基于强化学习的多阶段网络分组路由方法

高远翔[1]，罗 龙[1]，孙 罡[1*]

(1. 电子科技大学光纤传感与通信教育部重点实验室 成都 611731)

【摘要】多阶段网络被广泛应用于机器学习集群，由于多阶段网络中可用路径多，分组的路由是一个组合优化难题。现有基于启发式的路由算法由于缺乏性能保证，严重影响分组传输延迟。提出了基于强化学习的多阶段网络分组路由方法，使用一个新颖的策略迭代算法，通过学习的方式计算出最佳路由策略。算法通过在策略评估步骤中使用价值函数的最大似然估计器，克服了强化学习方法中蒙泰卡罗 (MC) 或时间差分 (TD) 价值估计器样本效率低的问题。为了应对组合优化时计算复杂度高的问题，算法在策略改进步骤中将组合动作空间上的优化分解为各组成动作的序列优化，以提高求解效率。基于 NS-3 网络模拟器的仿真实验结果表明，相较于现有最优的启发式路由策略，该算法学习到的路由策略降低了 13.9% 的平均分组延迟。

关 键 词 集群网络； 策略迭代； 分组路由； 强化学习

中图分类号 TN915 **文献标志码** A doi:10.12178/1001-0548.2021260

Packet Routing Method for Multi-Stage Networks Based on Reinforcement Learning

GAO Yuanxiang[1], LUO Long[1], and SUN Gang[1*]

(1. Key Lab of Optical Fiber Sensing and Communications, University of Electronic Science and Technology of China Chengdu 611731)

Abstract Multi-stage networks are widely used in machine learning clusters. Due to the large number of available paths in a multi-stage network, packet routing is a combinatorial optimization problem. Existing routing algorithms based on heuristics lack performance guarantee, which seriously affects the packet transmission delay. This paper proposes a packet routing method based on reinforcement learning for multi-stage networks, using a novel policy iteration algorithm to compute an optimal routing policy by learning. In the policy evaluation step, this algorithm uses the maximum likelihood estimator of the value function, which overcomes the low sample efficiency problem of Monte Carlo (MC) or Temporal-Difference (TD) value function estimators in reinforcement learning. To deal with the high computational complexity of the combinatorial optimization problem in the policy improvement step, this algorithm decomposes the optimization over a combinatorial action space into a sequential optimization of each action. Experiments based on NS-3 network simulator show that the routing policy learnt by the algorithm reduces 13.9% of the average packet transmission delay compared to existing best routing heuristics.

Key words cluster network; policy iteration; packet routing; reinforcement learning

近年来，机器学习方法在包括图像识别[1]、机器翻译[2] 等多个领域得到了广泛应用。由于数据量及算法复杂度的增长，机器学习应用通常在一个由大量计算资源组成的集群系统上分布式运行。机器学习应用周期性地在集群网络中产生跨计算资源的数据分组，这些数据分组在网络中的传输延迟对机器学习应用的时间效率具有关键影响。

机器学习集群常用多阶段 Clos 网络[3-5]，其通过多个阶段的交换机将各个计算资源互联起来。多阶段网络在两个计算资源间提供了大量可供选择的路径，且网络中同时有大量分组需要路由决策，所以分组的路由是一个组合优化难题。

基于最短路径的拟静态路由算法[6]，如贝尔曼–福特算法、狄克斯特拉算法，无法跟随集群网络负载状态的迅速变化。而多阶段网络通常采用基于启发式的动态路由算法[3,7]，目前，广泛使用的是基于

收稿日期：2021－09－14；修回日期：2021－10－30

基金项目：国家重点研发计划 (2019YFB1802800)

作者简介：高远翔 (1991－), 男，博士生，主要从事大数据集群资源分配与调度、强化学习等方面的研究.

*通信作者：孙罡，E-mail：gangsun@uestc.edu.cn

“加入最短队列”策略[3]的启发式路由算法。该算法将分组转发到具有最少排队分组的下一跳交换机。这些启发式算法基于网络局部的负载信息进行路由决策，常导致网络全局负载不均衡，无法保证最小的平均分组传输延迟。

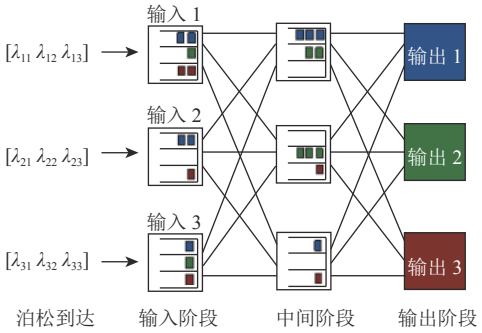
本文提出一种基于强化学习的分组路由算法，将多阶段网络的路由问题建模为一个马尔科夫决策过程[8](markov decision process, MDP)，这是该问题的首个 MDP 模型。为了求解该 MDP 的最佳路由策略，本文提出了最大似然策略迭代 (maximum likelihood policy iteration, MLPI) 算法。该算法在策略评估步骤中使用最大似然价值函数估计器，该价值函数估计器克服了现有强化学习方法[9-10]中蒙泰卡罗 (Monte Carlo, MC) 或时间差分 (temporal-difference, TD) 价值函数估计器样本效率低的问题。为了应对 MLPI 算法策略改进步骤中涉及的组合优化难题，本文提出了一个序列最小化的方法，通过将组合优化分解为一系列可求解的简单优化子问题来进行有效的策略改进。

基于 NS-3 网络模拟器的仿真实验结果表明，本文的 MLPI 算法找到的路由策略较“加入最短队列”启发式策略减少了 38.3% 的平均排队分组数目，同时减少了 17.6% 的平均分组延迟。此外，MLPI 算法的学习效率远高于基于蒙泰卡罗 (MC) 或时间差分 (TD) 价值函数估计器的强化学习算法。

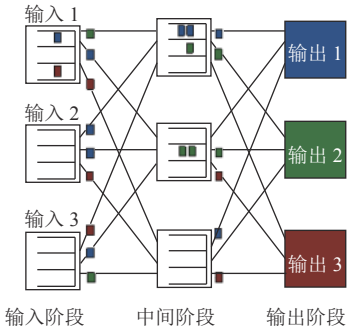
1　多阶段网络的分组路由

如图 1a 所示，在一个三阶段网络中，分组路由问题的模型是一个离散时间排队系统的控制问题。在一个特殊的时刻t，由计算资源产生的数据分组到达输入阶段的交换机。一个输入或输出交换机通常连接多个计算资源。为了简洁表达，连接到输入输出交换机上的计算资源没有在图中呈现。在第i个输入交换机，目的地为第j个输出交换机的到达分组数目服从一个到达率为λ_{ij}的泊松分布，且这些到达分组排在第i个输入交换机的第j个队列中，网络状态是该网络中各队列分组的数目。

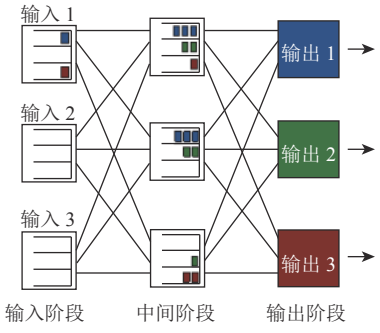
在每个时刻t，不同阶段交换机之间的每条链路负责将分组从上游交换机传输到一个下游交换机，具有一个单位的容量。假设使用先入先出排队准则，路由算法需要为每一个队列中的队首分组选择一个下游链路来传输它。如图 1b 所示，所有队首分组的路由选择可视作一个全局的路由动作。遵循这个路由动作，队首分组在选择的链路上传输。在下一个时刻$t+1$，如图 1c 所示，传输中的分组同时到达下游交换机的相应队列。所有分组到达下游交换机后，到达输入阶段交换机的新一轮分组遵循同样的泊松分布。然后类似的路由动作和分组传输重复进行。



a. 在时刻 t 泊松到达后的网络状态



b. 在时刻 t 的路由动作



c. 在时刻 t+1 泊松到达前的网络状态

图 1　三阶段网络分组路由的离散时间排队系统模型

2　MDP 模型

本节将多阶段网络分组路由问题建模为一个马尔科夫决策过程 MDP。该 MDP 由一个四元组$\mathbb{S},\mathbb{A},c,\boldsymbol{P}$指定，其中$\mathbb{S}$是状态空间，$\mathbb{A}$是动作空间，$c$是代价函数，$\boldsymbol{P}$是状态转移概率。该 MDP 的具体定义如下。

状态：该 MDP 在时刻t的状态表示为s_t，是一个 3 维矩阵，其元素表示为$n_{sij}^{(t)}$，表示第i个交换机上第j个队列在第s个阶段的分组数目。

动作：假设网络中有M个队首分组，该 MDP 在时刻t的动作是为每一个队首分组选择的链路所组成的集合$\{a_1,a_2,\cdots,a_M\}$。动作产生的顺序是从在最上游阶段的最低指标输入交换机上最低指标队列中的队首分组开始，逐渐轮询到同一个输入交换机上更高指标队列中的队首分组，然后轮询到同一个阶段中更高指标的输入交换机上的各队首分组，最终轮询到更下游交换机上的各队首分组。为第m个队首分组选择的链路a_m是没有被其他队首分组选择的空闲下游链路中的一个。当一个交换机上的某些队列没有队首分组时，为了充分利用链路，该交换机上其他队列中的非队首分组也可能在队首分组选择链路后获得一个链路分配。

代价：该 MDP 在时刻t的代价是在网络中排队分组的总数目，为$c(s_t)=\sum_{s,i,j} n_{sij}^{(t)}$。由 Little 定律[11]，网络中排队分组的平均总数目与分组在通过这个网络时的平均传输延迟成正比，减少网络中的平均排队分组总数能减小平均分组传输延迟。

状态转移方程：由于该问题的转移概率矩阵是无穷维的，且其表达式较繁琐，本文用等价的状态转移方程来表述该 MDP。非输入阶段交换机的状态转移由如下随机差分方程所给定：

$$n_{sij}^{(t+1)}=n_{sij}^{(t)}-d_{sij}^{(t)}+e_{sij}^{(t+1)}\quad s\neq 1\ \ \forall i,j \tag{1}$$

式中，$d_{sij}^{(t)}$为在时刻t离开第sij队列的分组数目；$e_{sij}^{(t+1)}$表示在时刻$t+1$进入第sij队列的分组数目。输入阶段交换机的状态转移由如下随机差分方程所给定：

$$n_{sij}^{(t+1)}=n_{sij}^{(t)}-d_{sij}^{(t)}+h_{ij}^{(t+1)}\quad s=1\ \ \forall i,j \tag{2}$$

式中，$h_{ij}^{(t+1)}$是在时刻$t+1$到达第i个输入交换机上第j个队列中的分组数目。给定一个路由策略π和一组泊松参数$\boldsymbol{\Lambda}$，状态转移概率$\boldsymbol{P}$可以从以上的状态转移方程中推导出来。最大似然策略迭代算法交替执行策略评估步骤和策略改进步骤来迭代地求解上述 MDP。

3 基于最大似然价值估计的策略评估

一个状态的价值定义为开始于状态s_t的一个足够大的窗口内未来状态的总折扣代价的期望值，如式 (3) 所示[10]：

$$V_\pi(s_t)=c(s_t)+\mathbb{E}_{\pi,\boldsymbol{\Lambda}}\left[\gamma c(S_{t+1})+\gamma^2 c(S_{t+2})+\cdots+\gamma^w c(S_{t+w})\right] \tag{3}$$

式中，S_{t+w}表示时刻$t+w$状态的随机变量，w是窗口大小。窗口大小通常设为能使得更远未来状态的代价可以忽略不计的值，如在$\gamma=0.99$时设为 500。$\mathbb{E}_{\pi,\boldsymbol{\Lambda}}[\bullet]$是在策略$\pi$和泊松参数$\boldsymbol{\Lambda}$下，相对于$S_{t+w}$的概率分布的期望。由于到达参数通常是未知的，作为其函数，上述价值函数需要从采样的状态样本轨迹中估计得到。

现有强化学习算法使用蒙泰卡罗 (MC) 或时间差分 (TD) 价值函数估计器[9-10]，但 MC 和 TD 价值估计样本效率低[9]。本文使用价值函数的最大似然估计器，推导如下。

给定一个策略π，式 (3) 中价值函数是未知泊松到达参数$\boldsymbol{\Lambda}$的函数。给定一个时间长度为T的样本轨迹，参数$\{\lambda_{ij}\}$的最大似然估计 (maximum likelihood estimate, MLE) 由如下的平均到达率给出[12]：

$$\hat{\lambda}_{ij}=\frac{1}{T}\sum_{t=1}^{T} h_{ij}^{(t)}\quad \forall i,j \tag{4}$$

根据点估计理论[12]，未知参数函数$f(\boldsymbol{\Lambda})$的最大似然估计是将该函数作用到未知参数的最大似然估计$f(\hat{\boldsymbol{\Lambda}})$。利用最大似然估计的这种函数不变性，价值函数的最大似然估计，表示为$\hat{V}_\pi^{\mathrm{ML}}(s_t)$，是把$\hat{\boldsymbol{\Lambda}}=\{\hat{\lambda}_{ij}\}$代入价值函数的定义中，如下所示：

$$\hat{V}_\pi^{\mathrm{ML}}(s_t)=c(s_t)+\mathbb{E}_{\pi,\hat{\boldsymbol{\Lambda}}}\left[\gamma c(S_{t+1})+\gamma^2 c(S_{t+1})+\cdots+\gamma^w c(S_{t+w})\right] \tag{5}$$

式中，$\hat{V}_\pi^{\mathrm{ML}}(s_t)$是相对于充分统计量$\hat{\boldsymbol{\Lambda}}$的一个条件期望，根据 Rao-Blackwell 定理[12]，$\hat{V}_\pi^{\mathrm{ML}}(s_t)$具有最小均方误差性质。所以，其能取得比 MC 和 TD 价值估计器更高的样本效率。式 (5) 中的期望是相对于策略π和估计的到达参数$\hat{\boldsymbol{\Lambda}}$下采样的状态取期望。然而计算最大似然价值估计需要对所有可能的状态轨迹取期望，对于该 MDP 庞大的状态空间，理论上准确地计算上述最大似然估计是不实际的。本文使用一种类似基于模型的强化学习[13-14] 的方法来近似计算最大似然价值估计。

首先从一条状态样本轨迹中估计到达参数$\hat{\boldsymbol{\Lambda}}$，这相当于建立了一个估计的状态转移模型。在这个估计的模型下，每个时刻到达网络输入阶段交换机的分组数目服从参数为$\hat{\boldsymbol{\Lambda}}$的泊松分布。因此，可以在不与网络交互的情况下，通过模拟该 MDP 来产生大量仿真的状态转移。然后，用这些模拟状态转移下的总折扣代价值作为对最大似然价值函数估计的近似，表示为$\tilde{V}_\pi^{\mathrm{ML}}(s_t)$，如下所示：

$$\tilde{V}_{\pi}^{\mathrm{ML}}(s_t)=c(s_t)+\gamma c(s_{t+1})+\cdots+\gamma^{w}c(s_{t+w}) \quad (6)$$

式中，样本轨迹$\{s_t,s_{t+1},\cdots,s_{t+w}\}$是在估计的到达参数和策略$\pi$下模拟的状态转移。

4　最大似然策略迭代

4.1　基于序列最小化的策略改进

在计算出价值函数估计$\tilde{V}_{\pi}^{\mathrm{ML}}(s_t)$后，一般的策略迭代方法[10]通过求解如下优化问题来得到一个更好的策略：

$$\pi'(s)=\underset{\{a_1,a_2,\cdots,a_M\}}{\operatorname{argmin}}\sum_{s'}p(s'|s,a_1,a_2,\cdots,a_M)\tilde{V}_{\pi}^{\mathrm{ML}}(s') \quad (7)$$

式中，s'是在状态s采取动作序列$\{a_1,a_2,\cdots,a_M\}$所导致的下一个状态。式 (7) 中的最小化问题是一个困难的组合优化问题，其搜索空间随网络中队首分组数目呈指数式增加。为了快速求解问题，本文使用一种在神经动态规划文献 [15] 中被称为动作空间复杂度与状态空间复杂度折衷的方法，通过引入一系列描述每个动作后果的人工状态使得策略改进步骤能序列地对每一个动作进行。

具体而言，本文通过求解如下最小化问题来计算相对于$\tilde{V}_{\pi}^{\mathrm{ML}}(s_t)$的最佳动作$a_1^*$：

$$a_1^*=\underset{a_1}{\operatorname{argmin}}\tilde{V}_{\pi}^{\mathrm{ML}}(s'(s,a_1)) \quad (8)$$

式中，$s'(s,a_1)$是在状态s采取动作a_1所导致的中间状态，即将第一个队首分组移动到动作a_1所选择的下游交换机后的那个中间状态。类似地，本文通过求解如下一系列最小化问题来计算第m个队首分组的最佳动作a_m^*：

$$a_m^*=\underset{a_m}{\operatorname{argmin}}\tilde{V}_{\pi}^{\mathrm{ML}}\left(s'\left(s,a_1^*,a_2^*,\cdots,a_{m-1}^*,a_m\right)\right)\ \forall m \quad (9)$$

式中，$s'(s,a_1^*,a_2^*,\cdots,a_{m-1}^*,a_m)$是在状态$s$对前$m-1$个队首分组采取最佳动作$a_1^*,a_2^*,\cdots,a_{m-1}^*$，并且对第$m$个队首分组采取动作$a_m$后所对应的中间状态。

4.2　最大似然策略迭代算法

如算法 1 所示，该 MLPI 算法首先初始化一个近似价值函数估计的卷积神经网络 (convolutional neural network, CNN)，V_{θ_0}作为初始的价值函数估计。初始路由策略π_0是相对于V_{θ_0}的ε-贪婪策略。具体地，在每步序列最小化时以概率为$1-\varepsilon$选取最佳动作而以概率ε随机选取链路。在第n次策略迭代时，该算法观察网络进行T步状态转移，且累加到达输入交换机各队列的分组总数。然后，该算法计算泊松参数的最大似然估计$\hat{\boldsymbol{\Lambda}}$。

接下来，MLPI 算法跟随策略π_n执行$K(K\gg T)$步模拟的状态转移。模拟状态转移中经历的状态作为输入集$S=\{s_l,l=1,2,\cdots,L\}$，对应的折扣样本总代价$C=\{c(s_l)+\gamma^1c(s_{l+1})+\cdots+\gamma^Wc(s_{l+W})\}$作为输出集构成了一个庞大的数据集$D_n$。该数据集用来训练$V_{\theta_n}$几个来回 (epochs)，得到的价值网络$V_{\theta_{n+1}}$是最大似然价值函数的近似。之后，在下一次迭代中遇到的每个状态，该算法通过ε-贪婪地相对于$V_{\theta_{n+1}}$求解式 (9) 中的序列最小化来产生新策略π_{n+1}。

算法 1：最大似然策略迭代算法 (MLPI)

设价值网络V_{θ_0}、相对于V_{θ_0}的ε-贪婪策略π_0

设分组到达总数$h_{ij}=0$

for(n = 0, 1, 2, $\cdots$)

　观察网络进行T步状态转移

　统计到达输入阶段交换机的分组总数$\sum_{t=1}^{T}h_{ij}^{(t)}$

　更新$h_{ij}=h_{ij}+\sum_{t=1}^{T}h_{ij}^{(t)},\forall i,j$

　计算$\hat{\lambda}_{ij}=h_{ij}/[(n+1)T],\forall i,j$

　执行$K(K\gg T)$步模拟的状态转移

　存储模拟路由过程产生的数据集D_n

　使用D_n训练V_{θ_n}几个来回得到$V_{\theta_{n+1}}$

　对$V_{\theta_{n+1}}$网络ε-贪婪地路由分组，得到π_{n+1}

end for

5　实　验

5.1　测试网络和路由代理

NS-3 网络模拟器是一个广泛使用的分组级别的离散事件仿真器[16]。本文基于 NS-3 搭建了一个多阶段的网络测试环境，参考强化学习中环境–代理 (agent) 的交互框架[10]搭建了一个网络路由代理。该网络路由代理使用由 MLPI 算法训练完成的价值网络产生最佳的路由动作序列。

具体来说，该测试网络是一个按时隙产生控制和进行传输的网络，在每个时隙t，该测试网络中各交换机将各队列的分组数目上报给网络路由代理。网络路由代理得到该时刻的网络状态，作为产生路由决策的初始状态。从该状态开始，路由代理相对于训练好的价值网络逐步求解序列最小化问题 (无需ε-探索)，来得到该状态下所有队首分组的最佳路由向量$\{a_1^*,a_2^*,\cdots,a_M^*\}$。之后，路由代理将该最佳路由动作序列下达给各交换机，各交换机按照指定的路由动作发送各队列的队首分组。当发送的

各分组到达下游交换机后，网络进入下一个时隙$t+\Delta t$并重复上述交互过程。

5.2 实验设定

本文在一个每阶段包含 16 个或 20 个交换机的三阶段网络中测试 MLPI 算法。在实验前，分组到达率$\{\lambda_{ij}\}$由独立同分布的 0～1 之间的均匀分布产生，网络负载ρ定义为总到达率除以单阶段的总链路容量。

卷积神经网络 (CNN) 的输入是表示网络队列状态的 3 维矩阵，其元素为$\{n_{sij}^{(t)}\}$。对每阶段有 20 个交换机的网络，输入通过 6 个连续的卷积层和 2 个全连接层处理之后，输出该状态的价值估计值。对每阶段有 16 个交换机的网络，最后两个卷积层被移除。该训练损失度量是均方误差，CNN 的超参数如表 1 所示，MLPI 算法的超参数如表 2 所示。

表 1 卷积神经网络的超参数

网络层	核数目	核尺寸	激活函数
卷积层1	64	7×7	relu
卷积层2	64	5×5	relu
卷积层3～6	64	3×3	relu
全连接层1	128	—	relu
全连接层2	1	—	relu

表 2 MLPI 算法的超参数

参数名	参数值
L2正则化系数	0.000 1
优化器	Adam[17]
学习速率	0.000 3
批大小	256
窗口大小	500
折扣因子	0.99
探索系数	0.4

5.3 对比方案

将 MLPI 算法与典型及现有最优的路由启发式算法进行对比。

1) 随机路由 (Rand) 算法：交换机随机选择一个空闲链路来传输队首分组。

2) 加入最短队列 (join-the-shortest-queue, JSQ)[3,6] 算法：对于交换机上第j个队列的队首分组，交换机在所有空闲链路里选择其下游交换机上第j个队列最短的链路来传输该分组。

3) Power-of-two-choices(Po2)[7,18] 算法：对于交换机上第j个队列的队首分组，交换机首先随机选取两个空闲链路作为候选链路，再在候选链路中选择其下游交换机上第j个队列更短的链路来传输该分组。

4) 基于蒙泰卡罗价值估计的强化学习 (MC)[10] 算法：该方法遵循相对于价值网络ε-贪婪的策略来与测试网络进行交互，在交互过程中产生的状态转移样本集合被用来在线训练价值网络。对每个状态，价值网络的训练目标为在一个窗口范围内未来状态的总折扣代价值。在产生一批训练示例后，该算法相对于价值网络参数执行一步随机梯度下降。

5) 基于n-步 TD 价值估计的强化学习 (TD)[10] 算法：类似于 MC，对每个状态，价值网络的训练目标是在一个大小为$n(n=100)$的窗口内的未来状态的总折扣代价，再加上第$n+1$个未来状态在γ^{n+1}折扣后的价值估计值。在产生一批训练示例后，该算法执行一步随机梯度下降。

5.4 实验结果

在实验中，MLPI 算法执行一系列的策略迭代步骤直到策略不再改进。在第n次策略迭代时，MLPI 算法观察测试网络，进行 20 个时隙的状态转移来更新对泊松参数的估计。然后，MLPI 算法执行 3 200 步模拟的状态转移，这个过程中收集到的训练示例 (接近 1 000 000 个，包含中间状态) 用来训练价值网络 10 个来回。训练完成后，从一个空的网络状态开始，路由代理使用现有价值网络对应的路由策略来控制测试网络 200 个时隙，且将平均排队分组总数和平均分组延迟记录下来，作为现有策略π_n的性能度量。上述迭代持续进行，直到策略的性能不再改进。

5.4.1 平均排队分组总数

图 2 的结果显示，对于每阶段 16 个交换机的网络，在 6 次最大似然策略迭代之后，MLPI 找到的路由策略的平均排队分组总数达到最低点，其相对于 JSQ 和 Po2 算法分别减少了约 26.6% 和 21.1% 的平均分组总数。图 3 的结果显示，对于每阶段 20 个交换机的网络，在 7 次最大似然策略迭代后，MLPI 找到的路由策略相对于 JSQ 和 Po2 分别减少了约 20.7% 和 17.2% 的平均排队分组总数。然而，

MC 或 TD 算法的平均排队分组总数始终保持在较高的程度，几乎没有从经验中学习的迹象。

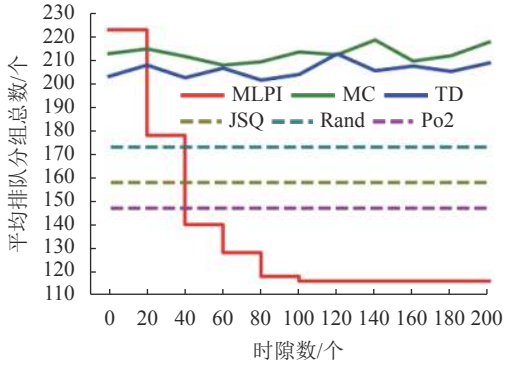


图 2　每阶段 16 个交换机时的平均排队分组总数

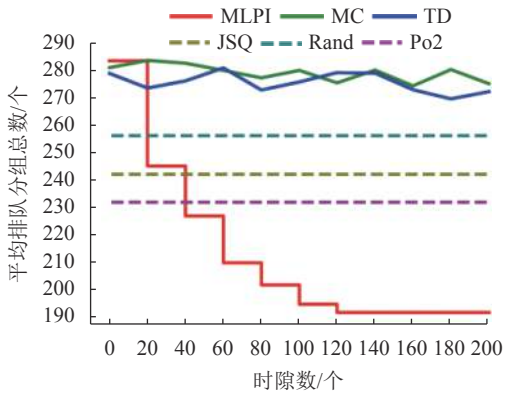


图 3　每阶段 20 个交换机时的平均排队分组总数

5.4.2　平均分组延迟

如图 4 所示，经过 8 次最大似然策略迭代，MLPI 收敛到的路由策略相对于 JSQ 和 Po2 分别减少约 17.6% 和 13.9% 的平均分组延迟。如图 5 所示，经过 8 次最大似然策略迭代，MLPI 算法收敛到的路由策略相对于 JSQ 和 Po2 分别减少了约 13.0% 和 10.3% 的平均分组延迟。可以观察到平均分组延迟的下降趋势与平均排队分组总数的下降趋势一致。

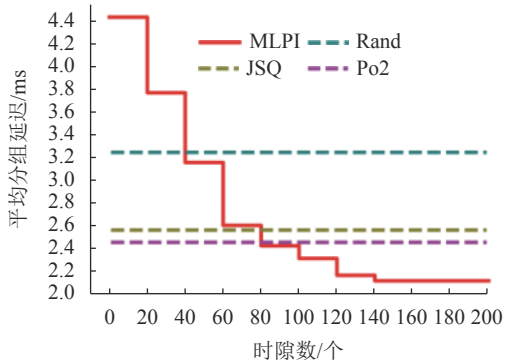


图 4　每阶段 16 个交换机时的平均分组延迟

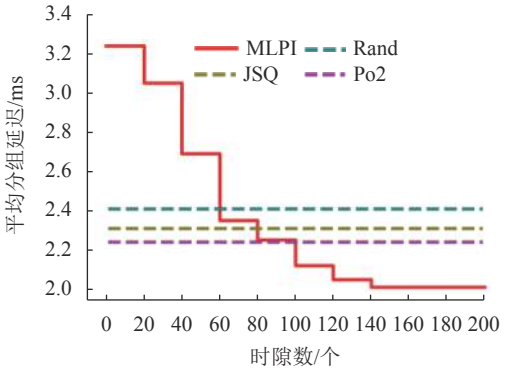


图 5　每阶段 20 个交换机时的平均分组延迟

5.4.3　网络负载的影响

图 6 展示了 MLPI 算法在各负载条件下收敛到的路由策略的平均排队分组总数。不论负载条件如何变化，MLPI 算法找到的路由策略的平均排队分组总数都显著低于启发式路由策略。在越重的负载条件下，MLPI 算法找到的路由策略相对于其他对比方案的平均排队分组总数减少量越大。当网络负载为 0.8 时，MLPI 算法相对于 JSQ 和 Po2 算法的平均排队分组总数减少量分别约为 38.3% 和 28.9%。

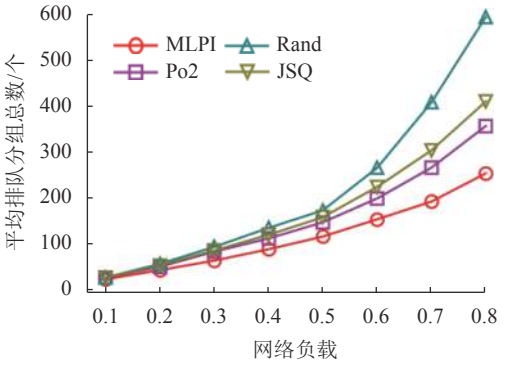


图 6　不同负载条件下的平均排队分组总数

5.4.4　负载均衡的效果

在对不同路由算法的测试运行中，记录在每个时隙泊松到达事件之前的网络状态，且对所收集的状态取平均来得到各路由算法下总体的排队行为。如图 7 所示，每一个热图代表一个 16×16 矩阵，其第ij个元素表示在某个阶段中第i个交换机上的第j个队列的平均排队分组数。

图 7 的结果显示，在第一个阶段，在记录状态的时刻各队列中没有分组累积。在第二个阶段，MC 算法找到的路由策略导致了一些拥塞的队列和不均衡的负载分布。在第三个阶段，Po2 路由算法导致了显著的负载不均衡，这是因为其将队首分组路由到一些拥塞队列中而让其他队列保持空载。这种对

链路资源的欠利用降低了网络吞吐量，导致分组滞留在网络中。对比之下，MLPI 算法通过最大似然策略迭代学会了达到近乎理想的负载均衡状态。

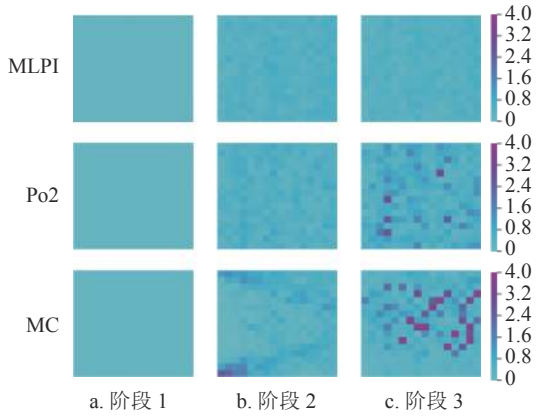


图 7 不同路由策略下各队列的平均排队分组数

强化学习算法如 MC 或 TD，需要与实际网络进行大量的交互并收集大量的试错数据，这会在训练过程中显著损害网络的延迟性能。MLPI 算法通过模拟的状态转移来学习路由策略，其训练过程不会对实际网络的正常运行产生干扰。由于集群网络需要不间断地为用户提供低延迟的传输服务，MLPI 算法是一种更加实用的路由策略学习方法。

6 结 束 语

本文提出最大似然策略迭代 (MLPI) 算法来求解多阶段网络分组路由问题，MLPI 采用了高效的最大似然价值估计器来进行策略评估。为了有效地改进策略，MLPI 采用序列最小化的方法将复杂的组合优化问题分解为一系列简单的优化子问题进行高效求解。基于 NS-3 的实验证实相较于现有最优的启发式算法，MLPI 学习到的路由策略能将网络中的平均排队分组总数和平均分组延迟分别降低约 21.1% 和 13.9%。

参 考 文 献

[1] HE K M, ZHANG X Y, REN S Q, et al. Deep residual learning for image recognitions[C]//IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas: IEEE, 2016: 770-778.

[2] BAHDANAU D, KYUNGHYUN C, BENGIO Y. Neural machine translation by jointly learning to align and translate[C]//International Conference on Learning Representations. San Diego: ACM, 2015: 1-15.

[3] HOJABR R, MODARRESSI M, DANESHTALAB M, et al. Customizing Clos network-on-chip for neural networks[J]. IEEE Transactions on Computers, 2017, 66(11): 1865-1877.

[4] GEBARA N, GHOBADI M, COSTA P. In-network aggregation for shared machine learning clusters[C]// Proceedings of Machine Learning and Systems. San Jose: ACM, 2021: 1-16.

[5] SINGH A, ONG J, AGARWAL A, et al. Jupiter rising: A decade of Clos topologies and centralized control in Google's datacenter network[C]//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. London: ACM, 2015: 183-197.

[6] ZHENG W, CROWCROFT J. Analysis of shortest-path routing algorithms in a dynamic network environment[J]. ACM SIGCOMM Computer Communication Review, 1992, 22(2): 63-71.

[7] GHORBANI S, GODFREY B, GANJALI Y, et al. Micro load balancing in data centers with DRILL[C]//Proceedings of the ACM Workshop on Hot Topics in Networks. Philadelphia: ACM, 2015: 1-7.

[8] PUTERMAN M L. Markov decision process: Discrete stochastic dynamic programming[M]. New Jersey: John Wiley & Sons, 2014.

[9] PENEDONES H, RIQUELME C, VINCENT D. Adaptive temporal-difference learning for policy evaluation with per-state uncertainty estimates[C]//Advances in Neural Information Processing Systems. Vancouver: MIT, 2019: 1-22.

[10] SUTTON R S, BARTO A G. Reinforcement learning: An introduction[M]. Cambridge: MIT, 2018.

[11] LITTLE J. A proof for the queuing formula: L=λw[J]. Operations Research, 1961, 9(3): 383-387.

[12] CASELLA G, GRAYBILL F A, BOES D C. Statistical inference[M]. Pacific Grove: Duxbury, 2002.

[13] RAJESWARAN A, MORDATCH I, KUMAR V. A game theoretic framework for model-based reinforcement learning[C]//International Conference on Machine Learning. Vienna: JMLR, 2020: 7953-7963.

[14] JANNER M, FU J, ZHANG M, et al. When to trust your model: Model-based policy optimization[C]//Advances in Neural Information Processing Systems. Vancouver: MIT, 2019: 1-18.

[15] BERTSEKAS D P, TSITSIKLIS J N. Neuro-dynamic programming[M]. Belmont: Athena Scientific, 1996.

[16] ABHIJITH A. NS-3[EB/OL]. [2021-09-08]. http://www.nsnam.org/.

[17] KINGMA D P, BA J L. Adam: A method for stochastic optimization[C]//International Conference on Learning Representations. San Diego: ACM, 2015: 1-15.

[18] MITZENMACHER M. The power of two choices in randomized load balancing[J]. IEEE Transactions on Parallel and Distributed Systems, 2001, 12(10): 1094-1104.

编 辑 刘飞阳