专题：智算网络

基于深度强化学习的算网协同动态路由调度算法

越奇强[1]，田乐[1,2,3]，魏帅[1]，胡宇翔[1,2,3]，冯旭[1]，董永吉[1,2,3]，陈博[1,2,3]
（1. 信息工程大学信息技术研究所，河南 郑州 450002；
2. 先进通信网全国重点实验室，河南 郑州 450002；
3. 网络空间安全教育部重点实验室，河南 郑州 450002）

摘　要：针对算力网络中算网资源协同不足、任务需求适配性差的问题，将算力路由问题建模为序列决策问题，提出了基于深度强化学习的算网协同动态路由调度算法。该算法借鉴混合专家模型思想，针对时延敏感型、普通型以及计算密集型3类典型场景，设计了基于编码器–解码器结构的差异化专家网络进行专项优化，并通过动作屏蔽机制约束路由选择空间，实现高效的逐跳决策，输出包含最优计算节点的路径。仿真实验结果表明，相较于其他路由调度算法，该算法在服务成功率上提升约17%，降低了端到端时延，优化了节点间的负载均衡度，展现出良好的网络拓扑适应性，能够有效满足多样化计算任务的差异化需求。

关键词：算力路由；算网融合；多场景优化；序列决策；深度强化学习
中图分类号：TP393
文献标志码：A
doi: 10.11959/j.issn.1000−0801.2025171

Computing-network collaborative dynamic routing and scheduling algorithm based on deep reinforcement learning

YUE Qiqiang[1], TIAN Le[1,2,3], WEI Shuai[1], HU Yuxiang[1,2,3],
FENG Xu[1], DONG Yongji[1,2,3], CHEN Bo[1,2,3]
1. Information Engineering University, Zhengzhou 450002, China
2. National Key Laboratory of Advanced Communication Networks, Zhengzhou 450002, China
3. Key Laboratory of Cyberspace Security, Ministry of Education, Zhengzhou 450002, China

Abstract: To address the issues of insufficient collaboration among computing resources and poor adaptability to task requirements in computing power networks, the computing power routing problem was modeled as a sequential decision problem. A deep reinforcement learning-based computing-aware routing algorithm was proposed for dynamic routing scheduling of computing network collaboration. The idea of hybrid expert models was drawn on and a differentiated expert network was designed based on an encoder-decoder structure for specialized optimization in three typi-

收稿日期：2025−03−31；修回日期：2025−07−06
通信作者：田乐，xxgctianle@163.com
基金项目：国家重点研发计划项目（No.2024YFB2906704）；河南省重大专项课题项目（No.22110021090003）
Foundation Items: The National Key Research and Development Program of China (No.2024YFB2906704), The Major Special Projects in Henan Province (No.22110021090003)

cal scenarios: delay-sensitive, ordinary, and computationally intensive. The routing selection space was constrained through an action masking mechanism to achieve efficient hop-by-hop decision-making and output a path containing the optimal computing node. The simulation experiment results show that compared with other routing scheduling algorithms, the proposed algorithm improves service success rate by about 17%, reduces end-to-end latency, optimizes load balancing between nodes, demonstrates good network topology adaptability, and can effectively meet the differentiated needs of diverse computing tasks.

Key words: computing-aware routing, computing-network integration, multi-scenario optimization, sequential decision-making, deep reinforcement learning

0 引言

随着信息技术的快速发展，特别是大数据、人工智能、云计算和边缘计算的广泛应用，各行业对算力的需求呈现出高度动态化、复杂化的趋势，为了应对这些日益增长的需求，算力网络应运而生。算力网络[1]是指将分布在不同地域和节点的计算资源通过网络连接起来，实现计算、存储、通信等服务的协同调度和资源共享的新型网络。

算力网络应用场景如图1所示，它由客户端、通信链路、边缘节点、云节点和算力路由节点构成。边缘节点作为基于硬件基础设施的虚拟化边缘应用平台，能提供计算和存储资源以处理用户任务。云节点即中心节点，拥有充足的计算与存储资源，通常是部署在离用户较远位置的大型服务器集群。

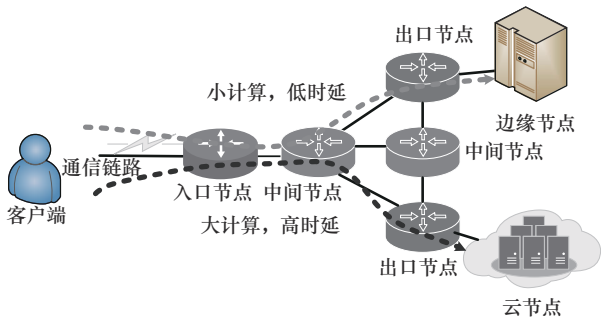


图1 算力网络应用场景

在算力网络中，算力路由[2]扮演着至关重要的角色，它通过将算力信息引入路由域，实时感知业务特征、算力分布及网络态势，动态生成“最优传输路径+适配算力节点”的联合调度方案，将计算任务沿最优传输路径调度至目标服务节点，实现分布式计算资源的高效协同[3]。用户可通过终端设备发起计算任务请求，经无线接入点将请求传输至基站，再由算力路由入口节点接入算力路由中间节点。该中间节点负责路由控制和数据转发，之后计算任务经算力路由出口节点，被路由转发至不同的计算节点。若计算任务及时响应，会被转发到边缘节点处理；若需大规模计算处理，则通过算力路由节点调度至中心节点执行[4]。

算力路由可以通过智能调度与资源分配优化，提高计算任务的处理效率，但仍面临诸多挑战，特别是在应对动态变化的算网环境、计算任务类型的多样性以及不同用户的需求等问题时，现有路由调度算法往往只关注计算任务的时延，忽略了算网环境下节点间资源的负载均衡问题[5]。为了高效利用算力资源，本文针对3种不同场景下的计算任务，综合考虑算网资源状态以及用户的计算需求，在满足计算需求的前提下，对时延以及负载均衡2个性能指标进行联合优化，并针对不同计算任务的需求进行侧重优化，旨在满足多样化计算任务的差异化需求。本文主要贡献如下。

（1）针对3种不同场景下的计算任务需求，设计了一种基于深度强化学习的算网协同动态路由调度（deep reinforcement learning computing-aware routing，DRL-CAR）算法，该算法不仅能

够自适应地处理不同任务的资源需求，还能在实际运行中根据系统的实时状态动态调整调度策略，设计差异化权重策略，实现时延与负载均衡的联合优化。

（2）将算力路由问题定义为序列决策问题，其中输入序列是由计算任务特征、算网资源组成的特征矩阵，输出序列是路由决策的序列，借鉴混合专家模型思想，设计了基于编码器-解码器结构的差异化专家网络进行专项优化，并通过动作屏蔽机制实现逐跳路由决策，最终输出包含最优计算节点的路径。

（3）通过在 GEANT、Abilene、Switch 3 个真实世界网络拓扑上进行模拟测试来评估效率，并与贪婪策略及随机策略 2 个启发式路由调度算法以及深度 Q 网络（deep Q-network，DQN）强化学习算法进行对比试验验证，结果表明，本文所提算法在计算任务满足计算资源需求的基本条件下，实现了对时延以及负载均衡 2 个性能指标的优化，不同计算任务可以根据需求调度到合适的计算节点上进行计算处理，适应算网资源的动态变化，满足不同任务的计算需求。

1　相关工作

在算力网络[6]环境下，节点的计算资源分布在不同的地理位置，数据在不同节点间的传输会产生时延，从用户角度来看，多样化业务需求对时延的敏感性不同，从整个网络层次来看，多个任务会同时竞争有限的网络资源和计算资源，导致网络拥塞和资源争抢，从而增加数据传输和处理的时延[7]。与此同时，不同节点间的计算资源存在较大差异，在资源分配不均时会影响节点资源间的负载均衡[8]。因此算力网络下的算力路由问题主要以时延和负载均衡为优化目标。

在感知算网资源信息的路由方面，文献[9]提出的服务感知网络（service aware network，SAN）架构通过语义服务标识实现高效服务连接，结合算网需求感知动态获取资源需求，并利用两级两层路由进行联合编排，提供位置无关、资源协同的服务互联解决方案。文献[10]分析了包含算力业务的路由策略与实现机制，探索在东数西算场景下，面向业务和算力资源分布的路由策略，提出了一种基于业务属性与算力资源分布的混合式路由策略，将全国分成若干“算力区域”，在不同算力区域之间采用集中式路由决策，在算力区域内部采用分布式路由决策。该策略能够提供一种安全、灵活、高效的全局路由方法。

在时延优化方面，文献[11]创新性地提出了基于 Floyd 算法的算力感知路由模型，通过建立以通信时延为核心的目标函数，系统性地解决计算任务调度过程中的时延最小化问题。而从计算资源负载均衡方面考虑，文献[12]重构了大数据计算架构体系，在算力网络框架内设计了资源编排层，结合粒子群优化算法实现算力资源的智能封装与动态调度，有效提升了资源分配的均衡性。在同时考虑时延和负载均衡 2 个指标方面，文献[13]提出了一种动态路由控制策略来确定分布式计算能力网络中的路径和计算节点，通过引入计算度量参数实现路径与算力节点的联合优化，在实证研究中展现出多维性能指标的同步提升。但这些传统启发式算法适合处理那些规则较为清晰、环境相对稳定的资源调度问题，在复杂、动态变化的算力网络中，存在明显局限性，尤其是在面对大规模任务和时延敏感型任务时，往往难以提供全局最优解，且无法动态适应环境变化。

在此背景下，深度强化学习（deep reinforcement learning，DRL）作为一种先进的人工智能方法，凭借其能够在未知环境中通过与环境的互动来自主学习最优策略的特点，在资源分配、任务调度以及智能网络路由方面已经取得了很不错

的应用成果，成为解决算力路由以及资源调度问题的新思路[14-17]。当前，已有一些基于深度强化学习的方案，在算力路由调度方面取得了一定成果。例如，文献[18]构建了算网融合环境下的DRL调度模型，通过实时感知设备状态、算力容量及网络拓扑等信息，实现系统全局成本最优的智能决策。为了解决算力网络中计算任务的时延敏感性问题，文献[19]用基于深度强化学习方法的改进RBDQN算法优化门控，结合贪婪策略进行时延敏感型任务的联合优化，创新性地建立了多维度效用函数以平衡时延、能耗与用户满意度。文献[20]提出了基于多目标优化的强化学习–遗传混合算法，通过自适应参数调整机制实现路由规划的多目标协同优化，显著提升了算力网络的整体服务效能。

综上所述，传统方法虽在特定场景下取得局部最优解，但其基于预设规则的静态建模方法难以适应计算任务类型的动态异构性，缺乏对算力、网络、存储等多维资源耦合关系的全局建模能力。而新兴的DRL方案虽然通过环境交互机制实现了策略动态优化，但在多目标联合优化层面仍面临挑战。因此，如何构建具备动态环境适应性、多目标自优化能力及资源协同调度机制的智能算法体系，仍是算网融合场景下进行算力路由调度亟待突破的核心问题。

2 系统模型及问题描述

2.1 系统模型

算力网络系统框架如图2所示，本文设计的算力网络系统框架由算网管控中心和算网基础设施构成。算网管控中心由集中控制器和智能体两部分组成。集中控制器负责收集算网状态信息以及入口节点发送的计算任务请求信息，并将信息上传至智能体，智能体负责进行决策，并经集中控制器将决策下发到入口节点，入口节点实施源路由机制，在入口节点即确定计算任务完整的调度路径。

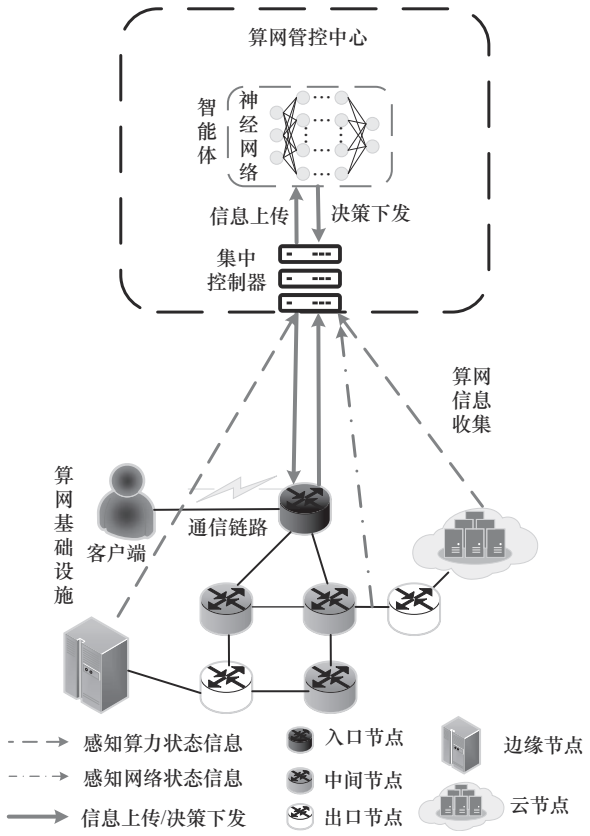


图2 算力网络系统框架

算网基础设施由网络节点及其连接链路构成，该网络可以用无向图$G=(V,E)$来表示，其中V是网络中所有节点的集合，可用$V=\{v_1,v_2,\cdots,v_n\}$来表示，包括算力路由节点v_r以及计算节点v_c。算力路由节点v_r无计算资源，分为面向用户接收计算任务请求的入口节点、负责控制转发的中间节点和调度任务到不同计算节点的出口节点。计算节点v_c包含边缘节点和中心节点，具备计算资源。

计算节点的总计算资源量用参数$S_v^{\text{comp}}=\{s_{v_1}^{\text{comp}},s_{v2}^{\text{comp}},\cdots,s_{v_n}^{\text{comp}}\}$来表示，剩余可用计算资源可以用集合$R_v^{\text{comp}}=\{r_{v_1}^{\text{comp}},r_{v2}^{\text{comp}},\cdots,r_{v_n}^{\text{comp}}\}$来表示，其中$v_j$代表第$j$个计算节点，计算节点处的处理能力可以用$v_j^{\text{com}}$来表示。对于计算节点间通信链路，引入链路集合$E$，其中链路$l\in E$拥有的总带

宽资源量由参数B_l表示，而数据在链路l上的传输时延则由参数D_l表示。本文涉及的主要变量含义见表1。

不同场景下的计算任务特性差异明显，其中自动驾驶及工业控制等场景对时延具有很高的敏感性，需降低时延以确保实时性；深度学习训练、大数据处理等大规模计算任务则对计算资源要求高，对时延要求不敏感。假设有n个计算业务，计算任务可以用集合$K=\{k_1,k_2,\cdots,k_n\}$来表示，每个计算任务请求可统一形式化为六元组$M=\{u_i,\text{type},C_k,B_k,\tau_k,T_k\}$。其中$u_i$表示用户$i$接入计算任务的入口节点，type代表不同的计算请求类型，C_k代表每种计算任务k所需的计算资源量，$C_k=\{C_1,C_2,\cdots,C_n\}$表示每个不同的计算任务$k$所需要的中央处理器（central processing unit，CPU）周期数，$B_k=\{B_1,B_2,\cdots,B_n\}$表示不同计算

表1　主要变量含义

变量	含义	变量	含义
V	网络节点集合	s_t^{network}	算力网络中的网络拓扑信息
v_r	算力路由节点	s_t^{node}	计算节点的计算资源量信息
v_c	计算节点	Path	算力路由路径
S_v^{comp}	计算节点的总计算资源量	$\boldsymbol{e_t}$	当前时刻输入信息
R_v^{comp}	剩余可用计算资源	$\boldsymbol{h_{t-1}}$	上一时刻的隐藏状态
E	链路集合	$\boldsymbol{h_t}$	传递到当前时刻的隐藏状态
B_l	链路l拥有的总带宽资源量	$\tilde{\boldsymbol{h}}_t$	候选隐藏状态
C_k	计算任务所需CPU周期数	r_t	重置门
D_l	数据在链路上的时延	z_t	更新门
K	计算任务集合	tanh	tanh函数
M	计算任务请求构成的六元组	$\boldsymbol{h}_t^{\text{enc}}$	编码器输出上下文向量
u_i	用户i接入的计算任务的入口节点	$\boldsymbol{h}_t^{\text{dec}}$	解码器输出隐藏向量
type	计算任务请求类型	w、b	可学习参数
B_k	计算任务所需要的带宽量	σ	sigmoid函数
τ	计算任务的最大容忍时延	v_{start}	起始节点
node	网络拓扑中节点的度	$\boldsymbol{h}_{t-1}^{\text{dec}}$	解码器$t-1$时刻的输出
T_k	计算任务请求到达的时间	v_{visited}	已访问节点
v_j^{com}	计算节点v_j处的处理能力	$\boldsymbol{o_t}$	解码器最后输出向量
T_{ij}^{trans}	网络传播时延	$\boldsymbol{M_t}$	掩码向量
$T_{v_j}^{\text{com}}$	计算时延	$\boldsymbol{o}_t^{\text{masked}}$	调整后的解码器输出
v	信号在链路介质中的传播速度	λ_1,λ_2	奖励函数权重系数
L_{ij}	网络链路长度	$P_{\text{bwd}},P_{\text{res}}$	惩罚项
$\phi_{v_j}^k$	计算任务k的调度策略	α,β_1,β_2	惩罚因子
ξ_L	不同时刻的链路L带宽占用情况	θ_v,θ_w	神经网络参数
δ_{v_j}	计算节点v_j的资源占用情况	$\boldsymbol{x_i}$	特征向量
Var	计算节点间的方差	$\boldsymbol{e_i}$	嵌入向量
ω	计算节点剩余计算资源的平均占用率	$\boldsymbol{E}$	嵌入矩阵
σ_1,σ_2	时延及负载均衡指标	ΔG	加权综合目标差距
$\pi_{\theta_v}(a\mid s_t)$	策略网络的函数表示	ΔT	整体时延差距
$b(s_t,\theta_w)$	价值网络的函数表示	ΔL	负载均衡差距
s_t^k	计算任务k的特征	μ_1,μ_2	实验权重配置

任务k所需要的带宽资源量，τ_k代表每种计算任务的最大容忍时延，即计算任务服务完成时间的上限，T_k代表计算任务k请求的到达时间，系统将按照计算任务请求的到达时间进行调度处理。

本文根据上述变量来建立计算任务模型。当用户发起计算任务请求时，需综合考虑计算任务类型、网络状态以及各计算节点的计算处理能力做出决策，决策使用变量$\phi_{v_j}^k$来表示，$\phi_{v_j}^k$表示计算任务k的调度策略，当$\phi_{v_j}^k=1$时，计算任务k被调度到计算节点v_j执行，当$\phi_{v_j}^k=0$时，计算任务k在算力路由节点进行转发，具体表达形式见式（1）：

$$\phi_{v_j}^k=\begin{cases}1, \text{计算任务}k\text{被路由调度到计算节点}v_j\text{执行}\\0, \quad \text{计算任务}k\text{在算力路由节点}v_i\text{进行转发}\end{cases} \tag{1}$$

在计算任务请求进行调度的过程中可能占用同一条链路带宽，因此，本文用ξ_L来表示在某时刻链路L带宽是否被占用，具体表达形式见式（2）：

$$\xi_L=\begin{cases}1, \quad \text{链路}L\text{带宽被占用}\\0, \text{链路}L\text{带宽未被占用}\end{cases} \tag{2}$$

在算网融合的网络环境中，不同类型计算节点的计算资源种类和数量不同，同一计算节点在不同时刻可能接收多个计算任务请求，本文用δ_{v_j}来表示计算节点v_j的计算资源是否被占用，并根据节点v_j剩余的计算资源数对其他计算任务请求进行资源分配，具体表达形式见式（3）：

$$\delta_{v_j}=\begin{cases}1, \quad \text{计算节点}v_j\text{当前时刻计算资源被占用}\\0, \quad \text{计算节点}v_j\text{当前时刻计算资源未被占用}\end{cases} \tag{3}$$

2.2 问题描述

在算网融合下的分布式计算系统中进行算力路由调度，考虑算网资源的动态变化，不同类型的计算任务根据任务需求动态选择最佳传输路径以及计算节点，以满足在算网资源约束的条件下达到最优的时延和负载均衡，这是本文所要优化的2个性能指标。

本文聚焦于算网资源的统一路由调度，因此需要分别考虑计算资源以及网络资源。对于算力网络中的计算资源，首先要求其能够满足计算任务k的需求，基于这一原则，给出关于计算资源方面的约束，见式（4）：

$$R_1: \sum_{k\in K} C_k\cdot\phi_{v_j}^k \le R_{v_j}^{\text{comp}}\cdot\delta_{v_j} \tag{4}$$

在算网资源进行算力路由过程中，网络状态主要由时延和带宽2个指标进行表征，从计算任务k所需带宽B_k不能超出链路剩余可使用带宽B_l这一出发点，给出关于带宽方面的约束，见式（5）：

$$R_2: \sum_{k\in K} B_k\cdot\xi_L \le B_l\cdot\xi_L \tag{5}$$

时延是算力网络的核心优化目标，在计算任务k进行路由调度时，产生的时延T由2部分组成，即网络传播时延T_{ij}^{trans}以及在计算节点v_j的计算时延$T_{v_j}^{\text{com}}$，见式（6）：

$$\sigma_1=T=T_{ij}^{\text{trans}}+T_{v_j}^{\text{com}} \tag{6}$$

T_{ij}^{trans}表示计算任务k通过网络链路传输到计算节点v_j的传播时延，它由网络链路长度L_{ij}以及信号在链路介质中的传播速率v决定，见式（7）：

$$T_{ij}^{\text{trans}}=L_{ij}/v \tag{7}$$

T_{ij}^{com}表示计算任务k在计算节点v_j处进行处理的时延，由计算任务所需的CPU周期数C_k与计算节点v_j处的计算能力v_j^{com}决定，见式（8）：

$$T_{ij}^{\text{com}}=C_k/v_j^{\text{com}} \tag{8}$$

在将计算任务路由调度到计算节点的过程中，由于各计算节点接收的计算任务所需的计算资源量存在差异，节点负载各不相同，进而出现部分节点过载、部分节点空闲的状态。为此，需要

对计算节点间的负载均衡进行评估，其中负载均衡可以用方差Var这一数学指标来进行衡量，方差Var越小，表明计算节点的剩余计算资源分布得越均匀，方差的计算过程见式（9）~式（10）：

$$\omega = \frac{\sum_{v \in V} \frac{S_{v_j}^{\text{comp}} - R_{v_j}^{\text{comp}}}{S_{v_j}^{\text{comp}}}}{|N|} \tag{9}$$

$$\sigma_2 = \text{Var} = \frac{\sum_{v \in V} \left(\frac{S_{v_j}^{\text{comp}} - R_{v_j}^{\text{comp}}}{S_{v_j}^{\text{comp}}} - \omega\right)^2}{|N|} \tag{10}$$

其中，ω表示计算节点剩余计算资源的平均占用率，N表示计算节点数量。

考虑多场景下的计算任务具有不同特性，本文在追求最小化时延和优化负载均衡的基础上，针对3种不同类型的计算任务有侧重地进行优化。对于时延敏感型任务，侧重于优化时延指标；对于计算密集型任务，侧重于考虑负载均衡指标，对于普通型任务，则在两个指标间实现均衡优化。基于上述分析，本文构建了在时延以及负载均衡约束下的数学模型，具体见式（11）：

$$\min\{\sigma_1, \sigma_2), \text{s.t.} R_1, R_2 \tag{11}$$

其中，σ_1、σ_2代表优化的2个性能指标。

3　基于深度强化学习的算力路由调度算法

针对第2.2节提出的问题，本文提出基于深度强化学习的DRL-CAR算法，该算法借鉴混合专家模型思想，设计专家神经网络输出动作策略，并通过强化学习训练该策略，从而实现对高维度和复杂问题的有效求解。针对动态计算任务的差异化需求，算法设计了具备环境自适应特征的智能路由调度策略，在满足端到端时延约束的同时，实现网络负载均衡优化目标。本节后续小节将详细介绍图2中所提出的算力网络系统框架中智能体的算力路由调度算法DRL-CAR及神经网络模型的设计。

3.1　DRL-CAR算法

DRL-CAR算法架构如图3所示，由算力网络系统构成的环境、智能体以及经验缓冲区组成。

环境：由云–边–端组成的算力网络基础设施，环境接收智能体的动作a_t，执行动作a_t并生成奖励r_t和下一状态s_{t+1}反馈给智能体。

智能体：DRL-CAR算法的决策核心，包含2个关键神经网络。

（1）策略网络$\pi_{\theta_v}(a|s)$

接收状态s_t作为输入，输出经决策后产生的动作a_t路径序列Path。策略网络借鉴混合专家模型思想，设计3组独立但结构相同的编码器–解码器作为专家神经网络，根据计算任务类别$\text{type} \in \{1, 2, 3\}$来分别处理时延敏感型、普通型及计算密集型3种计算任务的任务需求，其中每个专家子网络独立输出基于各自计算任务类型的算力路由路径结果。

（2）价值网络$b(s, \theta_w)$

价值网络$b(s, \theta_w)$采用与策略网络相同的编码器–解码器结构，其输出作为基线（baseline），用于计算优势函数，评估状态价值，降低策略梯度更新的方差，提升训练稳定性。

经验缓冲区：存储智能体与环境交互时的经验样本(s_t, a_t, r_t, s_{t+1})，每隔N个完整回合使用经验样本求进行平均更新，用于训练和更新网络参数θ_v和θ_w。

DRL-CAR算法执行流程如下。

（1）状态感知：智能体从环境中获取当前状态s_t，其中状态$s_t = (s_t^k, s_t^{\text{network}}, s_t^{\text{node}})$由计算任务特征及算网信息组成。$s_t^k$包括计算任务$k$的计算需求$C_k$、任务类型type、最大容忍时延$\tau_k$以及所需带宽$B_k$；$s_t^{\text{network}}$包括算力网络中的网络拓扑信息即节点的度node、可用网络带宽B_i；s_t^{node}包括计算节点的剩余可用计算资源信息$R_{v_j}^{\text{comp}}$。

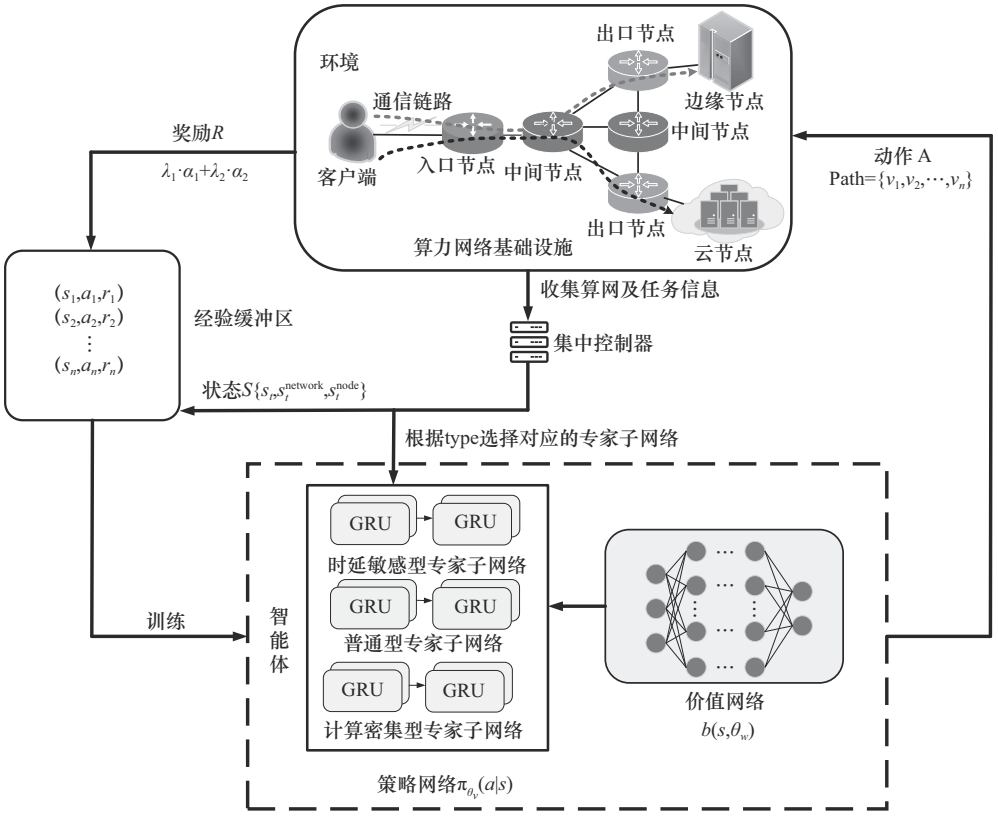


图3　DRL-CAR算法架构

（2）专家子网络选择：根据任务类型type ∈ {1, 2, 3}，策略网络$\pi_{\theta_v}(a|s)$激活对应的专家神经网络分支。

（3）路径决策：策略网络进行决策选取完整算力路由路径的过程主要分为2个阶段，在每一次迭代中，编码器处理输入特征矩阵，生成包含任务和资源信息的上下文向量$\boldsymbol{C}$，该向量完整地描述了计算任务的资源需求、时延约束等关键特性。

解码器基于编码器上下文$\boldsymbol{C}$和前序节点生成当前隐藏状态向量$\boldsymbol{h}_t^{\mathbf{dec}}$，经全连接层转换为节点得分向量$\boldsymbol{o}_t$，最后对向量$\boldsymbol{o}_t$应用动作屏蔽排除无效选项后，通过softmax生成概率分布并采样选择下一跳节点。迭代此过程构建完整路径，最终评估所有候选路径的时延与负载均衡指标，选取最优路径作为算力路由的最终路径。

具体流程见算法1。

算法1　基于策略网络$\pi_{\theta_v}(a|s)$的算力路由路径生成算法

输入　状态$s=(s_t^k, s_t^{\text{network}}, s_t^{\text{node}})$//特征矩阵（计算任务特征，算网资源状态）

输出　算力路由路径$\text{Path}=[v_0, v_1, \cdots, v_c]$//节点序列，$v_c$为计算节点

（1）初始化空路径$\text{Path}\leftarrow\varnothing$

（2）选择专家网络：根据任务类型type选择对应的编码器-解码器专家网络

（3）$\boldsymbol{C}=\text{Encoder}(s)$//编码器处理，输入特征矩阵$\boldsymbol{s}$，生成上下文向量$\boldsymbol{C}$

（4）$\boldsymbol{h}_0^{\mathbf{dec}}\leftarrow\boldsymbol{C}$, $v_0\leftarrow v_{\text{start}}$//解码器初始化，输入起始节点$v_0$

（5）for $t=1$to $T_{\max}$ do //$T_{\max}$为最大路径长度

（6）　$\boldsymbol{h}_t^{\mathbf{dec}}=\text{Decoder}(\boldsymbol{h}_{t-1}^{\mathbf{dec}}, v_{t-1})$//输入$v_{t-1}$和$\boldsymbol{h}_{t-1}^{\mathbf{dec}}$，通过解码器更新隐藏状态$\boldsymbol{h}_t^{\mathbf{dec}}$

（7）　$\boldsymbol{o_t}=W_o\cdot\boldsymbol{h_t^{dec}}+b_o$ //解码器输出向量$\boldsymbol{o_t}$

（8）　$\boldsymbol{M_t}=\text{masked}(v_{\text{visited}})$ //动作屏蔽，根据邻居节点中的已访问节点v_{visited}生成掩码$\boldsymbol{M_t}$

（9）　$\boldsymbol{o_t^{masked}}=\boldsymbol{o_t}+\log(\boldsymbol{M_t}+\boldsymbol{\varepsilon})(\varepsilon\to 0^+)$ //调整向量$\boldsymbol{o_t}$

（10）　$\pi_{\theta_v}(a|s)=\text{softmax}(\boldsymbol{o_t^{masked}})$ //计算概率分布

（11）　$a_t\sim\pi_{\theta_v}(a|s)$ //采样动作

（12）　$\text{Path}\leftarrow\text{Path}\cup\{v_t\}$ //更新路径

（13）　if $v_t\in v_c$ and $C_k\geqslant R_v^{\text{comp}}$ *and* $B_k\geqslant B_i$ //最后节点为计算节点且资源充足

（14）　　break

（15）　end if

（16）end for

（17）返回算力路由路径Path

值得注意的是，当某个计算任务到达计算节点进行计算时，如果该节点剩余计算资源能够满足该计算任务的需求，此任务将在该节点进行计算，这种情况下，任务可以顺利执行，计算节点将分配所需的资源来完成任务。如果该节点剩余计算资源不能满足新到达任务的需求，则服务失败。此时，该任务需要寻找其他计算节点进行计算，需要重新调度任务到其他有足够资源的节点。

（4）动作执行与奖励

策略网络$\pi_{\theta_v}(a|s)$输出包含最优算力路由路径的动作a_t到环境中，环境发生改变，基于时延及负载均衡指标计算奖励信号r_t并反馈给智能体，智能体通过策略梯度调整网络参数，迭代优化策略。

其中奖励函数R根据任务的路由决策结果进行评估，旨在最小化端到端时延和优化节点间的负载均衡。同时，对于无法满足任务资源需求的情况，施加惩罚项以确保资源约束的满足。在某个时刻t，将环境状态s_t智能体所采取的动作a_t作为一组状态-动作对记为$\mu=(s_t,a_t)$，是智能体与环境交互得到的轨迹，$\mu=\{(s_1,a_1),(s_2,a_2),\cdots,(s_t,a_t)\}$每组$\mu$都会产生一个奖励值，可以记为$R(\mu)$。

本文采用稀疏奖励的方式，基于回合制进行奖励值的计算，稀疏奖励指的是智能体只有在达成某个明确目标时才会获得奖励，其余时间没有奖励或惩罚。在该算法的奖励函数实现中，仅聚焦当前决策的即时收益，当整条路径选取结束后，才会基于完整路径给出奖励用于反馈优化策略。

因此，本文奖励函数设置为：

$$R(\mu)=\begin{cases}\lambda_1\cdot\sigma_1(\mu)+\lambda_2\cdot\sigma_2(\mu), & \text{满足需求}\\ \alpha\left(\beta_1\cdot P_{\text{bwd}}(\mu)+\beta_2\cdot P_{\text{res}}(\mu)\right), & \text{其他}\end{cases}\tag{12}$$

其中，σ_1、σ_2是式（10）中优化的2个性能指标，智能体执行状态–动作对$\mu=(s_t,a_t)$中的动作a_t生成完整路径后，通过环境反馈得到调度总时延T、计算节点v_j的总计算资源量$S_{v_j}^{\text{comp}}$、计算节点v_j的剩余可用计算资源$R_{v_j}^{\text{comp}}$，并使用式（6）和式（10）计算得到σ_1、σ_2的值。λ_1、λ_2为权重系数，且$\lambda_1+\lambda_2=1,\lambda_1,\lambda_2\in(0,1)$，用来平衡时延以及负载均衡的重要性，通过加权得到最后的奖励函数，其中λ_1、λ_2具体取值需要根据不同类型的计算任务进行调整，最终通过策略优化，为3类任务适配权重因子。P_{bwd}、P_{res}为惩罚项，α、β_1、β_2为惩罚因子。

当计算资源或带宽无法满足需求时，系统将施加相应的惩罚机制。在奖励函数设计中，惩罚项被赋予最高优先级，其惩罚因子α设置较大，以突出资源不足导致服务失败的严重性。在这种情况下，即使时延和负载均衡指标表现优异，系统仍会因资源不足而被判定为不可接受。为了消除时延和负载均衡指标单位不统一对训练过程造

成的影响，本文研究通过权重调整方法，使两者数值处于同一数量级范围，从而确保各优化目标的均衡性。

（5）经验存储与网络更新

将交互经验（s_t, a_t, r_t, s_{t+1}）存储经验缓冲区，每隔N个完整回合使用经验样本求进行平均更新，用于训练和更新网络参数。本文通过参数化的策略函数$\pi_{\theta_v}(a|s)$来进行估计，$\pi_{\theta_v}(a|s)$也对应着DRL-CAR算法中的神经网络模型，且$\pi_{\theta_v}(a|s)$与$\pi(a|s)$的关系为：

$$\pi_{\theta_v}(a|s) \approx \pi(a|s) \tag{13}$$

具体而言，训练后的神经网络策略$\pi_{\theta_v}(a|s)$能够逼近最优策略$\pi(a|s)$，$\pi_{\theta_v}(a|s)$表示神经网络参数为θ_v时在状态s下选择动作a的概率分布，针对本文涉及的3类计算任务，设置$v=3$种不同的最优路由策略。通过梯度下降方法迭代优化参数θ_v，使$\pi_{\theta_v}(a|s)$逐渐收敛至最优策略π^*。此外，为提高训练稳定性，算法引入了基线机制，以降低学习过程中的方差，提升训练过程的稳定性。具体实现细节详见算法2。

算法2 基于策略梯度的算力路由调度训练算法

输入 策略神经网络$\pi_{\theta_v}(a|s)$，价值神经网络$b(s, \theta_w)$，更新参数步长N

输出 优化过的神经网络参数θ_v和θ_w

（1）随机策略网络及价值网络参数θ_v和θ_w

（2）初始化经验缓冲区为一个空列表D

（3）**for** $t=1,2,3,\cdots,$ **do**

（4）收集当前环境状态为状态s_t

（5）根据$\pi_{\theta_v}(a|s)$采样动作a_t

（6）执行动作a_t，获得奖励r_t和下一个状态s_{t+1}

（7）保存经验样本(s_t, a_t, r_t, s_{t+1})到D中

（8） **if** $t \bmod N == 0$

（9） 从经验缓冲区D内随机采样一个批量的经验样本

（10） //计算价值网络的损失L_w和梯度

（11） $R = r_t + V(s_{t+1}; \theta_w)$

（12） $L_w = (R - V(s_t; \theta_w))^2$//MSE损失函数

（13） $\nabla_{\theta_w} L_w = 2(V(s_t; \theta_w) - R) * \nabla_{\theta_w} V(s_t; \theta_w)$

（14） //更新价值网络参数

（15） $\theta_w \leftarrow \theta_w - \alpha_w * \nabla_{\theta_w} L_w$

（16） //计算策略网络的损失L_v和梯度

（17） advantages $= R - V(s_t; \theta)$

（18） $L_v(\theta_v) = -\log(\pi_{\theta_v}(a|s_t)) *$advantages

（19） $\nabla_{\theta_v} L_v = -\nabla_{\theta_v} \log(\pi_{\theta_v}(a|s_t)) *$advantages //更新策略网络参数

（20） $\theta_v \leftarrow \theta_v - \alpha_v * \nabla_{\theta_v} L_v$

（21） $D \leftarrow$ 空列表

（22） end if

（23） end for

下面对算法2中的伪代码参数更新计算式进行解释，首先初始化策略网络$\pi_{\theta_v}(a|s)$及价值网络$b(s, \theta_w)$参数θ_v和θ_w，价值神经网络$b(s, \theta_w)$用于估计在状态s下累计奖励的期望值，为策略梯度计算提供基线，算法2中伪代码的（10）~（15）行是价值网络的损失L_w和梯度计算过程和价值网络参数θ_w更新过程。策略神经网络$\pi_{\theta_v}(a|s)$用于预测在状态s下选择动作a的概率分布，算法中伪代码的第（16）~（20）行是策略网络的损失L_v和梯度计算过程和策略网络参数θ_v更新过程。

在本文提出的DRL-CAR算法结构中，神经网络训练周期长度指神经网络模型在进行一次完整的参数更新过程中所花费的时间，当训练周期短于任务到达间隔时，模型在每次任务到来前都能完成更新，可基于最新策略做出决策。当训练周期长于任务到达间隔时，基于最近一次更新的策略做出决策。

3.2 神经网络模型设计

针对第3.1节DRL-CAR算法中提到的智能体

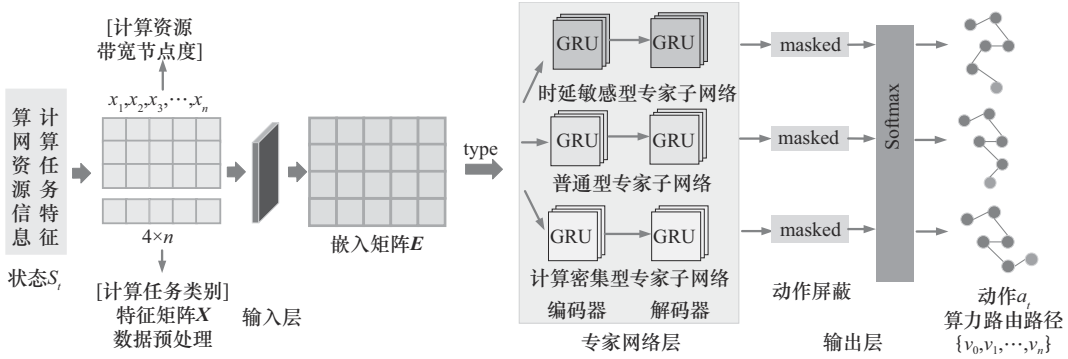


图4　神经网络模型设计

策略神经网络$\pi_{\theta_v}(a|s)$，本文设计了一个3层结构的神经网络模型，该模型由输入层、专家网络层和输出层组成。神经网络模型设计如图4所示。

数据预处理：算网资源信息包括计算节点的计算资源数量、网络链路的带宽状况以及节点的度信息，计算任务特征包括计算任务类型。算网资源信息和计算任务特征经处理之后分别转换成$3\times n$和$1\times n$的矩阵，数据预处理主要负责将其拼接成一个$4\times n$的特征矩阵$\boldsymbol{X}=\begin{bmatrix}\text{type}\\ R_{v_j}^{\text{comp}}\\ B_i\\ \text{node}\end{bmatrix}$，即$\boldsymbol{X}\in\mathbb{R}^{4\times n}$，其中$n$为网络中的节点数量，矩阵的每一列对应一个节点$v_i$的4维特征向量$\boldsymbol{x_i}$。

输入层：主要负责对数据预处理后的特征矩阵$\boldsymbol{X}$进行处理，将每个节点的特征向量$\boldsymbol{x_i}$转换为嵌入向量$\boldsymbol{e_t}$，构成嵌入矩阵$\boldsymbol{E}=[\boldsymbol{e}_1,\boldsymbol{e}_2,\cdots,\boldsymbol{e}_n]$。$\boldsymbol{E}\in\mathbb{R}^{n\times e}$作为序列输入专家网络层的编码器，用于提取高层次的特征表示。

专家网络层：作为策略网络$\pi_{\theta_v}(a|s)$的核心计算部分，该层负责根据输入状态s_t和当前路径状态即已选择的节点，迭代生成下一跳节点，最终将构建的完整算力路由路径Path输出为动作a_t。该层包括3组结构相同、彼此独立的专家子网络。每个子网络均由多层门控循环单元（gated recurrent unit，GRU）[21-23]构成的编码器–解码器组成，用于处理和分析算网资源信息。每个子网络专用于一种计算任务类型，并拥有特定的网络参数θ_v（$v=1,2,3$）。在推理阶段，系统根据计算任务类型type={1,2,3}，激活对应的专家子网络进行处理，从而实现任务感知的自适应路由决策。GRU中各变量说明见表1。

在专家网络层中，GRU的门控机制是编码器与解码器处理序列信息的核心模块，编码器可以利用其强大的变长序列处理能力、上下文编码能力和特征提取能力，将动态规模、非时序的节点特征集合转化为上下文感知的节点嵌入和固定维度的网络状态摘要，作为后续解码器的输入，为模型理解当前网络整体状况和个体节点状态提供了更丰富的信息基础。具体可以分为以下步骤。

（1）编码器处理

编码器通过GRU逐个处理嵌入矩阵$\boldsymbol{E}$的列向量序列$[\boldsymbol{e}_1,\boldsymbol{e}_2,\cdots,\boldsymbol{e}_n]$，每个输入$\boldsymbol{e_t}$包含了计算任务类型type、计算资源量$R_{v_j}^{\text{comp}}$、带宽状况$B_i$以及节点的度node 4个特征维度，能够全面描述该节点在当前任务下的状态。

GRU通过更新门z_t和重置门r_t 2个神经网络结构来处理嵌入向量$\boldsymbol{e_t}$，具体见文献[22-24]。每个GRU在处理当前输入时，会结合前一步的隐藏状态来更新当前的隐藏状态，按序处理所有n个节点。更新门z_t根据计算任务类型type自动调节历史路径状态$\boldsymbol{h_{t-1}}$与当前节点实时特征$\boldsymbol{e_t}$的融合权重。重置门r_t响应节点资源变化，通过遗

忘历史负载信息避免路由到过载节点，保障计算任务的资源可靠性。GRU使用重置门r_t来计算候选隐藏状态$\tilde{\boldsymbol{h}}_t$，候选隐藏状态$\tilde{\boldsymbol{h}}_t$融合过滤之后的历史状态$r_t \cdot \boldsymbol{h}_{t-1}$和当前节点实时特征$\boldsymbol{e}_t$，旨在时延与负载均衡的联合优化。最终GRU通过$z_t$加权融合历史与实时状态，输出序列最后一步的隐藏状态$\boldsymbol{h}_t^{\mathbf{enc}}$即上下文向量$\boldsymbol{C}$。该向量是整个序列特征的聚合表示，融合了任务需求与网络资源信息，蕴含了节点间拓扑关系以及计算任务与节点资源间的关联，为后续解码器提供关键的决策上下文。具体处理流程详见式（14）~式（17）：

$$z_t=\sigma(W_z\cdot[\boldsymbol{h}_{t-1},\boldsymbol{e}_t]+b_z) \tag{14}$$

$$r_t=\sigma(W_r\cdot[\boldsymbol{h}_{t-1},\boldsymbol{e}_t]+b_r) \tag{15}$$

$$\tilde{\boldsymbol{h}}_t=\tanh(W_h\cdot[r_t\cdot\boldsymbol{h}_{t-1},\boldsymbol{e}_t]+b \tag{16}$$

$$\boldsymbol{h}_t=z_t\cdot\boldsymbol{h}_{t-1}+(1-z_t)\cdot\tilde{\boldsymbol{h}}_t \tag{17}$$

（2）解码器处理

解码器的输入为上下文向量$\boldsymbol{C}$和路径历史状态$\boldsymbol{h}_{t-1}$，其结构采用与编码器相同的多层GRU网络，负责路径的自回归生成。通过式（14）~式（17）进行运算，在解码过程中，解码器GRU接收上一跳节点$\boldsymbol{h}_{t-1}$的嵌入向量及上一时刻解码器的隐藏状态$\boldsymbol{h}_{t-1}^{\mathbf{dec}}$为输入，迭代输出解码器的隐藏向量$\boldsymbol{h}_t^{\mathbf{dec}}$，该隐藏向量融合了当前路径状态信息和全局上下文信息。最后隐藏状态$\boldsymbol{h}_t^{\mathbf{dec}}$经由一个全连接层映射为原始向量$\boldsymbol{o}_t$，表示选择每个节点作为下一跳的原始得分，即：

$$\boldsymbol{o}_t=W_o\cdot\boldsymbol{h}_t^{\mathbf{dec}}+b_o \tag{18}$$

输出层：接收解码器输出的向量$\boldsymbol{o}_t$和当前状态信息，应用动作屏蔽机制，排除无效节点并从概率分布中抽样选取动作a_t，每次迭代生成一个节点v_i，将多个动作选取节点的序列$v_0,v_1,\cdots,v_n$作为一条完整的算力路由路径进行输出。

在深度强化学习中，动作屏蔽是一种关键机制，用于限制智能体在特定状态下可选择的动作，从而避免无效或非法的动作。在该算法进行算力路由的逐跳路由场景中，动作屏蔽可以通过屏蔽不可达节点和已访问节点显著提升训练效率和策略稳定性。该算法在输出层使用动作屏蔽机制的具体步骤如下。

（1）生成二进制动作掩码

首先根据当前节点的邻居列表中已访问节点集合V_{visited}，创建掩码向量$\boldsymbol{M}_t$，如果节点v_i是已访问节点，则$\boldsymbol{M}_t=1$，否则$\boldsymbol{M}_t=0$。具体见式（19）：

$$\boldsymbol{M}_t=\begin{cases}1, & \text{节点}v_i\text{是已访问节点}\\ 0, & \text{其他}\end{cases} \tag{19}$$

（2）调整解码器输出的向量$\boldsymbol{o}_t$

将掩码取对数后与原始向量$\boldsymbol{o}_t$相加，使无效动作向量$\boldsymbol{o}_t$变为−∞。具体见式（20）：

$$\boldsymbol{o}_t^{\mathbf{masked}}=\boldsymbol{o}_t+\log(\boldsymbol{M}_t+\boldsymbol{\varepsilon})(\boldsymbol{\varepsilon}\to 0^+) \tag{20}$$

（3）应用softmax计算概率分布

通过softmax转换，无效动作概率严格归零，有效动作概率自动归一化。具体见式（21）：

$$\pi_{\theta_v}(a|s)=\text{softmax}(\boldsymbol{o}_t^{\mathbf{masked}}) \tag{21}$$

（4）动作选择

从概率分布$\pi_{\theta_v}(a|s)$中采样动作a_t。具体见式（22）：

$$a_t\sim\pi_{\theta_v}(a|s) \tag{22}$$

通过上述步骤，智能体在每一步仅能选择有效连接的未访问节点，显著减少无效探索，加速策略收敛，最终经输出层输出一条完整的算力路由路径。

在本文提出的DRL-CAR算法中，神经网络的复杂度可以从时间复杂度和空间复杂度2个维度分析。

针对时间复杂度，输入层处理算网资源信息以及计算需求，因此输入层的时间复杂度为$O(n)$，

其中n为节点数量，由于专家网络由3组结构相同的编码器–解码器构成，且每个编码器–解码器组合由多个GRU组成，并考虑GRU的结构，假设路径生成的最大步长为T，每个GRU的时间复杂度为$O(d)$，此部分时间复杂度为$O(3\cdot T\cdot d)=O(T\cdot d)$，输出层的时间复杂度为$O(n)$，因此总的时间复杂度为$O(n+T\cdot d)$。

针对空间复杂度，在输入层，网络中n个节点，每个节点有d维的特征，则输出层的空间复杂度为$O(n\cdot d)$。在专家网络层，每个GRU需要存储一个d维的隐藏状态，如果有T个时间步，总的空间复杂度为$O(3\cdot T\cdot d)$。最后是输出层，网络中有n个节点，概率分布和掩码向量的大小都是n，则输出层的空间复杂度是$O(n)$。因此总空间复杂度为$O(n\cdot d+T\cdot d)$。

4　实验设置与评估

本文仿真实验使用Pycharm编辑器，配置所需的Conda环境，代码基于Pytorch深度学习框架实现，运行在具有Intel(R) Core(TM) i9 12 900H 3.70 GHz CPU、32 GB RAM以及RTX 3060 GPU的Windows计算机上。

为确保实验结果的真实性和可靠性，本文从Topology Zoo中选取现实世界的GEANT全球网络拓扑作为主要实验对象，并与Abilene、Switch等典型网络拓扑进行对比实验。实验网络拓扑以及节点设置见表2。本文基于3种不同场景设置不同类型的计算任务，分别为时延敏感型、普通型以及计算密集型。具体而言，针对自动驾驶以及工业控制等实时应用场景，计算任务的时延需求在10~30 ms，对于一些音视频传输等应用场景，计算任务的时延需求在30~60 ms，针对某些计算密集型需求，计算任务时延设置为60~100 ms，因此选取10~100 ms作为实验模拟的时延标准。

表2　实验网络拓扑以及节点设置

网络拓扑	节点数量	链路数量	算力路由节点	计算节点
Abilene	12	30	8	4
GEANT	23	74	15	8
Switch	34	98	24	10

本文采用指数分布来模拟计算请求的到达过程，其中任务到达时间间隔服从独立同分布的随机过程，可以用计算式$f(t)=\lambda e^{-\lambda t}$来表示。其中$\lambda$为任务到达率，即单位时间内任务到达的平均次数，$t$是时间间隔，并根据计算任务请求的服务成功率来判断算力路由的结果，仿真参数设置见表3。

表3　仿真参数设置

参数	含义
计算任务请求数量	1 000
计算任务请求到达时间	按指数分布时间到达
时间片段	200
每个时间片段的平均任务请求数量	5
链路带宽资源/GHz	500~1 000
中心云节点计算资源量	10
边缘云节点计算资源量	6~10
计算任务k所需CPU资源数/CU	2~10
计算任务k最大容忍时延/ms	10~100
数据传输所需带宽数/GHz	10~100
神经网络模型数量	50
神经元层数	4
更新步长	10
初始学习率	0.02
学习率变化轮数	50

为了验证本文所提算法的效果，将所提算法与1个深度强化学习算法DQN、基于贪心算法的节点选择、2个启发式算法随机选择算法进行对比。

（1）本文算法：基于深度强化学习算法联合算力资源及网络资源信息，针对不同场景下的计算任务进行路由，得到相应的路由策略。

（2）DQN算法：基于深度学习的Q学习算法，主要结合了价值函数近似与神经网络技术，

并采用目标网络和经验回放等方法进行网络训练。

（3）基于贪心算法的网络节点选择策略：采用经典贪心算法，对网络路径进行最短路径优先和计算资源最大化匹配策略的组合优化。

（4）随机执行策略：随机选择网络中的节点进行路由转发，直到路由到合适的计算节点进行执行。

选取GEANT网络拓扑，基于上述4个算法进行对比评估，对计算任务请求的服务成功率进行分析，取时延以及负载均衡2个优化目标作为实验性能指标进行考量，同时选取2个不同的网络拓扑来验证算法的泛化能力。

实验结果表明，本文算法在训练过程中表现出较快的收敛性，GEANT网络拓扑下训练损失如图5所示。神经网络从初始化开始，在约75个训练周期后进入快速收敛阶段，经110个训练周期后达到稳定状态。此时，系统损失值稳定在8~10，显示出较好的收敛性。

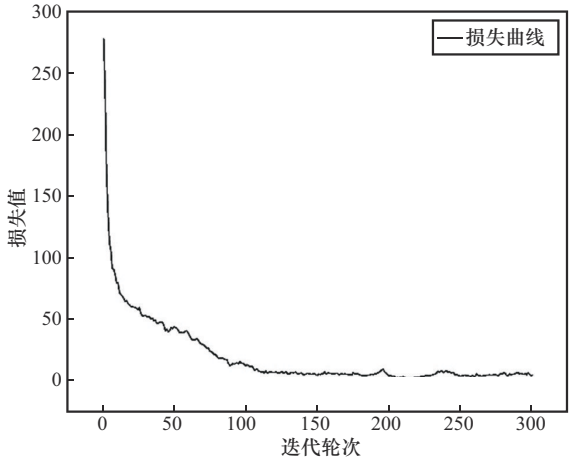


图5　GEANT网络拓扑下训练损失

在动态的算网环境中，路由调度时可能同时面临多个计算任务请求的服务需求。为计算节点设置不同的服务时间，在计算执行结束后释放计算资源，当计算资源尚未释放导致资源不足时，判定服务失败，并通过服务成功率判断计算任务是否成功路由至计算节点执行。

GEANT网络拓扑下服务成功率如图6所示，实验分析表明，本文算法在相同时间片段内的服务成功率最高，约为96.1%，较DQN算法提升约9%，较基于贪心算法提升约8%，较随机策略提升约17%。随机策略由于其高度的不确定性，在节点选择时往往无法保证路由调度的稳定性，服务成功率最低。

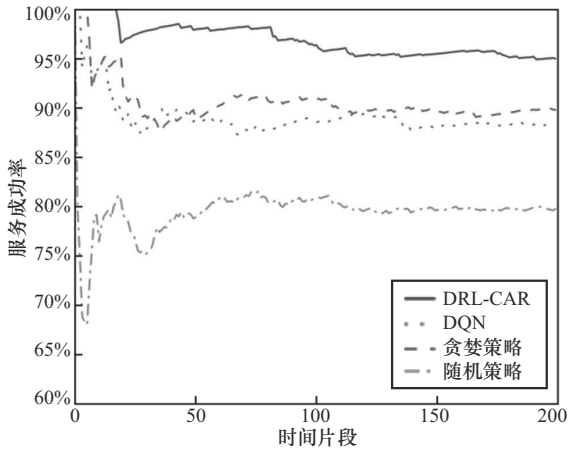


图6　GEANT网络拓扑下服务成功率

在网络状态初始化时，网络中可利用总资源量较为充足，此时服务成功率较高。但随着计算任务按泊松分布持续到达，部分计算资源被占用且尚未释放，可利用总资源量少，计算任务因资源不足而无法满足需求的概率逐渐上升，此时服务成功率呈一定的下降趋势，然而随着时间推移，计算资源释放与新任务到达的频率逐步达到动态平衡，此时服务成功率也随之趋于稳定。

在多目标优化问题中，通常不存在单一的解能同时在所有目标上都达到绝对最优，目标之间往往是冲突的。由于时延和负载均衡是本文优化的2个不同的性能指标，可以通过构建帕累托最优解集的方法来系统评估算法性能，且该解集内的每个解都不存在让一个优化方向没有变差的条件下让另一个指标变得更好。不同算法的帕累托

最优解分布对比如图 7 所示，本文 DRL-CAR 算法的帕累托最优解集整体上更靠近坐标原点，这表明，在相同端到端时延条件下，DRL-CAR 能实现更低的负载方差。其帕累托解集整体位于其他 3 种算法左下方，表明 DRL-CAR 能够在时延最小化和负载均衡优化这 2 个相互冲突的目标之间，找到一组更优的平衡解，其整体性能边界显著优于其他算法。

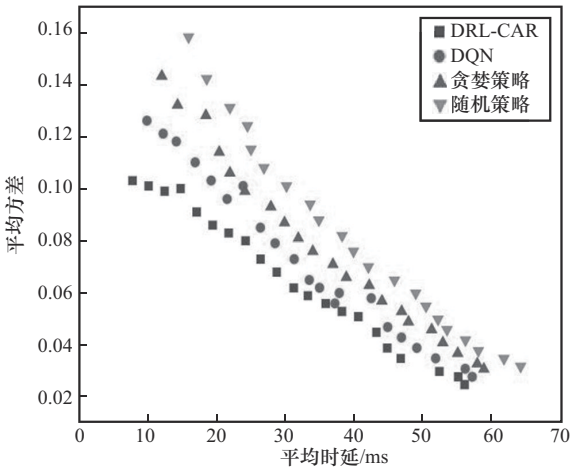


图 7　不同算法的帕累托最优解分布对比

针对 3 种不同计算任务的时延以及负载均衡性能优化需求，本文采用系统化的权重调参方法调整时延与负载均衡的权重，每隔 0.02 步长训练一个模型，基于服务成功率、时延和负载均衡的综合评估，分别选取 λ_1、λ_2 为（0.7，0.3）、（0.5，0.5）、（0.42，0.58）作为时延敏感型、普通型和计算密集型的最优实验权重。

实验结果表明，该算法在处理时延敏感性和计算密集型任务时具有显著优势，在保障高服务成功率的同时，有效优化了端到端时延和节点间负载均衡。对于普通型任务，由于权重设置的平衡性，随机策略在负载均衡方面表现出一定优势，但总体上，本文所提 DRL-CAR 算法在多场景下均表现出较强的适应性和优化能力。GEANT 网络拓扑不同算法下的平均时延及平均方差如图 8 所示。

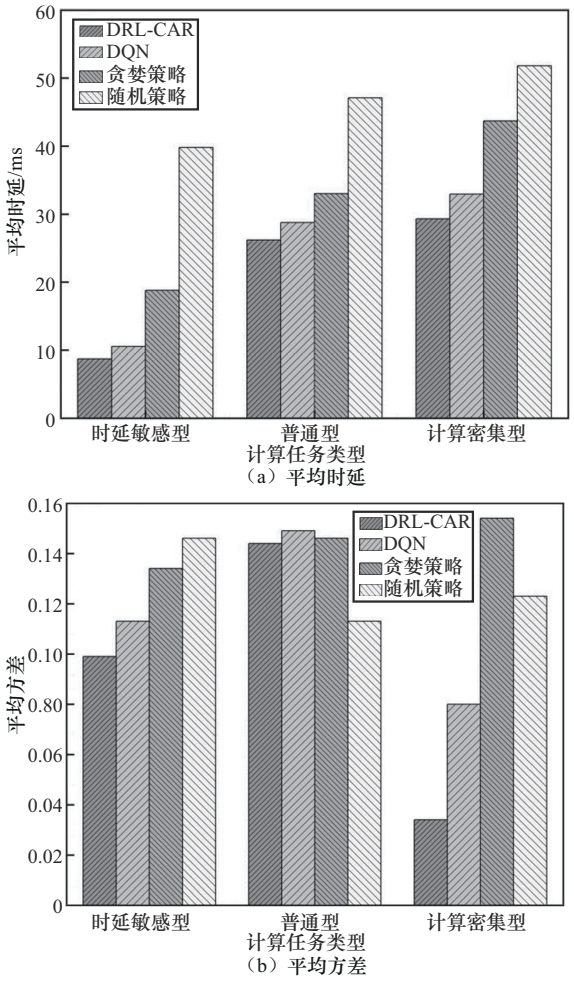


图 8　GEANT 网络拓扑不同算法下的平均时延及平均方差

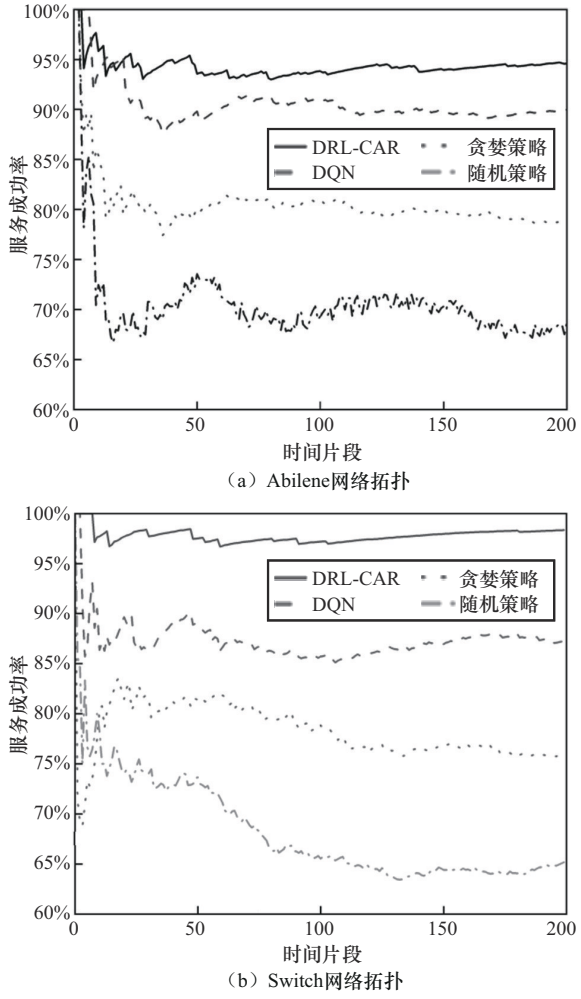
为了验证算法的泛化能力，本文进一步在 Abilene 网络（节点数量较少）和 Switch 网络（节点数量较多）上进行对比实验。实验结果表明，本文 DRL-CAR 算法在不同的网络拓扑均表现出较高的服务成功率，表明该算法能够适应不同的网络环境，具有良好的稳定性和泛化能力，Abilene 及 Switch 网络拓扑下服务成功率如图 9 所示。

为了验证本文提出的 DRL-CAR 算法选取的路径与全局最优路径之间的差距，本文选取 3 种计算任务在 GEANT 网络拓扑下进行验证，通过遍历这些计算任务在网络拓扑中所有“计算节点-路径”对，求解出全局最优路径，通过整体时延差距、负载均衡差距进行加权设定综合目标差距 ΔG 来判断该算法与最优解之间的差距，并

通过箱线图来显示DRL-CAR算法得到的解与全局最优解之间的差距百分比分布情况。

$$\Delta G=\frac{\mu_1\Delta T+\mu_2\Delta L}{\mu_1+\mu_2} \tag{23}$$

其中，ΔG为综合目标差距，ΔT为整体时延差距，ΔL为负载均衡差距，μ_1、μ_2为实验权重配置。



（a）Abilene网络拓扑

（b）Switch网络拓扑

图9　Abilene及Switch网络拓扑下的服务成功率

全局最优差距对比如图10所示，本文提出的DRL-CAR的逐跳决策策略在3类任务下得到的解与全局最优解的平均综合性能差距均在6%上下，其中时延敏感型任务差距最小，因时延信息更易通过局部状态捕获；计算密集型任务的差距最大，因负载均衡优化需要跨节点协调。理论上，通过穷举所有“计算节点–路径”组合可以求得严格的全局最优解，但该方法时间复杂度较高，在大型网络中实际不可行。相比之下，DRL-CAR算法的实现，在保持接近全局最优性能的同时，降低了时间复杂度，实现了优化效果与执行效率的良好平衡，这一特性使其特别适合复杂动态网络环境中的实时路由决策。

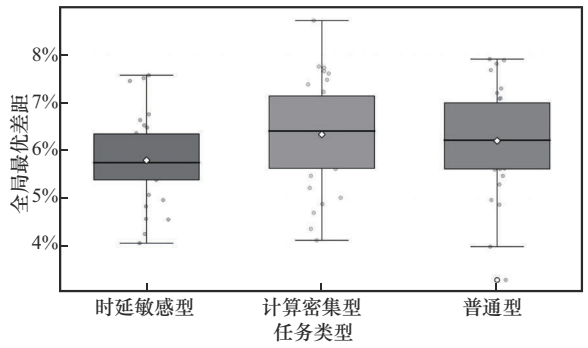


图10　全局最优差距对比

5　结束语

本文提出了一种基于深度强化学习的算网协同动态路由调度算法，旨在应对算力网络中计算任务路由调度的复杂性和多样性挑战。通过结合深度强化学习的优势，本文设计了一个能够自适应处理不同计算任务需求的路由调度算法，并对多种计算任务提出了针对性的优化策略。仿真实验表明，该算法在服务成功率方面表现优异，稳定在96%左右，较启发式算法提升了17%左右。针对不同计算任务的时延以及负载均衡，在满足基本要求的情况下，均有相当提升。通过对不同网络拓扑的对比实验，进一步验证了算法的泛化能力。

未来的研究可以进一步拓展该算法的应用场景，如支持更加复杂的异构计算任务、实现算网资源的更加精细化调度，并结合更多实时反馈机制以进一步提升系统的鲁棒性和性能。此外，考虑算网环境的持续演化和多样化需求，如何进一步提高深度强化学习模型的泛化能力，仍然是未来工作中值得深入探讨的方向。

参考文献：

[1] SUN Y K, LEI B, LIU J L, et al. Computing power network: a survey[J]. China Communications, 2024, 21(9): 109-145.

[2] 姚惠娟, 陆璐, 段晓东. 算力感知网络架构与关键技术[J]. 中兴通讯技术, 2021, 27(3): 7-11.
YAO H J, LU L, DUAN X D. Computing power perception network architecture and key technologies[J]. ZTE Technology Journal, 2021, 27 (3): 7-11.

[3] 庞冉, 易昕昕, 辛亮, 等. 算力网络路由调度技术研究[J]. 电信科学, 2023, 39(8): 149-156.
PANG R, YI X X, XIN L, et al. Research on routing and scheduling technology in computing power networks[J]. Telecommunications Science, 2023, 39 (8): 149-156.

[4] GONG X M, REN S Y, WANG C J, et al. Research on computing resource measurement and routing methods in software defined computing first network[J]. Sensors, 2024, 24(4): 1086.

[5] 雷波, 刘增义, 王旭亮, 等. 基于云、网、边融合的边缘计算新方案: 算力网络[J]. 电信科学, 2019, 35(9): 44-51.
LEI B, LIU Z Y, WANG X L, et al. New edge computing scheme based on cloud, network and edge integration: computational network[J]. Telecommunications Science, 2019, 35(9): 44-51.

[6] 杨烨. 基于算网一体化演进的算力网络技术研究[J]. 现代传输, 2022, (4): 45-48.
YANG Y. Research on computing power network technology based on the evolution of computing network integration[J]. Modern Transmission, 2022, (4): 45-48.

[7] 贾庆民, 丁瑞, 刘辉, 等. 算力网络研究进展综述[J]. 网络与信息安全学报, 2021, 7(5): 1-12.
JIA Q M, DING R, LIU H, et al. A review of the research progress of computing power networks[J]. Journal of Network and Information Security, 2021, 7(5): 1-12.

[8] 段晓东, 姚惠娟, 付月霞, 等. 面向算网一体化演进的算力网络技术[J]. 电信科学, 2021, 37(10): 76-85.
DUAN X D, YAO H J, FU Y X, et al. Computing power network technology for the evolution of integrated computing network[J]. Telecommunications Science, 2021, 37(10): 76-85.

[9] 任杰, 王洪超, 王钦定, 等. 服务感知网络[J]. 电子学报, 2025, 53(2): 371-384.
REN J, WANG H C, WANG Q D, et al. Service aware network[J] Acta Electronica Sinica, 2025, 53(2): 371-384.

[10] 魏汝翔, 刘琦, 赵广, 等. 东数西算下面向业务的路由策略分析与探索[J]. 中兴通讯技术, 2023, 29(4): 14-18.
WEI R X, LIU Q, ZHAO G, et al. Analysis and exploration of business-oriented routing strategy under eastern data and western computing[J]. ZTE Technology Journal, 2023, 29(4): 14-18.

[11] 孙钰坤, 张兴, 雷波. 边缘算力网络中智能算力感知路由分配策略研究[J]. 无线电通信技术, 2022, 48(1): 60-67.
SUN Y K, ZHANG X, LEI B. Research on intelligent computing aware routing allocation strategy in edge computing networks[J]. Radio Communications Technology, 2022, 48 (1): 60-67.

[12] 金天骄, 栗蔚. 基于算力网络的大数据计算资源智能调度分配方法[J]. 数据与计算发展前沿, 2022, 4(6): 29-37.
JIN T J, LI W. Intelligent scheduling and allocation method for big data computing resources based on computing power network[J]. Frontiers of Data and Computing Development, 2022, 4(6): 29-37.

[13] GUO L J, GUO F X, PENG M G. A novel routing method for dynamic control in distributed computing power networks[J]. Digital Communications and Networks, 2024, 10(6): 1644-1652.

[14] 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.

[15] FRANCOIS-LAVET V, HENDERSON P, ISLAM R, et al. An introduction to deep reinforcement learning[J]. Foundations and Trends® in Machine Learning, 2018, 11(3-4): 219-354.

[16] MOUSAVI S S, SCHUKAT M, HOWLEY E. Deep reinforcement learning: an overview[C]//Proceedings of SAI Intelligent Systems Conference. Cham: Springer International Publishing, 2016: 426-440.

[17] TANG F X, MAO B M, FADLULLAH Z M, et al. ST-DeLTA: a novel spatial-temporal value network aided deep learning based intelligent network traffic control system[J]. IEEE Transactions on Sustainable Computing, 2019, 5(4): 568-580.

[18] 张维庭, 孙呈蕙, 王洪超, 等. 算网资源智能适配与融合调度方法[J]. 电信科学, 2023, 39(9): 12-20.
ZHANG W T, SUN C H, WANG H C, et al. Intelligent adaptation and fusion scheduling method for computing network resources[J]. Telecommunications Science, 2023, 39(9): 12-20.

[19] 孙国玮, 许方敏, 朱瑾瑜, 等. 算力网络中的确定性调度与路由联合智能优化方案[J]. 北京邮电大学学报, 2023, 46(2): 9.
SUN G W, XU F M, ZHU J Y, et al. Deterministic scheduling and routing joint intelligent optimization scheme in computing

power networks[J]. Journal of Beijing University of Posts and Telecommunications, 2023, 46(2): 9.

[20] XIE Y P, HUANG X Y, LI J C, et al. Computing power network: multi-objective optimization-based routing[J]. Sensors, 2023, 23(15): 6702.

[21] 赵丹丹, 黄德根, 孟佳娜, 等. 基于BERT-GRU-ATT模型的中文实体关系分类[J]. 计算机科学, 2022, 49(6): 319-325.
ZHAO D D, HUANG D G, MENG J N, et al. Chinese entity relationship classification based on BERT-GRU-ATT model[J]. Computer Science, 2022, 49(6): 319-325.

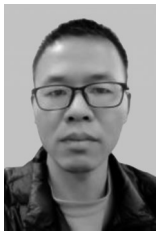
[22] 后同佳, 周良. 基于双向GRU神经网络和注意力机制的中文船舶故障关系抽取方法[J]. 计算机科学, 2021, 48(S2): 154-158.
HOU T J, ZHOU L. Chinese ship fault relation extraction method based on bidirectional GRU neural network and attention mechanism[J]. Computer Science, 2021, 48 (S2): 154-158.

[23] HE X, ZHAO W L, GAO Z J, et al. Short-term load forecasting by GRU neural network and DDPG algorithm for adaptive optimization of hyperparameters[J]. Electric Power Systems Research, 2025, 238: 111119.

[作者简介]

越奇强（2000– ），男，信息工程大学信息技术研究所硕士生，主要研究方向为算力网络和强化学习等。

田乐（1987– ），男，博士，信息工程大学信息技术研究所副研究员，主要研究方向为网络空间安全、软件定义网络。



魏帅（1984– ），男，博士，信息工程大学信息技术研究所副研究员，主要研究方向为高性能计算、网络安全、计算机软件网络架构和机器学习。

胡宇翔（1982– ），男，博士，信息工程大学信息技术研究所教授、博士生导师，主要研究方向为网络空间安全、新型网络架构。

冯旭（1997– ），男，信息工程大学信息技术研究所博士生，主要研究方向为零信任网络、下一代互联网等。

董永吉（1983– ），男，博士，信息工程大学信息技术研究所研究员，主要研究方向为新型网络架构和网络空间安全。

陈博（1989– ），男，博士，信息工程大学信息技术研究所助理研究员，主要研究方向为网络空间安全、软件定义网络。