Deep Reinforcement Learning based Optimal Routing with Software-Defined Networking

ABSTRACT

长期以来，拥塞避免和路由优化一直都是现代通信网络中流量工程的重要研究课题。但随着用户数量和网络规模的飞速增长，网络结构越来越复杂，传统的解决方案面临着越来越大的挑战。近些年来，归功于硬件计算能力的提高和深度学习（DL）的成功，深度强化学习（DRL）在很多领域都取得了突破性的进展。与此同时，软件定义网络（SDN）的出现和发展使得网络的控制变得更加高效和方便。这些都为实现更加智能的路由策略提供了新的机遇。在本文中，我们提出了一种结合DRL策略优化和SDN流量控制的最优路由控制方案。该方法使用基于DRL的方法来学习不同流量模式下网络的最优路由策略，而SDN controller则为基于DRL的决策提供网络流量信息，并根据DRL的决策输出实现流的路由控制。仿真实验的结果表明，与使用传统负载均衡方式的网络相比，使用基于DRL最优路由决策的网络能够在具有非对称拓扑结构的网络中更显著地降低拥塞的发生率，改善网络的质量。

Key: Deep reinforcement learning, Software-Defined Networking, Routing optimization, congestion avoidance, Load balance

1. INTRODUCTION

new page {

网络中高度动态变化的流量业务和分布不均的流量密度是引起网络拥塞的主要原因，为了解决网络拥塞，已经有很多解决方案被提出，常见的解决方案主要有两种，一种是根据网络状态来优化网络的路由策略（e.g. 流量感知路由算法），另一种方案是对可能引起网络拥塞的流量进行多路径分流以防止流量造成的负载过度集中（e.g. ECMP），虽然这些解决方案能够在一些情况下降低网络拥塞，但它们都有着各自的问题：流量感知路由算法基于潜在的流量状况独立地为每个数据包选择最优的path，但不能保证所有个体组成的全局网络路由策略是最优的；ECMP简单地将流量均分给所有等价path而不考虑网络中流量的分布，这导致其在具有非对称拓扑和流量的网络中表现不尽如人意。考虑到以上两种方案各自的问题，我们需要寻找一种可以新的解决方案：一方面能够基于网络全局流量状态进行路由策略优化的同时，另一方面还可以适用于各种复杂拓扑结构的网络中。

SDN为我们实现第一个条件提供了一个平台。与传统的网络架构不同，SDN提出了网络数据平面与控制平面分离的思想。而网络数据平面与控制平面之间的通信可以通过一种开放的协议——Openflow协议来实现。在数据片面中，基于Openflow的交换设备可以与控制平面进行信息交互并受到控制，而在控制平面有一个管理整个网络区域的SDN控制器。在部署了SDN的网络中，SDN控制器可以掌握网络的全局信息，同时也可以实现网络的全局路由控制。这就意味着借助SDN我们可以实现网络的全局策略最优化

DRL-DQN三条件

在本文中，我们提出了DQN-routing，一种基于DRL中Deep Q learning方法的路由策略优化方案。

实现一个效果足够好的DQN-routing需要大量的训练，但我们不需要过于担心训练的问题。因为我们完全可以将DQN-routing的运行过程和训练过程分离开来。DQN-routing的运行需要借助部署了SDN的网络，但DQN-routing的训练并不依赖于SDN，甚至可以不依赖于任何实际的网络。作为替代，我们需要为DQN建立一个用于训练的环境模型，这个环境模型应该是一个flow-level的通用网络流量-负载模型，通过这个模型我们可以得到网络流量与链路负载的关系。在本方案中，我们简单地将链路负载看作该链路上所有流量的线性累计。但如果要实现更精细的网络控制，我们就必须基于实际的网络数据来建立更准确的流量-负载模型。一些最新的网络分析手段，如基于P4的INT检测[参考文献]，以及一些统计学习方法可以帮助我们建立我们想要的模型。

在本文接下来的章节中我们将首先介绍基于SDN框架实现的DQN-routing控制系统的整体架构以及DRL的相关技术背景。之后，我们将给出对网络拥塞和路由优化问题的描述并具体阐述如何设计DQN以将其用于处理网络拥塞和路由策略优化。最后，我们通过在多种拓扑结构网络下的对比仿真实验论证基于DRL的路由方案在实现网络负载均衡和拥塞避免上的有效性。

}

传统的通信网络中，网络的服务质量(QoS)往往会受到网络拥塞的影响。这是由于传统网络的路由协议大多使用最短路径的路由方式，这种路由方式无法避免网络拥塞的发生。为了解决网络拥塞问题，我们可以通过升级硬件提升网络设备处理能力的方式，但更经济和有效的方法是通过优化网络的路由策略来避免网络拥塞发生。

传统网络中解决网络拥塞的方法是负载均衡。但由于一般通信网络中存在流量密度分布不均，流量高度动态变化等问题，极大地增加了网络负载均衡的难度。一个传统的负载均衡解决方案是ECMP（Equal-Cost Multipath Routing）。当在网络中的源地址和目的地址之间存在多条不同的等价链路时，支持ECMP的网络协议可以同时使用多条链路进行源地址和目的地址之间数据流量的传输。虽然借助ECMP可以在一定程度上降低网络拥塞的发生，但无法从根本上解决网络拥塞产生的原因，因为ECMP无法感知全局网络中的流量状态，因此不能有效地应对网络中流量分布不均而产生的拥塞问题。

软件定义网络（SDN）框架的出现为解决上述问题提供了一个新的可能，与传统的网络架构不同，SDN提出了网络数据平面与控制平面分离的思想。而网络数据平面与控制平面之间的通信可以通过一种开放的协议——Openflow协议来实现。基于Openflow协议的交换设备除了可以进行普通的数据流量转发和传输之外，还能够将采集的实时网络状态信息上传给网络控制平面的控制器。控制器可以收集其所管理的网络区域内的Openflow交换机所上传的信息并进行汇总，并根据收集到的网络状态信息来制定相应的路由策略和转发机制。SDN框架与传统的网络相比具有很多优势，依靠SDN框架我们可以实现网络功能的虚拟化（NFV），而其中一个重要的优势是可以利用控制器实现网络的全局路由控制。这就意味着我们可以从整体的角度出发控制网络中流的路由策略和流量分配以解决网络的拥塞问题，这当然也包括我们之前提到的网络中流量密度分布不均引起的拥塞问题。

当我们可以通过SDN控制器掌握网络状态信息和实现网络路由控制后，我们所需要做的就是对采集到的实时网络状态信息进行分析并采取相应的路由控制操作来实现我们的目标，如流量控制。这个分析——决策的过程可以是由一系列规则构成的专家系统，也可以是由一些公式组成的精确算法，在本文中，我们将会使用基于DRL训练的DNN来实现这个分析——决策过程。

Deep Reinforcement Learning（DRL）是我们所认为的目前机器学习方法中最适合应用于网络路由控制的一种方法。DRL源于传统机器学习方法中的Reinforcement Learning，结合了采用深度神经网络的deep learning，于近几年在一些领域（Atari video games[]，AlphaGo [29]）取得了令人瞩目的成功，击败了这些领域最优秀的人类对手。但遗憾的是，目前DRL的成功仅停留在少数领域，而尚未在网络控制，工业控制等方面得到足够应用。我们回顾了DRL成功的案例，总结了DRL在不同应用场景下获得成功所需的三个条件：一个具有明确规则和定义的环境，一个能够帮助我们实现目标的奖励函数，以及一个能够得到正确和及时反馈的训练系统。我们的一个重要工作就是设计一个满足上述三个条件的方案，使得可以通过DRL方法处理网络路由策略优化问题。

我们认为网络的流量控制和路由决策问题是符合以上条件的，也就是说，使用DRL来实现网络的智能控制和路由决策是可行的，而我们也在本文中提出了基于DRL的网络路由策略优化方案，我们以避免网络拥塞为目标设计了相应的奖励函数，使训练后的DNN可以根据输入的网络流量状态信息输出最优的路由策略，并证明了该方案在实现负载均衡和避免网络拥塞上的有效性。

在本文中，我们提出了DQN-routing，一种基于DRL中Deep Q learning方法的路由策略优化方案。在本文接下来的章节我们将首先介绍基于DRL和SDN框架的网络控制系统的整体架构以及相关技术背景。之后，我们将具体阐述如何设计DRL中的DQN以将其用于分析网络信息进行路由决策优化。最后，我们通过在多种拓扑结构网络下的对比仿真实验论证基于DRL的路由方案在实现网络负载均衡和拥塞避免上的有效性。

1. METHODOLOGY AND IMPLEMENTATION

在本节中，我们首先提出基于SDN和DRL的网络控制系统的整体架构。Figure1展示了这个系统是如何在部署openflow switch的网络中运作的。整个网络控制系统可以分为三个部分:数据平面，控制平面和决策模块，其中数据平面，控制平面来自于SDN架构中的基本概念，决策模块则是由经过DRL训练后的DNN构成，是整个网络控制系统中的关键部分。决策模块独立于数据平面和控制平面之外，不受数据平面和控制平面中采用的协议影响，仅仅与控制平面中的SDN controller进行信息交互。

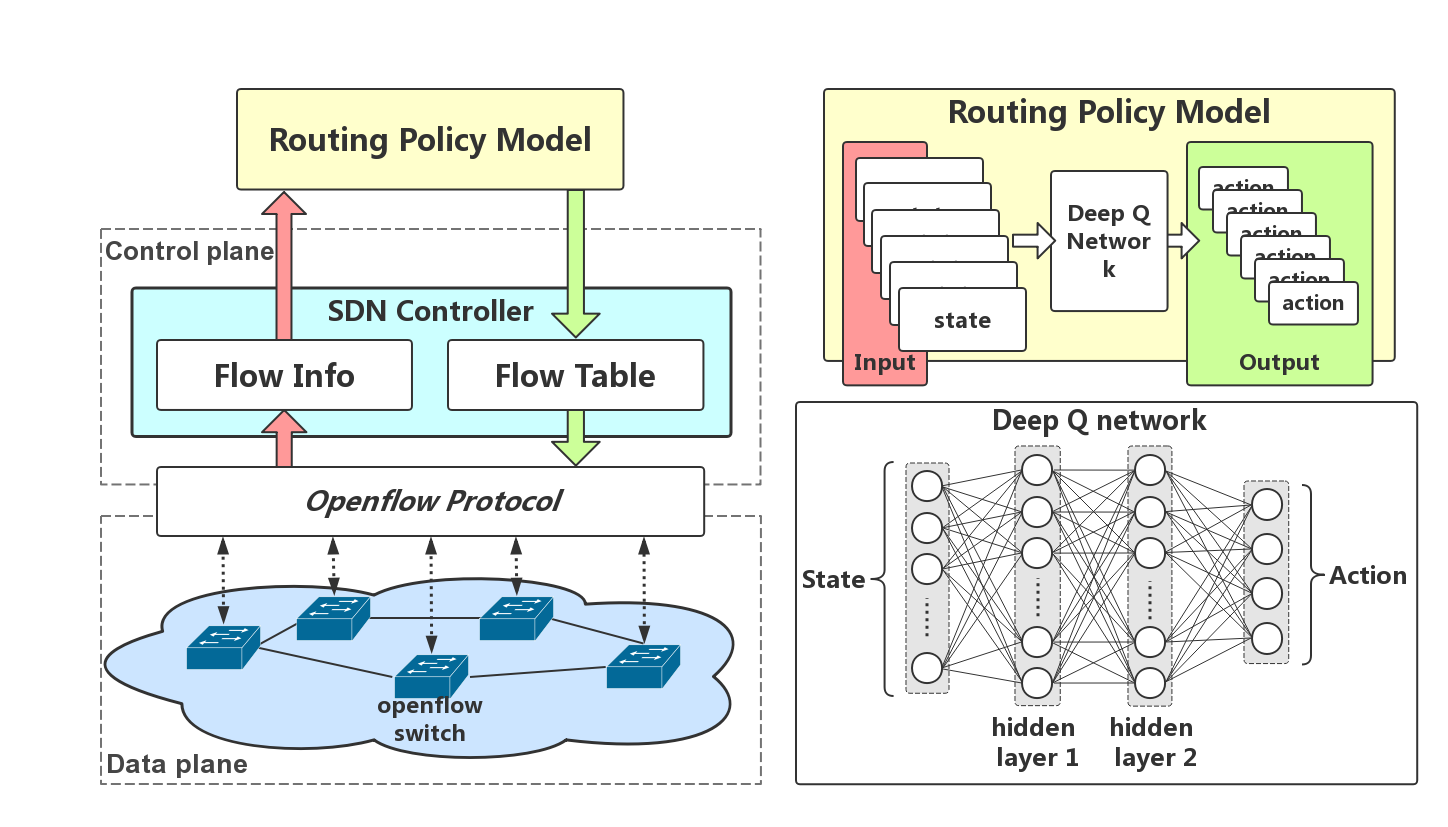


Figure 1: Overview of DRL based load balance with SDN framework

为了对网络进行路由控制，SDN controller会收集网络数据平面openflow switch中的实时流量信息，当采集完成后，SDN controller会将网络流量信息汇总并送达给决策模块。决策模块将接收到的网络流量信息作为DNN的输入，经过训练的DNN会输出用于路由决策的控制信息，在将这些控制信息传送给SDN controller后，SDN controller会根据控制信息修改网络的流表，然后将流表下达给数据平面的openflow switch，使网络中的每个openflow switch根据流表进行相应的流转发动作。

在网络中，SDN controller每收集一次网络实时流量信息，就可以进行一次路由决策和流表更新。SDN controller收集网络实时流量信息的时间间隔可以根据网络的具体情况决定，例如在流量高度动态变化的网络中，我们可能需要一个较小的时间间隔以保证控制系统能够及时调整最优路由策略。

下面我们将详细讨论整个网络控制系统中核心的决策模块。

* 1. DRL

首先我们介绍一下本文所使用的DRL技术的一些基本概念和背景。

强化学习（RL）处理的任务通常使用马尔可夫决策过程（Markov Decision Process，MDPs）来描述：在一个确定的环境E中，有一个状态空间S以及一个动作空间A，强化学习会使用一个代理在该环境中进行决策，状态空间S中的每一个状态s对应着代理所感知的当前环境，动作空间A中的每一个动作a对应着在每种状态下可供选择的动作。当代理在状态s下执行某个可选动作a后，代理所处的状态会按照概率P发生转移。代理所使用的策略决定了代理在各种状态s下选择执行的动作a。而在代理所处的状态发生转移后，环境会根据此次状态转移给予代理一个奖励r。当代理根据策略由初始状态开始执行一系列动作（即进行一系列状态转移）后，代理会得到一个累计奖励。强化学习的目标就是找到一个最优的策略（这里\*表示最优），能够最大化代理得到的累计奖励。。

为了寻找一个最优的策略（一个任务可能会有多个最优策略同时存在），我们需要评估不同的策略，通常我们可以使用一个状态-动作值函数来表示代理从状态s出发，执行动作a后再根据策略选择以后的动作所得到的累计奖励。根据以上定义，我们有：

(1)

其中表示求随机变量的期望，表示起始状态，表示起始状态时执行的第一个动作，表示后续执行动作的步数，表示由第步状态转移到第步状态得到的奖励，是折扣因子(0)。 当我们得到某个策略下所有状态下的状态-动作值函数,后，我们就可以对该策略进行评估，最优的策略应当能够最大化该策略下所有状态s的累计奖励，换句话说，最优策略为我们每个s选择的动作都是s的所有动作中最大的那个，即：

(2)

当策略满足(2)式时，该策略即为最优策略，否则我们就需要对策略进行改进，策略改进的方式主要有策略迭代和值迭代两种，其基本思想都是通过迭代逐步更新策略或者来使策略逐渐接近最优策略。以值迭代为例，根据(1)式中的累加关系和状态转移的马尔可夫性，我们可以使用迭代的形式表示出最优策略下状态-动作值函数的值（该式也称为Bellman方程），由(1)式我们可以得到：

(3)

通过(3)我们可以更新当前策略下所有状态s的，然后根据得到的，我们再使用(2)式来更新当前策略，当我不断更新策略直到当前策略已经满足(2)时，即:

(4)

我们就找到了最优策略。

以上这种最优策略的方法就是RL中的有名的Q-learning。虽然Q-learning可以帮助我们找到RL任务中的最优策略，但当状态空间过于庞大时，求解各个状态下的过程会变得十分困难。为了解决这个问题，一个理想的想法是使用深度神经网络（DNN）通过函数approximate的方式求取。这一想法成功得到应用的开端是DeepMind使用DQN（Deep Q-Network）在玩Atari游戏的表现上达到了人类的水平[参考文献]。DQN的主要结构为一个神经网络——我们称之为Q网络(结构如Figure 2所示)，Q网络可以将高维状态信息s作为输入，输出s状态下可选动作的值，由于输出的是Q网络approximate的，因此我们需要训练Q网络的参数来使得其approximate的更精确，在[参考文献]中，更新Q网络所使用的损失函数的表达式为：

(5)

在确定了神经网络的损失函数后，我们就可以通过常见的反向传递和梯度下降法来更新Q网络的参数。值得一提的是，在Q网络输出所有动作的后，代理会以概率执行所有动作中最大的动作，而以的概率执行随机动作，这里的作用是用于保证Q网络在更新当前策略的Q值的同时还能有机会探索其他可能的策略，一般我们将称为。当Q网络在经过足够多的探索和更新之后，Q网络输出的就会接近最优状态-动作值函数，同时当前策略也会逼近最优策略。

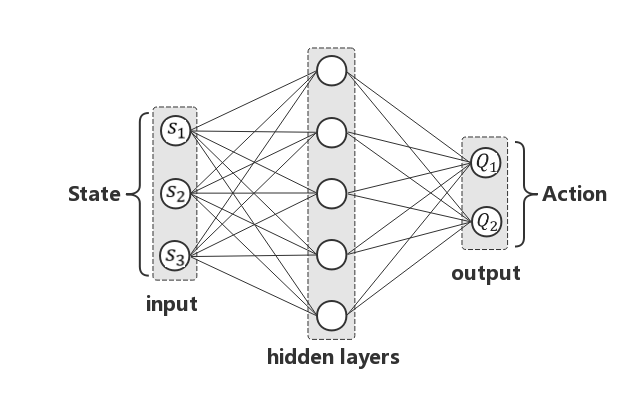


Figure 2 : Structure of a simple Q network

DeepMind在DQN取得成功之后，又不断提出DQN的改进方法，[Nature参考文献]中在基本的DQN基础上增加了Experience Replay和Target Net，Experience Replay将用于训练数据储存到储存区里，之后使用随机采样的方法从储存区中选择用于训练DNN的数据，这降低了用于训练的数据之间的相关性，有效提升了性能。Target Net则是为DQN添加与原来的Q网络结构相同的目标网络，目标网络不进行运算，每经过一定回合会储存一次原来的Q网络中的参数，储存的参数用来为原来的Q网络的更新提供目标值。在这之后，DeepMind又提出了多种新的改进方法，例如Double DQN[参考文献]，Prioritied Replay[参考文献]，Dueling Network[参考文献]等等。Double DQN改进了目标Q值的计算，减少了使用maxQ值引起的过度估计问题（over estimation），Prioritied Replay使用优先采样取代了随机采样，在采样时会更大概率选择目标Q值与当前Q值差别较大的样本。Dueling Network修改了Q网络结构，将Q网络的输出分为两个部分，分别评估动作价值和状态价值，最后再合起来得到Q值。

DQN使用基于Q值的更新方式，可以用来处理动作空间离散的任务，而DRL中另一类基于策略的方法PG[参考文献]则可以处理动作空间连续的任务，典型的有Actor-Critic[参考文献]，A3C（Asynchronous Advantage Actor-Critic）[参考文献]，DDPG（Deep Deterministic Policy Gradient）[参考文献]等。这些方法与DQN基于Q值进行策略更新不同，是采用策略梯度的方式进行策略更新。在本文提出的基于DRL进行路由策略优化的方案中，作为输出动作的路由选择是离散而非连续的，因此方案中采用了DQN的方法，具体的路由决策问题描述和如何使用DQN优化路由策略将在下面进行详细的介绍。

* 1. Problem statement

本节我们将详细介绍如何将网络路由决策优化定义为一个可以使用强化学习方法解决的问题。

首先我们确定网络的模型：我们考虑一个具有若干个通信节点，m条物理链路的通用通信网络，在这里网络中的每个通信节点对应着本方案中SDN架构下数据平面中的每个openflow switch。所有的通信节点可以分为两种：源节点和转发节点，源节点是在网络中产生和最终接收数据流的节点，网络中所有的数据流都由源节点产生，由一个源节点产生的数据流最终都会到达另一个源节点。源节点对应着网络中连接host，服务器等其他设备或网络的openflow switch（对应于传统网络中的边缘路由器Provider Edge router），网络中源节点的个数为n，我们用表示源节点。转发节点是网络中负责转发数据流的节点，转发节点不产生数据流，它们仅仅对由其他节点传输过来的数据流基于流表进行转发操作，在本文的方案中我们不关心网络中转发节点的个数。

然后我们定义流（flow）和流的路由方式：在上述的网络模型中，我们将网络中所有由相同的源节点出发，最终到达相同的目的节点（流的源节点和目的节点不能为同一个）的数据包归为一类，此类数据包共同组成一条流。（可以推断出，在一个具有n个源节点的网络模型中最多有条流）为了定量地描述每一条流的在网络中的流量需求，我们定义：一个以节点为源节点，节点为目的节点的流在链路中正常传输所占用的链路带宽为（在本文的后面部分我们都使用来描述每条流）。对于每条流，我们会为其在源节点和目的节点间确定条备选路由，每条备选路由指定了流由源节点出发到达目的节点所经过的所有链路。这条备选路由应满足以下条件：(1)无环，(2)各条路由经过的路径不能完全相同，(3)到达目的节点的距离尽可能短（距离可以使用跳数或其他传统网络中常用的度量方式）。在本文的方案中，系统进行路由决策的基本方式就是对于网络中每一条流，为其从备选路由中选择一条作为实际的路由。需要强调的是，在本方案中流是我们在路由决策中所控制的最小单位，这意味我们对网络的控制不会深入到流以下的等级，虽然将流细分为流以下的等级可能能够取得更好的效果，但以流为最小单位进行网络控制对使用流表控制网络转发的SDN controller来说是易于实现的。

最后我们描述网络的拥塞问题。对于网络模型中的m条物理链路，我们为每条物理链路指定两个参数：链路的最大可用带宽的阈值和链路的实时流量负载（与使用相同的度量单位）。在上面流的定义中，我们将流在链路中正常传输所需要的链路带宽表示为。对于一条链路，如果当前状态下有多条流（例如有三条流）的路由经过该条链路，那么我们就定义该链路上的实时流量负载等于这多条流正常传输所需的链路带宽之和（即链路实时流量负载=）。如果的值超过了链路的最大可用带宽阈值，那么我们认为此时链路上会有拥塞发生，值超过阈值的程度对应着链路上发生拥塞的严重程度，拥塞的严重程度越高，链路的吞吐量越低，流经链路的流的时延也越高（即的传输延迟越高）；如果的值不超过链路的最大可用带宽阈值，那么我们则认为此时该链路上不会有拥塞发生，此时链路的吞吐率量着链路实时流量负载的增加而线性增加，同时流经链路的流能够在可接受的时延范围内进行传输。

至此我们已经详细描述了路由决策问题中的网络模型，流和流的路由方式以及网络拥塞的界定。对于这个路由决策问题，我们想要实现的目标是：在一个具有n个源节点，m条物理链路的网络模型中，在得到某一时刻网络中所有流（共有条）的所需链路带宽向量（若此时刻某条流不存在，其所需链路带宽为0）的条件下，为各条流选择最合适的路由，使得该路由策略下网络中发生拥塞的链路数量最少。我们对于整个问题的流程描述如下：

Given flow demand vector ,where

for in:

Choose in for every

Then we get routing vector )

Based on flow demand vector and routing vector ) we get traffic load vector of links

for in links:

Congestion

Minimize

可以看出，我们这里要解决的路由决策问题是一个限定条件下（即网络中所有流所需链路带宽向量），在离散的有限解空间（每条流的备选路由）内寻找最优解（使网络发生拥塞的链路数量最少的路由向量）的问题。该问题的求解可以通过设计巧妙的精确算法（exact algorithm）来实现，但使用精确算法求解来实现网络路由策略优化会产生两个关键的问题：第一个问题是，我们所研究的网络流量状况是随时间动态变化的，这意味着我们在问题中给出的限定条件是在不断改变的，为了保证网络始终能够采取最优路由策略以避免网络拥塞，我们需要在每次网络流量状况产生变化后都使用精确算法重新计算一次最优解，以前得到的最优解的结果并不能对我们之后的计算最优解过程产生帮助，这种方式无疑是繁琐且低效的。另一个更重要的问题是，随着网络规模的增大，我们所面对的问题的解空间和寻找最优解所需的计算量都会迅速增加，这意味着对于规模较大的网络，我们必须始终为网络配置足够的计算资源以保证每次网络流量变化后都能够在可容忍的时间内找到最优解，这会为网络控制系统增加额外的资源和能量消耗，同时对网络运营商来说也是不经济的。与使用精确算法不同，通过使用DQN我们可以很好的解决上述的两个问题，这是因为一个使用DQN训练过的DNN能够根据输入的网络实时流量状态给出迅速的策略输出，而且这个过程并不会消耗太多的计算资源。

* 1. DQN based routing policy optimization

在前文中我们提出DRL方法的成功应用需要三个基本条件，2.2中我们对网络路由策略优化问题进行了详细的定义，这为使用提供了第一个条件：一个具有明确规则和定义的环境。在本节接下来的部分中，我们将详细介绍DQN－routing的设计并说明我们的设计是如何满足另外两个条件的。

使用DRL处理任务时我们必须要确定DRL的输入状态，输出动作，初始状态和结束状态的定义以及用于策略优化的奖励函数。Figure 3详细地illustrates了本文中DQN输入输出的设计，以及在一个回合中，由初始状态开始执行动作进行状态转移直到结束状态的全部过程。每个回合的初始状态由该回合我们输出的网络流量状态信息（即flow demand vector (）所确定，每次输出的动作会为我们确定网络中某一条流的路由，当回合结束后，我们在本回合中所输出的全部动作就为我们确定了网络中所有流的路由（即routing vector )），然后我们就可以将路由信息转换为SDN controller中的流表，从而控制网络中的openflow switch按照流表进行相应的转发操作。这就是DQN根据网络流量状态信息进行路由决策的基本过程，接下来我们先介绍DQN的状态输入，动作输出的设计。

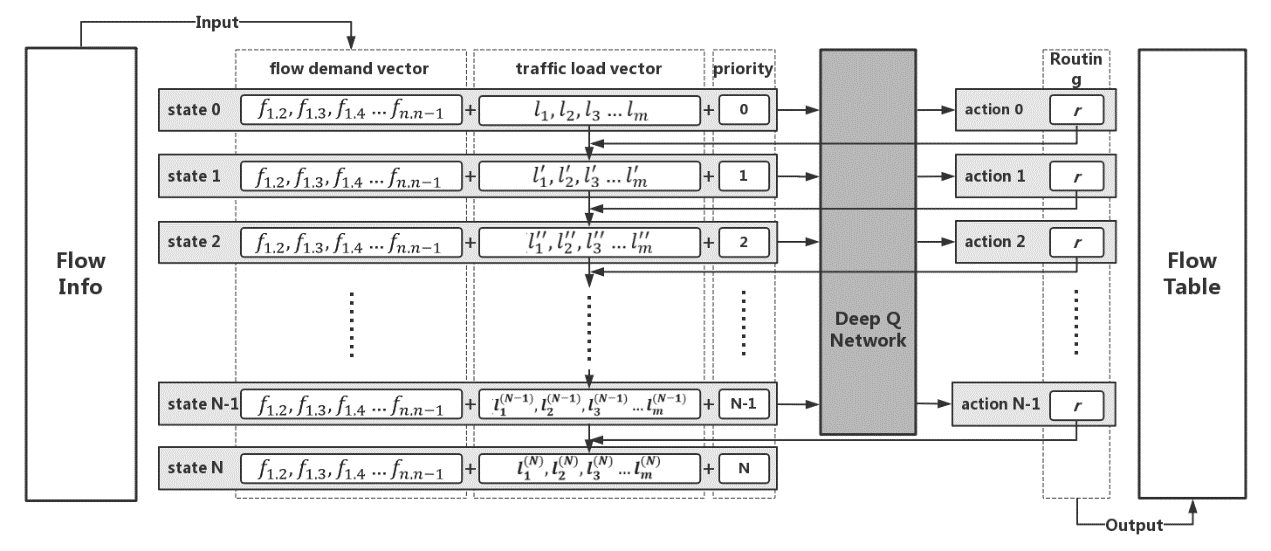


Figure 3 决策模块从网络流量信息输入到路由策略输出的一个完整过程

Figure 3中我们可以看出，DQN的输入状态由三个部分构成：flow demand vector (, length is ), links traffic load vector , length is m) 以及 priority（one number from 0 to n）value. 其中flow demand vector来源于网络流量状态信息。在一个回合中一系列的状态转移过程中，flow demand vector是不会随着状态转移而改变的，这是因为在一个回合的训练中我们所进行的路由决策以及得到的奖励只针对这一组flow demand vector，对于其他flow demand vector条件下的路由决策和奖励将会在其他回合进行。输入状态中的links traffic load vector部分是用于描述当前状态下各条链路上负载情况的，与flow demand vector不同，输入状态中的links traffic load vector部分的值是随着状态转移而不断变化的，每次状态转移中links traffic load vector的变化会由上一个状态和上一个状态所选择的动作来决定。Priority是用来表示网络中流的路由决策优先级的，同时也用来确定各个状态的顺序，初始状态下Priority的值为0，之后每执行一次动作，状态转移后的新状态的Priority值就加1，，关于流的路由决策优先级的问题，我们将在后面详细叙述。

然后我们讨论DQN的输出。使用DQN处理网络路由策略优化时我们会遇到一个问题，DQN的动作输出决定了其不适合处理拥有高维度动作空间的控制问题，而对于一个规模较大的网络来说，网络中所有流的routing vector 〖(r〗\_1,2^\* 〖,r〗\_1,3^\*,r\_1,4^\*…r\_(n,n-1)^\*)组成的向量空间会十分庞大。对DQN来说，每次都将一个高维度的routing vector作为输出是不可接受的，正是由于这个原因，[参考文献][参考文献]等基于DRL的网络控制方法中往往会使用Actor-Critic，DDPG等能够处理高维度连续动作空间的方法。但在本文中，我们参照RL中处理序贯决策问题的思想，将单次动作输出高维度的routing vector〖(r〗\_1,2^\* 〖,r〗\_1,3^\*,r\_1,4^\*…r\_(n,n-1)^\*)的方式分解为每次动作输出一个r\_(i,j)，连续多次动作构成一组routing vector，从而使得路由决策问题变成一个DQN可以解决的低维度动作空间问题。在将状态输入到DQN后，DQN会输出一个动作r，这个动作表示我们为网络中的一条流选择的路由。假设对于网络中每条流，我们多为其确定了4条备选路由，那么这里输出的动作r的值就可能是0，1，2或是3，如果动作输出为2，就代表我们为该条流选择了编号为2的那条备选路由。由于网络中最多会有条流，所以在一个回合中我们总共会执行次动作，进行次状态转移。在状态转移中，每次选择的动作r都会影响下个状态中的links traffic load vector，在初始状态中links traffic load vector (l\_1,l\_2,l\_3…l\_m)中所有的值均为0，这是因为初始状态时我们还未确定任何一条流的路由，而当我们确定了一条流f\_(i,j)的路由r\_(i,j)后，通过这条流的所需链路带宽f\_(i,j)和路由r\_(i,j)的路径，我们可以计算出这条流给网络中链路增加的负载，然后根据得到的新的links traffic load vector (l\_1^',l\_2^',l\_3^'…l\_m^')我们就能够确定下一个状态。当所有的次动作都执行完之后，我们就为网络中的所有条流确定了各自的路由。

在确定了DQN的输入状态和输出动作之后，我们需要定义用于策略优化的奖励函数。由于我们的目标是尽可能地减少网络拥塞的发生几率和拥塞的程度，因此我们需要面对两种情况，当网络中各链路上的负载低于链路最大可用带宽阈值时，我们需要使得负载尽可能的远离阈值，而当某条链路的负载超过阈值时，我们需要使得该链路的负载尽可能的接近阈值。为了实现这个目标，我们要定量地描述各链路上负载与阈值的关系，因此我们定义了一个网络的最大负载值maximum loading Value（MLV），MLV的定义如下：

(4)

MLV给出了一个网络中负载最重的链路上最大可用带宽阀值与链路负载的差值，通过MLV的值我们可以定量地描述网络负载均衡的状态，当MLV的值为正时，网络中所有链路上的负载均小于阈值，网络中没有拥塞发生，此时MLV的值越大，我们认为网络中的负载就越均衡，当MLV为负时，网络中有某条链路上的负载超过了阈值，即网络中某处发生了拥塞，此时MLV越小，就表示网络的拥塞越严重。由于MLV的这种性质，在DQN的训练中，我们使用MLV作为reward函数。Reward只会在每一个回合结束时被给出，reward的值根据结束状态中的traffic load vector得到。每一个正的reward都会奖励此回合DQN所选择的路由策略（使Q值增大）每一个负大的reward则会使DQN此回合的路由策略得到惩罚（Q值减小），当DQN逐渐学会如何选择输出动作来得到更大的reward之后，我们基于DQN的网络路由策略也会逐渐得到优化。之所以选择使用最值而非均值来当作描述网络拥塞状况的reward，是因为考虑到网络拥塞的特殊性，由于网络拥塞往往是网络负载不均导致的，因此我们需要对任何可能导致网络负载过度集中的路由策略进行惩罚，通过寻找网络中负载状态最极端的链路，我们可以很容易判断哪些策略是好的（使网络负载均衡），哪些策略是不好（使网络负载集中），而使用平均值会使得好的策略和不好的策略更加难以区分。事实上，在之后的仿真实验中，我们也会使用这个reward来当作衡量网络拥塞状况的指标。

最后我们将讨论DQN为网络中各条流选择路由策略的顺序问题（即上文提到的Priority）。在本方案最初的仿真实验中，DQN为网络中各条流选择路由策略时使用的是默认顺序，即首先将源节点编号，然后根据源节点的顺序为各条流依次排序。当我们在一个拥有对称拓扑的网络中进行路由决策时，无论采用默认顺序还是随机顺序对于DQN路由决策的训练来说效果都是一样的，但当我们使用非对称拓扑的网络来进行路由决策时，我们发现为不同的流选择路由的顺序对DQN的训练有着显著的影响，一些情况下DQN的训练速度会显著提升，而另一些情况下DQN的训练速度十分缓慢甚至难以收敛。通过多组实验的对比，我们发现这与不同流的备选路由有关：当某条流的备选路由中有一条“理想”路由时（这里“理想”是指这条路由通过的路径上其他流可能带来的负载很小，即该条路由所流经的路径不易发生拥塞），优先进行这些流的路由选择可以让DQN更快的训练和优化路由决策，而如果优先对那些备选路由均非“理想”路由的流进行路由选择，会使得DQN需要更多的训练时间，而且训练效果也很不理想。这很容易理解：当一条流的备选路由中有一条明显优于其他路由时，DQN很容易就能学习到这条流的最优路由策略，这条流在回合中的顺序越靠前，DQN就越快能学习到这条流的最优路由策略，同时优化整个网络的路由策略所需要探索的解空间也就越小，训练也就越容易。基于这个原因，我们在DQN中流的路由策略选择顺序上提出了流的优先级算法：首先假设网络中的某条流有条备选路由，其中某条备选路由所经过的链路为，而在该网络中所有其他流的备选路由经过的链路中出现的总次数分别为，那么我们定义这条路由评价值为，值越大，就意味着这条路由经过的链路的使用率越低，选择这条路由造成拥塞的可能性越小。同理对于的其他备选路由我们也能够得到相应的评价值，当确定了所有备选路由的值后，我们就可以给出优先级的参考值：

(5)

其中，式子中减号左边的项为流f\_(i,j)所有备选路由的组成的集合中的最大值，减号右边的项为集合E中第二大的元素，通过计算两者的差值我们确定流所有备选路由中最“理想”的路由优于其他路由的程度。最后计算出的参考值越大，流的优先级就越高。将所有的流按照优先级排序后（Priority值从0到），DQN训练时就可以按照输入状态中的Priority值为所有的流按照优先级选择路由策略。

训练： 对于任何网络来说，我们只需要根据网络可能的flow demand vector空间来为DQN生成大量随机的flow demand vector作为训练过程的输入，而当训练结束，将DQN用于路由决策的时候，我们根据SDN controller为决策模块所提供的flow demand vector确定初始的输入状态，然后利用训练好的DNN进行路由决策输出即可。一般来说，对于网络中所产生的任何可能的flow demand vector，只要其大小在我们之前用于训练的随机生成的flow demand vector的空间范围之内，且能够保证我们的训练次数足够多，那么训练后的DQN都能够对于该组flow demand vector给出很好的路由策略。

在本文的方案中，我们所使用的DQN是DeepMind所使用的Nature版本的DQN，即使用了Experience Replay和Target Net的DQN。除此之外，我们还使用了上文中提到的Prioritied Replay和Dueling Network改进方法以提升DQN的训练速度和效果。值得一提的是，在仿真实验中我们发现，使用Prioritied Replay来提升DQN训练速度的同时也能有效地改善我们上文提到的顺序问题。一些实验结果表明，在非对称网络拓扑中，Prioritied Replay能够有效改善不使用优先级算法排序所导致的收敛困难问题，而如果我们同时使用优先级算法排序和Prioritied Replay，最终DQN的训练效果会比单独使用Prioritied Replay或优先级算法排序的情况都要好。因此为了取得最好的训练效果，我们在方案中同时将使用优先级排序和Prioritied Replay。

1. EVALUATION

为了评估本文所提出的基于DRL的路由策略优化方案的效果，我们进行了大量的仿真实验，在本节的其他部分我们将详细分析我们的对比实验的条件和结果。我们基于python为本方案中的路由决策模块搭建了DQN神经网络，并实现了DQN的训练和测试的功能。除此之外，为了能够准确地实现我们在2.2节problem statement中描述的网络流量控制环境，我们还使用python编写了流级别的网络仿真环境，用以模拟SDN架构中SDN controller采集网络中流量信息和修改流表控制流量转发路径的行为。这不仅保证了我们的网络流量控制仿真环境能够与基于DQN的路由决策模块完美配合，而且可以让我们不用关心仿真环境与决策模块繁琐的数据交互过程。

* 1. network topology and hyper-parameters

首先说明我们在仿真实验中希望达到的目标，本文提出的方案的最终目的是通过优化路由策略来实现网络的负载均衡，避免网络拥塞的发生，因此我们需要和传统的网络负载均衡方案进行对比，在现代通信网络中使用最广泛的一种负载均衡方案是ECMP，即将一条流的流量负载均匀分配到该流的所有可用等价路径上。在本文中，由于在之前的定义网络中流的路由方式时已经为每条流确定了多条备选路由，因此我们可以将一条流的流量负载均匀分配给所有的备选路由。不过我们并未要求所有的备选路由等价，所以这种负载均衡方式并不是严格意义上的ECMP，在仿真实验中我们称这种负载均衡方式为MPR。对于大部分拥有随机拓扑网络来说，我们为每条流确定的备选路由的数量是不少于该网络中流的等价路径的，因此我们可以认为MPR的效果是近似或好于ECMP的。因此在我们的仿真实验中，我们使用MPR来作为DQN based Routing的对比，我们的目标是证明经过训练的DQN路由决策方案在避免网络拥塞的能力上优于MPR。除此之外，作为MPR和DQN based Routing负载均衡效果的对照，我们还额外设置了一组使用随机路由策略的方案，该方案会为网络中每条流从备选路由中随机选择一条（这种随机路由策略可以看作是未经过训练的DQN-Routing）。

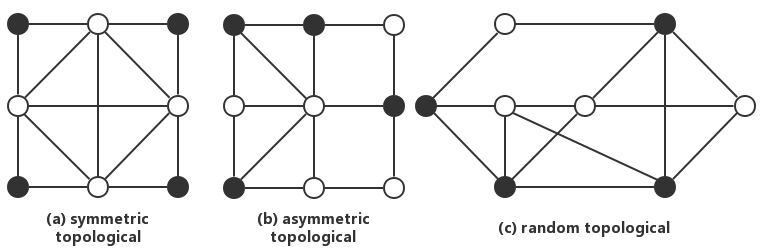


Figure 3 用于仿真实验的三种不同拓扑结构网络

在仿真实验中我们使用了如Figure 3所示的三种拓扑结构的网络，其中我们使用黑色的节点来代表网络中的源节点。(a)网络是拥有一个4个源节点和14条链路的对称拓扑网络，(b)网络是在(a)的基础上修改了拓扑结构的非对称拓扑网络，(c)网络是一个随机生成的具有4个源节点和14条链路的网络。除此之外，为了验证DQN-Routing的一般性，我们还使用了一个拥有NSFNET拓扑结构的网络，并在NSFNET14个节点中选择6个作为源节点。之所以使用不同拓扑结构的网络进行实验，是由于我们考虑到网络拓扑会对使用MPR的负载均衡效果产生影响，因此我们使用网络(a)和(b)来研究对称拓扑结构网络和非对称拓扑结构网络中DQN based Routing和MPR负载均衡的效果。然后我们使用网络(c)和NSFNET来对比DQN based Routing和MPR在网络中不同流量需求下的表现，

在(a)和(b)中，我们将所有链路的最大可用带宽threshold设为50Mbps，同时使用一组随机向量来模拟网络的实时flow demand vector，随机向量的值服从[0,20]Mbps范围内的均匀分布。在(c)和NSFNET中我们同样将链路带宽threshold设为50Mbps，但与(a)和(b)不同的是，为了更准确地分析不同流量需求大小下MPR和DQN based Routing的表现，我们将生成flow demand vector值的均匀分布的范围缩小为[0,10]，并控制范围窗口按照每次1Mbps的步长向上滑动（从[0,20]Mbps到[1,21]Mbps），从而逐步增加产生的flow demand vector。在这两组实验中我们都使用100000次随机flow demand vector条件下产生网络拥塞的概率和我们之前定义的最大负载值(MLV)均值来作为衡量MPR和DQN based Routing负载均衡能力的标准。

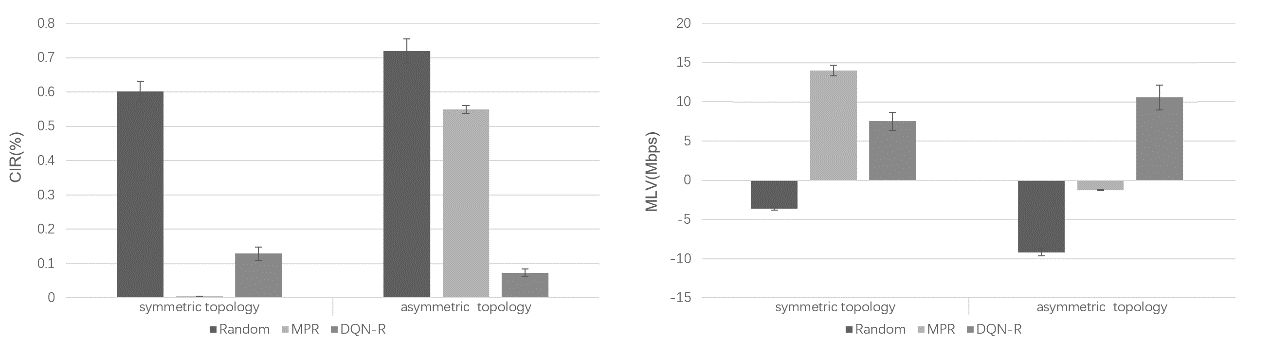
最后我们给出仿真实验中DQN的模型结构以及超参数。一般来说，神经网络的隐藏层的层数和隐藏层中神经元的个数与神经网络的输入输出大小有关，输入的特征值的数量越多，神经网络的隐藏层的层数与神经元的个数就越多。在我们的方案中，DQN的输入状态中flow demand vector和links traffic load vector的大小都与网络规模有关，因此我们的DQN的结构需要根据我们所使用的网络的规模来确定。在本文的仿真实验中，我们使用的是只拥有一个隐藏层的DQN，隐藏层的神经元个数为50个，我们的实验结果表明，这种结构的DQN足以处理小规模网络（源节点数量不超过6，链路数小于20的网络）中路由策略的优化问题，当需要对更大规模的网络进行路由策略优化时，我们就需要适当的增加DQN的隐藏层的层数和隐藏层中神经元的个数。除了隐藏层层数和神经元个数之外，使用DQN进行训练的过程中我们还需要设定一些其他的超参数，这里我们给出本文在实验中所使用的效果较好的超参数，如表1所示，其中-greed increment代表每回合结束后的增加值（初始为0），-greed increment的值是根据我们设定的训练回合数所确定的，为了达到足够的训练效果，仿真实验中我们设定每组网络环境下DQN的训练回合数为1million。

Table 1：hyper-parameter of DQN training

|  |  |
| --- | --- |
| hyper-parameters | value |
| learning rate | 0.001 |
| reward decay | 0.9 |
| -greed | 0.9 |
| -greed increment | 0.000002 |
| Q target replace interval | 500 |
| memory size | 10000 |
| batch size | 32 |

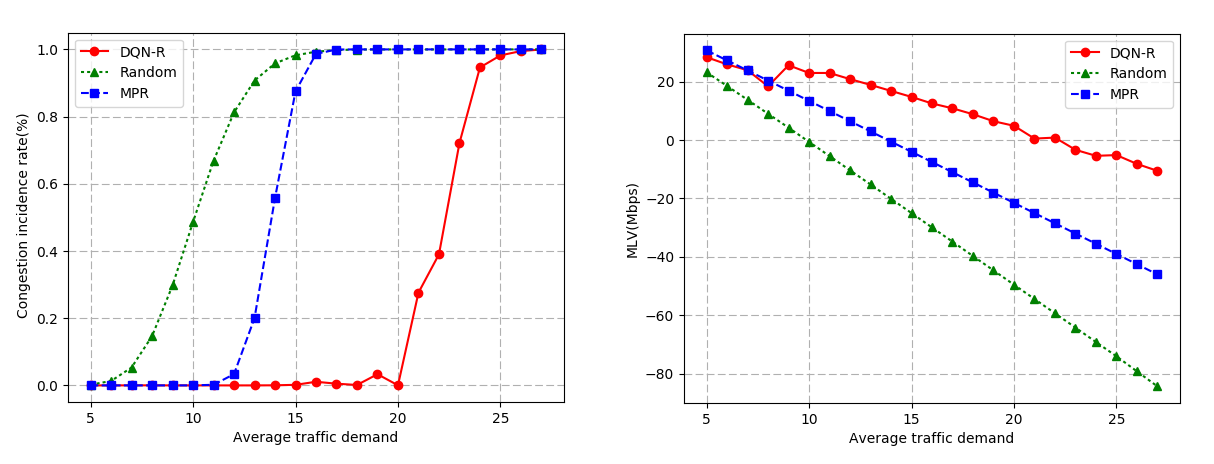
* 1. Results

Figure 5 显示了我们在对称拓扑网络(a)和非对称拓扑网络(b)中分别使用Random routing，MPR和DQN based Routing的结果。从Figure 5 a 中我们可以看出，MPR在对称拓扑网络中有着非常优秀的表现，100000次测试中拥塞的发生率仅仅只有不到1%，与此同时使用DQN based Routing的拥塞发生率大约在13%左右，但与训练前使用Random routing方式（拥塞发生率高达61%）相比拥塞发生率大幅减少，从Figure 5 b中我们可以看出对称拓扑网络下MPR的Maximum loading value是DQN based Routing的两倍。很明显在对称拓扑网络我们更适合使用MPR来进行网络的负载均衡，这是由于在对称拓扑网络中MPR能够通过分流的方式更加充分地使用每条链路，而DQN based Routing虽然能优化流的路由策略，但无法进行流级别以下的流量分配，因此对于数据中心网络等具有对称拓扑结构的网络来说，使用DQN based Routing进行负载均衡的效果还未能超过MPR。但是在Figure 5 a的非对称拓扑网络中，DQN based Routing的效果却远超过MPR，实验中使用MPR的拥塞发生率达到了55%，而DQN based Routing却只有7%，当Figure 5 b中DQN based Routing的Maximum loading value达到10时MPR却甚至无法超过0。这种显著的差异体现出了MPR在非对称拓扑网络的问题：即当网络中流量分布差异较大时，仅仅通过分流难以解决流量集中处可能产生的拥塞问题，而DQN based Routing却可以通过更有效的路由策略来改善网络中流量的分布，因此对于大多数具有非对称拓扑结构的通信网络来说，我们认为使用DQN based Routing能够比MPR能有效地避免网络拥塞问题。



1. Congestion incidence rate (b) Maximum loading value

Figure 4 Random routing，MPR和DQN based Routing在(a)(b)两种拓扑网络中的表现



(a)Congestion incidence rate (b) Maximum loading value

Figure 5 Random routing，MPR和DQN based Routing在随机拓扑网络(c)中的表现

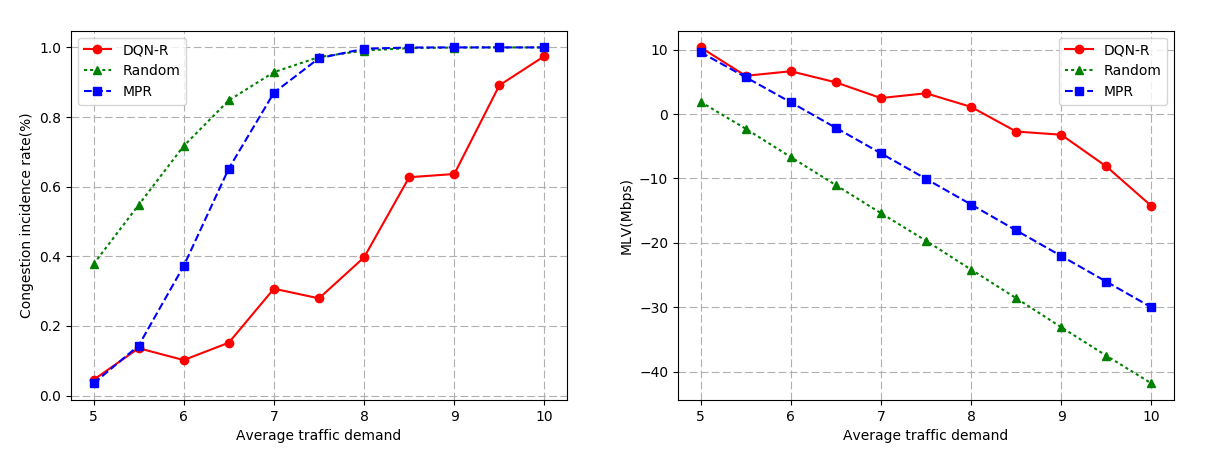


Figure 6 Random routing，MPR和DQN based Routing在随机拓扑网络(c)中的表现

然后我们在随机拓扑结构的网络(c)和NSFNET中对比测试不同网络流量需求下DQN based Routing和MPR应对网络拥塞的能力，其结果如Figure 5所示。由于我们使用了一个大小为10的窗口作为均匀分布的范围（向前滑动）来为网络生成随机的flow demand vector，因此在Figure 5a中我们将窗口的中间值作为横坐标。正如Figure 5a所表现的，我们可以看出在网络中总网络流量需求还不大时，random routing下的网络拥塞发生率就开始增加，在average traffic demand值为15时网络拥塞就已经接近100%，而MPR虽然在average traffic demand达到11之后才开始产生网络拥塞，但随着流量需求继续增加，在average traffic demand值超过17时就已经完全无法避免网络拥塞的发生了。相比random routing和MPR，DQN based Routing在average traffic demand值超过20之前都能够有效地避免网络拥塞的发生。DQN based Routing拥有这种优秀的表现的原因我们可以从Figure 6b中发现：随着average traffic demand的增加，三种方法的MLV都是线性下降的，但下降速率并不相同。在网络流量需求较低时，DQN based Routing，MPR以及random routing有着近似的MLV均值，随着average traffic demand增长，MPR和random routing的MLV均值下降的速率明显快于DQN based Routing。MLV描述了网络中负载最重的链路的情况，当MLV的均值接近0时，意味着该average traffic demand条件下，网络中未发生拥塞（MLV>0）的概率和发生拥塞（MLV<0）的概率相当，也意味着此时网络中的拥塞已经难以避免。通过对比DQN based Routing，MPR和random routing的MLV的下降速率，我们可以

认为DQN based Routing相比于MPR能够更加有效地应对网络中traffic demand较高的情况。

另外，为了验证我们在前文中提出的路由决策中流的优先级算法，我们在非对称拓扑网络(b)中进行了一组对比实验：在其他实验环境完全相同的条件下，我们对比了使用流的优先级算法排序的DQN和不使用流的优先级算法排序（使用默认顺序）的DQN的训练过程，结果如Figure 7所示，使用流的优先级算法排序的DQN随着训练次数增加reward稳步上升并趋于收敛，相比之下不使用流的优先级算法排序的DQN则难以收敛，reward甚至趋于下降，因此我们认为使用流的优先级算法排序对本方案中DQN的训练和优化来说是十分必要的。

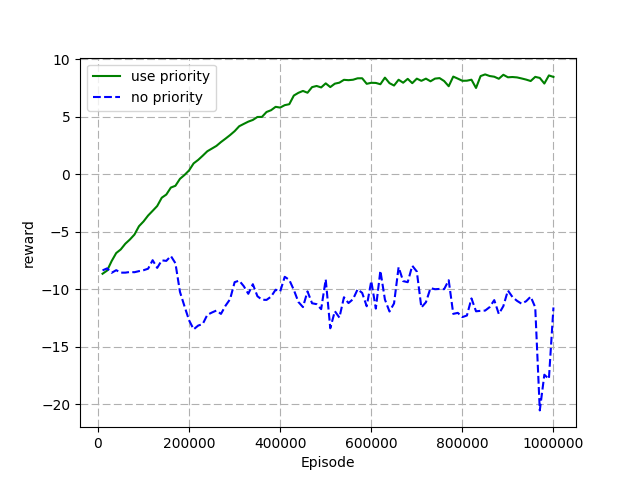


Figure 7 网络(b)中使用流的优先级算法排序和不使用流的优先级算法排序的DQN的训练表现

通过大量的仿真实验中，我们还发现DQN based Routing在网络冗余链路较多时效果更好，同时DQN based Routing的效果也与我们选择的备选路由有关：当网络中所有流的备选路由构成的空间越大时，DQN可能找到的最优路由方案就越好，但同时训练的难度也越大，但备选路由构成的空间并不是越大越好，因为一点点微小的路由策略的优化可能会伴随着指数级的训练难度提升，因此在实验中我们为每条流确定的备选路由为4条，考虑到实验所使用的网络的规模，我们认为这个数量是合适的。

1. RELATED WORK

随着DRL在各个领域不断取得成果，这种强大的技术越来越受到人们的重视。理所当然地，将DRL应用在流量工程（TE）上的研究也正在迅速增加。在[参考文献]的研究中，Zhiyuan Xu等人提出了一种基于Actor-Critic和prioritized experience replay的DRL方法用于优化流量工程来以提高网络利用率。在[参考文献]中，作者在TE上使用了监督学习来预测网络流量需求，然后使用DRL生成链路权重用于路由决策。[参考文献On Removing]中的作者提出了结合CNN的DNN架构，通过DNN判断输出的路由组合是否会引发拥塞。除了对于传统网络中使用DRL处理TE的研究外，在SDN框架下进行TE问题的优化也是研究的热点。例如等人在文献[2]中提出了在SDN框架网络中TE问题的近似算法。而最近等人也提出了结合SDN和DRL进行网络路由优化的方法[参考文献]，但与本文不同的是，他们将网络链路权重作为输出来控制网络的路由。

由于网络控制问题的复杂性，上述大多数基于DRL实现网络控制的方法中往往都使用维度较高的输出，这无疑会增加训练的难度，同时也会限制系统的可扩展性，本文提出的通过使用组合动作输出的方式简化了网络中对路由的控制，为基于DRL的网络控制问题提供了一种新的思路。

1. CONCLUSION AND FUTURE WORK

在本文中，我们提出了一种使用SDN框架和DRL方法对网络进行路由策略优化的方案，该方案可以根据通信网络中的流量信息动态地调整路由策略，从而降低网络拥塞的发生。在方案中我们使用了DRL的经典方法DQN来训练用于网络路由决策的DNN，同时为了降低动作空间的维度，我们创新性地将一次动作为网络中所有的流分配路由的方式分解为每次动作为一条流选择路由，多次动作完成所有的路由选择，从而使得基于DQN的训练方式能够实现。除此之外，我们还提出了路由决策时为所有流按照优先级排序的算法，以加快训练速度。最后我们通过仿真实验，证明了与传统的负载均衡ECMP相比，基于DQN的路由决策优化方案在非对称拓扑网络中能更加有效地实现负载均衡并避免网络拥塞的发生。虽然我们能看到我们所提出的DQN-routing在仿真实验取得了中令人鼓舞的效果，但是其还有很多不足之处和可以改进的空间。我们相信在以后的研究中，我们能够在DQN-routing的基础上实现一个实用性更高，可扩展效果更好的通用网络优化方案。

REFERENCES

deepmind-mastering-go

DQN nature nature14236 Human-level control through deep reinforcement learning

DQN PlayingAtariwithDeepReinforcementLearning

Double Deep Reinforcement Learning with Double Q-Learning.

[Prioritized Experience Replay](https://link.zhihu.com/?target=http%3A//arxiv.org/abs/1511.05952" \t "_blank)

Dueling Dueling Network Architectures for Deep Reinforcement Learning

PG Policy Gradient Methods for Reinforcement Learning with Function Approximation

AC Off-Policy Actor-Critic

[14] V. Konda and J. Tsitsiklis, Actor-critic algorithms, *NIPS’2000*. pp. 1008–

1014.

A3C [Asynchronous Methods for Deep Reinforcement Learning](http://proceedings.mlr.press/v48/mniha16.pdf" \t "_blank)

A3C GPU [Reinforcement learning through asynchronous advantage actor-critic on a gpu](https://openreview.net/forum?id=r1VGvBcxl&noteId=r1VGvBcxl" \t "_blank)

Ddpg1 **ICML 2014 DeepMind** **Deterministic Policy Gradient Algorithms**

DDPG2 CONTINUOUS CONTROL WITH DEEP REINFORCEMENT

Experience-driven Networking A Deep Reinforcement Learning based Approach

Learning to route

On Removing Routing Protocol from Future Wireless Networks10.1109@MWC.2017.1700244

The Deep Learning Vision for Heterogeneous

S. Agarwal, M. Kodialam and TV. Lakshman, Traffic engineering in software defined networks, *IEEE INFOCOM’2013*, pp. 2211–2219.

A Deep-Reinforcement Learning Approach for Software-Defined Networking Routing Optimization



RL-DQN

SDN

Load-balance traffic-engineer flow-control