变长，因此，采用多层神经网络捕获特征，将网络和应用指标映射到平均意见评分（MOS, mean opinion score）。利用该模型，可以根据数据流状态快速得到 MOS 值，支持流量调度模型的训练。Huang 等[13]进一步利用 DDPG（deep deterministic policy gradient）算法将 SDN 中的数据分流问题构建为一个深度强化学习模型，该模型将 QoS 指标作为奖励函数用于训练神经网络训练，以优化 QoS 性能。

Zhang 等[18]提出了一种时延优化的多播树封装问题，即交付时延最小化多播树封装（DDMMTP, delivery delay minimized multicast tree packing）。该问题的目标是根据可用网络资源，在网络带宽和最大源–目的时延约束下，最小化平均传输时延。通过利用批量多播树封装算法并根据需要调整现有多播路径的网络容错容量改进算法来近似求解，提出一种基于可用网络资源的源–目的时延改进算法，以进一步减少传输时延。

Wu 等[19]提出了一种多信道重分配和流量控制框架，通过在神经网络中增加 LSTM（long short-term memory）层来提取信道的时序信息，将每个信道的预测流量负载作为链路的状态信息之一，结合丢包率、时延、吞吐量等数据，由多智能体 DRL 模型根据局部状态选择决策，从而实现最大化吞吐量、最小化丢包率和时延。

Saha 等[20]针对通用拓扑提出了一种 QoS 感知的流规则聚合方案，该方案考虑了网络应用的不同 QoS 需求和交换机的流规则容量，自适应聚合流规则。实验结果显示，该方案能够减少 22%的平均端到端传输时延。

在复杂的 SDN 中，往往存在多个控制器应对请求。合理的控制器分配方法可以最小化数据流设置时延，进而影响数据平面性能。Filali 等[21]将控制器的分配问题表述为基于请求数量的一对多匹配博弈，并使控制器在满足最小资源利用率和容量约束的前提下实现负载均衡，以最小化控制平面的响应时间。Savas 等[22]考虑了网络故障后需要多个阶段恢复交换机和控制器之间的控制路径的问题，提出了一种多级控制路径恢复方法。Wang 等[23]结合随机固定水平控制框架，提出了一种结合匹配理论和联盟博弈的控制器分配方法，降低控制器响应时间。Bera 等[24]提出动态控制器分配方案，考虑了特定的流量需求，利用 FlowVisor 模型，构建了一个虚拟平台，作为 SDN 架构的控制面和数据面之间的管理器，使用动态稳定匹配机制，通过定义偏好列表以最小化数据流设置时延和相关控制开销。Bouzidi 等[25]动态计算控制器的最优数量，确定它们的最优位置，同时将交换机集划分为集群，利用 DRL 解决分配控制器的优化问题。Lin 等[26]提出了最小化控制器选择机制，保证控制器的区域覆盖率，并利用改进的多目标人工蜂群算法，根据实时流量判断需要打开哪个控制器进行数据传输，大幅降低了传输时延。

1.2 平衡链路负载

对于网络管理者来说，在满足用户需求的前

提下，还需要考虑链路负载均衡，保证网络性能长期稳定。Zhang 等[11-12]考虑网络中由于频繁重路由带来的负面影响，提出了一种针对 SDN 流量调度的强化学习方法，该方法可自动选择流量矩阵中的关键数据流，通过有选择地重路由少数关键流量，以平衡网络的链路利用率。

Huang 等[14]针对 SDN 和遗留设备共存的混合环境给路由策略带来挑战的问题，提出一种 QoS 优化的近似最优流量控制方法，利用 DRL 输出多路可拆分路由的流量分流比，在链路利用率方面取得了显著改善。

Zhang 等[15]提出了使用 DL 进行网络内容感知以及使用 DRL 进行流量调度的方法。作者认为 SDN 提供对流的控制粒度，但不是对内容的适当抽象，如带宽要求很小的图片和带宽要求很大的视频可能具有相同的源地址和目的地址、相同的端口 ID 和传输协议。因此，提取网络传输内容的属性是合理分配带宽等网络资源的关键，在 SDN 架构中，没有深度数据包检测，控制器很难获取内容属性。通过 DL 模型建立起带宽需求和内容属性的关联性，将对带宽的预测输入基于 DRL 的流量调度模块，输出数据流的传输路径。该方法在网络吞吐量、带宽利用率和负载平衡方面显著提高了网络性能。

Maity 等[27]针对防止控制器过载和优化分配流量的问题，根据马尔可夫预测器对设备移动性的预测结果，实现了流量感知的规则缓存机制和主控制器分配方案，降低了 23.08%控制流量峰值强度。

以上方法针对不同的优化目标，利用深度模型强大的特征表达能力和强化学习的自主学习机制，可不需要精确的环境建模，为数据流请求实时计算调度方案。然而，随着流表项数量、设备负载率和连接主机数等因素变化，SDN 交换机具有不同的转发效能，进而影响数据传输。因此，将 SDN 交换机的转发效能作为影响数据传输的重要因素，同时考虑多维优化目标，才能够输出更合理的流量调度策略。

2　转发效能感知流量调度算法

基于 OpenFlow 协议的 SDN 端到端的数据传输模型如图 1 所示。

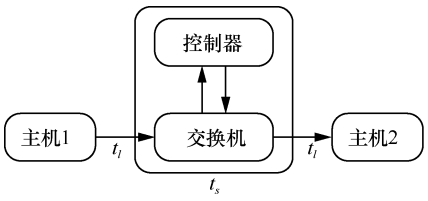


图 1　基于 OpenFlow 协议的 SDN 端到端的数据传输模型

当主机 1 向主机 2 发送的数据流经过交换机时，交换机将数据包的匹配域与自身保存的流表项进行对比。如果有匹配流表项，根据对应动作执行；如果没有匹配流表项，交换机和控制器交互 PACKET_IN 和 PACKET_OUT 消息，数据流首个报文由交换机发送至控制器，控制器在接收到报文后，根据转发策略产生数据转发路径，并发送到路径上的各个交换机安装更新规则。因此，端到端的数据传输时延可表示为

$$T = \sum_{l \in L} t_l + \sum_{s \in S} t_s \tag{1}$$

其中，t_l 为链路传输时延，L 为链路集合，t_s 为交换机转发时延，S 为交换机集合。数据传输时延 T 为链路传输时延和交换机转发时延的总和，且 t_s 与交换机状态紧密相关。

本文除了考虑网络资源和链路状态等因素外，还综合考虑了 SDN 交换机的转发效能对数据传输效率的影响，结合深度学习和深度强化学习的特点，提出了 FEATS 算法求解该问题。FEATS 算法的整体架构如图 2 所示。首先，对于 SDN 流量调度问题，定义网络拓扑为无向图 Graph=(N, E)，N 为节点集合，E 为链路集合，从源节点 src 到目的节点 dst 的链路为 $e_{\text{src,dst}}$，$\forall e_{\text{src,dst}} \in E$。接下来，将对算法中的核心模块和算法流程进行阐述。

2.1　转发效能估计模块

由于 SDN 交换机的转发效能与其本身的性能及实时状态有关，且这种映射关系很难通过数学多项式精确表达。因此本文设计了一种基于深度学习的 SDN 交换机转发效能估计模块。该模块的输入向量 $\mathbf{In}_{\text{trans}}$=[entries$_i$, table$_i$, hostes$_i$, lr$_i$, rf$_i$]为 SDN 交换机状态参数，其中，entries 为流表项数量，table 为流表数量，hostes 为连接主机数，lr 为设备负载率，rf 为出入口流量。模块的输出向量为 $\mathbf{Out}_{\text{trans}}$，表示模块预测数据包从进入交换机到输出的完整时延，其中包含了流表项的匹配时间，以及当接收新数据流时交换机与控制器的交互时间。模块主要由一个 3 层 sequential 神经网络实现。该神经网络第一层包含 5 个神经元，

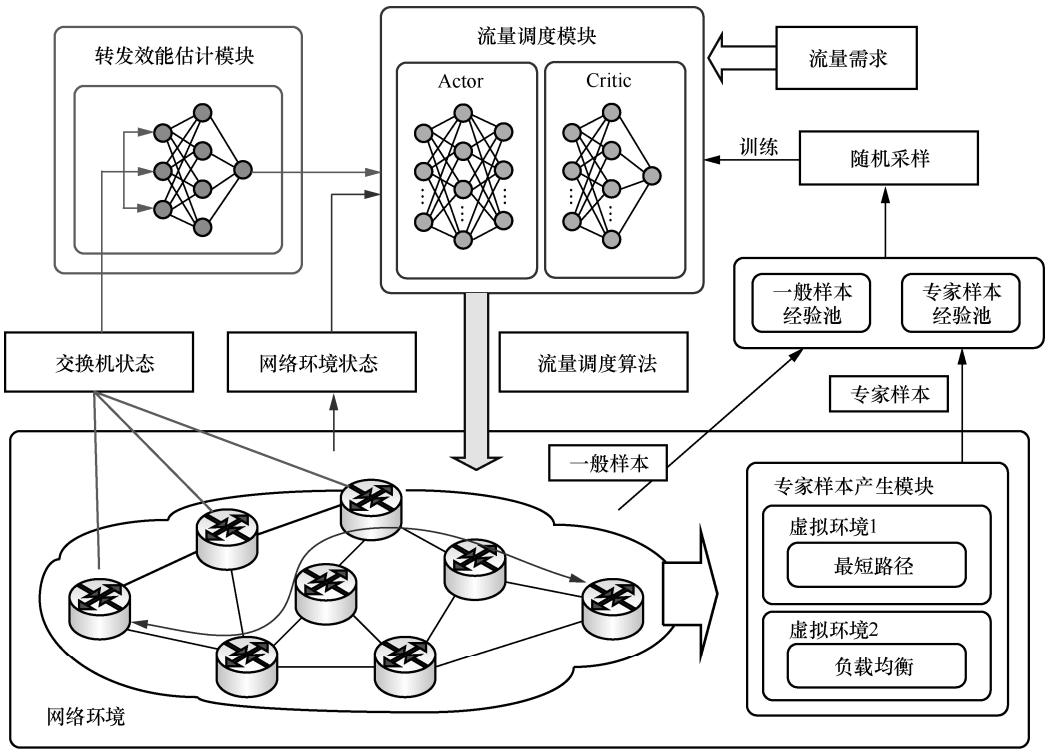


图 2　FEATS 算法的整体架构

用于接收输入向量；第二层为全连接层，分别包含 64 个神经元，与第一层全连接，激活函数为 ReLU；最后一层包含一个神经元，输出对交换机转发时延的估计值。模型以均方误差（MSE, mean square error）作为损失函数，表示为

$$\text{loss}=\frac{1}{n}\sum_{i=1}^{n}\left(\text{Out}_{\text{trans},i}-(t_{\text{in},i}-t_{\text{out},i})\right)^2 \tag{2}$$

其中，n 为样本数量，$\text{Out}_{\text{trans},i}$ 为模型输出估计值，$t_{\text{in},i}$ 和 $t_{\text{out},i}$ 分别为数据包进出交换机的时刻。随着最小化损失函数通过反向传播更新模型参数，模型估计的转发时延与真实值逐渐接近。

2.2　流量调度模块

为了解决动态复杂环境下的流量调度问题，本节将引入基于深度强化学习的流量调度算法。在一个典型的强化学习模型中，智能体与环境交互，利用状态（state）、策略（policy）、动作（action）和奖励（reward）逐步实现优化目标。定义智能体的状态空间为 $\boldsymbol{S}$，动作空间为 $\boldsymbol{A}$。具体来说，在每个时间步 t，智能体首先观察当前状态 $\boldsymbol{s}_t\in\boldsymbol{S}$，结合自身策略 π 产生对应动作 $\boldsymbol{a}_t\in\boldsymbol{A}$，通过执行该动作实现智能体与环境的互动，获得奖励 $\boldsymbol{r}_t$ 并同时观察到下一个状态 $\boldsymbol{s}_{t+1}$。通过构建四元组<$\boldsymbol{s}_t$, $\boldsymbol{a}_t$, $\boldsymbol{r}_t$, $\boldsymbol{s}_{t+1}$>以最大化未来的累计奖励的期望 $\text{E}(R_t)$为目标调整模型参数。$\text{E}(R_t)$可表示为

$$\text{E}(R_t)=\text{E}\left(\sum_{t}^{T}\gamma^t\boldsymbol{r}_t\right) \tag{3}$$

其中，T 为迭代上限，t 为当前时间步，$\gamma\in[0,1]$为折扣因子，γ 越大表示算法越重视长期的累计奖励，γ 越小则反之。在本文的研究场景中，状态、动作和奖励的具体含义如下所示。

状态（state）。状态是指在时刻 t 时，SDN 交换机的转发效能 $\textbf{Out}_{\text{trans}}$、链路的传输时延 d_t、丢包率 l_t 和网络抖动 j_t，用向量 $\boldsymbol{s}_t$ 表示，即 $\boldsymbol{s}_t$=[$\textbf{Out}_{\text{trans}}$, d_t, l_t, j_t]。若令拓扑中交换机数量为 m，源节点 src 和目的节点 dst 间可用的链路数为 e，则 $\boldsymbol{s}_t$ 为 $3e+m$ 维向量。

动作（action）。动作是指智能体根据策略 π 和状态 $\boldsymbol{s}_t$ 生成的流量调度方案，即数据流最优的转发路径，动作向量可表示为策略 π 和状态 $\boldsymbol{s}_t$ 的函数，如式(4)所示。$\boldsymbol{a}_t$=[e_1,e_2,⋯,e_l]用于表示输出的可用传输路径。

$$\boldsymbol{a}_t=\pi(\boldsymbol{s}_t) \tag{4}$$

奖励（reward）。奖励是指环境针对智能体的行为做出的反馈，用于表示所执行动作的好坏，同时也体现了模型训练的目标。流量调度模块中奖励函数定义为

$$\boldsymbol{r}_t=\frac{1}{\dfrac{D_{\mathrm{src,dst}}-\min\{\boldsymbol{D}\}}{\max\{\boldsymbol{D}\}-\min\{\boldsymbol{D}\}}+\max\{\boldsymbol{U}\}} \tag{5}$$

其中，$\boldsymbol{U}$ 表示当前网络环境下各链路利用率的矩阵，$\boldsymbol{D}$ 表示对应流量需求的传输时延矩阵，$D_{\mathrm{src,dst}}$ 表示从源节点 src 到目的节点 dst 的特定数据流的传输时延。算法的目标是使奖励最大化，即使当前网络中的最大链路利用率和端到端传输时延尽可能小。

深度强化学习是深度学习和强化学习结合的产物。传统的强化学习算法分为基于策略的算法和基于价值的算法，而将 2 种方法结合起来就是 Actor-Critic 算法。该算法由 Actor 和 Critic 这 2 个神经网络组成，Actor 网络负责生成动作并和环境交互，产生四元组<$\boldsymbol{s}_t$, $\boldsymbol{a}_t$, $\boldsymbol{r}_t$, $\boldsymbol{s}_{t+1}$>作为训练数据，而 Critic 网络负责评估 Actor 的表现。

具体来说，此模块中的 Actor 网络由 3 层构成，分别包含 3e+m、128 和 e 个神经元，其中，e 和 m 分别为源节点 src 和目的节点 dst 间可用的链路数和拓扑中包含的交换机数。第一层 3e+m 神经元数量与 DRL 状态向量 $\boldsymbol{s}_t$=[$\mathbf{Out}_{\mathrm{trans}}$, d_t, l_t, j_t]的维度对应。模块后两层为全连接层，激活函数分别为 ReLU 和 softmax。Critic 网络由 3 层构成，分别包含 e、128 和一个神经元，后两层为全连接层，激活函数使用 ReLU，损失函数为 Huber。令 Actor 网络的参数为 θ，Critic 网络的参数为 w，则 Actor 网络的参数更新计算式为

$$\theta=\theta+\alpha\nabla_\theta\log\pi_\theta(\boldsymbol{S}_t,\boldsymbol{A}_t)\delta(t) \tag{6}$$

$$\delta(t)=\boldsymbol{r}_{t+1}+\gamma Q_w(\boldsymbol{s}_{t+1},\boldsymbol{a}_{t+1})-Q_w(\boldsymbol{s}_t,\boldsymbol{a}_t)) \tag{7}$$

其中，$\nabla_\theta\log\pi_\theta(\boldsymbol{S}_t,\boldsymbol{A}_t)$ 是策略梯度的分值函数，α 是学习率，$\delta(t)$是 TD-error，γ 是折扣率。Critic 网络使用 Huber 损失函数作为参数 w 的梯度更新，该损失函数为

$$L_\lambda(O_{\mathrm{Cri}},R_t)=\begin{cases}\dfrac{1}{2}(O_{\mathrm{Cri}}-R_t)^2, |O_{\mathrm{Cri}}-R_t|\leqslant\lambda\\ \lambda|O_{\mathrm{Cri}}-R_t|-\dfrac{1}{2}\lambda^2, \text{其他}\end{cases} \tag{8}$$

其中，λ 为超参数，当$|O_{\mathrm{cri}}-R_t|\leqslant\lambda$ 时，损失函数等价为 MSE；反之则等价于平均绝对误差（MAE, mean absolute error），该方法同时具备 MSE 和 MAE 这 2 种损失函数的优点，能够降低奇异点数据对拟合效果的影响。

2.3　专家样本产生模块

本文提出的 FEATS 算法中的流量调度模块是基于 DRL 实现的。在复杂环境下，面临奖励稀疏的问题，即模型在训练初期执行随机策略，难以学习到有效的知识，导致模型训练时间增加。

FEATS 算法设计了专家样本产生模块，即构建和网络拓扑具有相同结构、参数和状态的并行虚拟环境。令初始网络拓扑为 Graph=(N, E)，则平行网络拓扑 $\mathrm{Graph}_{v_i}=\mathrm{Graph}=(N, E)$。在 Graph_{v_i} 中运行最短路径和负载均衡算法，生成具有相同格式的<$\boldsymbol{s}_t$, $\boldsymbol{a}_t$, $\boldsymbol{r}_t$, $\boldsymbol{s}_{t+1}$>四元组作为专家样本，其中

$$\boldsymbol{a}_t=\pi_{\mathrm{ex}}(\boldsymbol{s}_t) \tag{9}$$

其中，$\boldsymbol{a}_t$ 是控制器利用最短路径和负载均衡算法的策略 π_{ex}，根据当前状态 $\boldsymbol{s}_t$ 产生的流量调度方案。

将专家样本和由智能体与环境交互产生的一般样本分别放入专家样本经验池和一般样本经验池中，利用随机采样方法从上述 2 个经验池中抽取样本组成 mini-batch 用于模型训练。通过控制随机采样因子逐渐递减，实现 mini-batch 中专家样本数量随着训练过程逐渐降低。该模块的主要作用介绍如下。1) 在训练初期，利用更多的专家样本引导模型训练，一方面，可加速模型在训练初期的学习速度。由于 RL 智能体在训练初期由于缺乏知识，自身策略接近于随机策略，输出的动作无法从环境中获得足够的正反馈，因此仅靠智能体与环境交互产生的样本进行训练的效率较低。另一方面，专家样本也可辅助模型学习到专家样本中的知识，实现多目标优化。2)在训练后期，即在智能体学习到了专家样本的知识后，降低专家样本在 mini-batch 中的比例，发挥 RL 算法在环境中探索的能力，增加智能体学习到综合性能高于单一性能优秀的专家样本知识的概率，实现 FEATS 算法的优化目标。

3　算法设计与实现

为了解决上述流量调度问题，本文提出 FEATS 算法，首先使用基于 DL 的转发效能估计模块预测 SDN 交换机转发时延，之后将该估计值作为输入变量传递给基于 DRL 的流量调度模块，使控制器输出的最优决策受到该估计值的影响。

3.1　交换机转发效能估计

在 FEATS 中，利用 DL 强大的表示能力拟合交换机的转发效能和多维状态数据之间映射关系。转

发效能估计算法的训练流程如算法 1 所示。

算法 1 转发效能估计算法

1) 初始化网络 Graph 节点和链路参数、转发效能估计模块 M、流量需求矩阵 $\boldsymbol{F}$、流量调度模块 S

2) 清空缓存空间 B

3) for each $f \in \boldsymbol{F}$:

4) for each $s \in S$:

5) 获取数据包进出时刻 t_{in}, t_{out}，计算 $t_l = t_{\text{out}} - t_{\text{in}}$

6) 获取交换机状态信息 $\mathbf{In}_{\text{trans}}$=[entries$_i$, tables$_i$, hosts$_i$, lr$_i$, rf$_i$]

7) 计算估计值 $\mathbf{Out}_{\text{trans}} = M(\mathbf{In}_{\text{trans}})$

8) 最小化 MSE(Out$_{\text{trans},i}$, t_l)更新模型参数

9) end for

10) end for

随着对流量需求的执行，可以获取数据流传输路径上交换机的状态信息及转发时延，根据模块输出估计结果，通过最小化损失函数调整模型参数以提升预测准确度。

3.2 转发效能感知流量调度

在估计交换机转发效能后，FEATS 根据该估计值和当前网络状态，针对流量需求产生对应的调度策略。基于深度强化学习的智能流量调度算法流程如算法 2 所示。定义当前的训练时长为 timer，模型训练时间上限为 DT，一次迭代内最大时间步上限为 T。定义 random(α)为随机采样方法，随机采样因子 $\alpha \in (0, 1)$，表示以 α 为概率抽取样本。

算法 2 基于深度强化学习的智能流量调度算法

1) 初始化网络 Graph 节点和链路参数、转发效能估计模块 M、流量调度模块 S、虚拟环境和流量需求、随机采样因子 α

2) 初始化专家样本经验池 B_e 和一般样本经验池 B_g

3) for each episode:

4) for $t = 0$ to T:

5) 在虚拟环境 1 中执行最短路径算法

6) 获取四元组$<\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{s}_{t+1}> \rightarrow B_e$

7) 在虚拟环境 2 中执行负载均衡算法

8) 获取四元组$<\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{s}_{t+1}> \rightarrow B_e$

9) 在网络环境中执行 FEATS 算法

10) for each $s \in S$:

11) 控制器获取 SDN 交换机状态 $\mathbf{In}_{\text{trans}}$ 和网络状态参数

12) 计算估计值 $\mathbf{Out}_{\text{trans}} = M(\mathbf{In}_{\text{trans}})$

13) end for

14) 形成 $\boldsymbol{s}_t$=[$\mathbf{Out}_{\text{trans}}$, d_t, l_t, j_t]传入流量调度模块 S 并输出 $\boldsymbol{a}_t = [e_1, e_2, \cdots, e_l]$

15) 执行 $\boldsymbol{a}_t$，观察 $\boldsymbol{s}_{t+1}$ 并计算 $\boldsymbol{r}_t$

16) 获取四元组$<\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{s}_{t+1}> \rightarrow B_g$

17) end for

18) 从 B_e 中根据 random(α)采样

19) 从 B_g 中根据 random(1−α)采样，与步骤 18)的样本组成 mini-batch

20) if α>=0.1 则 α−=0.01%

21) 利用梯度下降法更新 Actor 网络，如式(6)和式(7)所示

22) 利用 Huber 误差更新 Critic 网络，如式(8)所示

23) 清空经验池 B_e 和 B_g

24) if timer ≥ DT

25) end for

FEATS 在同参数同状态的虚拟环境中运行最短路径和负载均衡算法，由此产生专家样本及其自身与环境交互产生的数据共同组成训练集，在随机因子 α 的作用下从中采样。随着 α 逐渐降低，算法抽取到专家样本的概率减小，即专家样本在训练初期对模型的影响大，不仅可以加速训练速度，也可使模型学习到专家样本中的知识，而在训练后期更强调模型自身对环境的“探索”。模型根据交换机状态预测数据包经过该设备转发时产生的时延，输出综合考虑传输时延和网络负载均衡的流量调度方案。

4 仿真分析

为了验证 FEATS 算法的性能，本节通过实验仿真将其与迪杰斯特拉（Dijkstra）算法、轮询（Round Robin）调度算法及加权最小连接（Weighted Least Connection）调度算法的评价指标对比。

1) Dijkstra 算法是 OSPF 协议的核心算法，使用广度优先搜索解决赋权有向图或者无向图的单源最短路径问题。

2) Round Robin 是一种以轮询方式将负载请求分配给设备的负载均衡调度算法。

3) Weighted Least Connection 可为设备赋值权重，算法通过连接数和权重调整设备被轮询到的概率，以实现更平均的负载均衡。

本节实验利用 Mininet 和 Python 构建实验环境及采集数据，使用 Ryu 作为 SDN 控制器，通过 Iperf

工具产生数据流。FEATS 模型采用 Keras 实现，Keras 是一种基于 Tensorflow 的高阶 API。服务器硬件配置及软件版本如表 1 所示。

表 1　服务器硬件配置及软件版本

属性	配置
操作系统	Ubuntu 16
CPU	Intel(R) Core(TM) i5-10400F
内存	16.0 GB
Python	Ver. 3.7
Tensorflow	Ver. 1.14
Mininet	Ver. 2.3.0
Ryu	Ver. 4.34
Iperf	Ver. 2.0.9

本文从 SNDLib 获取 GEANT 网络拓扑和流量需求，该拓扑具有 22 个节点及 72 条链路。初始化实验参数，随机采样因子 α=0.9，各节点根据高斯分布随机产生流表项及流表数量，设置最小传输时延为 1 ms，最大传输时延为 200 ms。

4.1　算法评价指标

本文提出的 FEATS 算法由于考虑了设备转发效能，并且由最短路径和负载均衡算法产生的专家样本训练，可输出综合考虑了网络负载均衡的最短传输时延的流量调度方案。因此，实验部分针对以下评价指标进行对比分析。令 $\boldsymbol{F}$ 为流量需求矩阵，$f\in\boldsymbol{F}$ 为单个流量需求命令，n 为流量需求命令的数量，$\boldsymbol{L}$ 为全部交换机负载率矩阵，$\mathrm{lr}\in\boldsymbol{L}$ 为交换机负载率，m 为交换机数量。

平均跳数为

$$\mathrm{Avg}_{\mathrm{hop}}=\frac{\sum_{f\in\boldsymbol{F}}\mathrm{num}_{\mathrm{hop}}}{n} \tag{10}$$

其中，$\mathrm{num}_{\mathrm{hop}}$ 为完成流量需要跳转的设备数量，每经过一台交换机，$\mathrm{num}_{\mathrm{hop}}$ 加 1。

平均传输时延为

$$\mathrm{Avg}_{\mathrm{delay}}=\frac{\sum_{f\in\boldsymbol{F}}t_{\mathrm{end}}-t_{\mathrm{start}}}{n} \tag{11}$$

其中，t_{end} 和 t_{start} 分别为数据包从发送端发出和到目的端接收的时刻。

最大负载率为

$$\mathrm{Max}_{\mathrm{lr}}=\max\{\boldsymbol{L}\} \tag{12}$$

其中，max 函数为获取 $\boldsymbol{L}$ 中的最大值，即计算各算法在执行流量需求的过程中全部交换机的负载率最大值。

负载率方差为

$$\mathrm{Var}_{\mathrm{lr}}=\frac{\sum_{i=1}^{m}(\mathrm{lr}_i-\overline{\mathrm{lr}})^2}{m-1} \tag{13}$$

式(13)用于计算各算法在执行流量需求的过程中各交换机负载率方差的最大值。该指标越大，表示各交换机之间负载率的差异越大，反之则表示负载率越均衡。

4.2　专家样本产生模块支持模型训练效果分析

FEATS 算法中设计了专家样本产生模块，在同结构同参数的虚拟环境中，利用成熟的算法产生专家样本，帮助算法学习到对应方面的知识，加速模型在训练初期的训练效率。本节实验通过控制有无专家样本产生模块参与训练，验证该模块对模型训练的影响。无专家样本模块支持的 FEATS 算法在实验中用“FEATS 无专家样本”表示。同时，本节实验还与同属 Actor-Critic 架构的 A3C（asynchronous advantage actor-critic）和 DDPG（deep deterministic policy gradient）算法进行了更广泛的分析对比。在实验环境中部署上述 4 种算法，分别进行 20 次实验，收集模型从环境中获取的累计奖励值。平均奖励值对比结果如图 3 所示。

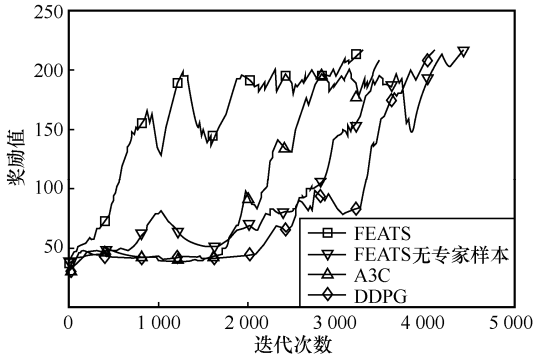


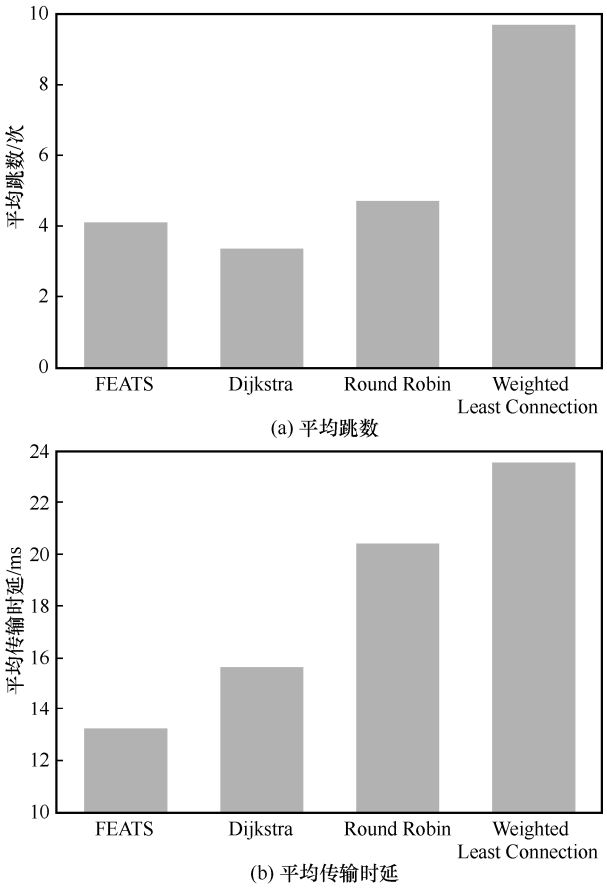
图 3　平均奖励值对比结果

从图 3 可以看出，FEATS 算法的训练效率高于 FEATS 无专家样本、A3C 和 DDPG 算法的情况。FEATS、A3C 和 DDPG 都是基于 DRL 的算法，在复杂环境中同样面临奖励稀疏的问题，表现为模型训练的前中期很难获得有效的正反馈，使奖励值难以扩展至环境的大部分状态。图 3 中的结果显示，FEATS 无专家样本、A3C 和 DDPG 算法在

第 1~2 000 次迭代期间获取的奖励值在较低水平浮动，说明模型输出的动作难以获得环境给予的正反馈，导致模型在前中期训练效率低，并分别在迭代 4 439 次、3 485 次、4 101 次后收敛。对于有专家样本支持的情况，由成熟的算法产生的动作在训练初期就可以获得较高的奖励值，缩短了 FEATS 在没有足够知识的情况下盲目探索的过程，使模型在迭代 3 295 次后完成训练。DDPG 具有经验回放功能，但训练样本来自自身与环境交互，相当于 FEATS 算法中的一般样本，其功能更多是为了打破数据相关性，在训练初期对加速模型收敛的影响不大。A3C 通过异步方式执行多个 Actor 进行学习，并行的方式对加速模型训练起到了一定的积极效果，但在初期依然难以获得正反馈。本节实验证明了专家样本产生模块使 FEATS 模型训练效率提升了 25.78%。

4.3　数据传输性能对比分析

路由跳数和数据传输时延是评价数据传输性能的重要指标。本节实验在 GEANT 拓扑中执行流量需求指令，记录各算法完成流量需求的平均跳数和平均传输时延，实验结果如图 4 所示。



(a) 平均跳数

(b) 平均传输时延

图 4　数据传输性能对比

图 4(a)显示出 Dijkstra 作为一种最短路径算法，在平均跳数方面取得了最好的性能，为 3.77；负载均衡算法 Round Robin 和 Weighted Least Connection 的平均跳数分别为 4.72 和 9.75。在平均传输时延方面，如图 4(b)所示，Dijkstra 为 15.61 ms，Round Robin 和 Weighted Least Connection 分别为 20.38 ms 和 23.56 ms。

本文提出的 FEATS 算法的平均跳数为 4.13，平均传输时延为 13.22 ms，较 Dijkstra 算法分别增加了 9.55%及降低了 15.31%，呈现出跳数高、传输时延低的结果。这是因为在数据转发过程中，交换机具有不同数量的流表项和连接主机数以及不同程度的负载等因素，使其转发效能呈现较大差异。FEATS 算法的转发效能估计模块预测的转发时延最小值为 0.9 ms，最大值为 9.3 ms。因此，存在在跳数最少的数据传输路径上某些交换机的转发时延较高的情况。FEATS 算法可以有效感知交换机的转发效能，在预测到交换机的转发时延较高时，选择跳数多但交换机可以更快完成数据转发的路径，从而实现更低的传输时延。

4.4　负载均衡性能对比分析

最短路径算法往往仅考虑数据传输效率，导致不同重要程度的链路和节点之间负载率差距较大，不利于网络整体性能和稳定性。各算法执行流量需求指令时，交换机的负载均衡性能对比如图 5 所示。

图 5(a)显示了 FEATS、Dijkstra、Round Robin 和 Weighted Least Connection 的最大负载率分别为 78.92%、75.73%、79.07%和 85.14%。其中，Dijkstra 算法的最大负载率最低，主要原因是该算法的目标为经过数量最少的交换机完成数据传输，因此在执行相同的流量需求时，全部交换机的负载率总和最低。Round Robin 和 Weighted Least Connection 需要考虑负载均衡，输出的传输路径通常要比 Dijkstra 算法的更长，导致网络整体的负载率有所增加。

在负载均衡方面，如图 5(b)所示，负载率方差反映了交换机负载率的差异性。FEATS、Dijkstra、Round Robin 和 Weighted Least Connection 的最大负载率方差分别为 7.39%、8.93%、7.21%和 6.51%。Dijkstra 算法虽然产生最短传输路径，但输出的路径可能频繁经过在网络拓扑中占据重要位置的节点，而边界节点常处于闲置或低负载状态，造成节点负载率差距较大，在图 5(b)中表现为最大负载率方差最大。Round Robin 按顺序调度流量，起到了

负载均衡的作用，其最大负载率方差与 Dijkstra 相比有显著的降低。Weighted Least Connection 算法由于存在加权系数，使流量可以更均匀地调度到低负载的交换机，实现各设备之间负载率更均衡，因此其最大负载率方差最低。FEATS 算法在负载均衡方面取得了和 Round Robin 算法接近的水平。由于在训练过程中，模型有利用负载均衡算法产生的样本参与训练，且 r_t 与最大负载率紧密相关，因此，产生的决策也具备较好的负载均衡效果。

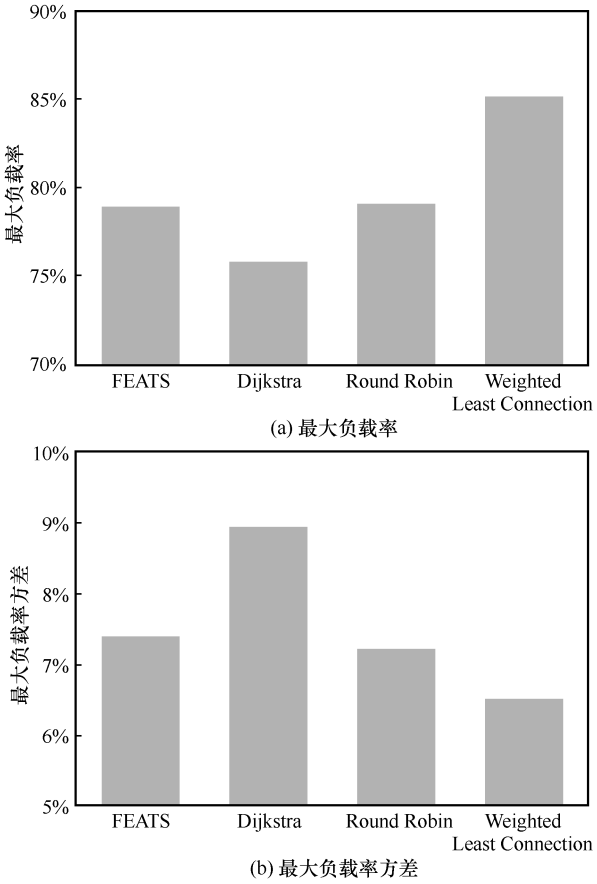


图 5　负载均衡性能对比

综合以上结果分析，本文提出的 FEATS 算法可以根据交换机状态有效估计交换机转发时延，并基于转发时延规划更合理的数据传输路径。通过利用专家样本缩短了 25.78%的训练时间，并且实现了在具备一定负载均衡效果的基础上平均降低 15.31%的端到端传输时延。

5　结束语

在 SDN 架构中，流表项数量、设备负载率及连接主机数量增加，会导致 SDN 交换机的转发效能降低，进而影响端到端数据传输时延。本文提出的 FEATS 算法利用 DL 的强大表示能力，建立起 SDN 交换机的多维状态数据与转发效能之间的映射关系，并根据转发效能、网络状态和流量需求输出兼顾最低时延和网络负载均衡的流量调度策略。实验表明，FEATS 算法的专家样本产生模块可提升 25.78%的模型训练效率。同时，由于 FEATS 算法可以有效评估交换机转发效能，端到端传输时延比 Dijkstra 算法降低 15.31%，负载均衡性能与 Round Robin 算法接近。说明 FEATS 算法学习到了最短路径和负载均衡专家样本的知识，并在自身奖励函数的引导下可以输出综合考虑了传输时延和网络负载均衡的流量调度方案。FEATS 算法对解决 SDN 的流量调度问题、提升网络性能有一定的实用价值。

需要注意的是，FEATS 算法在实验环境下验证了有效性。在实际场景中，交换机转发效能还会额外受到硬件性能和状态等多种因素的影响，在针对实际场景调整模型输入维度后，可更好地满足实际使用需求。

参考文献：

[1] 周桐庆, 蔡志平, 夏竟, 等. 基于软件定义网络的流量工程[J]. 软件学报, 2016, 27(2): 394-417.
ZHOU T Q, CAI Z P, XIA J, et al. Traffic engineering for software defined networks[J]. Journal of Software, 2016, 27(2): 394-417.

[2] MENDIOLA A, ASTORGA J, JACOB E, et al. A survey on the contributions of software-defined networking to traffic engineering[J]. IEEE Communications Surveys & Tutorials, 2017, 19(2): 918-953.

[3] 杨思锦, 庄雷, 宋玉, 等. 多模态网络中时间敏感网络模态的智能调度机制[J]. 通信学报, 2022, 43(5): 82-91.
YANG S J, ZHUANG L, SONG Y, et al. Intelligent scheduling mechanism of time-sensitive network modal in polymorphic network[J]. Journal on Communications, 2022, 43(5): 82-91.

[4] SARASWAT S, AGARWAL V, GUPTA H P, et al. Challenges and solutions in software defined networking: a survey[J]. Journal of Network and Computer Applications, 2019, 141: 23-58.

[5] 黄韬, 刘江, 汪硕, 等. 未来网络技术与发展趋势综述[J]. 通信学报, 2021, 42(1): 130-150.
HUANG T, LIU J, WANG S, et al. Survey of the future network technology and trend[J]. Journal on Communications, 2021, 42(1): 130-150.

[6] LECUN Y, BENGIO Y, HINTON G. Deep learning[J]. Nature, 2015, 521(7553): 436-444.

[7] SUTTON R S, BARTO A G. Reinforcement learning: an introduction[M]. Massachusetts: MIT Press, 1998.

[8] XIAO Y, LIU J, WU J W, et al. Leveraging deep reinforcement learning for traffic engineering: a survey[J]. IEEE Communications Surveys & Tutorials, 2021, 23(4): 2064-2097.

[9] XIE J F, YU F R, HUANG T, et al. A survey of machine learning

techniques applied to software defined networking (SDN): research issues and challenges[J]. IEEE Communications Surveys & Tutorials, 2019, 21(1): 393-430.

[10] HARTMAN T, HASSIDIM A, KAPLAN H, et al. How to split a flow? [C]//Proceedings of IEEE INFOCOM. Piscataway: IEEE Press, 2012: 828-836.

[11] ZHANG J J, YE M H, GUO Z H, et al. CFR-RL: traffic engineering with reinforcement learning in SDN[J]. IEEE Journal on Selected Areas in Communications, 2020, 38(10): 2249-2259.

[12] ZHANG J J, GUO Z H, YE M H, et al. SmartEntry: mitigating routing update overhead with reinforcement learning for traffic engineering[C]//Proceedings of the Workshop on Network Meets AI & ML. Piscataway: IEEE Press, 2020: 1-7.

[13] HUANG X H, YUAN T T, QIAO G H, et al. Deep reinforcement learning for multimedia traffic control in software defined networking[J]. IEEE Network, 2018, 32(6): 35-41.

[14] HUANG X H, ZENG M, XIE K. Intelligent traffic control for QoS optimization in hybrid SDNs[J]. Computer Networks, 2021, 189: 107877.

[15] ZHANG Q Y, WANG X W, LV J H, et al. Intelligent content-aware traffic engineering for SDN: an AI-driven approach[J]. IEEE Network, 2020, 34(3): 186-193.

[16] XU Z Y, TANG J, MENG J S, et al. Experience-driven networking: a deep reinforcement learning based approach[C]//Proceedings of IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2018: 1871-1879.

[17] KATO N, FADLULLAH Z M, MAO B M, et al. The deep learning vision for heterogeneous network traffic control: proposal, challenges, and future perspective[J]. IEEE Wireless Communications, 2017, 24(3): 146-153.

[18] ZHANG X C, WANG Y L, GENG G G, et al. Delay-optimized multicast tree packing in software-defined networks[J]. IEEE Transactions on Services Computing, 2021, PP(99): 1.

[19] WU T, ZHOU P, WANG B H, et al. Joint traffic control and multi-channel reassignment for core backbone network in SDN-IoT: a multi-agent deep reinforcement learning approach[J]. IEEE Transactions on Network Science and Engineering, 2021, 8(1): 231-245.

[20] SAHA N, MISRA S, BERA S. Q-flag: QoS-aware flow-rule aggregation in software-defined IoT networks[J]. IEEE Internet of Things Journal, 2022, 9(7): 4899-4906.

[21] FILALI A, KOBBANE A, ELMACHKOUR M, et al. SDN controller assignment and load balancing with minimum quota of processing capacity[C]//Proceedings of IEEE International Conference on Communications (ICC). Piscataway: IEEE Press, 2018: 1-6.

[22] SAVAS S S, TORNATORE M, DIKBIYIK F, et al. RASCAR: recovery-aware switch-controller assignment and routing in SDN[J]. IEEE Transactions on Network and Service Management, 2018, 15(4): 1222-1234.

[23] WANG T, LIU F M, XU H. An efficient online algorithm for dynamic SDN controller assignment in data center networks[J]. IEEE/ACM Transactions on Networking, 2017, 25(5): 2788-2801.

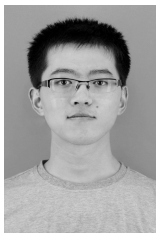
[24] BERA S, MISRA S, SAHA N. Traffic-aware dynamic controller assignment in SDN[J]. IEEE Transactions on Communications, 2020, 68(7): 4375-4382.

[25] BOUZIDI E H, OUTTAGARTS A, LANGAR R, et al. Dynamic clustering of software defined network switches and controller placement using deep reinforcement learning[J]. Computer Networks, 2022, 207: 108852.

[26] LIN N, ZHAO Q, ZHAO L, et al. A novel cost-effective controller placement scheme for software-defined vehicular networks[J]. IEEE Internet of Things Journal, 2021, 8(18): 14080-14093.

[27] MAITY I, MISRA S, MANDAL C. CORE: prediction-based control plane load reduction in software-defined IoT networks[J]. IEEE Transactions on Communications, 2021, 69(3): 1835-1844.

[作者简介]



沙宗轩（1990− ），男，回族，安徽蚌埠人，北京工业大学博士生，主要研究方向为未来网络、网络人工智能深度学习、强化学习等。

霍如（1988− ），女，黑龙江哈尔滨人，博士，北京工业大学讲师，主要研究方向为未来网络、工业互联网、边缘计算、网络资源管理、区块链等。

孙闯（1989− ），男，黑龙江哈尔滨人，博士，清华大学在站博士后，主要研究方向为传感器测量与测试技术、初始仪器精密标定方法和惯性导航技术等。

汪硕（1991− ），男，河南灵宝人，博士，北京邮电大学讲师，主要研究方向为数据中心网络、软件定义网络、网络流量调度等。

黄韬（1980− ），男，重庆人，博士，北京邮电大学教授，主要研究方向为未来网络体系架构、软件定义网络、网络虚拟化等。