分类号:　　学校代码：10389

密　级:　　学　　号：12311093016

福建农林大学

硕士学位论文

（学术学位）

数据中心网络中基于动态子网划分的协同路由优化研究

学 习 形 式：全日制

一级学科名称：计算机科学与技术

二级学科名称:

研 究 方 向：计算机网络与信息安全

研究生姓名：林庆杰

指 导 教 师：舒兆港 副教授

完 成　时 间：二〇二六年五月

Research on Collaborative Routing Optimization Based on Dynamic Subnet Partitioning in Data Center Networks

By

Qingjie Lin

Supervised by Associate Professor Zhaogang Shu

A Dissertation Submitted to

Fujian Agriculture and Forestry University

in Partial Fulfillment of the Requirements

for

Master Degree of Engineering

College of Computer and Information Science

Fujian Agriculture and Forestry University

Fujian, P.R. China

Completion Date (May 30, 2026)

Commencement Date (December 13, 2024)

独创性声明

本人声明，所呈交的学位（毕业）论文，是本人在指导教师的指导下独立完成的研究成果，并且是自己撰写的。尽我所知，除了文中作了标注和致谢中已作了答谢的地方外，论文中不包含其他人发表或撰写过的研究成果。与我一同对本研究做出贡献的同志，都在论文中作了明确的说明并表示了谢意，如被查有侵犯他人知识产权的行为，由本人承担应有的责任。

学位（毕业）论文作者亲笔签名：林庆杰　　日期：2026年5月30日

论文使用授权的说明

本人完全了解福建农林大学有关保留、使用学位（毕业）论文的规定，即学校有权送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

延期公开，在　　年后可适用本授权书。 □

公开，本论文属于不延期公开论文。 ☑

学位（毕业）论文作者亲笔签名：林庆杰　　日期：2026年5月30日

指导教师亲笔签名：　　日期：2026年5月30日

目 录

图 录

表 录

摘 要

近年来，软件定义网络（Software-Defined Networking, SDN）与数据中心网络（Data Center Network, DCN）的深度融合为全局资源调度与路由优化提供了可编程基础。随着云计算、边缘计算与人工智能等数据驱动型应用加速落地，DCN 业务流呈现出高并发、强突发等特征，对服务质量（Quality of Service, QoS）保障、负载均衡、能耗控制以及在线可解性提出了更高要求。现有全网尺度路由优化方法虽能统一刻画多类约束，但在大规模拓扑和动态负载场景下面临复杂度高、时效性不足的问题；而结构简化方法虽能降低求解开销，但对划分后的动态演化，以及跨域协同中负载、能耗等多维指标之间的耦合关系刻画仍不充分。

针对上述问题，本文首先围绕结构层协同优化展开研究，提出了一种基于动态子网划分与网关部署协同的路由优化框架（Routing Optimization Framework based on the Collaboration of Dynamic Subnet Partitioning and Gateway Deployment, ROCDSG），以在可控复杂度下实现 QoS、负载与能耗的协同优化。具体而言，框架首先引入基于 Louvain 算法的动态子网划分机制，在流量、负载与能耗约束下自适应聚合节点，实现结构降维以缩小搜索空间；随后，针对子网间转发的边界网关配置，构建网关部署的混合整数非线性规划（Mixed-Integer Nonlinear Programming, MINLP）模型，并采用解空间优化萤火虫算法（Solution Space-Optimized Firefly Algorithm, SOFA）进行高效求解；最后，基于子网与网关骨架设计延迟感知的分层路由算法，为子网内与跨子网流量构建低延迟、稳定路径。仿真结果表明，ROCDSG 的端到端平均延迟较次优算法降低 25.2%，平均跳数与执行时间分别降低 14.4%与 25.6%，同时能够有效保持负载与能耗均衡。

针对所设计的子网网关骨架在多业务场景下仍存在多维 QoS 约束细粒度保障不足、跨子网候选路径构造能力受限以及路径下发后共享链路状态波动难以及时调节等问题，本文进一步从运行层在线增强出发，提出了 QoS 感知的波动反馈增强路由算法（QoS-Aware Volatility Feedback Enhanced Routing Algorithm, QVFER）。该算法首先基于加权主成分分析（Weighted Principal Component Analysis, WPCA）完成请求流优先级划分，并建立统一的路由优化模型，将总流量接纳收益与路径综合代价纳入同一决策框架；随后，通过融合路径摘要索引表（Path Summary Index, PSI）与有效多跳延迟转移图（Effective Multi-hop Delay Transfer Graph, eMDT）扩展跨子网候选路径集合，并在多维 QoS 约束下依据目标函数对可行候选路径进行评分择优，从而完成路径下发；最后，引入链路波动率（Link Volatility Score, VS）作为反馈控制量，对共享链路与关键瓶颈进行持续感知和轻量调节，实现面向动态负载环境的 QoS 增强路由决策。仿真结果表明，QVFER 相较次优算法总体平均接受率提升约 1.43%，平均跳数下降约 8.58%，请求流执行时间缩短约 42.82%，并有效降低了端到端延迟、抖动、丢包率和 SLA 违反率，表现出较好的综合性能优势。

关键词：软件定义网络；数据中心网络；动态子网划分；网关部署；路由优化

ABSTRACT

In recent years, the deep integration of Software-Defined Networking (SDN) and Data Center Network (DCN) has provided a programmable foundation for global resource scheduling and routing optimization. With the rapid deployment of data-intensive applications such as cloud computing, edge computing, and artificial intelligence, traffic flows in DCN have exhibited characteristics including high concurrency and strong burstiness, thereby imposing increasingly stringent requirements on Quality of Service (QoS) assurance, load balancing, energy efficiency, and online solvability. Existing network-wide routing optimization approaches are capable of jointly modeling multiple constraints; however, they often suffer from high computational complexity and insufficient responsiveness under large-scale topologies and dynamic traffic conditions. In contrast, structure simplification methods can reduce optimization overhead, but they remain insufficient in capturing the dynamic evolution after partitioning as well as the coupling relationships among load, energy consumption, and other multidimensional metrics during inter-domain collaboration.

To address these issues, this paper first investigates structural-layer collaborative optimization and proposes a Routing Optimization Framework based on the Collaboration of Dynamic Subnet Partitioning and Gateway Deployment (ROCDSG), aiming to achieve coordinated optimization of QoS, load balancing, and energy efficiency under controllable computational complexity. Specifically, the framework first introduces a dynamic subnet partitioning mechanism based on the Louvain algorithm, which adaptively aggregates nodes under traffic, load, and energy constraints to perform structural dimensionality reduction and shrink the routing search space. Subsequently, for inter-subnet forwarding, a Mixed-Integer Nonlinear Programming (MINLP) model is formulated for gateway deployment optimization, and a Solution Space-Optimized Firefly Algorithm (SOFA) is employed to improve solving efficiency. Finally, a delay-aware hierarchical routing algorithm is designed based on the subnet–gateway backbone architecture to construct low-latency and stable paths for both intra-subnet and inter-subnet traffic flows. Simulation results demonstrate that, compared with the suboptimal algorithm, ROCDSG reduces the average end-to-end delay by 25.2%, while decreasing the average hop count and execution time by 14.4% and 25.6%, respectively, while effectively maintaining load and energy balance.

To further overcome the limitations of the constructed subnet–gateway backbone in multi-service scenarios, including insufficient fine-grained QoS assurance under multidimensional constraints, limited inter-subnet candidate path construction capability, and the lack of timely adjustment to shared-link state fluctuations after route deployment, this paper further proposes a QoS-Aware Volatility Feedback Enhanced Routing Algorithm (QVFER) from the perspective of online runtime enhancement. First, the algorithm employs Weighted Principal Component Analysis (WPCA) to classify request-flow priorities and establishes a unified routing optimization model that jointly considers total traffic admission utility and comprehensive path cost within a single decision framework. Subsequently, by integrating the Path Summary Index (PSI) and Effective Multi-hop Delay Transfer Graph (eMDT), the algorithm expands the set of inter-subnet candidate paths and evaluates feasible candidates under multidimensional QoS constraints according to the objective function, thereby selecting the optimal route for deployment. Finally, the Link Volatility Score (VS) is introduced as a feedback control metric to continuously monitor and lightly regulate shared links and critical bottlenecks, enabling QoS-enhanced routing decisions in dynamic traffic environments. Simulation results show that, compared with the suboptimal algorithm, QVFER improves the overall average acceptance ratio by approximately 1.43%, reduces the average hop count by 8.58%, shortens request-flow execution time by 42.82%, and effectively decreases end-to-end delay, jitter, packet loss rate, and SLA violation rate, demonstrating superior overall performance.

KEYWORDS: Software-defined networking; Data center network; Dynamic subnet partitioning; Gateway deployment; Routing optimization

1 绪论

1.1 研究背景及意义

随着云计算、边缘计算与人工智能等数据驱动型应用在数据中心内加速落地，数据中心网络（Data Center Network, DCN）所承载的业务流呈现出高并发、强突发与多等级并存的典型特征[1]。在线交互、分布式存储访问以及大规模分布式训练与推理等场景，对服务质量（Quality of Service, QoS）提出了更为严格的要求：在追求毫秒级甚至亚毫秒级端到端延迟的同时，还需在负载波动与热点迁移下维持抖动与丢包的可控水平，避免性能长尾恶化[2]。这意味着网络必须在更短时间尺度内感知并响应流量与负载变化，且多维约束并行叠加[3]。由此，网络状态变化更快、约束更复杂、决策窗口更窄，使路由与资源配置从可达性问题转向在线多目标协同优化问题。

然而，现有 DCN 的路由与资源管理机制在规模扩展与业务动态叠加下暴露出突出矛盾。传统分布式控制依赖局部信息与静态策略，难以形成全局一致的决策视角，容易导致跨域路径次优并诱发拥塞扩散[4]；以机架或集群为单位的静态分域与固定资源配置虽能降低计算复杂度，却缺乏对流量时空演化的适应性，当跨域通信占比升高时，域边界处易出现汇聚效应并形成关键瓶颈，进而引发排队积压与绕路增多，造成 QoS 持续退化[5]。与此同时，能耗已成为数据中心运行不可忽视的约束，相关研究指出数据中心用电占比已处于约 1.5%–2%的量级并仍在持续增长[6]。但现有能耗管理多为设备级、慢周期的粗粒度调控，难以与在线流量变化及 QoS 需求同频联动，从而使能耗控制与低延迟保障、负载均衡需求相互掣肘。因此，在业务与规模扩展条件下，如何在保证严格 QoS 的同时实现负载均衡与能耗约束，并保持在线可解性与结果稳定性，已成为 DCN 迈向智能化服务的关键挑战。

面向上述矛盾，业界与学术界普遍借助软件定义网络（Software-Defined Networking, SDN）[7]等可编程架构增强全局可视与策略下发能力。SDN 为全网状态感知（Sensing）、统一编排（Orchestration）与快速执行（Enforcement）提供了基础条件，使得从全局角度开展路径与资源协同成为可能；但在大规模拓扑与多目标、多约束并行的条件下，直接在全网层面进行集中式求解往往面临计算复杂度高、在线收敛困难的问题[8]，难以满足实时决策时限。相反，单纯依靠静态结构简化虽然能够降低计算开销，却会牺牲对业务演化的适应性[9]，使路径质量随时间退化，并在热点增强与跨域通信上升时产生瓶颈累积效应。由此可见，DCN 需要一种兼顾“可扩展的结构化抽象”与“在线稳健的路由决策”的综合方法：既要在结构层面有效压缩搜索空间、降低求解难度，又要在运行层面能够感知负

载波动、稳定 QoS，并与能耗目标形成可操作的协同机制。

鉴于此，本研究提出一种面向 DCN 的协同路由优化框架，将复杂的全网多约束问题转化为可控的层次化决策过程。框架首先从结构侧入手，基于流量、负载与能耗等多维约束进行动态子网划分，以结构性降维缩小路径搜索空间；在此基础上，面向跨子网通信的关键瓶颈进一步设计弹性网关部署机制，使跨域转发能够随业务变化动态调整，避免对少量固定边界节点的长期依赖而形成汇聚热点。随后，依托子网—网关骨架构建延迟感知的分层路由算法，分别为子网内与跨子网流量生成低延迟、稳定的候选路径与转发策略，从而为大规模拓扑下的可解性与可控性提供支撑。

需要强调的是，结构化抽象并非终点，而是支撑动态优化有效性的基础前提：即便完成子网划分、网关骨架与分层路由设计，强动态负载下仍可能出现绕路倾向、局部新热点与性能波动。为此，本研究进一步构建面向在线业务的闭环路由增强机制。在候选层面，通过提高候选路径生成的多样性与覆盖度，减弱传统分层路由因候选集受限而对少数热点通道的依赖。在决策层面，采用可行性筛选与代价驱动结合的方式择优选择路径。与此同时，在运行层面引入链路波动率（Volatility Score, VS）作为监测与反馈信号，对代价权重或选择偏置进行轻量自适应调节，从而在网络状态显著变化时及时抑制热点迁移与负载波动带来的性能退化。总体而言，本研究以结构降维与骨架化组织保障在线可解性，以闭环反馈增强提升动态稳定性与鲁棒性，实现复杂度与性能的协同兼顾。

1.2 国内外研究现状

数据中心网络的路由优化设计是提升网络效能与资源利用的关键环节，长期以来受到学术界与工业界的持续关注。近年来，面向多约束路由决策与在线资源调配已形成大量理论模型与工程实践。围绕精细联合优化与规模可扩展性这一主线，现有研究大体可归纳为两类代表性技术路径：其一从全网视角出发，将多类约束与资源分配纳入统一模型，通过全局搜索、松弛分解或近似优化提升解质量与可行性；其二利用网络结构特征开展结构性降维（如分域、分层与骨架化），以缩小搜索空间、分解决策过程，从而降低计算复杂度并提升在线响应能力。基于此，本文将相关工作按“全局多约束路由优化”与“结构性降维路由优化”两类进行梳理与评述，为后续方法设计与对比分析提供研究脉络与评价基准。

1.2.1 全局多约束路由优化

在传统优化模型方面，研究普遍以全网尺度的统一建模为主线，将端到端 QoS 与负载均衡、能耗控制等目标共同纳入优化框架，并通过启发式、松弛近似、

分解策略或并行加速来缓解在线求解压力。围绕能耗与负载的协同管理，He 等人[10]提出融合能耗优化与负载均衡的流量管理策略，通过将多路径路由与设备激活最小化问题建模并设计启发式算法实现高效求解；Pathan 等人[11]进一步引入流优先级因素，构建多目标整数线性规划模型并提出 PEMA、PEDL 双算法，分别侧重能耗最小化与负载均衡优化；Lu 等人[12]从策略执行层面提出最小能耗启发式，通过流量分类与设备启停联动在节能目标与 QoS 满意率之间实现折中，并强调高负载场景下的可用性与稳定性。在 SDN 场景中，研究进一步利用集中式控制带来的全局视图与算力优势提升全局建模的可在线性：Naeem 等人[13]开发 GPU 加速的并行路由框架，将多约束 QoS 路由转化为最大流与最小成本模型并以贪婪策略实现路径计算并行化；肖慧等人[14]针对多约束条件下的路由优化问题，构建了相应的路由模型，并结合候选路径筛选与优化算法提升路由寻优能力和计算效率。与此同时，Wu 等人[15]采用先分级、再寻优的分层调度思路，通过 ToS 字段划分流量优先级，高优先级流以延迟与丢包复合成本驱动动态选路，低优先级流基于剩余带宽实施负载均衡，从而缓解多目标冲突并增强关键业务保障。除上述建模与工程加速外，元启发式与松弛近似也被广泛用于降低组合复杂性并提高可解性，例如尹凤杰等人[16]提出改进人工蜂群多目标优化（ABC-MOP），通过将搜索个体映射为端到端路径并以突变与选择机制迭代优化以获得更优路由；Niu 等人[17]提出 YWDSR，通过平衡全局搜索与局部搜索改善最小跳数选路在拥塞场景下的性能退化；BinSahaq 等人[18]提出 MODLARAC，通过减少 Dijkstra 调用次数并结合小规模子图补充搜索，在解质量与运行开销之间获得更可控的折中。虽然该类方法能够在一定规模下取得解质量与求解效率的折衷并支持面向不同目标偏好的策略设计，但由于仍依赖全网状态收集与迭代重优化，当网络规模扩大与业务动态增强时往往面临候选空间膨胀以及控制开销与策略波动风险放大等问题，从而对大规模网络的实时决策能力形成约束。

随着人工智能的发展，数据驱动方法逐渐将路由优化从静态规则与固定权重推进到状态感知与策略学习的自适应决策范式，其关键在于利用链路与队列等运行状态在线生成或更新路由策略，以提升对流量波动与拓扑变化的适配能力。围绕这一思路，Sanchez 等人[19]采用深度强化学习（Deep Reinforcement Learning, DRL），以链路状态与队列深度为观测构建多目标奖励函数并自主训练路由策略；Wang 等人[20]将预测机制引入决策闭环，利用长短期记忆网络（Long Short-Term Memory, LSTM）刻画流量趋势，并构建 MILP–DDPG 混合框架以实现带宽分配与设备能耗策略的动态调整。针对 DRL 在在线训练中可能出现的收敛慢与迁移成本高等问题，Kim 等人[21]引入 M/M/1/K 队列模型构建离线预训练框架，通过仿真生成链路权重优化策略并由 SDN 控制器下发至交换机，从而降低拓扑变化带来的重训练开销；Chen 等人[22]聚焦网络状态的时空依赖建模，提出融合注意力机制与时空相关性的近端策略优化算法（Proximal Policy Optimization, PPO），

并结合门控循环单元与图注意网络提取链路时序演化与节点空间依赖，同时引入跳跃连接增强状态表征，使 DRL 代理在多场景下具备更强的泛化与自适应能力。为了进一步提升动态环境下的鲁棒性与可用性，Wang 等人[23]提出 DFRDRL，将模糊逻辑的在线路径评估与 DRL 结合，并引入实时流量预测与拥塞延迟机制以强化延迟保障；Wu 等人[24]采用 PPO 与图神经网络（Graph Neural Network, GNN）协同建模，通过候选路径筛选压缩动作空间并利用 GNN 学习拓扑表征，使得在拓扑变化或节点故障条件下仍能自适应输出分流决策以满足多业务 QoS。与此同时，亦有研究从工程可部署性与方法扩展性的角度，采用更轻量的强化学习或协同优化框架开展路由设计。She 等人[25]围绕业务需求驱动的动态 QoS 按需路由，通过面向服务需求构建优化目标并降低控制器侧路径搜索复杂度，实现路由的统一计算与下发；杨宋亮等人[26]在 Q-Learning 与 SARSA 框架上通过动态学习率与吞吐-延迟联合奖励加速收敛并提升中高负载下的路由性能；刘佩鑫等人[27]面向数据中心强动态流量提出 DRL-NSA 与 MDRL-NSA，分别通过网络态势预测增强状态感知与多智能体协同缓解高维动作空间问题，以提升策略收敛性与适应性。综合来看，数据驱动方法能够在复杂环境中实现动态路由策略生成并提升自适应能力，但整体仍存在对状态表征与超参数较为敏感、以及在严格 SLA 约束下的策略稳定性不足等问题，因而仍需与约束化建模与稳定性机制进一步融合以增强工程可信度。

需要指出的是，现有方法大多侧重于路径生成与初始下发，对路径运行过程中共享链路热点形成及其引起的性能偏移考虑不足。当网络状态发生变化时，既有路径容易偏离初始评估结果，而现有研究普遍缺乏对已下发路径进行轻量反馈调节和快速微调重构的能力。

1.2.2 结构性降维路由优化

在传统优化建模与数据驱动学习两类路线之外，学术界也形成了从网络结构层面提升路由可扩展性与在线效率的研究思路，即通过结构性降维重构路由决策空间。子网划分作为其中的典型手段，通常将全网问题组织为域内局部优化与域间协同控制的分层过程，从而在缩小搜索空间的同时降低控制开销并提升求解效率。相关研究中，El-Hefnawy 等人[28]采用盒覆盖法与 K-means 聚类对拓扑进行分域，并在划分结果上结合蚁群算法与变异算子来实现动态路由优化；Tiwari 等人[29]提出基于子网划分的 SDN 架构，利用三态内容可寻址内存（TCAM）的通配符匹配实现基于前缀的 IP 地址匹配，以降低交换机存储压力并减少交换机到控制器的交互开销。面向特定业务场景，Wang 等人[30]针对 5G 工业物联网设计 QoS 与数据隐私感知的路由协议，通过改进信息映射并结合负载、延迟约束构造最优子网；Zhang 等人[4]提出基于盒覆盖的路由算法，利用分形维数重整化构建

子网层级结构，并在 Mininet 平台验证其在超大规模 SDN 中的适用性。总体来看，这类基于盒覆盖与聚类的划分方法能够有效实现结构降维，但也呈现出两类具有代表性的不足：其一，子网数量或划分粒度往往依赖预设尺度，难以随拓扑演化与业务态势变化自适应调整[4,28]；其二，部分工作对子网划分与能耗等运行指标的耦合刻画相对不足（如 Wang 等人的研究[30]更侧重负载与延迟约束而未显式引入能耗因素），从而在多维约束协同场景下仍有提升空间。另一方面，面向能效目标，Dev 等人[31]在无线传感器网络中通过 K-means 生成初始簇结构，并基于灰狼优化动态调整簇头（网关）位置，以节点剩余能量、距基站距离及负载为目标实现能量均衡划分；此外，Zou 等人[32]从社区发现与关键节点约束出发改进聚合划分算法，以提升分区结果的可用性，Wang 等人[33]将子网划分与优先级增强的 Q-learning 融合，通过子网重要度引导策略选择与奖励设计，从而在恢复与调度过程中优先保障关键子网的运行韧性。总体而言，子网划分为大规模网络路由提供了有效的结构化组织方式，但如何实现面向动态业务的自适应划分，并将 QoS、负载与能耗等指标与域内域间协同机制更紧密地耦合，仍是该方向进一步深化的关键问题。

在完成子网划分实现结构性降维之后，跨子网通信由域间边界节点承载汇聚与转发职能，因而网关的数量、位置与容量配置会显著影响跨域路径代价、瓶颈形成位置及端到端 QoS 的波动特性。围绕这一关键环节，现有研究大体可归纳为三类思路：其一是全局尺度的联合建模与求解，Torkzaban 等人[34]构建联合优化卫星网关部署与 SDN 控制器放置的混合整数线性规划模型，通过问题分解与子模函数转化实现复杂约束下的高效求解，并进一步面向地面网络设计兼顾总成本与平均延迟的路由协议；其二是面向动态场景的启发式与近似优化，Matni 等人[35]针对动态物联网建立成本支出与 QoS 的联合优化模型，通过相似设备分组实现网关数量与部署位置的协同确定，Cao 等人[36]提出容量约束下的卫星网关部署方法，结合枚举与启发式贪心策略缩小搜索空间并加速最优网关组合发现；其三是将关键节点选择、受约束选路与智能优化相结合的结构化方案，例如 Rafique 等人[37]提出 SoftCaching 框架，通过全网流量矩阵来识别关键节点并在延迟与链路利用率约束下求解受约束最短路径以降低检索延迟并缓解链路负载，Choi 等人[38]从成本最小化角度构建网关部署模型并利用分治与近似策略支撑大规模快速求解，Huang 等人[39]将几何 K 中心问题与竞争群体优化相结合，通过竞争机制与自适应搜索提升收敛性与鲁棒性以改善覆盖效果并提升 QoS 指标。综合来看，这些工作验证了网关部署与路由联动在降低路径代价与缓解跨域瓶颈方面的有效性，但多数研究仍以成本、覆盖或宏观延迟为主要度量，对数据中心场景中能耗关联以及网关队列管理延迟、协议栈处理延迟等细粒度延迟来源的统一刻画相对不足，同时与子网划分后的域内域间协同过程缺乏同步演化的联动设计。因此，面向数据中心跨域通信场景，有必要构建与子网划分同步迭代的弹性网关部

署机制，并与分层路由协同优化，在同一框架内实现 QoS、负载与能耗的可控权衡，以提升策略稳定性与工程一致性。

1.2.3 研究现状小结

针对上述研究现状，可归纳为以下两点：

(1) 在全局多约束路由优化方面，现有研究主要沿两条路线推进，一类以全网统一建模为核心，通过启发式、松弛近似、分解与并行加速等手段在 QoS、负载与能耗目标间进行显式权衡；另一类以数据驱动学习为核心，利用链路与队列状态在线生成或更新路由策略，并借助预测、预训练与时空特征建模提升环境适配能力。总体上，两类方法在中小规模场景中能够取得较好的解质量或自适应效果，但当拓扑规模扩大、业务动态增强时，往往需要频繁依赖全网状态与迭代式优化或训练过程，导致在线开销上升，进而对实时决策效率与策略稳定输出形成约束。此外，现有研究多聚焦于初始路径决策，对路由下发后可能出现的性能漂移与局部共享热点演化缺乏有效干预能力。

(2) 在结构性降维路由优化方面，研究者通过子网划分与骨架化将全网问题转化为域内优化与域间协同，从而显著压缩搜索空间，并进一步引入网关部署与路由联动以缓解跨子网汇聚瓶颈。尽管该路线在可扩展性上更具优势，但现有工作对子网划分粒度的动态演化、跨域协同中的多指标耦合刻画仍不充分，尤其是对能耗关联以及队列管理延迟、协议栈处理延迟等端到端细粒度延迟来源的统一建模不足，难以支撑面向数据中心场景的稳定可控优化目标。

1.3 研究内容

结合前述研究现状梳理可以看出，两类代表性思路各有侧重：全局多约束优化与学习式决策能够体现全局权衡或自适应能力，契合多维 QoS 保障的设计需求，但在大规模强动态场景下往往难以维持可接受的在线开销；结构性降维能够显著提升可扩展性与求解效率，但现有工作对 QoS、负载与能耗等多指标的统一刻画以及跨域协同支撑仍不充分。换言之，要同时满足算得快和控得稳，仅沿单一路线推进难以闭合工程需求，因此有必要构建融合结构降维、跨域骨架与多约束路由决策的协同框架，在统一闭环内实现可扩展求解与多指标可控权衡。基于上述动机，本文研究内容聚焦于以下两方面：

(1) 面向结构层降维设计，并围绕 QoS、负载与能耗等多维目标实现协同建模与权衡，本文提出了基于动态子网划分与网关部署协同的路由优化框架，称为 ROCDSG（Routing Optimization Framework based on the Collaboration of Dynamic Subnet Partitioning and Gateway Deployment）。该框架以协同闭环将全局视图转化

为可执行决策。首先，基于 Louvain 算法，在流量、负载与能耗等多维约束驱动下进行动态子网划分，从逻辑层面对全网分域，从而缩小路径搜索空间。其次，针对动态分域后产生的跨子网通信（即源节点与目的节点分属不同子网的端到端业务流）在子网间转发时通常依赖边界网关与子网间链路、易在网关处形成汇聚效应并演化为关键性能瓶颈，从而推高端到端延迟并加剧拥塞扩散的问题，本文在各子网内进行弹性网关部署，联合考虑延迟、能耗与负载均衡，并采用解空间优化萤火虫算法（Solution Space-Optimized Firefly Algorithm, SOFA）进行高效求解，以保证大规模场景下的可解性与部署稳定性。最后，在形成的子网与网关骨架上执行延迟感知分层路由，按最小延迟与稳定性准则构建跨域路径，实现服务质量与路径效率的兼顾，并通过仿真实验验证该协同框架的有效性与优越性。

(2) 在 ROCDSG 设计的分层路由中，跨子网通信通常依赖边界网关完成转发。该机制虽能通过结构降维压缩搜索空间并提升路由转发效率，但当跨网业务增多且链路负载快速波动时，仍易出现路径绕行、局部汇聚等问题。同时，其关注重点主要在于转发组织与搜索压缩，对多业务环境下的差异化 QoS 保障、链路状态演化以及路由运行期稳定性考虑不足，因此仅依赖骨架化路径组织仍难以满足在线场景下的实时动态需求。基于此，本文进一步提出 QoS 感知的波动反馈增强路由算法（QoS-Aware Volatility Feedback Enhanced Routing Algorithm, QVFER）。具体而言，首先，在业务侧对到达请求流采用加权主成分分析（Weighted Principal Component Analysis, WPCA）完成优先级划分，将各业务的延迟、抖动、丢包等 QoS 需求转化为分级判定门槛，用于指导不同优先级的可行性筛选与路径偏好。其次，基于该分级信息建立在线路由优化模型：以提升请求流总体接受率并在已接纳业务上最小化路径代价为优化目标，同时满足链路带宽容量约束与各类业务 SLA（Service Level Agreement, 服务等级协议）阈值等约束。最后，QVFER 围绕该模型给出在线求解流程：候选生成阶段先利用路径摘要索引表（Path Summary Index, PSI）完成快速检索与初筛，再通过虚拟坐标嵌入并基于 k 近邻（k-Nearest Neighbors, kNN）构建有效多跳延迟转移图（Effective Multi-hop Delay Transfer Graph, eMDT）补充跨子网低跳数候选，形成统一候选池；求解阶段对候选池执行多约束可行性判别与代价评分择优，输出可下发路径并更新资源占用；运行阶段以链路波动率（Link Volatility Score, VS）为反馈信号，在控制周期内识别热点与关键瓶颈，触发必要的路径微调或重构。在实验部分，本文统计请求流接受率、端到端 QoS、丢包率、SLA 违反率及运行效率等指标，以验证引入路由增强机制后的闭环框架在动态场景下对综合性能的提升效果。

1.4 论文章节安排

为系统阐述本文围绕数据中心网络动态路由优化所开展的研究工作，全文按

照由基础理论到结构优化、再到路由增强与总结展望的思路展开，共分为五章，各章内容结构如图 1-1 所示。

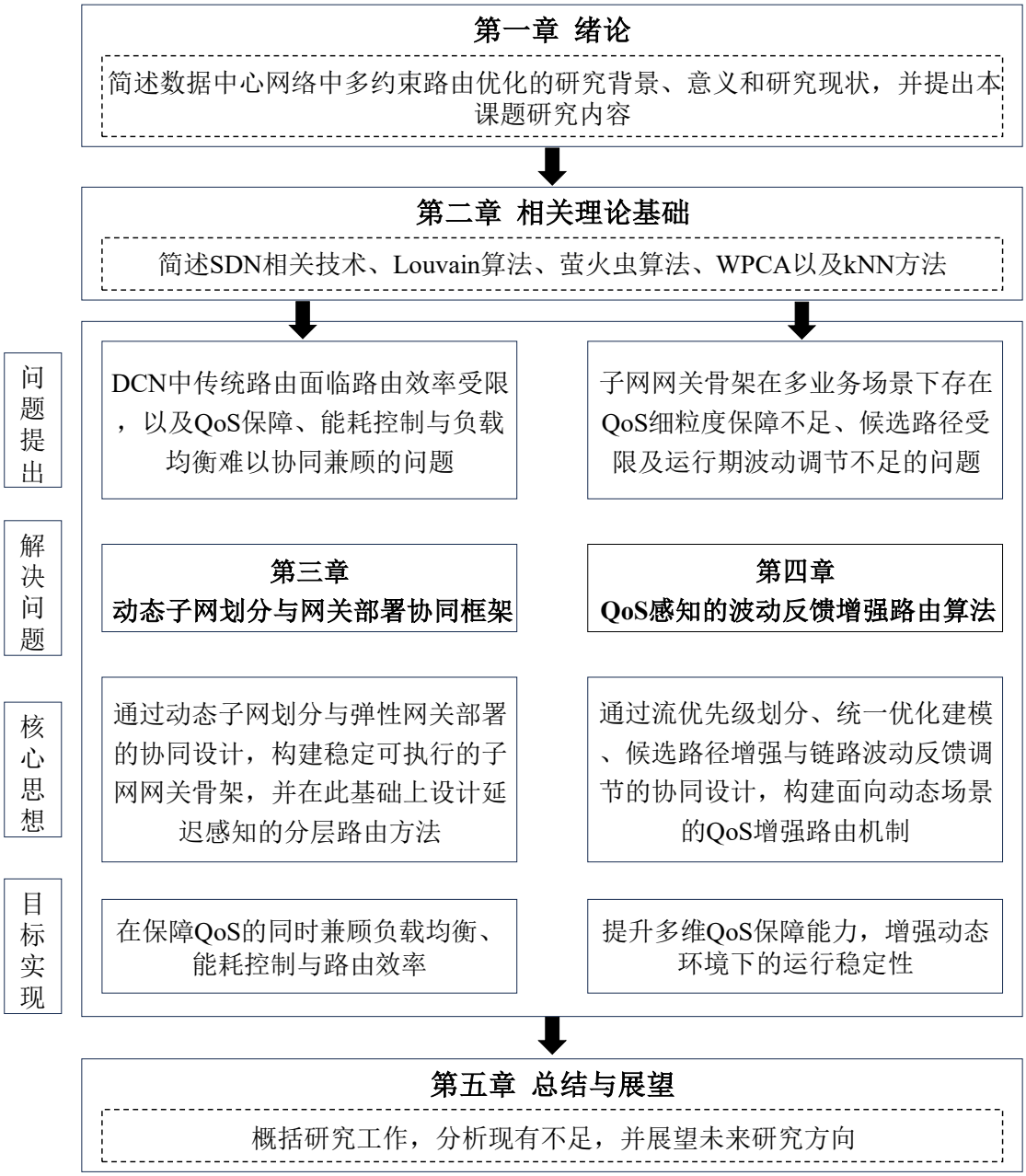


图 1-1 论文的组织结构与章节安排

第一章为绪论。本章首先介绍论文的研究背景与研究意义，围绕数据中心网络中多约束路由优化问题的研究现状，从全局多约束路由优化与结构性降维路由优化两个方面进行了综述，并对现有研究存在的不足进行了总结。在此基础上，进一步明确了本文的研究内容，并给出了全文的章节安排。

第二章为相关理论基础。在这一章中，本文系统介绍了研究所涉及的关键理论与技术基础，包括 SDN 相关技术、Louvain 算法、萤火虫算法、加权主成分分析法以及 kNN 方法，为后续章节中动态子网划分、网关部署优化与 QoS 增强路由算法的设计提供必要的理论支撑。

第三章为动态子网划分与网关部署协同框架。针对数据中心网络中传统路由在规模扩展与动态负载场景下面临路由效率受限，以及 QoS 保障、能耗控制与负载均衡难以协同兼顾的问题，提出了动态子网划分与网关部署协同的路由优化框架。本章分别建立动态子网划分与弹性网关部署的优化模型，并设计相应求解算法；在此基础上，进一步提出延迟感知的分层路由算法，最后通过仿真实验验证所提框架在延迟、负载均衡、能耗控制与路由效率等方面的有效性。

第四章为 QoS 感知的波动反馈增强路由算法。基于上一章动态子网划分与网关部署形成的子网网关骨架，针对其在多维 QoS 约束细粒度保障不足、跨子网路径构造方式较为单一等方面的局限，以及现有方法对路径下发后共享链路状态波动缺乏及时调节的问题，提出了 QoS 感知的波动反馈增强路由算法。本章首先介绍改进后的路由框架，随后围绕流优先级划分、优化模型构建与链路波动率机制设计展开研究，在此基础上给出 QVFER 算法的流程描述，最后通过仿真实验验证所提算法在多维 QoS 保障与动态路由性能提升方面的有效性。

第五章为总结与展望。本章对全文的研究工作进行了系统总结，同时指出了当前研究的局限性与不足，并对后续可能的研究方向进行了展望。

2 相关理论基础

本章旨在系统阐述本文研究所依赖的关键技术与理论基础。首先，从软件定义网络（SDN）的体系架构与控制机制出发，重点介绍 OpenFlow 协议及 Ryu 控制器的工作原理与实现要点；随后，对本文涉及的主要算法方法进行梳理，包括 Louvain 算法、萤火虫算法、WPCA 评权方法以及 k 近邻等，为后续模型构建与路由优化研究提供必要的理论支撑。

2.1 SDN 相关技术

2.1.1 SDN 体系架构

SDN 的提出源于传统网络架构在实际应用中暴露出的问题和不足。在传统网络中，交换机、路由器等设备通常同时承担控制决策与数据转发功能，网络控制依赖分布式协议协同完成，并通过逐设备的命令行方式实现策略配置与维护。这种控制与转发耦合、以设备为中心的运行模式在网络规模较小时尚可支撑，但当网络节点数量持续增长、业务类型日益多样且流量模式呈现强动态特征时，网络的全局可视性与一致性管理能力显著不足，进而导致运维成本上升、策略落地难以及时适配业务需求。为缓解上述矛盾，研究界与产业界逐步形成了“控制与转发分离、网络能力开放可编程”的设计思路[40]，并在 Ethane 与 OpenFlow 等工作的推动下得到工程化实现，随后在 ONF 等组织的标准化推进下形成较为完整的 SDN 技术体系[41]。

SDN 的基本思想是将控制平面从网络设备中抽离，由逻辑集中（可物理分布）的控制器统一维护网络全局状态并进行策略计算，而数据平面则保留高性能的数据包转发能力，按控制器下发的规则执行匹配-动作的转发行为。通过这种解耦设计，SDN 有效降低了控制逻辑与硬件设备之间的耦合程度，使网络能够以软件方式快速迭代与按需定制，从而显著提升网络的灵活性与可编程性。同时，控制器基于全局视图进行统一编排与路径优化，更易实现流量工程、QoS 保障与负载均衡等全局目标，并可结合状态回传构建策略调整的闭环机制，提升网络对动态业务的适应能力。

SDN 架构通常采用三层结构，自上而下分别为应用平面、控制平面和数据平面[27]，北向接口与南向接口则构成贯通各层的关键组件，如图 2-1 所示。具体来说：

(1) 应用平面：应用平面位于 SDN 架构的上层，主要承载网络应用与服务逻辑。其通过北向接口调用控制器提供的抽象网络能力，以完成策略定义、资源请

求以及网络状态与执行反馈的获取，从而实现面向业务需求的网络服务编排与快速调整。

(2) 北向接口（NBI）：北向接口作为应用平面与控制平面之间的交互桥梁，向上提供标准化的接口能力（如 API/意图接口），支持应用提交策略请求、查询网络状态并接收事件通知。

(3) 控制平面：控制平面以 SDN 控制器为核心，负责拓扑与状态感知、策略解析与冲突消解，以及路径计算与资源分配等关键控制功能。控制器基于全局视图生成网络控制决策，并通过南向接口将决策进一步转化为可执行的规则与配置，实现对数据平面的集中式管理与编排。

(4) 南向接口（SBI）：南向接口用于控制器与数据平面转发设备之间的通信与控制，主要承担转发规则下发与网络状态回传等功能。当前主流的南向接口协议包括 OpenFlow 协议、NETCONF 协议[42]和 PCEP 协议[43]等，其中 OpenFlow 协议应用较为广泛。控制器通过南向接口将流表项、队列参数与相关策略配置下发至设备侧，同时接收设备回传的链路状态、端口事件及流量统计信息，为控制平面的动态决策与闭环调优提供数据支撑。

(5) 数据平面：数据平面由交换机、路由器等转发设备构成，负责高速数据转发与规则执行。设备侧通常部署南向代理以实现与控制器的双向通信，按控制器下发的规则完成匹配与转发等操作，并将链路/端口状态与流量统计等信息回传至控制平面。

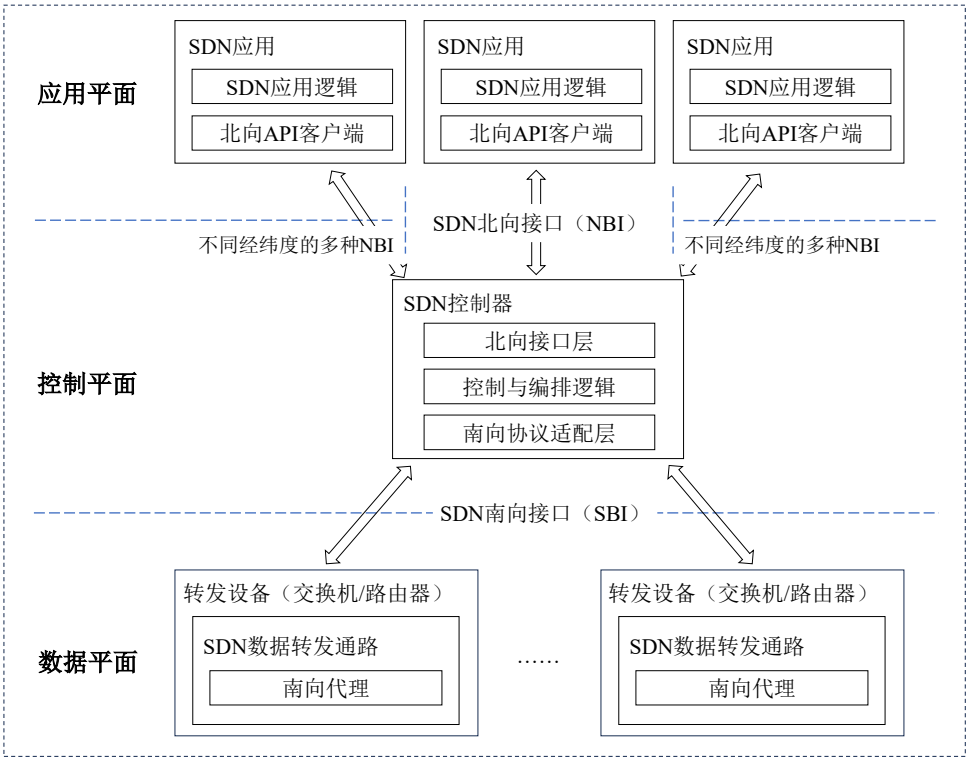


图 2-1 SDN 体系架构

2.1.2 OpenFlow 协议

OpenFlow 协议是 SDN 架构中典型的南向接口通信规范，用于实现 SDN 控制器与数据平面转发设备之间的控制交互。其核心思想在于将传统网络中耦合于设备内部的控制功能与数据转发功能进行解耦，实现控制平面与数据平面的分离，并通过标准化协议在二者之间建立统一的通信机制；数据平面则以流为基本粒度，采用匹配-动作（match-action）的方式执行转发，从而使网络转发行为能够由控制器集中编程与动态调整，提升网络的灵活性、可编程性与可管理性。

OpenFlow 架构主要由 OpenFlow 交换机、网络虚拟化层与 SDN 控制器构成，如图 2-2 所示。OpenFlow 交换机位于数据平面，负责依据控制器下发的规则完成高速转发；网络虚拟化层用于对底层网络资源进行抽象与隔离，为上层控制提供统一的逻辑视图；SDN 控制器位于控制平面，承担数据平面状态信息采集、全局视图维护以及转发策略制定与下发等功能。

在 OpenFlow 交换机内部，流表（Flow Table）与安全通道（Secure Channel）是两项关键组成。流表用于存储转发规则，交换机对进入的数据包进行特征匹配，命中相应表项后执行对应动作，实现基于流的规则化转发；安全通道用于承载控制器与交换机之间的控制报文交互，支持规则下发、配置更新、状态查询与统计回传等操作，为控制平面与数据平面之间的可靠通信提供保障[27]。进一步地，控制器与交换机之间的通信可归纳为三类消息：Controller-to-Switch、Asynchronous 与 Symmetric。其中，Controller-to-Switch 消息由控制器发起，主要用于配置管理、能力查询以及流表项的下发与修改；Asynchronous 消息由交换机主动上报，用于反馈端口状态变化、异常事件与统计信息等运行状态；Symmetric 消息由双方对等发送，通常用于连接建立与维护、保活检测及错误通知。基于上述机制 OpenFlow 为 SDN 控制器对数据平面实施细粒度控制与动态调优提供了协议层面的支撑。

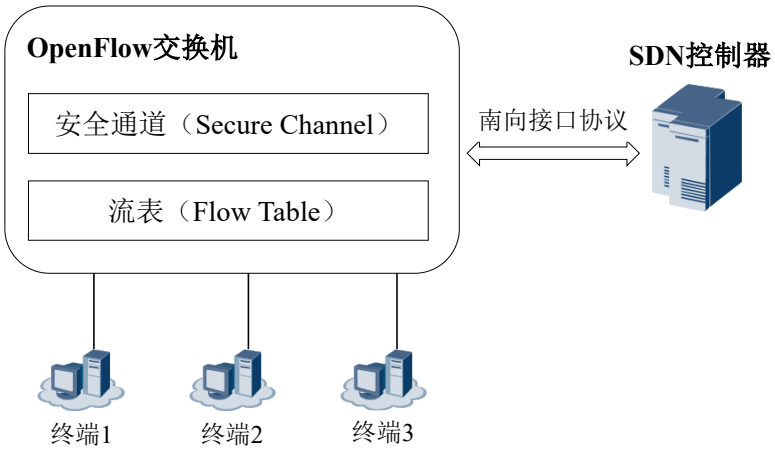


图 2-2 OpenFlow 协议架构

2.1.3 Ryu 控制器

当前主流 SDN 控制器类型较为丰富，既包括 OpenDaylight、ONOS 等工程化平台，也包括 Floodlight、POX、NOX 以及 Ryu 等轻量级开源控制器[44]，具体对比如表 2-1 所示。其中 Ryu 控制器是一款基于 Python 的组件化 SDN 控制框架，能够通过清晰的北向 API 快速构建网络控制应用，并对多版本 OpenFlow 协议提供良好支持。相比体量较大的工程化控制平台，Ryu 具有开发门槛低、部署轻量、原型迭代快、便于调试与二次开发等优势，更适合用于算法验证与实验评估。因此，本文后续实验选取 Ryu 作为控制平面实现与验证的平台，以支撑相关策略与优化算法的实现与对比分析。

表 2-1 常用 SDN 控制器的性能对比

控制器	语言	平台	描述
Ryu	Python	Linux	轻量级，易开发，兼容多版本 OpenFlow
OpenDaylight	Java	Win/Linux	模块化强，扩展性高，兼容多种南向协议
ONOS	Java	Linux	集群架构，高可用，面向大规模网络部署
Floodlight	Java	Win/Linux/Mac	跨平台，易扩展，稳定性较好
POX	Python	Linux	基于 NOX 扩展，上手快，适合快速原型开发
NOX	C++	Linux	体系较早，功能完整，部署成本较高

2.2 Louvain 算法

Louvain 算法是一种典型的社区发现方法，其通过迭代式的局部优化与层次聚合逐步生成多尺度的社区结构。该方法无需预设社区数量，并具有良好的可扩展性，因此常用于大规模复杂网络分析。为量化社区划分结果的质量，Louvain 算法引入模块度Q作为评价指标，并在迭代过程中不断提升该指标以获得更符合网络真实结构的社区划分[45]。模块度Q的计算公式如下：

$$Q = \frac{1}{2m}\sum_{i,j}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j) \tag{2-1}$$

其中，A_{ij}表示节点i与节点j之间的边权重，k_i与k_j分别为节点i、j的度（加权网络中对应加权度）；m为全网边权重总和，$\delta(c_i, c_j)$为指示函数，当i与j属于同一社区时取 1，否则为 0。A_{ij}描述真实网络连接，而$\frac{k_i k_j}{2m}$则反映在保持节点度分布不变的随机网络i与j的期望连接强度。若某一划分使得社区内部连接显著强于随机期望，则对应的Q值增大，表明社区更显著。

Louvain 算法的执行过程由“局部移动优化”与“社区聚合压缩”两阶段循环构成，并在每一层上反复迭代直至模块度不再提升。算法初始化时通常将每个节点视为一个独立社区，此时社区数目最多、粒度最细，为后续优化提供搜索起

点。在第一阶段（局部移动）中，算法逐个考察节点，将节点从其当前社区移动到其邻居节点所在的候选社区，并比较该移动所带来的模块度增益ΔQ，ΔQ的计算公式为：

$$\Delta Q = \frac{1}{2m}\left(k_{i,in} - \frac{\sum_{tot} k_i}{m}\right) \tag{2-2}$$

公式里的$k_{i,in}$为节点i与目标社区之间的权重，$\sum_{tot} k_i$表示目标社区所有连接的权重与节点i所有连接权重的乘积，当节点i迁入目标社区所带来的模块度增益$\Delta Q > 0$时，说明该迁移能够提升当前划分质量。因此，算法依次评估节点迁入各候选社区的ΔQ，若存在正增益方案，则将节点迁移至使ΔQ最大的社区，并重复迭代直至任一节点的单次迁移均无法进一步提高模块度为止。该阶段的关键在于ΔQ可由上述局部统计量快速求得，从而避免频繁全局重算式(2-1)，保证 Louvain 算法在大规模网络上的计算效率与可扩展性。

在第二阶段（社区聚合）中，算法将第一阶段得到的每个社区视为一个超级节点，把社区之间的边权重累加形成新的聚合网络；原社区内部的边权重则表现为该超级节点的自环权重。由于社区数量通常远小于原始节点数，聚合后的网络规模显著降低。随后在新的聚合网络上再次执行第一阶段的局部移动优化，从而形成优化到聚合到再优化的层次化迭代过程。随着层数上升，网络逐步压缩，社区结构从细粒度向粗粒度演化，最终得到多层次的社区划分结果。

2.3 萤火虫算法

萤火虫算法（Firefly Algorithm, FA）是一种典型的生物拟态群体智能优化方法，其核心机理模拟了夏夜萤火虫通过光信号传递信息、协同觅食的群体行为特征[46]。FA 将候选解视为萤火虫个体，通过亮度比较驱动的吸引移动机制引导种群在搜索空间中迭代演化，从而逐步逼近全局最优解。

在 FA 中，萤火虫个体的亮度用于衡量解的优劣，通常由目标函数（适应度函数）决定，并与其单调相关。以最小化问题为例，可将亮度设置为随目标函数值降低而增强，使得目标函数值更优的个体具有更强的吸引能力。基于该规则，亮度较弱的萤火虫会被亮度更强的萤火虫吸引，并向其位置靠近，以实现群体协同搜索。萤火虫之间的吸引度与个体间距离相关，常用指数衰减模型描述其随距离增大而减弱的规律：

$$\beta(d_{ij}) = \beta_0 e^{-\gamma d_{ij}^2} \tag{2-3}$$

其中，d_{ij}为萤火虫i和j之间的欧几里得距离，β_0表示当d_{ij}为 0 时的最大吸引度，γ是吸引度的衰减因子，用于控制吸引力随距离增长的衰减速度与算法收敛特性。距离d_{ij}一般定义为：

$$d_{ij} = \sqrt{\sum_{k=1}^{D}\left(x_{i,k} - x_{j,k}\right)^2} \tag{2-4}$$

其中，D是问题的维度，$x_{i,k}$和$x_{j,k}$分别表示萤火虫i和j在第k个维度的坐标。在迭代更新过程中，当萤火虫j的亮度高于萤火虫i时，萤火虫i将向j移动，其位置更新可表示为：

$$x_i^{t+1} = x_i^t + \beta(d_{ij})\left(x_j^t - x_i^t\right) + \psi S_k \odot (rand - 0.5) \tag{2-5}$$

其中，x_i^t和x_j^t分别为第t次迭代时萤火虫i与j的位置向量，ψ是随机扰动系数，用于控制解的多样性，S_k是第k堆的缩放因子，用于使各维度的扰动幅度与量纲相匹配，$rand$表示 0 到 1 范围内的均匀随机数。

萤火虫算法的运行流程如图 2-3 所示，其通过“亮度评价确定优劣、距离衰减吸引引导移动、随机扰动保持多样性”的协同机制，在迭代搜索中实现全局探索与局部开发的平衡，从而能够有效逼近优化问题的较优解。

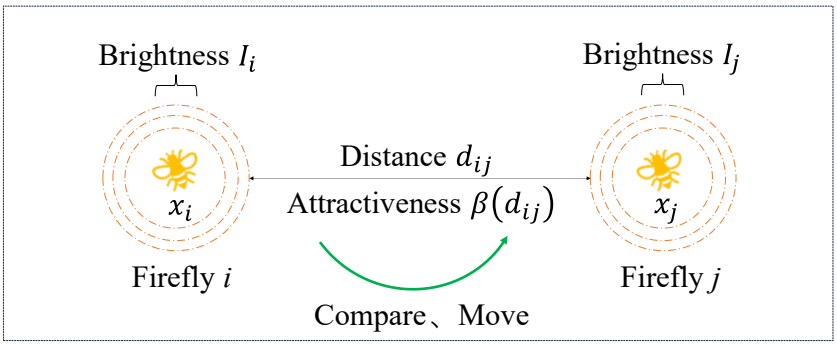


图 2-3 萤火虫算法的运行流程

2.4 加权主成分分析法

加权主成分分析法（WPCA）是在主成分分析（PCA）基础上引入权重机制的降维方法。PCA 的核心是通过样本的二阶统计结构（协方差）提取主要变化方向，但其默认各样本对统计量的贡献一致。当数据中不同样本在可靠性、代表性或重要性上存在差异时，等权处理会使协方差结构被平均化，从而主成分方向可能更多反映整体的平均变化，而对关键样本的结构特征刻画不足。WPCA 的基本思路是将这种差异以权重的形式显式引入统计建模过程，使主成分提取过程对高权重样本更敏感[47]。

假设数据集包含n个样本，每个样本为$x_i \in \mathbb{R}^p$，构成数据矩阵$X \in \mathbb{R}^{n\times p}$。对数据进行中心化处理后，引入样本权重$\omega_i > 0$。加权均值可定义为：

$$\mu_\omega = \frac{\sum_{i=1}^{n}\omega_i x_i}{\sum_{i=1}^{n}\omega_i} \tag{2-6}$$

在此基础上，WPCA 采用加权协方差矩阵描述数据的离散结构：

$$S_\omega = \frac{1}{\sum_{i=1}^{n}\omega_i}\sum_{i=1}^{n}\omega_i(x_i-\mu_\omega)(x_i-\mu_\omega)^{\mathrm{T}} \tag{2-7}$$

与 PCA 通过对等权协方差矩阵进行特征分解不同，WPCA 对S_ω进行特征分解以提取主成分方向：

$$S_\omega v_k = \lambda_k v_k,\ k = 1, 2, \dots, p \tag{2-8}$$

其中，v_k为第k个主成分方向，λ_k为对应的特征值，反映该方向上数据的加权方差贡献。通常将特征值按$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$排序，并选取前$r$个主成分构成投影矩阵$V_r = [v_1, \dots, v_r]$。样本$x_i$在低维空间中的表示（主成分得分）为：

$$z_i = V_r^{\mathrm{T}}(x_i - \mu_\omega) \tag{2-9}$$

主成分数r的确定一般依据累计解释方差比：

$$\eta(r) = \frac{\sum_{k=1}^{r}\lambda_k}{\sum_{k=1}^{p}\lambda_k} \tag{2-10}$$

当$\eta(r)$达到预设阈值时，可认为前r个主成分已保留数据的主要信息，从而实现有效降维。从优化角度来看，WPCA 等价于在正交约束下最小化加权重构误差，即通过最小化高权重样本的重构偏差来确定低维子空间：

$$\min_{V_r^{\mathrm{T}}V_r}\sum_{i=1}^{n}\omega_i\left\|(x_i-\mu_\omega) - V_rV_r^{\mathrm{T}}(x_i-\mu_\omega)\right\|_2^2 \tag{2-11}$$

该表达式清楚表明权重的作用方式：权重越大的样本，其重构误差在目标函数中的占比越高，因此学习到的低维子空间会优先保证对这些样本的解释能力。由此带来的直接结果是，WPCA 在保持 PCA 正交投影与可解释性框架不变的前提下，使主成分方向对关键样本的统计结构更敏感，并在样本质量不均或关注重点明确的场景中更容易得到符合任务需求的低维表示，因而常用于多指标综合评分、排序评价与分层划分等场景。

2.5 k 近邻方法

k 近邻（k Nearest Neighbors, kNN）是一类基于距离度量的近邻建模方法，其基本思想是在给定特征空间中，为每个对象保留与其距离最近的k个邻居[48]，从而刻画节点周围的局部几何结构。设对象集合为$V = \{v_1, \dots, v_n\}$，并定义任意两点之间的距离函数$d(v_i, v_j)$，则节点v_i的 k 近邻集合可表示为距离v_i最近的k个节点构成的集合，记为$\mathcal{N}_k(v_i)$：

$$\mathcal{N}_k(v_i) = \arg\min_{\substack{S\subseteq V\setminus\{v_i\}\\ |S|=k}}\sum_{v_j\in S} d(v_i, v_j) \tag{2-12}$$

若采用欧几里得距离，在嵌入坐标$p_i = (x_i, y_i)$下，节点v_i与v_j之间的距离可写为：

$$d(v_i, v_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2-13}$$

在此基础上，可进一步构建 kNN 图：对每个节点v_i，将其与$\mathcal{N}_k(v_i)$中的节点建立连边，即可得到一张稀疏邻接图。由于每个节点至多引出k条近邻边，图的边规模可由全连接情形下的$O(n^2)$降低至$O(nk)$。因此，kNN 图能够在保留局部邻域关系的同时，有效压缩图结构规模，为后续的邻接筛选与候选路径构建提供更紧凑的表示基础。

2.6 本章小结

本章围绕本文研究所需的关键理论基础展开阐述。首先介绍 SDN 的相关技术体系，包括 SDN 的分层架构、OpenFlow 协议的南向通信机制以及 Ryu 控制器的基本框架与开发特性，为后续实验平台选取与控制逻辑实现提供支撑。随后，针对动态子网划分与结构建模需求，系统说明 Louvain 算法的模块度优化思想及其两阶段迭代过程。进一步地，本章对萤火虫算法的吸引度机制与随机扰动搜索策略进行了概述，为后续组合优化问题的求解方法奠定理论依据。在此基础上，引入加权主成分分析法与 k 近邻方法，分别用于多指标综合评分与局部邻域关系建模，以支撑后续特征融合与候选结构构造。通过上述内容的铺垫，本章为后续章节的模型设计、算法实现与实验验证提供了必要的理论与技术支撑。

3 动态子网划分与网关部署协同框架

3.1 引言

由第一章相关理论与现状分析可知，数据中心网络在规模扩展与业务动态叠加下，路由决策需在保障端到端 QoS 的同时兼顾负载分布与能耗开销等运行目标。与此同时，端到端延迟除链路传播外还受设备排队等因素影响，拥塞波动亦会进一步放大 QoS 的不确定性。因此，如何在保证在线可用性的前提下实现多维目标的协同权衡，成为当前路由优化亟待解决的关键问题。

现有研究主要存在两方面不足。其一，全网尺度的多约束联合求解虽然能够统一刻画 QoS 与资源约束，但随着网络规模扩大，候选路径空间与状态同步开销快速膨胀，导致计算复杂度过高、决策时效性不足，难以支撑动态场景下的频繁请求到达。其二，结构层方法虽试图通过分域降低求解复杂度，但在跨层协同方面仍显不足。一方面，部分子网划分方案更侧重拓扑结构或流量分布特征，未将能耗因素有效纳入划分准则，对设备排队延迟、协议处理延迟等关键性能来源的刻画也不够充分，因而难以保证划分结果与实际 QoS 瓶颈相匹配。另一方面，网关部署通常采用静态配置或独立优化方式，缺乏与子网划分结果相协调的动态联动机制。当跨域通信比例上升或负载快速波动时，边界转发节点容易产生流量汇聚，进而引发排队积压以及 QoS 尾部性能恶化。由此可见，需要从结构层面对网络进行合理组织与压缩，在同时考虑 QoS 保障、负载均衡与能耗控制的基础上将子网划分与网关部署纳入统一的协同闭环，以此构建稳定可控的跨域转发骨架，进而降低问题规模并抑制瓶颈扩散。

为此，本文提出基于动态子网划分与网关部署协同的路由优化框架，称为 ROCDSG。该框架在流量、负载与能耗等约束驱动下进行动态子网划分以缩小搜索空间，并与弹性网关部署同步联动以缓解跨域汇聚瓶颈，最终在子网与网关骨架上构建延迟感知的分层路由生成可下发路径。

3.2 网络体系架构

本文提出的 ROCDSG 采用控制平面与数据平面解耦设计，通过多模块协同实现异构流量的智能化路由。整体网络架构如图 3-1 所示，架构主体由边缘接入层、可编程数据平面、集中式控制平面及应用层构成。

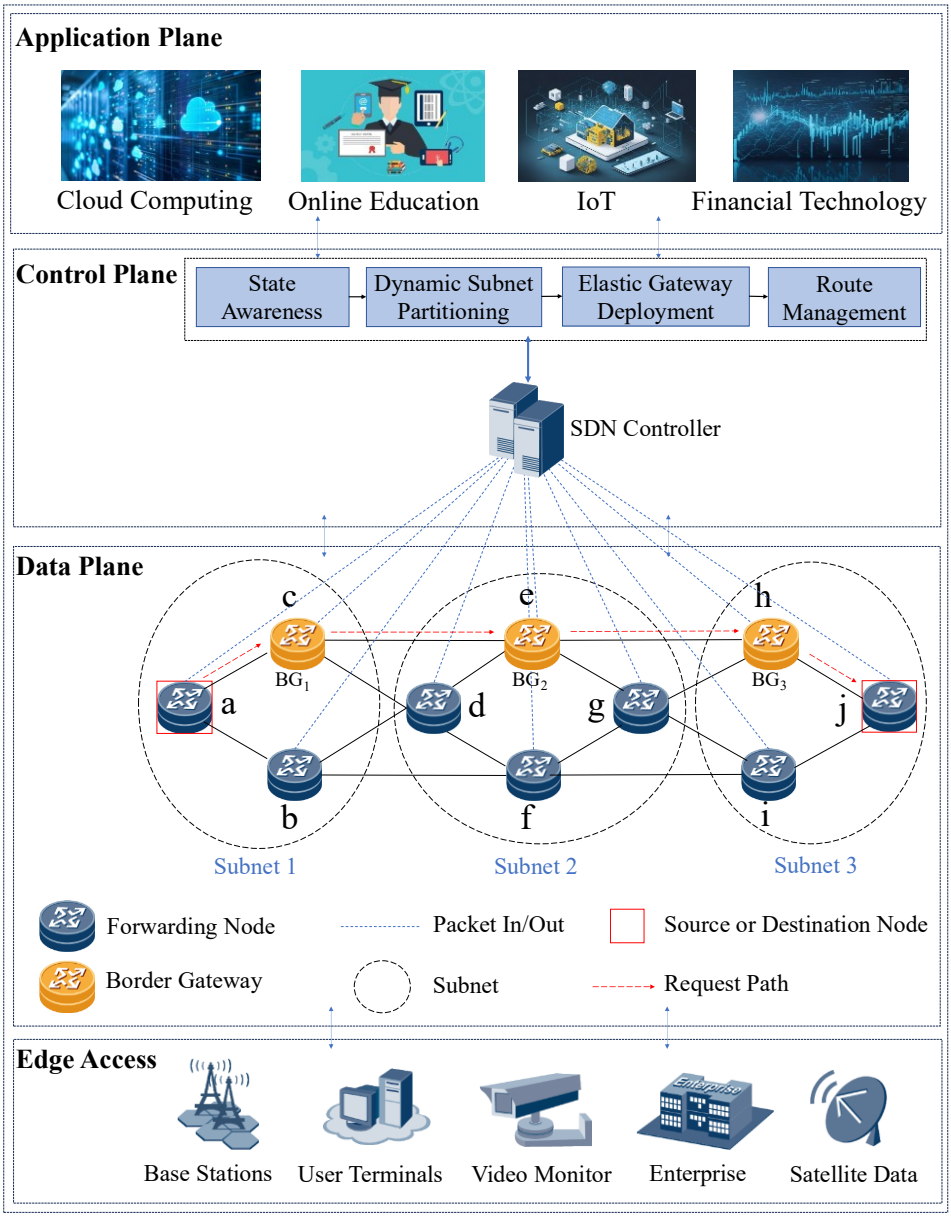


图 3-1 整体网络架构

在接入层方面，基站、用户终端、视频监控设备、企业机构及卫星地面站等多元数据源通过标准化数据传输接口将原始流量汇聚至底层 SDN 交换机；数据平面由支持可编程功能的智能交换机组成，主要包括流表匹配、数据包封装解封装，以及网络遥测数据采集等关键功能。控制平面主要部署状态感知、动态子网划分、弹性网关部署及路由管理四大核心模块，其中状态感知模块通过持续监听 Packet-In 消息与主动 LLDP 包探测，实时获取网络数据并构建带权拓扑图。动态子网划分模块基于 Louvain 算法，以流量、负载和能耗等约束为目标生成逻辑隔离的虚拟子网。本文中的子网并非按 IP 前缀聚合的一组终端主机，而是在 SDN 控制器的全局视图下，通过上述多维约束动态聚合形成的逻辑转发域，其成员为转发设备（SDN 交换机或路由节点，包含候选网关），该逻辑子网用于降低路由求解规模并实现分层调度；主机与 IP 前缀在本工作中被抽象化处理（不作为子

网划分依据）。为确保跨子网连通性，本文基线设定为每个子网恰好部署 1 个边界网关（Border Gateway, BG），用于跨子网中继。该设定由弹性网关部署模块实现：该模块根据子网划分结果，为每个子网动态实例化 BG，并基于 SOFA 算法优化其物理映射位置，以最小化跨域跳数。路由管理模块最终依据全局视图生成跨子网最优路径策略，经 Packet-Out 消息下发至数据平面。以图中 $a \rightarrow j$ 数据流为例，该架构首先识别源/目的节点所属子网（子网 1 与子网 3），在子网 1 内基于延迟感知算法选取 $a \rightarrow c$ 路径至边界网关 BG_1，跨子网阶段获取 BG_1 至 BG_3 的最短路径 $c \rightarrow e \rightarrow h$，最后在子网 3 内路由至目的节点 j，最终形成整体路径 a-c-e-h-j，以此降低路由复杂度。该架构通过子网自治和跨域协同的双层优化机制，能够有效支撑云计算、在线教育、工业物联网等场景中高并发、低延迟、差异化的服务质量需求。

3.3 网络模型定义

本文使用连通图$G(V, E)$来表示 ROCDSG 中的网络组成。其中，V代表节点集合，$E \subseteq V \times V$代表边集合。具体而言，每个节点$v_i \in V$表示一个设备（即 SDN 交换机或网关），而每条边$e_{ij} \in E$表示设备之间的通信链路，例如，e_{ab}是顶点a和b之间的连接（即$e_{ab} \in E$，其中$a, b \in V$且$a \neq b$）。从图 3-1 中可以看到，$V = \{a, b, c, d, e, f, g, h, i, j\}$，$E = \{e_{ab}, e_{ac}, e_{cd}, e_{bd}, \ldots\}$。本文假设，图中每次通信的源节点和目的节点都是唯一的，并且 SDN 交换机或网关都是具有相同功能的同构设备。

在支持 SDN 的数据中心网络中，数据流首先被发送至 SDN 交换机，随后由 SDN 控制器根据全局视图计算路径，并将数据流从源节点路由至目标节点。假设一条数据流用一个元组$F_k = \{s_k, d_k, p_k, a_k\}$表示，其中$s_k$和$d_k$分别表示该流的源节点和目的节点，且$s_k, d_k \in V$（$V$表示网络中所有节点的集合），$p_k$表示该流的数据包大小，$a_k$表示该流的到达时间。本章使用的主要符号及其含义列于表 3-1。

表 3-1 主要符号汇总

符号	说明
V	网络节点集合
E	网络边集合
v_i	网络设备节点
e_{ij}	节点间链路
F_k	数据流元组
G_b	边界网关集合
$f_{v_i}^{(T)}$	节点v_i的总流量
M_i	第i个子网的节点集合

表 3-1 主要符号汇总（续）

符号	说明
C	系统总容量
$L(M_i)$	子网M_i的总负载量
$d_{avg}(v_i)$	节点v_i的平均延迟
$d_{max}(v_i)$	节点v_i的最大延迟
$d_{avg}(v, g_i)$	各节点到边界网关g_i的平均延迟
$d_{max}(v, g_i)$	各节点到边界网关g_i的最大延迟
$d_{avg}(g_i, g_j)$	网关间的平均延迟
$d_{max}(g_i, g_j)$	网关间的最大延迟
τ_{intra}	子网内节点到边界网关的最大延迟上限
τ_{inter}	网关间的最大延迟上限
τ_{device}	节点的最大延迟上限
E_i^T	子网M_i内的总能耗
y_{ik}	网关选择变量
$z_{v,g}$	网关关联变量
$x_{i,j}^f$	链路使用变量
r_f	业务流规模
C_{ij}	链路带宽容量
E_{max}	网关的最大允许能耗上限
E_{g_i}	网关部署的实际能耗
L	网关部署的负载差异

3.4 问题描述

在数据中心网络中，本文研究如何在兼顾服务质量保障、负载均衡、能耗控制与路由效率的前提下，实现复杂场景下的高效路由决策。考虑到全网尺度联合优化在大规模动态环境中存在计算复杂度高、时效性不足等问题，本文从结构降维出发，关注如何通过动态子网划分对网络进行合理组织与压缩，进而在此基础上开展网关部署设计，以支撑跨子网通信并缓解边界汇聚瓶颈，最终协调子网内路由与跨子网流量转发，从而在降低求解开销的同时提升端到端传输性能与整体运行效率。

3.5 动态子网划分建模与求解

本节围绕动态子网划分的建模与求解展开。首先结合业务叠加场景，阐明动态分域的必要性与作用。随后构建划分目标与约束体系，并统一核心符号与变量定义。最后基于上述模型给出动态划分的求解流程与算法，为后续网关部署与分

层路由构建结构骨架基础。

3.5.1 约束条件建模

传统的静态子网划分方法依赖于固定的拓扑结构和预设的流量模型，在早期相对稳定的网络环境中具有一定的有效性。然而，随着数据中心业务需求的多样化与动态变化，这种方法存在高负载、高延迟、高能耗、资源利用率低下等问题。因此，需要引入一种能够动态适配业务需求的子网划分策略，并充分考虑负载均衡、能耗开销与通信延迟等多重约束。

子网划分应让各个子网的流量总和尽可能均衡。假设边界网关集合为$G_b = \{g_1, g_2, \dots g_k\}$，$f_{v_i}^{(in)}$表示节点$v_i$在子网内部处理的流量，$f_{v_i}^{(s)}$表示节点$v_i$向边界网关$g_i$发送的跨子网流量，$f_{v_i}^{(r)}$表示节点$v_i$从边界网关$g_i$接收的跨子网流量，那么节点$v_i$上的总流量$f_{v_i}^{(T)}$为：

$$f_{v_i}^{(T)} = f_{v_i}^{(in)} + f_{v_i}^{(s)} + f_{v_i}^{(r)} \tag{3-1}$$

整个子网划分需满足：

$$\left| \sum_{v_i \in M_i} f_{v_i}^{(T)} - \sum_{v_j \in M_j} f_{v_j}^{(T)} \right| \le \varepsilon^f \tag{3-2}$$

$$\sum_{i=1}^{K} \sum_{v_i \in M_i} f_{v_i}^{(T)} \le \alpha C \tag{3-3}$$

其中M_i=$\{M_1$,M_2,…,$M_K\}$表示第i个子网内的节点集合，ε^f表示流量均衡因子，α是安全裕度因子，表示系统容量 C 的最大允许使用比例（其中$0 < \alpha < 1$）。在网络中，每条链路都有其承载的负载量，表示为$l_{(v_i,v_j)}$，每个节点有当前需要处理的负载量$c(v_i)$，负载量是通过链路或节点的带宽利用率来衡量。假设在时间步t内，链路和节点会有即将到来的负载量$l_{(v_i,v_j)}{}'$和$c(v_i)'$，那么整个子网的负载量$L(M_i)$为：

$$L(M_i) = \sum_{v_i \in M_i} (c(v_i) + c(v_i)') + \sum_{v_i, v_j \in M_i, v_i \neq v_j} \left(l_{(v_i,v_j)} + l_{(v_i,v_j)}{}' \right) \tag{3-4}$$

衡量子网间的负载平衡约束定义为：

$$\left| L(M_i) - L(M_j) \right| \le \varepsilon^l \tag{3-5}$$

其中，ε^l表示负载平衡因子，ε^l越小表示子网间的负载越平衡，反之亦然。由于每个子网内都会部署一个边界网关，同时每个节点本身也会有处理延迟和排队延迟。因此，本文考虑的延迟有三类：节点延迟、节点到边界网关延迟、网关间延迟。节点延迟通常与节点的负载和处理能力相关，本文定义其平均延迟$d_{avg}(v_i)$和最大延迟$d_{max}(v_i)$为：

$$d_{avg}(v_i) = \frac{1}{|M_i|}\sum_{v_i \in M_i}\frac{c(v_i)}{\mu(v_i)} \tag{3-6}$$

$$d_{max}(v_i) = max\left\{\frac{c(v_i)}{\mu(v_i)}|v_i \in M_i\right\} \tag{3-7}$$

$\mu(v_i)$表示节点v_i的处理能力，即单位时间内可处理的数据量。本文定义子网内节点到边界网关的平均延迟$d_{avg}(v, g_i)$和最大延迟$d_{max}(v, g_i)$为：

$$d_{avg}(v, g_i) = \frac{1}{|M_i|}\cdot\sum_{v_i \in M_i} d(v_i, g_i) \tag{3-8}$$

$$d_{max}(v, g_i) = max\{d(v_i, g_i)\,|v_i \in M_i, g_i \in G_b\} \tag{3-9}$$

$d(v_i, g_i)$表示节点v_i到边界网关g_i的链路延迟。为保持符号表述的清晰性，本文将v_i定义为属于第i个子网的节点。本文定义网关间的平均延迟$d_{avg}(g_i, g_j)$和最大延迟$d_{max}(g_i, g_j)$为：

$$d_{avg}(g_i, g_j) = \frac{1}{K(K-1)}\cdot\sum_{i=1}^{K}\sum_{j=1, j\neq i}^{K} d(g_i, g_j) \tag{3-10}$$

$$d_{max}(g_i, g_j) = max\{d(g_i, g_j)\,|g_i, g_j \in G_b, i \neq j\} \tag{3-11}$$

其中，$d(g_i, g_j)$表示子网M_i和子网M_j的边界网关g_i和g_j之间的通信延迟。为了确保网络系统的高效运行和服务质量，应考虑子网内节点到边界网关的最大延迟以及网关间的最大延迟影响。因此，本文定义了子网内节点到边界网关的最大延迟以及网关间的最大延迟不超过一定阈值的约束条件，该约束用于后续子网划分的重触发判定，并作为网关部署模块的建模输入，其具体公式如下：

$$d_{max}(v, g_i) \leq \tau_{intra} \tag{3-12}$$

$$d_{max}(g_i, g_j) \leq \tau_{inter} \tag{3-13}$$

当请求流使用路径时，每个节点（即交换机或者网关）和链路在启用时都会产生一定的能量消耗，同时数据包在传输过程中，字节的封装和解封装也会消耗一定的能量。因此，子网内的能量消耗可定义为：

$$E_i^T = E_s + \sum_{i \in M_i}(k * E_t(v_i)) \tag{3-14}$$

其中，E_i^T表示子网M_i内的总能耗，E_s表示子网M_i内节点和链路的能耗（包括节点和链路的启动能耗，以及其他的静态能耗，例如处理、感知能耗），$k * E_t(v_i)$表示节点v_i传输 k 个字节的数据包时消耗的能量。子网间的能量消耗需满足：

$$\left|E_i^T - E_j^T\right| \leq \varepsilon^e \quad \forall i, j \in \{1, 2, \ldots, K\}, i \neq j \tag{3-15}$$

其中，ε^e表示能量平衡约束因子，其值越小越优，这有助于减少因能量不平衡导致的局部过载或欠载情况，从而降低系统故障和链路拥塞的风险。同时，对于任意子网M_i，其对应的子图$G_{M_i} = (V_{M_i}, E_{M_i})$，其中$V_{M_i} = M_i \cap V$，且$E_{M_i} = \{(u, v) \in E\,|u, v \in V_{M_i}\}$应满足连通性要求，即子网内的各节点必须处于一个连通的状态，避免孤岛现象的出现。

3.5.2 基于约束的改进 Louvain 算法

为了实现动态网络环境中的子网划分，本文将流量、负载和能耗等多维度约束引入到 Louvain 算法中。由第二章相关理论可知，Louvain 算法以模块度最大化为优化目标，通过迭代的节点局部迁移与社区层次聚合逐步提升模块度，从而获得能够反映网络连接结构的社区划分。

本文改进的 Louvain 算法在传统算法的基础上，不仅以模块度为目标优化子网划分，还结合各子网的流量均衡、节点和链路负载分布及能耗等进行约束调整。在算法执行过程中，首先初始化每个节点为独立社区，随后迭代进行节点移动与合并操作，并在每次迭代中引入约束条件对划分进行调节，最终形成满足多重约束的子网划分方案。

图 3-2 展示了如何将七个节点的网络划分为两个子网的示例。由(1)中节点与边的连接关系及权重可知网络的总权重m为 41。刚开始，本文将每个节点视为一个独立的社区。然后，依次尝试将每个节点移动到其邻居所在的社区，并计算模块度增量，模块度增量$\Delta Q > 0$时，表示节点可以移动到目标社区。以子图（1）为例，图中节点a为节点b和c的邻居节点。本文考虑将节点a移动到节点b所在的社区，此时$k_{i,in} = 8$，$\sum_{tot} k_i = 11*13$，那么其模块度增量为$\Delta Q =$（8-11*13/41）/82≈0.055。由于$\Delta Q > 0$，因此可以将节点a和节点b合并为一个社区，形成初步子网。随后，算法对其余节点重复进行模块度增益评估与节点迁移，直至全网不存在能够带来正增益的单节点移动，局部优化过程达到收敛。之后，将当前得到的社区划分进行聚合压缩，将每个社区视为一个超级节点，社区间边权按累加规则合并，社区内部边权以自环形式保留，构建新的加权图；并在该聚合图上再次执行同样的局部移动优化，迭代进行直至模块度不再提升。最终，检查所有子网是否满足式(3-2)、(3-3)、(3-5)、(3-15)以及连通性约束的定义，若全部满足，则完成最终子网划分。需要说明的是，受网络状态与约束阈值设置影响，基于约束的 Louvain 子网划分在个别场景下可能出现不可行情况，此时算法将返回“不可行”标志以交由系统层处理。

在系统实现层面，本文为子网划分设定控制周期。每个控制周期开始时，控制器依据全网实时状态运行改进的Louvain算法，生成当前周期的逻辑子网划分，并将其作为随后的网关部署与路由优化的输入参数。在当前周期内，各子网成员保持不变；仅当监测到网络状态出现显著偏离，即任一关键约束被违反（式(3-2)、(3-3)、(3-5)、(3-12)、(3-13)以及(3-15)所示），控制器才在下一控制周期边界触发子网重划分，从而避免周期内的结构震荡。若某一控制周期内的子网划分结果被判定为不可行，则系统沿用上一控制周期的可行子网划分作为回退方案；若连续多个控制周期仍判定不可行，则触发预设的保底策略（适度放宽相关约束阈值），以避免长期回退导致的系统停滞。

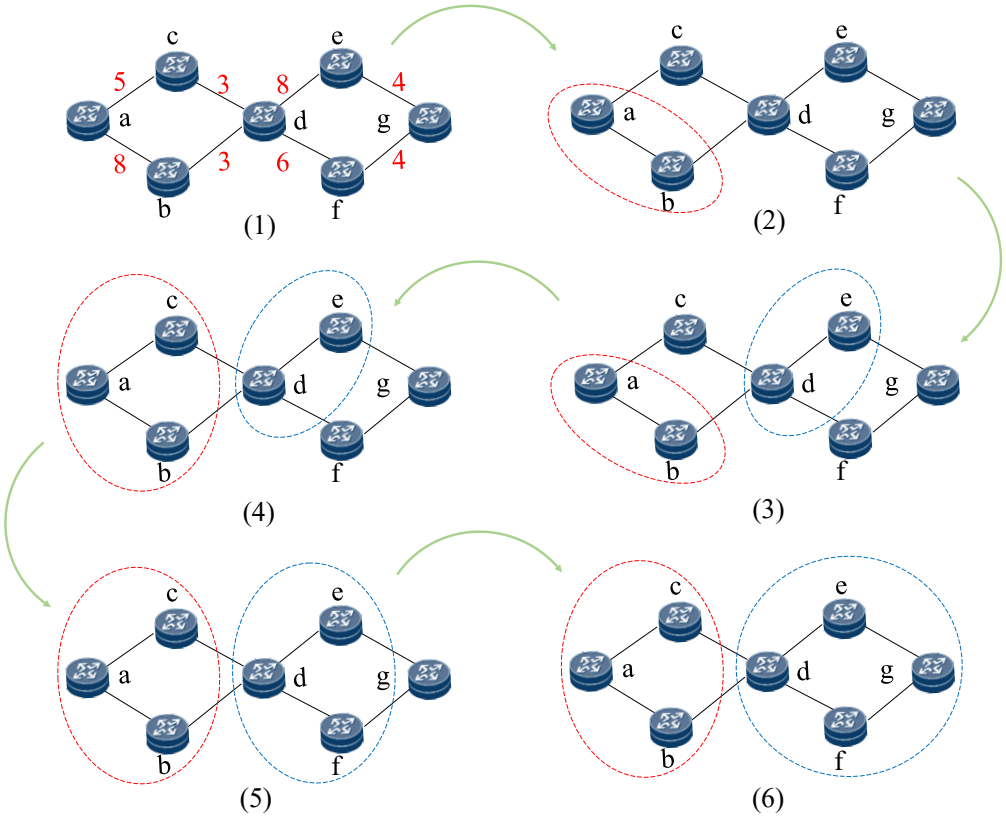


图 3-2 动态子网划分过程

算法 1 给出了基于约束的改进 Louvain 算法的执行流程。算法首先运行 Louvain 方法对网络图进行划分，得到一个候选的子网划分结果（见第 4–9 行）。随后进入迭代重试过程：对于每一个候选子网，算法计算其总流量、容量、负载与能耗等统计量（见第 11–13 行），并检查该子网是否违反预先设定的多维约束条件，包括容量上限、负载均衡、能耗平衡以及连通性要求（见第 14–19 行）。若存在任一子网不满足上述约束，则丢弃本轮迭代的划分结果，并在下一轮迭代中重新运行 Louvain 方法生成新的社区划分（见第 16 行和第 23 行；下一轮中的第 7–9 行）。该过程持续进行，直至满足以下终止条件之一：若所有子网均满足约束，则返回最终的子网划分结果（见第 20–22 行）；否则，若在达到最大迭代次数I后仍未找到可行划分，则返回“未找到可行的划分结果”。

算法 1：基于约束的改进 Louvain 算法

输入：包含交换机节点与链路边的网络图$G(V,E)$，参数ε^f、α、λ和ε^e，最大重试次数I
输出：划分结果 P

1: $P \leftarrow \emptyset$
2: $iter \leftarrow 0$
3: while $iter < I$ do
4: 　　if P 为空 then
5: 　　　　$P \leftarrow$在G上通过 Louvain 算法进行社区划分

```
    else
        P←在G上重新运行 Louvain 算法
    end
    valid←True
    for 每个社区S ∈ P do
        计算总流量 f(S)、总容量 C(S)、总负载 L(S)以及总能耗 E(S)
    end
    for 每个社区S ∈ P do
        if S 违反式(3-2)、(3-3)、(3-5)、(3-15)或连通性约束 then
            valid←False
            break
        end
    end
    if valid then
        return P
    end
    iter←iter+1
end
return “未找到可行的划分结果”
```

在算法 1 中，Louvain 算法本身的时间复杂度为$O(m)$，其中m表示网络中的边数。每一轮约束检查的时间复杂度为$O(n)$，最大社区重组重试次数为I。因此，算法 1 的总体时间复杂度为$O\left(I \cdot (m+n)\right)$。

3.6 弹性网关部署建模与求解

本节围绕弹性网关部署的建模与求解展开。在动态子网划分结果的基础上，首先给出网关部署的优化模型，明确决策变量、目标函数与约束条件，并将延迟、负载与能耗等因素统一纳入建模框架。随后给出相应的求解流程与算法，用于获得满足约束的网关部署方案，为后续分层路由提供可用的网关骨架基础。

3.6.1 部署目标建模

为了在多子网结构下兼顾路由分层的效率与模型的可解性，本文在建模中对每个子网仅选择一个边界网关作为跨子网通信节点。为统一表征网关选择、节点绑定及路径占用关系，本文建立如下二元决策变量体系。设每个子网内候选节点

$i \in M_k$均可作为潜在的网关。

定义网关选择变量$y_{ik} \in \{0,1\}$，当节点i被选为子网k的边界网关时，$y_{ik} = 1$，否则为 0：

$$y_{ik} = \begin{cases} 1, & \text{若节点}i\text{被选为子网}k\text{的边界网关} \\ 0, & \text{否则} \end{cases} \tag{3-16}$$

节点绑定关系由变量$z_{v,g} \in \{0,1\}$表示，若节点v的转发或汇聚流量由边界网关g处理，则$z_{v,g}$=1：

$$z_{v,g} = \begin{cases} 1, & \text{若节点}v\text{关联到边界网关}g \\ 0, & \text{否则} \end{cases} \tag{3-17}$$

为刻画请求流在链路层的转发行为，引入路径占用变量$x_{i,j}^{f} \in \{0,1\}$，当请求流f经过链路(i,j)传输时，$x_{i,j}^{f}$=1，否则为 0：

$$x_{i,j}^{f} = \begin{cases} 1, & \text{若请求流}f\text{经过链路}(i,j) \\ 0, & \text{否则} \end{cases} \tag{3-18}$$

为进一步确定每个子网最优的边界网关选址方案，需要在延迟、负载与能耗之间实现平衡，确保所选网关位置既能满足性能需求，又能避免潜在的单点瓶颈。由此，网关选址问题可归结为一个多约束优化模型，其目标与约束共同刻画了 QoS、网络效率与能耗之间的权衡关系。

直观而言，该模型旨在最小化子网内节点到边界网关的通信延迟以及网关之间的跨网传输延迟，同时降低边界网关的部署代价，从而反映控制器在动态子网结构下的全局决策逻辑。第一个优化目标聚焦于延迟最小化：

$$Minimize\left(\sum_{k=1}^{3}\frac{\omega_k * d_{avg}(k)}{d_{max}(k)}\right) \tag{3-19}$$

其中，$d_{avg}(k)$表示各维度的平均延迟，$k \in \{1,2,3\}$分别对应网关延迟（即节点延迟）、节点到网关的链路延迟以及网关间的延迟，$d_{max}(k)$表示各维度的最大延迟，ω_k是对应延迟项的权重。第二个优化目标最小化网关部署总能耗并实现负载均衡：

$$Minimize\left(\omega_4 * E_{g_i} + \omega_5 * L\right) \tag{3-20}$$

其中，E_{g_i}表示网关部署的实际能耗，L表示网关部署的负载差异，ω_4和ω_5分别为实际能耗和负载差异的权重。具体而言，E_{g_i}和L的定义如下：

$$E_{g_i} = E_s(g_i) + E_t(g_i) \tag{3-21}$$

$$L = \sum_{g_i \in G_b}\left(L_{g_i} - \frac{\sum_{g_i \in G_b} L_{g_i}}{|G_b|}\right)^2 \tag{3-22}$$

其中，$E_s(g_i)$表示边界网关的静态能耗，$E_t(g_i)$表示边界网关的传输能耗，L_{g_i}表示边界网关的实际负载量，$|G_b|$表示边界网关集合的元素数量，$\sum_{g_i \in G_b} L_{g_i}$表示所有边界网关集合的负载量。在上述两个目标的基础上，本文提出了一个联合优化公式，如下所示：

$$Minimize\ Z = \left(\sum_{k=1}^{3} \frac{\omega_k * d_{avg}(k)}{d_{max}(k)} + \left(\omega_4 * E_{g_i} + \omega_5 * L \right) \right) \tag{3-23}$$

这个优化公式受制于以下约束条件：

s.t. 式(3-12) – 式(3-13)

$$\sum_{k=1}^{3} \omega_k + \omega_4 + \omega_5 = 1 \tag{3-24}$$

$$d_{max}(v_i) \le \tau_{device} \tag{3-25}$$

$$\sum_{f \in F} r_f \, x_{ij}^{f} \le C_{ij}, \ \forall (i,j) \in E \tag{3-26}$$

$$\frac{L_{g_i}}{L(M_i)} \le \delta \ \forall i \in M_i, g_i \in G_b \tag{3-27}$$

$$\sum\nolimits_{i \in M_k} y_{ik} = 1, \forall k \tag{3-28}$$

$$E_{g_i} \le E_{max} \tag{3-29}$$

$$P_{uv}^{ij} \ge 1, \forall u \in M_i, v \in M_j, i \ne j \tag{3-30}$$

$$\forall k \in P(i \to j), k \notin M_i \tag{3-31}$$

$$\sum_{g \in G_b} z_{v,g} = 1 \, , z_{v,g} \le y_{gk}, \ \forall v \in M_k, g \in G_b \tag{3-32}$$

公式(3-24)用于在各项指标之间分配不同的权重，且这些权重的总和应为1。若仅将权重ω_4设置为1，则目标函数将仅聚焦于最小化网关部署的能耗，而忽略其他指标的影响。因此，必须在延迟、能耗和负载等多个因素之间进行合理的权衡，以确保优化方案能够综合考虑这些关键因素。公式(3-25)对网关本身的最大延迟、节点到网关的最大延迟以及网关间的最大延迟设定了上限，确保子网内与子网间通信的延迟控制在合理范围内。公式(3-26)控制跨域流量的总量不超过链路带宽上限，其中r_f为流量大小，C_{ij}为链路带宽限制。公式(3-27)限制了各网关的负载占比，防止个别网关因负载过重而成为性能瓶颈。公式(3-28)确保每个子网仅允许选择一个边界网关，保证网络的分层独立性与结构约束。公式(3-29)则规定了边界网关的能耗不能超过其所承受的最大能耗，其中E_{max}为所有网关的统一能耗上限，以避免因能耗过大而导致的设备故障或不可用情况。为了进一步提升网络的可靠性和连通性，公式(3-30)规定了每个边界网关至少能够通过一条有效路径与其他子网进行连接，其中P_{uv}^{ij}表示子网M_i与M_j之间从节点u到节点v是否存在可行路径。公式(3-31)进一步明确，对于子网间可达路径，路径上的节点不能属于边界网关所在子网的节点集合，其中$P(i \to j)$表示子网M_i到子网M_j所选跨网路径所包含的节点集合，k为路径中的任意中间节点。公式(3-32)确保每个节点仅能绑定至一个已选定的网关，且绑定关系限定在同一子网内，以避免跨网耦合。在网络负载突发或阈值设置过于严格的情况下，上述约束可能导致该优化问题出现不可行情形，因此本文不假设该优化问题在所有场景下始终可行。在求解过程中，对违反约束的候选部署方案施加惩罚项进行约束引导，并优先从满足全

部约束的可行方案中选取评价最优的方案作为最终输出；若可行方案集合为空，则选取约束违背程度最小的候选方案作为最优输出，以保证后续路由优化具备有效输入。

本文所提出的联合优化问题，旨在满足带宽、连通性、流量分配与能耗等约束下，综合考虑延迟并最小化边界网关能耗与负载不均衡，由于包含整数决策与非线性目标，可建模为混合整数非线性规划（MINLP）。

定理：本文所提出的优化模型是 NP-hard。

证明：MINLP 模型的求解复杂度会随着网络规模和约束条件的增加呈现出指数级的增长趋势。为证明该模型属于 NP-hard 类问题[49]，本文通过与已有经典 NP-hard 问题的等价性和复杂度分析来进行论证。在 MINLP 模型中，跨域流量的带宽限制、节点与网关的连通性要求，以及网关间通信的流量控制，共同构成了一个带有容量限制的多商品流问题[50]。多商品流问题涉及多源多目的路径选择和流量分配的组合复杂度，已被严格证明为 NP-hard 问题。由于 MINLP 模型的计算复杂性与多商品流问题相似，因此其具有 NP-hard 性质。

需要特别说明的是，本文所述弹性部署并不意味着对边界网关进行高频或持续的动态指派。其核心机制是在运维配置周期内，由控制器根据全网状态对已具备网关能力的候选节点进行低频优化选择，以确定每个子网的边界网关。设备的物理位置与硬件形态保持不变，重新选择仅在检测到全局状态发生重大变化时才会触发。为进一步避免不必要的频繁切换，本文将全局状态重大变化定义为子网划分结构的调整事件。当控制器重新执行动态子网划分后，若生成的新划分在拓扑关联上与上一周期存在差异，则自动触发网关部署模块重新求解，以在更新后的子网内确定最优的 BG 位置。换言之，网关的弹性部署与子网的动态划分保持同步：当子网结构稳定时，BG 配置保持不变；当子网重新划分时，BG 部署将随之更新。

3.6.2 SOFA

常规求解方法在面对 MINLP 模型时常因计算复杂度过高而难以有效应对。为此，本文选用萤火虫算法（Firefly Algorithm, FA）对其进行求解。相比于传统精确求解方法，萤火虫算法无需依赖目标函数连续平滑或凸结构等严格条件，能够较好适应网关部署问题中离散节点选择、非线性约束及多目标耦合并存的特点。同时，该算法具有较强的启发式搜索能力，适合在复杂候选解空间中快速获得较优可行解，因此更适用于本文网关部署问题的求解。然而，随着网络规模的扩大，网关部署组合的解空间呈现出百万级的规模，这使得萤火虫算法在面对如此庞大的解空间时，运行时间显著增加。为了解决这一问题，本文提出了一种基于解空间优化的萤火虫算法（SOFA），该算法首先对解空间进行优化，以减少搜索空间

规模来提高算法的效率。

解空间的优化过程如图 3-3 中第 2 阶段的上半部分所示。假设网络拓扑包含K个子网，其中每个子网M_i包含n_k个候选边界网关。本文通过笛卡尔积生成所有可能的边界网关组合，形成解空间$C = \{C_1, C_2, \dots, C_N\}$，其中$N = \prod_{k=1}^{K} n_k$。每个组合$C_i$表示从各子网选取一个候选网关的元组，$C_i = \{M_1^i, M_2^i, \dots, M_K^i\}$。对于每个组合$C_i$，定义其性能指标有以下五种：负载均衡度$f_L$、节点的延迟比$f_v$、节点到网关的延迟比$f_{v-g}$、网关间的延迟比$f_{g-g}$、能耗值$f_E$。基于这些指标，构建解空间矩阵$\mathcal{M} \in \mathbb{R}^{N\times 5}$，其每一行对应一个网关部署组合的特征向量[37]，具体定义如下：

$$\mathcal{M} = \begin{bmatrix} f_L^{(1)} & f_v^{(1)} & f_{v-g}^{(1)} & f_{g-g}^{(1)} & f_E^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_L^{(N)} & f_v^{(N)} & f_{v-g}^{(N)} & f_{g-g}^{(N)} & f_E^{(N)} \end{bmatrix} \tag{3-33}$$

在联合优化问题中，各指标的权重应根据数据本身的统计特性与内在关联性进行动态分配，而非依赖主观经验设定。接下来，将详细介绍动态分配权重的过程。为避免量纲差异对权重计算的干扰，首先对矩阵$\mathcal{M}$进行标准化处理：

$$\mathcal{M}_{std}[i,j] = (\mathcal{M} - \mu) \oslash \mathcal{L} \tag{3-34}$$

其中，$\mu \in \mathbb{R}^5$表示各列均值向量，$\mathcal{L} \in \mathbb{R}^5$为各列标准差向量，$\oslash$表示按列逐元素除法。接着，计算各指标方差$\theta_i \in \mathbb{R}^5$，并对其进行归一化得到指标的初始权重$\omega_i$：

$$\omega_i = \frac{\theta_i}{\sum_{k=1}^{5} \theta_k},\ i = 1,2,\dots,5 \tag{3-35}$$

为防止过拟合，设定最小权重阈值ω_{min}，从而得到调整后的权重ω'：

$$\omega_i{}' = max(\omega_i, \omega_{min}), \omega' = \frac{\omega_i{}'}{\sum \omega_i{}'} \tag{3-36}$$

然后，通过权重向量对原始矩阵进行加权处理，得到加权解空间矩阵$\mathcal{M}_{wtd} = \mathcal{M} \cdot diag(\omega')$。基于上述操作，利用奇异值分解（Singular Value Decomposition, SVD）方法提取解空间的主成分，从而实现解空间维度的压缩。具体来说，首先对加权矩阵$\mathcal{M}_{wtd}$进行奇异值分解，得到如下结果：

$$\mathcal{M}_{wtd} = U\Sigma V^T \tag{3-37}$$

其中，U、Σ和V分别表示左奇异矩阵、对角矩阵和右奇异矩阵。矩阵$U \in \mathbb{R}^{N\times r}$和$V \in \mathbb{R}^{5\times r}$满足$U^T U = I$和$V^T V = I$，左奇异向量和右奇异向量均为正交向量。对角矩阵$\Sigma \in \mathbb{R}^{r\times r}$的对角元素满足$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r \geq 0$，其中σ表示矩阵的对角系数[51]，其值反映了组合的重要性程度。SVD 本身具有以下性质：

$$\mathcal{M} = \sigma_1 u_1 x_1^T + \sigma_2 u_2 x_2^T + \cdots + \sigma_q u_q x_q^T \tag{3-38}$$

紧接着，计算前k个奇异值的累计方差占比，公式如下：

$$EV(k) = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2} \tag{3-39}$$

通过设定阈值ρ=0.9，选择最小的q，使得$EV(q) \geq \rho$，即保留前q个主成分，并将解空间投影到这些主成分上，从而得到降维后的矩阵$\mathcal{M}_{rd}$：

$$\mathcal{M}_{rd} = U_{[:,1:q]}\Sigma_{[1:q,1:q]} \tag{3-40}$$

通过上述定义，$\mathcal{M}_{rd}$保留了原始矩阵 90%以上的信息量。为了从降维后的解空间中筛选出具有较高代表性的候选解，本文对降维矩阵$\mathcal{M}_{rd}$的每一行计算其综合得分：

$$s_i = \sum_{j=1}^{q} \mathcal{M}_{rd}[i,j], i = 1,2,\dots,N \tag{3-41}$$

得分s_i反映了每个组合在主成分空间中的投影强度，本文按照得分从高到低进行排序，选择前$K = \varphi N$个候选组合作为下一阶段算法执行的解空间。

基于上述解空间优化结果，本文进一步采用萤火虫算法对网关的最优部署位置进行求解，由第二章理论介绍可知，萤火虫算法是一种基于生物拟态的群体智能优化方法，其通过亮度更优个体吸引较弱个体的群体协同搜索机制迭代逼近最优解。基于该基本原理，本章在求解弹性网关部署时，将每个萤火虫个体对应一个候选的网关部署方案，并以目标函数值作为亮度评价依据。结合公式(3-23)的联合优化目标，本文将萤火虫算法的适应度函数定义为：

$$f(x) = \left(\sum_{k=1}^{3} \frac{\omega_k * d_{avg}(k)}{d_{max}(k)} + \left(\omega_4 * E_{g_i} + \omega_5 * L \right) \right) + P_i \tag{3-42}$$

公式里的P_i表示违反第i个约束项所带来的惩罚，考虑到网关部署属于离散组合空间问题，标准迭代更新易在局部最优附近停滞。为增强搜索的全局性并维持种群多样性，本文在每轮位置更新后引入变异算子，对部分个体进行随机扰动，其形式为：

$$x_i = x_i + \eta \cdot (x_{best} - x_i) + \xi \cdot (rand - 0.5) \tag{3-43}$$

其中，x_{best}是当前种群中适应度最优的部署方案，η是变异强度，控制个体向最优解靠拢的强度，ξ是变异幅度，控制随机扰动幅度，$rand$为[0, 1]上的均匀随机数。

网关部署的基本过程如图 3-3 所示。在完成子网划分操作（阶段 1）后，本文使用 SOFA 算法执行网关部署。具体而言，阶段 2 包括解空间优化和萤火虫算法求解两个步骤。本文先对候选网关部署组合的解空间进行优化与降维，筛选出具有代表性的候选解，缩小搜索范围。接着，在精简后的解空间中应用萤火虫算法进行求解。萤火虫算法会根据目标函数计算每个方案的亮度，并依据“亮者吸引暗者”的原则，驱使较差解向较优解移动，同时通过变异操作增强解的多样性。该过程持续迭代，直至满足收敛条件，最终得到最优的网关部署组合（阶段 3）。

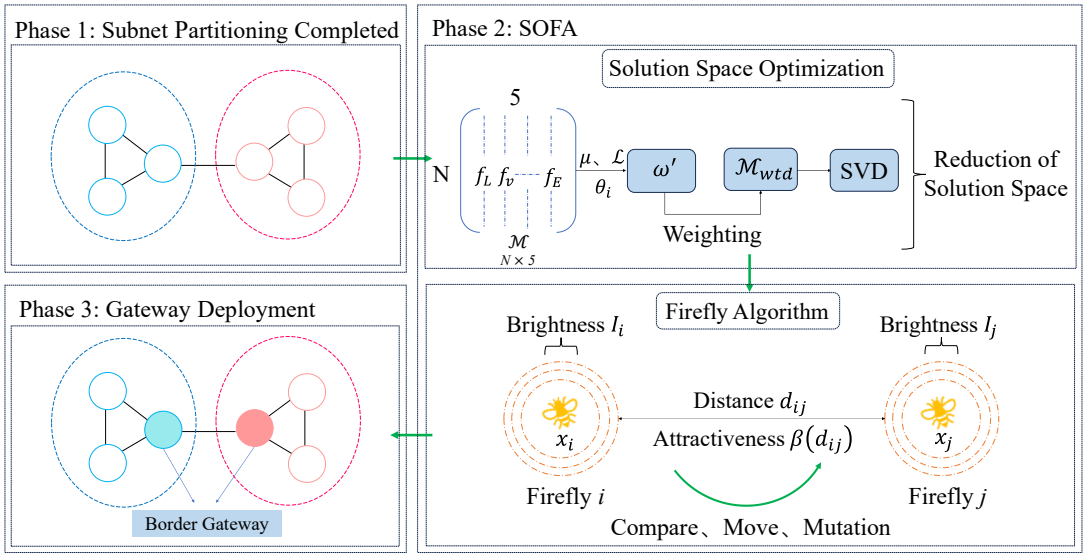


图 3-3 网关部署的基本过程

算法 2 描述了计算网关部署组合适应度值的具体流程，主要应用于 SOFA（算法 3）中。在网关集合非空的情况下，算法依次遍历每个网关计算其初步的适应度值（见第 2-5 行所示）。随后，检查该网关本身及其与其他网关的交互是否满足预设的约束条件。对于不满足约束的网关，适应度中将加入相应的惩罚项（见第 6-9 行所示），以降低该网关的评估分数。最终，算法返回当前网关部署组合的综合适应度值，作为后续优化过程的依据。

算法 2：适应度函数的计算

输入：包含交换机节点与链路边的网络图$G(V,E)$，网关部署方案G_b

输出：Fitness

```
1:  Penalty ← 0
2:  if G_b ≠ ∅ then
3:      for g_i ∈ G_b do
4:          for g_j ∈ G_b, i ≠ j do
5:              根据式(3-23)计算适应度值 Fitness
6:              if 式(3-12) – (3-13)、式(3-25) – (3-32)中的约束不满足 then
7:                  penalty ← penalty + penalty_value
8:              end
9:          end
10:     end
11:     Fitness←Fitness + penalty
12: end
13: return Fitness
```

SOFA 的执行流程如算法 3 所示。在第 1-9 行中，主要实现了解空间优化的

步骤。具体而言，算法基于笛卡尔积生成的候选解构建解空间矩阵，并通过标准化处理和方差归一化方法计算目标函数中的权重因子。随后，对标准化矩阵进行加权降维以保留关键特征信息，最后采用奇异值分解（SVD）确定主成分数量，生成低维近似矩阵并筛选出最具代表性的候选解作为初始种群。第 10-26 行展现了采用萤火虫算法求解最优网关部署组合的流程，在每一代中，萤火虫之间通过两两比较驱动个体向更优解迁移，同时引入概率性变异操作，避免算法陷入局部最优解，最终算法通过多次迭代达到最优解。

算法 3：SOFA

输入：子网数量K，$\{M_1^i, M_2^i, \dots, M_K^i\}$，$\omega_{min}$，用于 SVD 的累计方差阈值$\rho$，候选解比例$\alpha$，MaxIter，PopSize，MutationRate

输出：最优网关部署方案G_b，权重向量ω'

1: 将各子网候选网关集合$M_k^{(i)}$进行笛卡尔积，得到候选解空间C
2: 计算性能矩阵$\mathcal{M} \in \mathbb{R}^{N\times 5}$，$\mathcal{M}_{std}$
3: $\omega \leftarrow$max(Variance($\mathcal{M}_{std}$)/sum(Variance($\mathcal{M}_{std}$)), ω_{min})
4: $\omega' \leftarrow \omega$/sum($\omega$)
5: $\mathcal{M}_{wtd} \leftarrow \mathcal{M} \cdot$ diag(ω')
6: (U, Σ, _)$\leftarrow$SVD($\mathcal{M}_{wtd}$)
7: $q \leftarrow$min$\{q{:}EV(q) \geq \rho\}$并计算降维后的矩阵$\mathcal{M}_{rd}$
8: $s \leftarrow$rowSum($\mathcal{M}_{rd}$)
9: 根据得分s选择$\lfloor \alpha N \rfloor$个候选方案
10: 从筛选出的候选方案中初始化萤火虫种群$\{x_1, \dots, x_{Popsize}\}$
11: 依据算法 2 计算各候选方案的适应度值
12: for i=1 to *PopSize* do
13: $I_i \leftarrow$1/Fitness(x_i)
14: end
15: for t = 1 to *MaxIter* do
16: for i = 1 to *PopSize* do
17: for j = 1 to *PopSize* do
18: if $I_j > I_i$ then
19: $x_i \leftarrow x_i +$ Attraction(x_j, x_i)
20: Update I_i
21: end
22: end
23: if *Random()* < *MutationRate* then
24: $x_i \leftarrow x_i + \eta \cdot (x_{best} - x_i) + \xi \cdot (\text{Random}() - 0.5)$
25: Update I_i
26: end
27: end
28: end
29: return x_{best}，ω'，G_b

算法 2 包含两层嵌套的 for 循环，因此其时间复杂度为$O(n_g^2)$，其中$n_g = |G_b|$

表示每一种组合中网关的数量。对于算法 3，其时间复杂度可表示为$O(m^K + T \cdot N^2 \cdot n_g^2)$。其中，$m^K$来源于候选解空间的笛卡尔积；$m$表示每个子网的候选网关数量，$K$为子网数量，$T$表示萤火虫算法的迭代次数，$N$为种群规模（即 PopSize），而因子$n_g^2$用于刻画在每次适应度更新时调用算法 2 所产生的计算开销。

3.7 延迟感知路由算法设计

在动态子网划分与弹性网关部署的基础上，本文设计了一种基于延迟感知的分层路由算法。该算法的核心思想是根据数据请求的源节点与目的节点的逻辑子网隶属关系来动态选择路由策略。具体而言，若源节点与目的节点位于同一子网内，算法将基于 SDN 控制器实时维护的链路延迟权重矩阵，直接计算端到端最优路径，从而确保子网内通信的毫秒级响应。如图 3-4 的紫色虚线所示，节点 8 和节点 13 都属于子网 2，因此，节点 8 可以在子网内快速找到通往节点 13 的最优路径。

若源节点与目的节点属于不同子网，算法将采用分层优化机制。在子网内部，路由选择将依据已优化的网关部署组合，按照最小延迟原则在节点和网关之间确定最优路径；在跨子网通信场景中，数据包将通过最优网关节点进行中继转发，跨子网的路径选择仍然以最小延迟为优化目标，并依托 SDN 全局视图实现跨域链路状态的协同。最终，子网内的最优路径与跨子网的最优中继路径通过无缝整合构建了一个端到端的全局最优传输链路，从而满足数据中心网络中低延迟、高效率的传输需求。如图 3-4 粉色虚线所示，节点 8 先在子网 2 内计算并找到通向网关 5 的最优路径；随后，网关 5 依据跨子网路由表，定位至节点 4 所在的子网 3，并通过网关 10 进行跨子网转发；最后，网关 10 在子网 3 内查找到通往节点 4 的最优路径，完成端到端路径的全局构建。

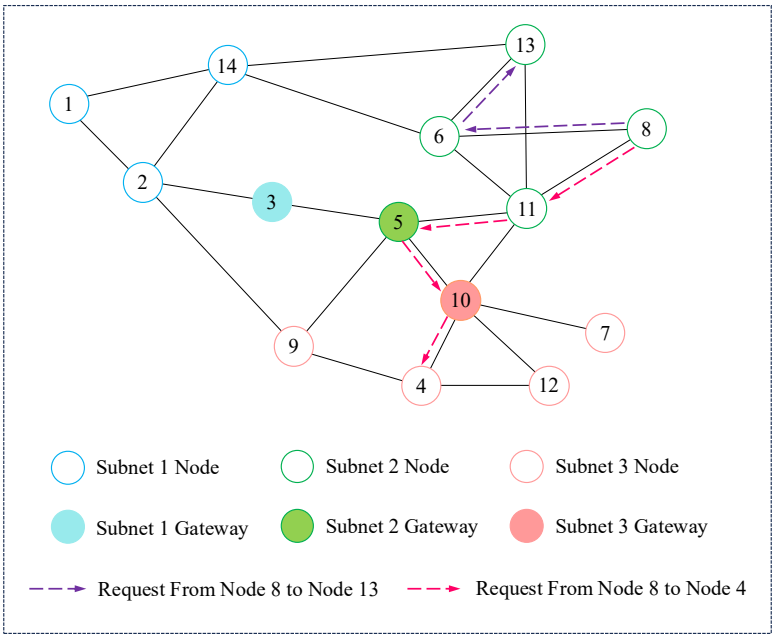


图 3-4 基于延迟感知的分层路由算法示例

算法 4 描述了基于延迟感知的分层路由算法执行流程。在第 1-3 行中，算法从请求流中提取了源节点和目标节点，并根据子网映射信息判断二者是否属于同一子网。若源、目标节点在同一子网内，算法基于实时更新的链路延迟权重计算它们之间的最小延迟路径，并直接返回结果（第 4 行所示）。若源、目标节点分属不同子网，算法将依次计算三段基于延迟权重的路径：源节点到源网关的最小延迟路径、源网关到目标网关的跨子网中继路径，以及目标网关到目标节点的最小延迟路径。随后，算法将三段路径拼接，并移除其中的重复节点，生成从源节点到目标节点的完整路径，作为跨子网的最终路由方案（第 6-9 行所示）。

算法 4：基于延迟感知的分层路由算法

输入：包含交换机节点与链路边的网络图$G(V,E)$，网关部署方案G_b，请求流f_k，节点所属子网映射 *SubnetInfo*，延迟权重ω_{delay}

输出：Path

1: $s \leftarrow s_k$

2: $d \leftarrow d_k$

3: if *SubnetInfo*[*s*] = *SubnetInfo*[*d*] then

4: Path ← LatencyOptimalPath(G, s, d, ω_{delay})

5: else:

6: path1 ← LatencyOptimalPath(G, s, s_{gw}, ω_{delay})

7: Path2 ← LatencyOptimalPath(G, s_{gw}, d_{gw}, ω_{delay})

8: Path3 ← LatencyOptimalPath(G, d_{gw}, d, ω_{delay})

9: Path ← RemoveDuplicates(path1 + path2 + path3)

10: end

11: return *Path*

在算法 4 中，其时间复杂度主要由基于延迟权重的最优路径计算过程决定。这里的路径计算虽然在形式上与最短路径算法一致，但采用的是动态更新的链路延迟作为权重，因此求解的是端到端最小延迟路径，而非传统基于跳数或静态权重的最短路径。

当源节点和目标节点位于同一子网内时，最小延迟路径仅在该子网内计算，时间复杂度为$O(m_i + n_i \log n_i)$，其中m_i和n_i分别表示子网内的边数和节点数。当源节点和目标节点分属不同子网时，需要计算三段基于延迟权重的最优路径，整体路径计算的时间复杂度$O(m_i + m_g + m_j + n_i \log n_i + n_g \log n_g + n_j \log n_j)$，其中$m_i$，$n_i$与$m_j$，$n_j$为源与目标子网内的边数与节点数，$m_g$，$n_g$为子网间的边数与节点数。

此外，算法去重操作的时间复杂度为$O(l)$，但通常该操作的复杂度远小于最小延迟路径的复杂度，因此算法 4 的整体时间复杂度仍然为$O(m_i + m_g + m_j + n_i \log n_i + n_g \log n_g + n_j \log n_j)$。考虑到实际划分中，单个子网的规模远小于全网

规模（即$n_i, n_j \ll n$，$n_g \ll n$），且子网间连接较稀疏，因此算法 4 的整体复杂度远低于传统集中式全局计算的复杂度。

3.8 仿真设计结果及分析

本节将详细介绍 ROCDSG 的实验环境设置与结果评估。首先说明仿真实验平台与实现配置，包括操作系统、控制器与仿真工具链等基础环境；随后给出实验所采用的网络拓扑与请求流量设置，并对实验中涉及的关键参数与资源约束进行统一定义与说明。最后，通过与其他三类对比算法在多种拓扑场景下的对照实验，系统展示 ROCDSG 的综合性能与优势。

3.8.1 实验方法及仿真参数设置

实验在虚拟机的 Ubuntu 18.04 操作系统上进行，基于 Ryu 控制器、Mininet 平台和 Python 对所提出的 ROCDSG 性能进行验证，其中 Ryu 控制器版本为 4.34，Mininet 版本为 2.3.1b4，python 版本为 3.11.7。实验采用的网络拓扑来自 The Internet Zoo[30]，并选取了以下三种典型拓扑结构进行测试：（1）BICS 拓扑，包含 33 个节点和 48 条边；（2）DFN 拓扑，包含 58 个节点和 87 条边；（3）Colt Telecom 拓扑，本文从该拓扑中选取了 100 个节点和 122 条边。本文假设激活交换机所需的能量为 118 个单位，激活链路所需的能量为 48 个单位，每个字节传输所消耗的能量为 $4e^{-10}$ 个单位。这些能耗参数来自不同来源：交换机激活能耗和链路激活能耗参考了文献[11]的能耗模型，而每字节传输能耗采用了主流数据中心网络研究中常用的经验值，用于模拟典型高性能交换设备的传输开销。实验中使用的交换机为千兆交换机，每条链路的带宽容量设定为 500 Mbps。

在实验中，本文采用泊松分布对数据流的到达进行建模[52]，以生成动态流量输入并评估所提出框架的性能。设随机变量$X(t)$表示时间步t内到达的数据流数量，则其概率质量函数为：

$$P[X(t)=k]=\frac{\lambda^k e^{-\lambda}}{k!}，\ k=0,1,2\ldots \tag{3-44}$$

其中，λ为单位时间内流的平均到达率，k表示在该时间步内实际到达的流数量，e为自然常数。为保证实验条件的一致性与可重复性，本文依据式(3-44)生成输入流；具体而言，数据流的到达过程服从参数为λ=5 flow/s 的泊松分布，每条数据流的数据包大小服从正态分布，均值为 500 KB，标准差为 50 KB。

针对萤火虫算法的参数设置如下：种群规模设为 100，最大迭代次数设为 200，吸引度β取值为 1.0，衰减因子γ设为 1.0，随机扰动系数ψ设为 0.3（扰动系数的衰减率为 0.97），变异强度η设为 0.2，变异幅度ξ设为 0.05[53]。实验的具体参数设

置详见表 3-2 相关说明。

表 3-2 实验具体参数设置

参数	值
BICS 拓扑	33 个节点、48 条边
DFN 拓扑	58 个节点、87 条边
Colt Telecom 部分拓扑	100 个节点、122 条边
交换机激活能量	118 个单位
链路激活能量	48 个单位
单字节传输能量	$4e^{-10}$ 个单位
交换机带宽容量	1000 Mbps
链路带宽容量	500 Mbps
平均到达率	5 flow/s
数据包大小均值	500 KB
数据包大小标准差	50 KB
种群规模	100
β	1.0
γ	1.0
ψ	0.3
η	0.2
ξ	0.05

3.8.2 对比算法介绍

为开展公平且具有代表性的评估，本文将所提出的方案与现有结构化降维路由算法领域的三类代表性方法进行对比：

(1) BCR[4]：一种面向大规模 SDN 的基于盒覆盖（box-covering）的路由算法。该方法通过重整化将网络划分为逻辑子网，并在子网间与子网内分别采用 Dijkstra 算法计算端到端路径。

(2) HGWO-Firefly[31]：一种面向簇结构异构传感器网络的群智能路由方案。该方案采用 K-means 与灰狼优化算法相结合的策略进行簇首选择，并利用萤火虫算法搜索传感器节点与基站之间的节能多跳路由。

(3) QoSPR[30]：一种面向 5G 赋能工业物联网（IIoT）的 QoS 与隐私感知路由协议。该方法对 Infomap 进行改进以实现动态子域划分，并结合深度强化学习与联邦强化学习，在延迟、负载均衡与隐私等约束下学习网关部署与路由策略。

上述三种方案分别代表了三类设计范式：拓扑感知的结构化路由（BCR）、结合聚类的元启发式路由（HGWO-Firefly），以及基于机器学习的网关部署与路由（QoSPR）。其构建思路与本文关注的“动态子网构建与网关部署协同优化”问题高度相关，因此具有较强的对比意义。

3.8.3 评价指标

在本文中，为了全面评估 ROCDSG 在动态子网构建与网关部署协同路由场景下的综合性能，参考相关研究的实验评估范式，本文从结构稳定性、延迟表现、均衡性与路由效率等维度出发，定义如下关键评价指标：

(1) **子网划分差异**：子网在负载、流量和能耗划分上的差异是评估子网划分合理性和有效性的关键度量。本文将各项指标的划分差异定义为最大值子网与最小值子网之间的差距。例如，负载的划分差异定义为具有最大负载量的子网与具有最小负载量的子网之间的差值。

(2) **平均延迟和最大延迟**：本文分别从三类通信关系统计延迟指标。节点的平均延迟和最大延迟分别由公式(3-6)和(3-7)计算，节点到网关的平均延迟和最大延迟分别由公式(3-8)和(3-9)计算，网关间的平均延迟和最大延迟分别由公式(3-10)和(3-11)计算。

(3) **负载均衡和能耗均衡系数**：由于本文提出的 ROCDSG 已经采用了负载约束和能耗约束实现了子网划分，因此，针对网络全局的负载和能耗性能评估，本文使用另外的指标来衡量，其公式定义如下：

$$\lambda_L = \sqrt{\frac{1}{M}\sum_{i=1}^{|M|}\left(L_i - \frac{L_T}{|M|}\right)^2} \tag{3-45}$$

$$\lambda_E = \sqrt{\frac{1}{M}\sum_{i=1}^{|M|}\left(E_i - \frac{L_E}{|M|}\right)^2} \tag{3-46}$$

其中，λ_L和λ_E分别表示网络的负载均衡系数和能耗均衡系数，L_i表示第 i 个子网的总负载，L_T表示整个网络的总负载，E_i表示第 i 个子网的总能耗，L_E表示整个网络的总能耗。这说明λ_L和λ_E越小，网络的负载和能耗越均衡。

(4) **端到端平均延迟**：该指标用于衡量数据在网络中从源节点到目标节点的整体传输时效性。设成功完成的流集合为$\mathcal{F}$，流f的端到端延迟为d_f，则端到端平均延迟可定义为：

$$\overline{D} = \frac{1}{|\mathcal{F}|}\sum_{f\in\mathcal{F}} d_f \tag{3-47}$$

(5) **平均跳数**：该指标用于衡量成功完成传输的请求流在端到端路径上经历的链路数（或中继次数）的平均水平，反映路由路径长度与转发开销。

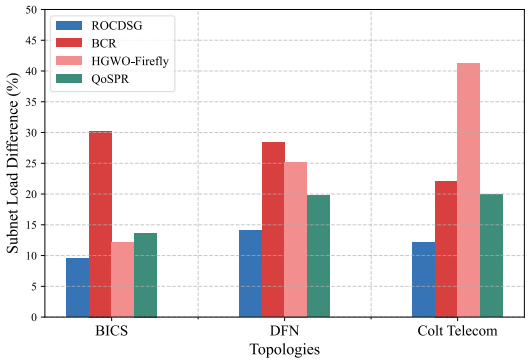
(6) **请求流执行时间**：指在一次实验周期内，算法完成全部请求流的路由决策及路径计算所消耗的总时间，用于衡量其计算开销与实际执行效率。

3.8.4 实验结果与分析

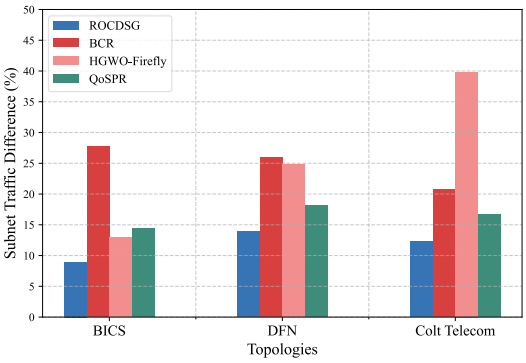
(1) 实验 1：子网划分差异

图 3-5 展示了不同算法在子网负载、流量及能耗分布差异上的对比结果。从(a)图可以看出，ROCDSG 在不同网络拓扑中均展现出优异的负载均衡能力。结合三类拓扑的具体结果可以发现，ROCDSG 的负载差异约介于 9.5%–14%之间，而各拓扑中的次优算法负载差异大多落在 12%–20%区间，相比之下，三种拓扑上的负载差异相对降低幅度约为 20%、29%和 39%，平均降幅约 29%，说明子网划分组件在负载均衡方面具有稳定而显著的性能优势。

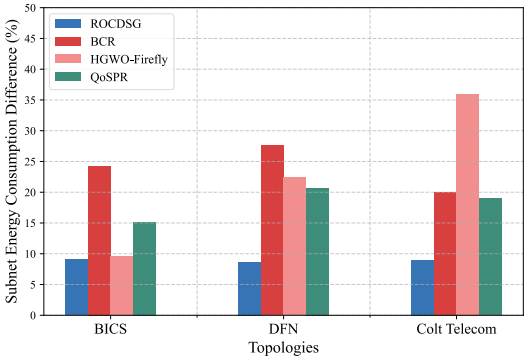
(b)和(c)图进一步验证了 ROCDSG 在流量和能耗均衡方面的优势。实验数据表明，ROCDSG 的流量差异分别约为 9%、13.9%和 12.3%，对应次优算法的流量差异约为 12%、16.8%和 16.6%；能耗差异方面，ROCDSG 分别约为 9%、8.6%和 9%，而次优算法约为 15.1%、20.7%和 19%。由此可见，ROCDSG 在三种拓扑上的流量差异相较次优算法平均降低约 23%，能耗差异平均降低约 50%。综合以上分析，ROCDSG 通过在子网划分过程中引入流量、负载和能耗的多维约束，使子网资源在三类拓扑上均实现了约 34%左右的综合均衡性提升。



a)子网负载分布差异



b)子网流量分布差异



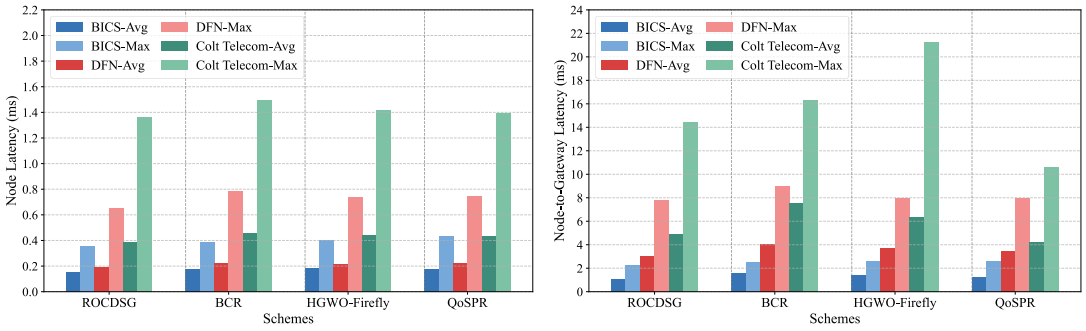
c)子网能耗分布差异

图 3-5 子网负载、流量、能耗分布上的差异比较

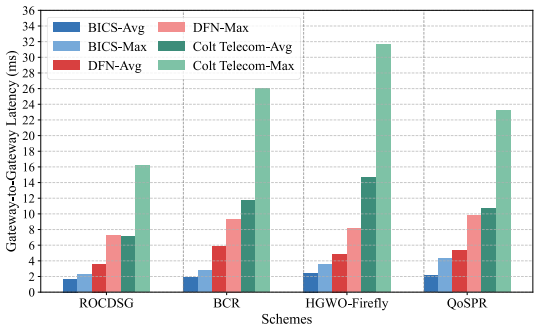
(2) 实验 2：平均延迟和最大延迟评估

图 3-6 通过多维度延迟评估揭示了不同算法在网关部署后的差异性表现：如

图 3-6(a)所示，ROCDSG 在 Colt Telecom 拓扑中的节点平均延迟和最大延迟分别稳定于 0.38 ms、1.35 ms 左右，较 BCR、HGWO-Firefly 和 QoSPR 算法均有所降低，说明 ROCDSG 能够在复杂的拓扑结构中有效地控制节点上的延迟堆积。进一步分析图 3-6(b)的对比结果，发现 ROCDSG 在 Colt Telecom 拓扑中的节点到网关平均延迟和最大延迟略高于 QoSPR 算法。这可能是因为，在网关部署过程中，由于考虑了多种约束条件，随着网络规模的扩大，子网数量增多，为了实现优化目标，部分子网的网关部署位置差异较大，导致到达网关的延迟增加。值得注意的是，在 BICS 和 DFN 拓扑上，ROCDSG 的节点到网关平均延迟和最大延迟均优于基准算法。图 3-6(c)显示了网关间延迟的评估结果，可以发现 ROCDSG 在跨网链路延迟控制方面表现出色，在三种不同拓扑中，ROCDSG 的网关间平均延迟和最大延迟均大幅低于其他算法。综上所述，ROCDSG 在网关部署过程中，通过综合考虑节点延迟、节点到网关延迟以及网关间延迟的最小化，能够有效保障数据中心网络中的 QoS 需求。



a)节点的平均延迟和最大延迟评估　　b)节点到网关的平均延迟和最大延迟评估



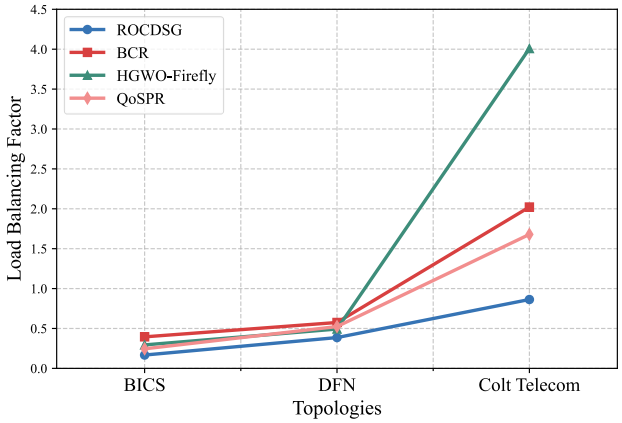
c)网关之间的平均延迟和最大延迟评估

图 3-6 不同拓扑下的多维度平均延迟和最大延迟比较

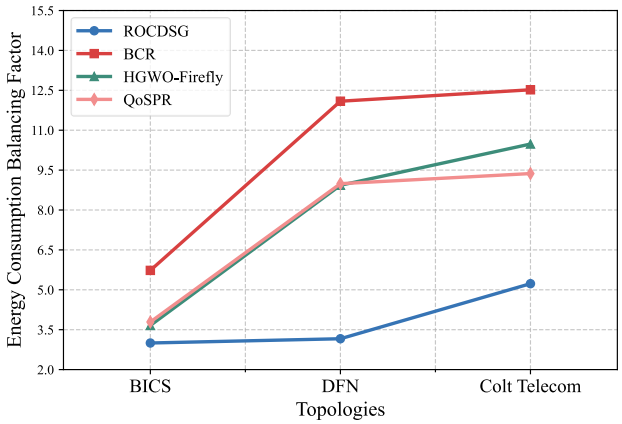
(3) 实验 3：负载均衡和能耗均衡

图 3-7 展示了不同算法在各拓扑下的整体负载均衡与能耗均衡性能。与基准算法相比，ROCDSG 在所有拓扑中均表现出更优的均衡能力。图 3-7(a)显示，在负载均衡方面，ROCDSG 在 BICS 和 DFN 拓扑中的性能与基准算法相对接近，但在 Colt Telecom 拓扑中优势显著：ROCDSG 的负载均衡系数始终维持在 1.0 以下的稳定区间，而基准算法的对应指标普遍超过 1.6，表明 ROCDSG 在较大规模

网络中的适应性更强。图 3-7(b)进一步显示，在能耗均衡方面，随着拓扑复杂度的提升，ROCDSG 的能耗均衡系数仅产生有限波动，而基准算法则呈现明显的性能下降趋势。



a)不同拓扑的负载均衡系数评估



b)不同拓扑的能耗均衡系数评估

图 3-7 网络整体的负载与能耗均衡性能比较

(4) 实验 4：端到端平均延迟评估

图 3-8 对比了各算法在不同拓扑中的端到端平均延迟性能。可以明显看出，ROCDSG 在所有拓扑中均显著优于基准算法。具体而言，ROCDSG 在三种拓扑中的端到端平均延迟分别稳定在 2.6 ms、9.5 ms 和 13.0 ms 左右，而次优算法的端到端平均延迟分别约为 3.3 ms、13.5 ms 和 17.3 ms。定量分析显示，相较于次优算法，ROCDSG 在三种拓扑中的端到端平均延迟分别降低了 21.2%、29.6%和 24.9%，整体降低幅度达 25.2%。同时可以观察到，随着网络拓扑规模的扩大，所有算法的端到端延迟整体均呈上升趋势；然而，ROCDSG 的增长幅度明显低于其他算法，表明其在网络规模扩展过程中能够更有效地抑制延迟开销的增加，从而展现出更优的可扩展性。

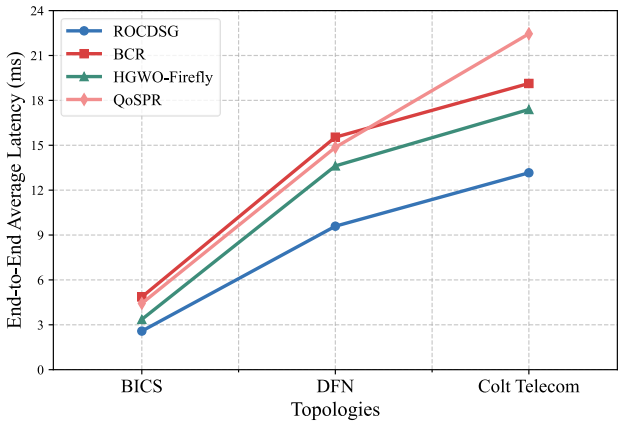
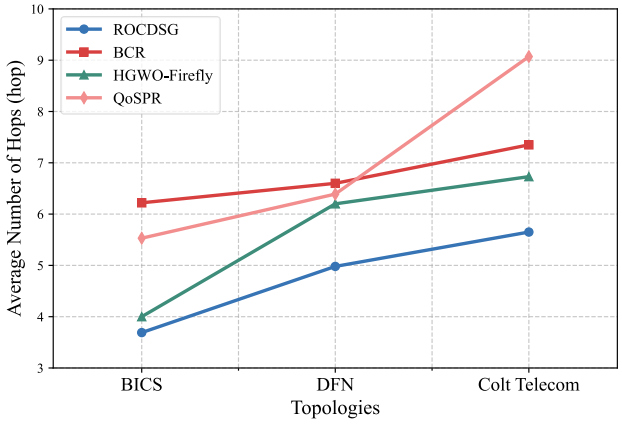


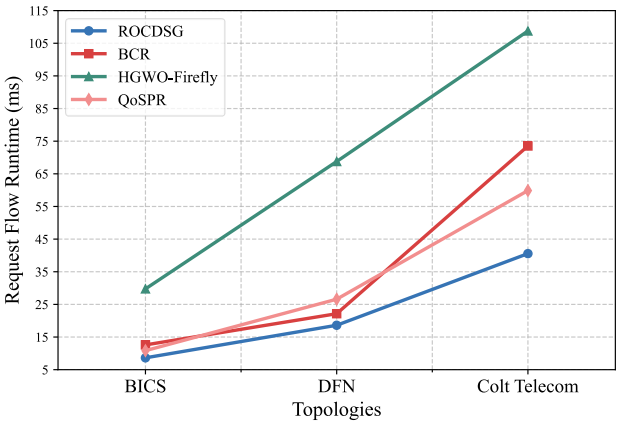
图 3-8 请求流端到端平均延迟比较

(5) 实验 5：路由效率评估

从图 3-9(a)中可以观察到，在路径跳数优化方面，ROCDSG 在三种拓扑中的平均跳数分别约为 3.7、5.0 和 5.6，而次优算法的平均跳数分别约为 4、6.2 和 6.7。这表明，ROCDSG 相较次优算法在三种拓扑中的平均跳数分别减少了 7.5%、19.4%和 16.4%，整体减少幅度达 14.4%。这种优化效果源于 ROCDSG 结合了子网划分与网关部署的联合优化，有效避免了次优算法中可能出现的路径迂回和不必要的跳数增加。在执行时间方面，图 3-9(b)显示，ROCDSG 在 BICS 和 DFN 拓扑中的执行时间分别约为 8.7 ms 和 18.7 ms，较次优算法的 12.5 ms 和 22 ms 分别降低了 30.4%和 15%。值得注意的是，在更复杂的 Colt Telecom 拓扑中，ROCDSG 的执行时间约为 41.2 ms，显著优于次优算法 QoSPR 的 60 ms，降低幅度达到 31.3%。总体而言，ROCDSG 在执行时间上较次优算法减少了 25.6%，展现出更好的扩展性与计算效率。相比之下，HGWO-Firefly 算法由于依赖于启发式算法、种群初始化及多轮迭代过程，在所有拓扑中均表现出较高的执行时间开销，进一步突显了 ROCDSG 的优势。



a)请求流平均跳数比较



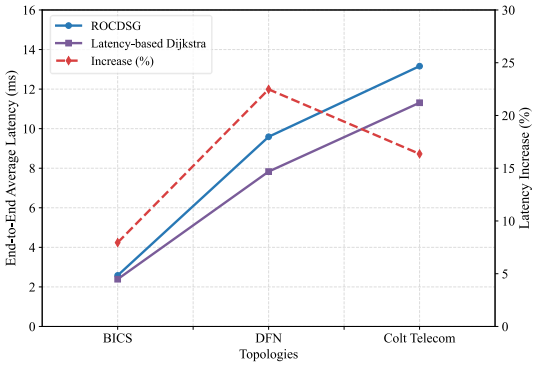
b)请求流执行时间比较

图 3-9 不同拓扑的路由效率指标评估

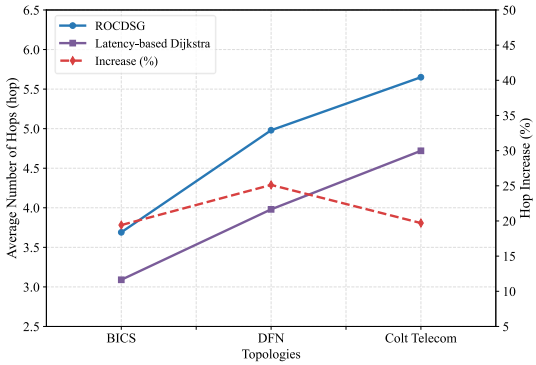
(6) 实验 6：结构降维与全局路径优化权衡分析

为进一步评估 ROCDSG 在结构降维场景下的实际性能收益，本文将其与基于延迟的 Dijkstra 算法进行对比分析，并从端到端平均延迟、平均跳数以及请求流执行时间三个方面验证结构降维机制对路径质量与计算效率的影响，如图 3-10 所示。实验结果表明，基于延迟的 Dijkstra 算法能够在全网范围内直接搜索最短路径，因此在端到端平均延迟与平均跳数方面整体表现更优。相比之下，ROCDSG 由于采用子网网关骨架进行跨子网转发，在部分场景下会引入一定程度的路径绕行，其在 BICS、DFN 与 Colt Telecom 拓扑中的端到端平均延迟分别增加约 7.9%、22.5%与 16.4%，平均跳数分别增加约 19.4%、25.1%与 19.7%。

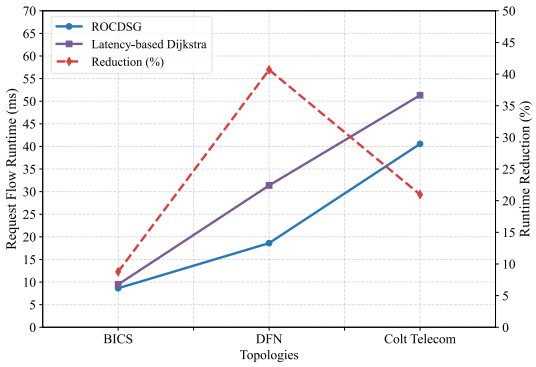
然而，从运行效率角度看，ROCDSG 表现出更明显的优势。在 DFN 与 Colt Telecom 拓扑中，其请求流执行时间分别降低约 40.7%与 20.9%，说明结构降维能够在中大规模复杂拓扑下有效降低全局路径搜索开销。综合来看，ROCDSG 并非以单一最短路径为唯一优化目标，而是在 QoS 保障、负载均衡、能耗控制与计算效率之间进行综合权衡，以更好支撑复杂动态场景下的在线路由决策与网络可扩展性。



a)请求流端到端平均延迟评估



b)请求流平均跳数评估



c)请求流执行时间评估

图 3-10 ROCDSG 与基于延迟的 Dijkstra 算法性能对比

3.9 本章小结

本章从结构降维的角度出发，研究数据中心网络中在 QoS 保障、负载均衡、能耗控制与路由计算复杂度之间的多维冲突及其协同优化问题。本文提出了 ROCDSG 框架，将基于 Louvain 的动态子网划分、优化的网关部署以及延迟感知的分层路由相结合。通过联合考虑延迟、负载与能耗等因素，ROCDSG 能够高效适应复杂的数据中心环境。实验结果表明，ROCDSG 在实现负载与能耗均衡分布的同时，显著提升了 QoS 性能与路由效率，验证了其在大规模数据中心应用中的有效性。

4 QoS 感知的波动反馈增强路由算法

4.1 引言

在上一章中，本文从结构降维与协同优化的视角提出 ROCDSG 框架，为数据中心网络在多目标约束下的路径决策构建了可执行的分层转发骨架，并通过实验验证了其有效性。该框架的核心收益在于先从结构层面对网络进行组织与压缩，从而显著降低路径搜索空间与控制开销，为在线决策提供可执行的骨架边界。也正因如此，上一章的路由判据与实现更侧重以端到端延迟作为主要评判标准，通过延迟感知的分层路由完成骨架上的路径选择，而对 DCN 中不同业务流的优先级差异以及抖动、丢包等多维 QoS 约束尚未形成统一的细粒度刻画与差异化保障策略。

需要指出的是，骨架化设计在降低路径搜索空间与控制开销的同时，也改变了跨子网业务的转发形态，使跨子网通信更集中地依赖边界网关等有限通道，网络逻辑由多路径分散承载转向少量骨架通道集中承载。当业务规模较小或负载变化较平缓时，该机制仍能保持较为稳定的 QoS；但在突发流量与动态负载等典型场景下，跨子网流量占比上升，链路状态波动增强，骨架通道中的共享竞争与路径绕行问题可能进一步放大排队效应，使网关汇聚更易形成持续性热点，从而增加端到端延迟与路径开销波动的风险。结合第三章实验结果可知，ROCDSG 在三种拓扑中的端到端延迟与跳数分别平均增加约 15.6%与 21.4%，表明跨子网转发会引入额外路径开销，并对端到端 QoS 稳定性产生一定影响。同时，由研究现状可知，现有方法对路径下发后共享链路上的热点演化关注不足，缺乏面向运行态的持续反馈与调节机制。

因此，本文旨在不破坏结构降维优势与骨架可执行性的前提下，引入更细粒度的在线调度与反馈调节机制，以提升路由对在线场景的适应能力。该机制需满足三点要求：其一，面对多业务优先级与多维 QoS 约束并存的在线决策，必须将是否接纳与如何选路统一刻画为同一权衡过程，避免仅以单一延迟判据造成的偏置，从而形成可比较、可优化的统一判据并实现差异化保障；其二，将跨子网直连边界链路与边界网关转发一并纳入候选生成，扩展跨子网可选通道与候选路径集合，降低候选受限与绕行风险；其三，在路径下发后持续监测关键链路与网关处的共享竞争状态，出现热点苗头时及时触发微调或重构，以抑制拥塞扩散并降低长尾风险。

基于此，本文提出一种 QoS 感知的波动反馈增强路由算法，称为 QVFER。对于在线到达的业务流，系统首先完成优先级划分，并据此确定差异化的 QoS 约束边界与资源偏好；随后，定义统一的目标函数，将流量接纳收益与路径综合代

价的权衡显式纳入同一决策准则；在该准则指导下，QVFER 生成统一的候选路径集合，并在多维 QoS 约束下依次进行可行性判别与代价评估，择优确定路径并在线下发；路径部署后，QVFER 将链路波动率 VS 作为反馈控制量嵌入闭环调节过程，在波动加剧或热点出现时触发路径微调或重构，从而在动态负载条件下提升端到端 QoS 的稳定性与保障能力。

4.2 改进架构介绍

本章在上一章动态子网划分与弹性网关部署的结果之上，进一步开展面向在线业务的路由增强设计。如图 4-1 所示，控制器在状态感知的支持下，利用既定的子网结构与边界网关集合构建跨子网转发骨架，并在路由管理模块中完成候选路径生成与下发。先前设计方案主要依赖子网网关骨架完成跨域通信，例如源节点 a 到目的节点 j 的基础跨域路径可沿 $a→c→e→h→j$ 转发，具备结构清晰与可控性强的优点，但在跨域流量占比上升或网关处出现汇聚时，易受到瓶颈链路与排队效应影响。为提升在线决策的适应性与鲁棒性，本文对路由管理模块进行功能增强，新增优先级划分、候选集融合与路径选择、VS 动态反馈机制三个子模块。其中，本文提出的 QVFER 算法由“候选集融合与路径选择“以及“VS 动态反馈机制”两部分协同构成。

具体而言，优先级划分子模块用于对在线到达的业务流进行分级管理，控制器接收流请求及其 QoS 约束参数（如延迟、抖动、丢包需求等），并据此完成优先级标记，为后续候选筛选与资源分配提供依据。候选集融合与路径选择子模块负责汇聚多来源候选路径并统一择优下发，以提升候选覆盖与拥塞绕避能力。该模块在骨架候选之外引入两类增强候选并与骨架候选统一融合比较：一方面，基于门户节点（网关与边界节点）形成逻辑跳转式的跨域替代候选，为骨架受阻时提供结构化绕行路径，例如图中紫色示意对应 a 到 c 后经子网 2 边界节点 g 进入子网 3 边界节点 i 并到达 j，形成 $a→c→e→g→i→j$ 的一条候选。另一方面，引入 PSI 路径摘要索引以复用历史优质路径，降低在线计算开销并提升稳定性，例如图中绿色示意可对应 $a→b→f→i→j$ 的复用候选。三类候选在控制器侧形成统一候选池后，模块对候选路径进行综合比较与择优，输出最终可下发的转发路径并更新资源占用状态。VS 动态反馈机制子模块用于在控制周期内监测链路波动并识别热点瓶颈，当关键链路触发阈值时，对受影响流进行路径微调或必要的重构，以分散汇聚压力并稳定端到端 QoS 表现。

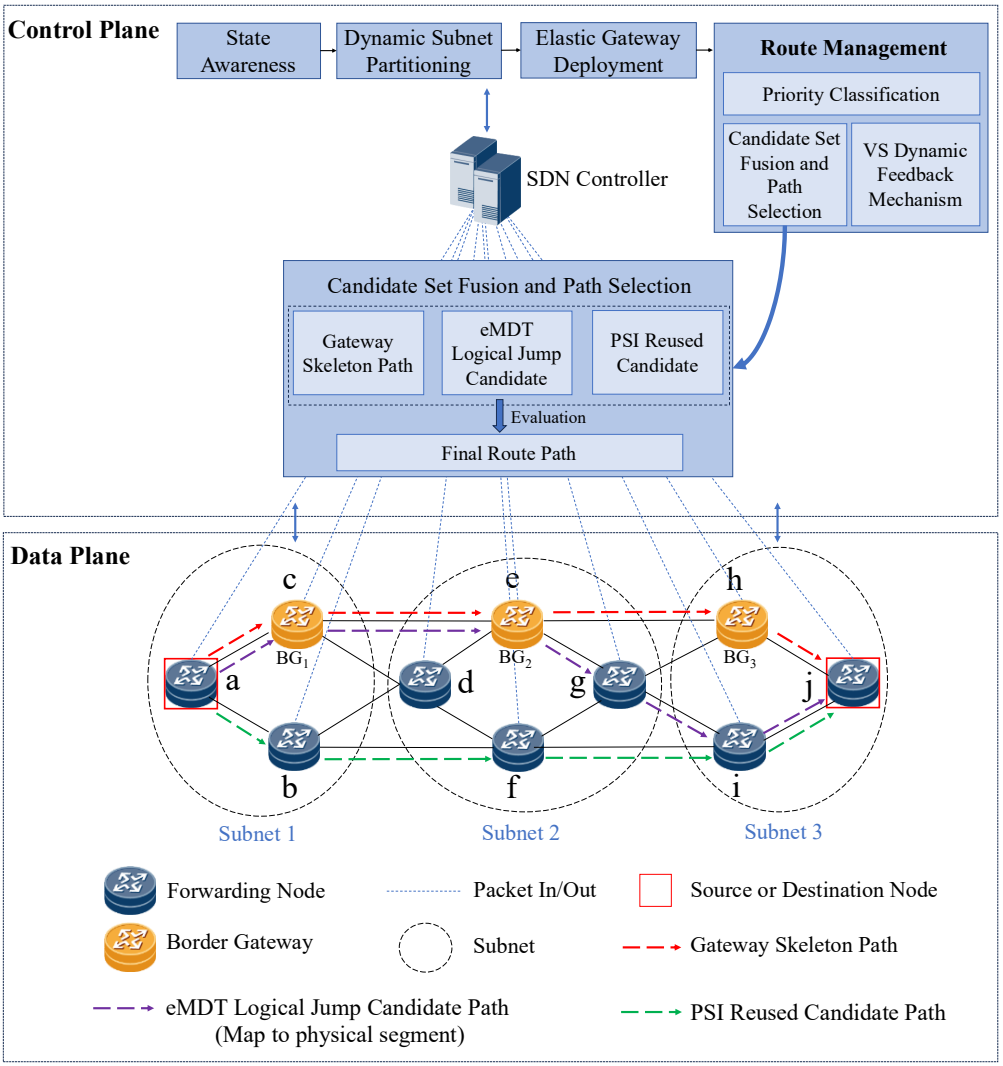


图 4-1 路由管理模块改进架构

4.3 网络模型定义

本文假设数据中心网络在物理层抽象为无向连通图$G(V,E)$，其中，V表示交换节点集合，E表示链路集合。对任意物理链路$e_{ij} \in E$，其静态资源参数包括链路带宽容量C_{ij}、基础传播延迟d_{ij}、背景丢包率p_{ij}与基线抖动j_{ij}。在上一章中已给出动态子网划分与边界网关部署结构，本文在此基础上开展在线路由增强。记子网集合为M_i={M_1,M_2,...,M_K}，满足$\bigcup_{i=1}^{K} M_i = V$且$M_i \cap M_{i'} = \emptyset$。每个子网$M_i$配置边界网关$g_i \in G_b$，用于承接跨子网通信并构成跨域转发骨架。

为刻画在线决策过程，本文采用离散控制周期对网络状态演化进行建模。设控制周期索引为$\tau = 1,2,\ldots$，周期长度为ΔT。在每个周期开始时，控制器从数据面采集链路运行状态，并在滑动窗口上计算用于稳定刻画链路质量的统计量。对任意链路e_{ij}在周期τ的状态向量定义为：

$$s_{ij}(\tau) = \left[u_{ij}(\tau), \tilde{d}_{ij}(\tau), \tilde{j}_{ij}(\tau), \tilde{\ell}_{ij}(\tau)\right] \tag{4-1}$$

其中$u_{ij}(\tau)$表示链路利用率，用于刻画资源占用水平；$\tilde{d}_{ij}(\tau)$、$\tilde{j}_{ij}(\tau)$、$\tilde{\ell}_{ij}(\tau)$为滑动窗口内的延迟统计量、抖动统计量与丢包统计量，用于反映链路的服务质量特征。

需要指出的是，在线路由选择不仅依赖当前观测到的链路状态，还必须考虑新流接入后对链路负载与排队的扰动。因此，在候选路径评估阶段，本文采用“分配后预测口径”对链路质量进行估计，即在给定候选分配方案$A(\tau)$的情况下，对链路的延迟、抖动与丢包进行前瞻性预测。为保持模型的通用性并避免绑定特定预测器，本文将预测过程抽象为状态驱动函数形式：

$$\tilde{d}_{ij}(\tau) = \Phi_d\left(s_{ij}(\tau), A(\tau)\right) \tag{4-2}$$

$$\tilde{j}_{ij}(\tau) = \Phi_j\left(s_{ij}(\tau), A(\tau)\right) \tag{4-3}$$

$$\tilde{\ell}_{ij}(\tau) = \Phi_\ell\left(s_{ij}(\tau), A(\tau)\right) \tag{4-4}$$

其中，$\Phi_d(\cdot)$、$\Phi_j(\cdot)$、$\Phi_\ell(\cdot)$分别表示对分配后链路延迟、抖动与丢包的预测映射，其输入由两部分组成：一是周期τ的链路观测状态$s_{ij}(\tau)$，二是当前周期的候选分配方案$A(\tau)$。在业务层面，本文以控制周期为单位组织到达请求。定义在周期τ内到达的业务流集合为$\mathcal{F}(\tau) = \{f_1, f_2, \ldots, f_K\}$，对任意到达流$f_m \in \mathcal{F}(\tau)$，其属性定义为：

$$f_m = \langle s_m, t_m, b_m, SLA_m, size_m \rangle \tag{4-5}$$

其中，s_m与t_m分别为源节点与目的节点；b_m为带宽需求；$size_m$为业务规模（MB），用于刻画传输数据量与资源占用强度。流的服务等级约束由三元组SLA_m给出：

$$SLA_m = (D_m^{max}, J_m^{max}, L_m^{max}) \tag{4-6}$$

其中$D_m^{\max}$、$J_m^{\max}$、$L_m^{\max}$分别表示延迟上限、抖动上限与丢包上限。本章使用的主要符号及其含义列于表 4-1。

表 4-1 主要符号汇总

符号	说明
V	交换节点集合
E	网络链路集合
τ	控制周期索引
$s_{ij}(\tau)$	链路e_{ij}在周期τ的状态向量
$\mathcal{F}(\tau)$	业务流集合
f_m	第m个业务流
ρ_m	业务流优先级标签
$\mathcal{P}_m$	业务流候选路径集
$x_{m,h}$	路径选择变量
λ_m	路径接入指示变量
$\alpha_{m,h}(i,j)$	边路径关联变量
$\alpha_m(i,j)$	链路占用变量

表 4-1 主要符号汇总（续）

符号	说明
$VS_{ij}(\tau)$	链路波动率指标
κ_m	PSI 列表模式键
$\mathcal{B}$	边界节点集合

4.4 问题描述

在控制周期τ内，给定基于动态子网划分与边界网关部署形成的子网网关转发架构、网络拓扑$G=(V,E)$、链路状态$\mathrm{s}_{ij}(\tau)$以及到达流集合$\mathcal{F}(\tau)$，本文研究 DCN 环境下多业务优先级并存的在线接入与路由选择问题。该问题需在子网内路由与跨子网骨架转发协同的基础上，为每条到达流联合确定接入决策与路径选择，并在路径部署后通过设计合理的反馈调节机制维持链路稳定性，以实现端到端 QoS 的持续保障。

4.5 问题建模

本节在前述定义基础上对在线路由决策进行建模。首先完成到达流的优先级划分，刻画差异化保障需求；随后构建在链路容量与多维 QoS 约束下的路由优化模型，明确决策变量与目标函数；最后设计基于链路波动率 VS 的动态反馈机制，为后续算法实现与性能分析提供理论支撑。

4.5.1 基于 WPCA 的流优先级划分

在资源竞争的在线场景中，不同业务对延迟、抖动与丢包的容忍度以及时效性要求存在显著差异，因而需要先对到达流的重要程度进行量化，以支撑后续的接入与路由决策[54]。本节采用基于 WPCA 的两阶段优先级划分方法。首先，针对每条到达流f_m，以其多维 SLA 门限与时效性参数构造特征向量：

$$q_m=[D_m^{max},J_m^{max},L_m^{max},\Delta_m]^{\mathrm{T}} \tag{4-7}$$

在第二章已给出 WPCA 的原理与求解过程，本文据此学习得到权重向量w，并将其用于综合刻画业务紧迫度。由此定义流f_m的优先得分：

$$pr_m=w^{\mathrm{T}}q_m \tag{4-8}$$

该得分越高表示业务对 QoS 约束与时效性的综合敏感性越强。进一步地，为便于在线策略执行与差异化保障，将连续得分pr_m通过分级映射函数$\psi(\cdot)$转换为离散优先级标签ρ_m：

$$\rho_m=\psi(pr_m) \tag{4-9}$$

其中$\psi(\cdot)$采用分位点规则实现，将业务流划分为高优先级（HIGH）与低优先级

（LOW）两类；得到的优先级标签将直接用于后续优化模型中的收益加权与调度约束设置，同时作为 VS 反馈调节时的保护依据，使高优业务在热点出现或波动加剧时获得更强的稳定保障。

4.5.2 优化模型构建

(1) 候选路径与决策变量描述

为保证在线求解的实时性与可实现性，在控制周期τ内，对每条到达流$f_m \in \mathcal{F}(\tau)$仅在有限候选路径集合上进行选择。设流$f_m$的候选路径集合为：

$$\mathcal{P}_m = \{p_m^1, p_m^2, \ldots, p_m^{H_m}\} \tag{4-10}$$

其中H_m为候选规模，p_m^h表示从源节点到目的节点的第h条候选路径。考虑跨子网通信的结构特征，$\mathcal{P}_m$由骨架约束下的跨域候选与子网内的局部增广候选共同构成，既保证跨域可达与可控，又为拥塞背景下的绕行保留必要的替代空间。为描述路径选择，引入二元决策变量：

$$x_{m,h} = \begin{cases} 1, & \text{若流} f_m \text{选择路径} p_m^h \\ 0, & \text{否则} \end{cases} \tag{4-11}$$

并据此定义接入指示量：

$$\lambda_m = \sum_{h=1}^{H_m} x_{m,h} \in [0,1] \tag{4-12}$$

其中$\lambda_m = 1$表示该流在本周期被接入且选定了一条路径，$\lambda_m = 0$表示拒绝接入。由于每条流至多选择一条路径，需满足：

$$\sum_{h=1}^{H_m} x_{m,h} \le 1, \forall f_m \in \mathcal{F}(\tau) \tag{4-13}$$

为将路径选择与链路占用建立对应关系，记$E(p_m^h) \subseteq E$为路径p_m^h所包含的边集合，并引入边路径关联指示函数：

$$\alpha_{m,h}(i,j) = \begin{cases} 1, & \text{若} e_{ij} \in p_m^h \\ 0, & \text{否则} \end{cases} \tag{4-14}$$

从而可将流f_m在链路e_{ij}上的占用指示写为：

$$\alpha_m(i,j) = \sum_{h=1}^{H_m} x_{m,h}\, \alpha_{m,h}(i,j), \forall e_{ij} \in E \tag{4-15}$$

显然，当且仅当流选择的路径经过链路e_{ij}时，$\alpha_m(i,j) = 1$，否则为 0。进一步考虑业务持续时间与控制周期离散化的关系。记T_m为流f_m的占用周期集合，引入占用状态变量：

$$y_m(\tau) \in \{0,1\}, \tau \in T_m \tag{4-16}$$

其中，$y_m(\tau)$用于表征流f_m在周期τ是否仍处于传输状态。由此，周期τ下链路e_{ij}的聚合占用可表示为：

$$\sum_{f_m \in \mathcal{F}(\tau)} \alpha_m(i,j)\, y_m(\tau), \forall e_{ij} \in E \tag{4-17}$$

(2) 优化目标建模

网络在高负载或跨域通信占比上升时，若仅追求总流量接入规模，容易将部分业务分配到高延迟、高丢包或高波动的共享链路上，进而诱发 SLA 违约与拥塞扩散；另一方面，若仅追求路径代价最小，则可能过度保守地拒绝可接入业务，降低吞吐与资源利用效率[55]。因此，本文采用总流量最大化与路径代价抑制相结合的加权目标，以统一刻画差异化保障与稳定性控制之间的权衡。本文的第一个优化目标是最大化总流量，其可定义为：

$$Maximize \sum_{m=1}^{K} \rho_m \lambda_m \tag{4-18}$$

其次，本文以最小化路径的综合代价作为第二个优化目标，用于抑制路径质量劣化并显式刻画共享链路波动带来的不稳定风险。对任意候选路径p_m^h本文定义其综合代价为：

$$c_{m,h}(\tau) = \omega_d \frac{\tilde{d}_{ij}(\tau)}{D_m^{max}} + \omega_j \frac{\tilde{j}_{ij}(\tau)}{J_m^{max}} + \omega_\ell \frac{\tilde{\ell}_{ij}(\tau)}{L_m^{max}} + \omega_{vs} \widetilde{VS}(\tau) \tag{4-19}$$

其中前三项分别对延迟、抖动与丢包进行 SLA 归一化约束化度量，使不同业务尺度可比[56]；最后一项用于度量路径共享链路的波动聚合风险，从而在候选选择阶段主动规避热点链路引发的 QoS 不稳定情况。基于式(4-19)，本文定义在候选集合上最小化路径代价为：

$$Minimize \sum_{m=1}^{K} \sum_{h=1}^{H_m} x_{m,h} c_{m,h}(\tau) \tag{4-20}$$

在上述两个目标的基础上，本文提出了一个联合优化公式，如下所示：

$$Maximize\ Z = \omega_1 \sum_{m=1}^{K} \rho_m \lambda_m - \omega_2 \sum_{m=1}^{K} \sum_{h=1}^{H_m} x_{m,h} c_{m,h}(\tau) \tag{4-21}$$

其中ω_1、ω_2分别用于调节流量接入规模与代价控制的权衡强度。优化公式受制于以下约束条件：

s.t. 式(4-13)

$$\omega_1 + \omega_2 = 1 \tag{4-22}$$

$$\sum_{f_m \in \mathcal{F}(\tau)} b_m \alpha_m(i,j)\, y_m(\tau) \le C_{ij}, \forall e_{ij} \in E, \forall \tau \in T \tag{4-23}$$

$$\tilde{d}_{ij}(\tau) \le D_m^{max}, \tilde{j}_{ij}(\tau) \le J_m^{max}, \tilde{\ell}_{ij}(\tau) \le L_m^{max} \tag{4-24}$$

$$\sum_{h=1}^{H_m} x_{m,h} \phi_{m,h} \ge \delta_m \lambda_m \tag{4-25}$$

公式(4-13)用于约束接入与路径选择的一致性，保证每条到达流在周期内至多选择一条候选路径且仅在接入时才会占用路径资源，从而避免多路径并占或未接入却占用资源的不合理情况；公式(4-22)规定两类目标的权重满足$\omega_1 + \omega_2 = 1$，

用于在总流量最大化和路径综合代价之间进行可控权衡，防止优化单纯追求吞吐而累积 QoS 风险或过度保守而降低承载能力；公式(4-23)限定任意周期内各链路上的在传流带宽占用之和不超过C_{ij}，从资源层面保证分配方案可执行且不发生链路过载；公式(4-24)确保被接入业务的预测延迟、抖动与丢包率分别不超过其 SLA 上限，以避免吞吐导向下接入不可满足服务质量的业务；公式(4-25)为跨域转发柔性可行性约束，用于表达跨域可走但不强制走网关的结构要求，对跨域流而言，需选择至少一条满足$\phi_{m,h} = 1$的候选路径；其中δ_m为跨域判定量，$\phi_{m,h}$为跨域段可行性标记，该约束保证跨域可达并避免网关成为硬性瓶颈。

本文所提出的联合优化问题，旨在在满足链路容量、连通性与多维 QoS 约束的前提下，对到达流集合在有限候选路径集上进行接入与路径选择，并综合权衡总流量收益与路径代价以获得稳定可控的在线转发决策。由于模型包含路径选择与接入判定等 0–1 决策变量，且约束主要由链路容量、跨域可达性及 SLA 门限等线性形式构成，因此该问题可建模为混合整数线性规划模型（MILP）。

定理：本文所提出的优化模型为 NP-hard。

证明：考虑该 MILP 的一个特例：忽略多维 QoS 门限与波动反馈等附加机制，仅保留链路容量约束并要求对多条业务流在其候选路径集合上选择可行路径以完成接入，则该特例可归约为带容量约束的多商品流/无分裂路由选择问题[50]；该类问题已被严格证明为 NP-hard。由于上述特例是本文模型的子问题，而本文模型在此基础上进一步引入跨域可达性约束、SLA 可行性判别与路径代价择优等限制，使得可行域不增反减、求解复杂度不低于该 NP-hard 子问题，因此本文所提出的 MILP 模型同样具有 NP-hard 性质。

4.5.3 VS 反馈机制设计

在多业务并发场景下，链路利用率$u_{ij}(\tau)$能够反映占用水平，但难以刻画短时间尺度的起伏与突变，而这类波动更容易通过队列积压表现为延迟抖动与丢包的波动[57]。为增强在线决策对链路态势变化的敏感性，本文引入链路波动率指标 VS，并将其在控制周期τ下对链路e_{ij}的取值记为$VS_{ij}(\tau)$，以提升路由策略在负载快速变化场景下的服务质量稳定性。

在每个控制周期τ内，控制器从数据平面采集链路e_{ij}的利用率序列，并在长度为W的滑动窗口上计算方差以刻画波动强度：

$$VS_{ij}(\tau) = Var\left(u_{ij}(\tau - W + 1), \dots, u_{ij}(\tau)\right) \tag{4-26}$$

其中$u_{ij}(\tau)$按链路容量C_{ij}归一化，以保证不同容量链路的波动量可比。考虑在线采样噪声与偶发尖峰对方差估计的影响，对$VS_{ij}(\tau)$进一步进行指数平滑，得到用于决策的稳态波动量：

$$\bar{VS}_{ij}(\tau) = \rho\bar{VS}_{ij}(\tau-1) + (1-\rho)VS_{ij}(\tau), \rho \in (0,1) \tag{4-27}$$

为将调节聚焦于对端到端质量影响更显著的共享瓶颈链路，本文采用波动强度与共享压力联合识别热点链路。首先，当链路满足$\bar{VS}_{ij}(\tau)$高于所设置的阈值η且利用率处于较高水平时，将其视为高风险候选链路。本文进一步定义周期τ内经过链路e_{ij}的在传业务集合为：

$$\mathcal{F}_{ij}(\tau) = \{f_m \mid \alpha_m(i,j) = 1, y_m(\tau) = 1\} \tag{4-28}$$

并以高优业务的聚合带宽刻画共享压力：

$$H_{ij}(\tau) = \sum_{f_m \in \mathcal{F}_{ij}(\tau)} \mathbb{I}(\rho_m = HIGH) \cdot b_m \tag{4-29}$$

其中$\mathbb{I}(\cdot)$为指示函数，用于筛选高优业务对共享压力的贡献，其定义为：

$$\mathbb{I}(\rho_m = HIGH) = \begin{cases} 1, & \rho_m = HIGH \\ 0, & \text{否则} \end{cases} \tag{4-30}$$

因此$H_{ij}(\tau)$实际上表示链路e_{ij}上在传高优业务的带宽需求总和，用于衡量该链路在关键业务层面的共享转发压力；当$|\mathcal{F}_{ij}(\tau)|$与$H_{ij}(\tau)$同时超过阈值时，将该链路纳入热点集合$\mathcal{E}(\tau)$。

在候选路径评估阶段，本文已将$\bar{VS}_{ij}(\tau)$以风险项并入路径综合代价，用于在满足 SLA 约束的可行集合内抑制高波动链路的选择倾向。路径部署后，控制器持续监测链路波动并构造热点集合$\mathcal{E}(\tau)$。当热点集合$\mathcal{E}(\tau)$非空时，控制器启动基于 VS 的在线调节过程，并按业务优先级分层执行。对经过热点链路的高优业务，系统优先选取其中一部分作为核心业务并保持其路径不变，以保证关键业务的稳定性；随后以低优业务作为主要调节对象，在既有候选路径集合内执行轻量级路径微调与在线重构。具体做法是，将热点链路加入禁用集合，在满足链路容量与 SLA 约束的可行候选中重新选择替代路径，使低优业务优先绕开热点链路以释放压力；若低优业务调整后热点仍未缓解，则再对未被保护的高优业务进行有限次数的重选，从而在不显著扰动核心业务的前提下完成进一步降压。对于候选集合内不存在可行替代路径的业务，维持原路径不变。为抑制短时间内反复切换引发的振荡，调节过程限制单个控制周期内的最大重选业务数。调节完成后，通过先占用新路径再释放旧路径的方式同步更新链路带宽占用与业务状态，为下一周期的链路指标预测与$VS_{ij}(\tau)$计算提供一致输入。

4.6 问题求解

4.6.1 QVFER 算法描述

尽管本文中已建立的 MILP 模型为解决优化问题提供了一个结构化的解决方案，但在实际应用中，特别是在面对动态变化的环境时，MILP 模型的求解效率和灵活性往往受限。为此，本文提出 QVFER 算法，在保证跨域可达性与计算

开销可控的前提下，通过候选池构造、可行性筛选、目标评分择优以及波动反馈调节等策略，实现对动态负载的自适应路由决策。

(1) 候选路径构造与融合机制

本文沿用子网内与子网间两级生成思路构造基础候选路径集：对于源/目的地址位于同一子网的请求流，仅在对应子网内生成若干条子网内候选；对跨子网请求流，则先在子网骨架上确定跨域转发段，再在两端子网内补全接入段，从而保证候选路径的结构可控与可达性一致。需要指出的是，该基础机制在候选覆盖率与在线开销之间仍存在折中，单一来源易导致候选同质化或在负载波动时可行性下降[58]。为扩大候选池并提升在线决策的鲁棒性，本文进一步引入两类增强候选：面向高频通信模式的 PSI 路径复用候选，以及基于虚拟坐标与 kNN 逻辑跳转的 eMDT 候选，并在控制器侧与基础候选进行融合筛选，以在可控规模内获得更丰富的跨域替代路径。

PSI 的目标是在不扩大在线搜索空间的前提下，将高频通信模式下已验证的优选路径以摘要形式预先组织起来，在线到达时通过键匹配、检索与轻量初筛直接得到可复用候选[59]，从而显著降低候选生成开销。对控制周期τ内到达的业务流$f_m = \langle s_m, t_m, \cdot \rangle$，本文以源宿所属子网与优先级类型构造模式键：

$$\kappa_m = \big(\sigma(s_m), \sigma(t_m), \rho(f_m)\big) \tag{4-31}$$

其中$\sigma(\cdot)$表示节点所属子网映射，$\rho(f_m)$表示优先级类型。由此 PSI 可抽象为从κ到 Top-K 路径摘要列表的映射：

$$PSI:\ \kappa \mapsto \{(P_1, \eta_1), \ldots, (P_K, \eta_K)\} \tag{4-32}$$

其中P_k为历史周期内在相同通信模式下被反复验证过的端到端路径，η_k为其轻量摘要。PSI 采用桶式组织方式，每个桶最多维护 *K* 条路径记录，并为每条记录维护质量分数 *Score*、最后更新时间戳 *t* 与子网划分版本号 *ver*。当路径成功下发后，控制器将其写入对应桶；若桶已满，则按最久未使用或质量最差策略淘汰一条记录以保持表规模恒定。为适应子网重划分带来的映射变化，本文不对 PSI 进行清空，而是维护全局版本号 Partition_version，在每次重划分后自增，并将写入时的版本号同步写入记录字段 *ver*。查询阶段优先返回 *ver* 与当前 Partition_version 一致的路径作为候选；若同版本未命中，则仅从旧版本中抽取少量路径作为弱候选，并先进行连通性与剩余容量的快速一致性检查，通过后才参与候选融合，从而在保留复用收益的同时避免旧路径被不加判别地直接采用。图 4-2 给出了 PSI 的桶式存储结构及对应的路径下发示意。

以图 4-2(a)中桶（S1→S3, HIGH）为例，PSI 维护了两条同版本路径记录 *a→b→f→i→j* 与 *a→c→e→h→j*（ver=2），以及一条旧版本记录 *a→c→d→g→i→j*（ver=1）。当 Partition_version 为 2 时，控制器优先命中同版本记录并选取其中评分最优的路径作为候选输出（即图中 *a→b→f→i→j* 这条路径）；当子网重划分使 Partition_version 更新为 3 后且同版本路径无法命中，则控制器从旧版本中抽

取少量路径进行快检，若在新骨架下仍连通且链路剩余容量满足需求，则以弱候选形式加入候选池，否则丢弃。

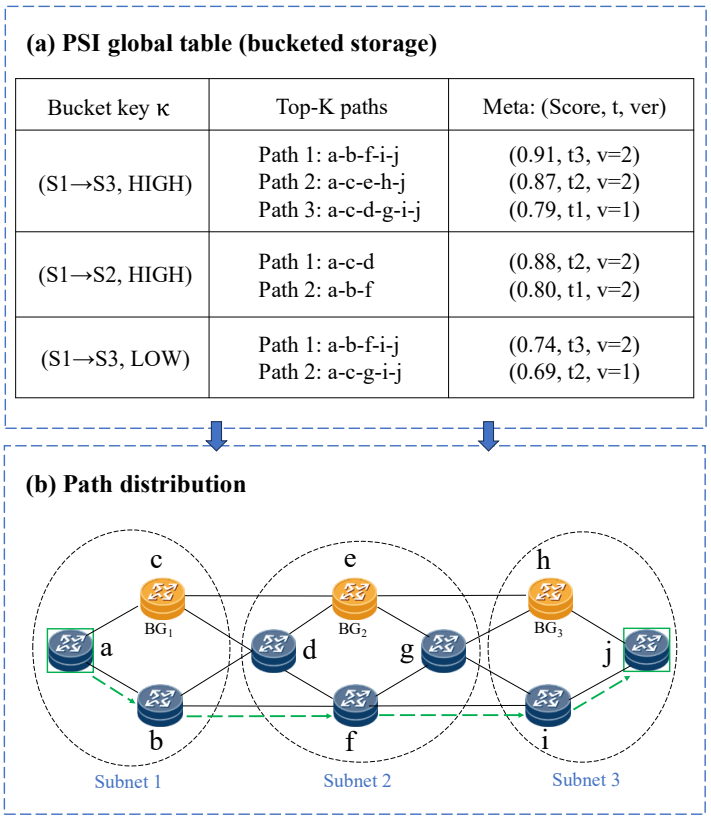


图 4-2 PSI 桶式存储结构及对应的路径下发

当 PSI 的复用候选覆盖不足，或网络状态变化导致复用路径在当前周期可行性下降时，需要补充结构可控且具备绕行能力的新候选。eMDT 的思路是将关键节点映射到虚拟坐标空间[60]，用延迟代价刻画节点间的邻近关系，并在此基础上构建跨网段逻辑跳转图；控制器在图上搜索少跳逻辑路径作为跳转骨架，再将逻辑跳转映射为满足骨架可达性与资源约束的物理可行段[61]，最终拼接生成可下发的端到端候选路径。

考虑到在线复杂度，eMDT 不对全网节点构图，而仅对关键节点集合$\tilde{V}$进行处理。本文将关键节点划分为网关集合$\mathcal{G}$与边界节点集合$\mathcal{B}$，并对任意$v \in \tilde{V} = \mathcal{G} \cup \mathcal{B}$定义虚拟坐标嵌入：

$$z_v = \Phi(v) \in \mathbb{R}^d, v \in \tilde{V} \tag{4-33}$$

其中$\Phi(\cdot)$为嵌入映射，用于在虚拟空间中保持节点延迟语义的相对结构。为使 kNN 的选边标准直接反映跳转代价，本文将 eMDT 的逻辑边权定义为当前控制周期下的预测延迟代价：

$$w(u,v) = \widehat{D_\tau}(u,v) \tag{4-34}$$

其中$\widehat{D}_\tau(u,v)$可由当前链路延迟估计得到的最短段延迟给出。基于上述定义 eMDT 采用约束 kNN 构建跨网段逻辑跳转边，并通过跨网段约束避免与网关骨架重复建边。记节点所属网段为$\sigma(\cdot)$，则对网关节点与边界节点分别采用如下选

边规则。

对任一网关节点$u \in \mathcal{G}$，其近邻仅在不同网段的边界节点集合中选取，选择k_g个最小延迟邻居：

$$\mathcal{N}_{k_g}^{\mathcal{G}}(u) = \arg \min_{\substack{S \subseteq \mathcal{B} \\ |S|=k_g, \sigma(v)\neq\sigma(u)}} \sum_{v \in S} w(u,v), u \in \mathcal{G} \tag{4-35}$$

该约束使网关节点不会优先选择网关节点作为近邻，从而避免产生与网关骨架重复的“网关-网关”逻辑边，并将逻辑跳跃聚焦于跨网段转移。对任一边界节点$u \in \mathcal{B}$，其近邻仅在不同网段的边界节点集合中选取，选择k_b个最小延迟邻居：

$$\mathcal{N}_{k_b}^{\mathcal{B}}(u) = \arg \min_{\substack{S \subseteq \mathcal{B} \setminus \mathcal{B}_{\sigma(u)} \\ |S|=k_b}} \sum_{v \in S} w(u,v), u \in \mathcal{B} \tag{4-36}$$

其中$\mathcal{B}_{\sigma(u)} = \{v \in \mathcal{B} \mid \sigma(v) = \sigma(u)\}$。由此可使边界到边界的逻辑跳跃仅发生在不同网段之间，避免在网段内部形成冗余的局部连接。综合两类选边，eMDT 的逻辑边集合可表示为：

$$\tilde{E} = \left\{(u,v) \mid v \in \mathcal{N}_{k_g}^{\mathcal{G}}(u), u \in \mathcal{G}\right\} \cup \left\{(u,v) \mid v \in \mathcal{N}_{k_b}^{\mathcal{B}}(u), u \in \mathcal{B}\right\} \tag{4-37}$$

据此本文可以得到延迟加权的逻辑跳转图$G_{\text{eMDT}} = (\tilde{V}, \tilde{E}, w)$。在逻辑图上，控制器针对跨子网业务流搜索跳数受限的逻辑路径$\omega = \langle v^{(0)}, \ldots, v^{(h)} \rangle$作为跳转骨架。由于逻辑边不要求对应物理拓扑中的单条链路，控制器需将每条逻辑跳转$\left(v^{(l)}, v^{(l+1)}\right)$映射为满足骨架可达性与当前周期资源约束的物理可行段。为保持表述简洁，将映射统一记为：

$$\text{map}\left(v^{(l)}, v^{(l+1)}\right) \Longrightarrow \text{a feasible physical segment} \tag{4-38}$$

映射实现采用子网内最短路段或骨架约束下的最小代价段，并在映射过程中同步检查容量与关键 SLA 阈值，最终将各物理段按序拼接形成可下发的端到端 eMDT 候选路径。

结合图 4-3 的示例可以更直观地理解上述过程。图中网关节点为 c、e、h，边界节点为 b、d、f、g、i，并给出了关键节点的虚拟坐标分布及约束 kNN 的连边结果，其中灰色连边表示网关到边界的逻辑跳转，蓝色连边表示边界到边界的跨网段跳转，边上数字为对应的延迟代价。在$k_g = 2, k_b = 1$的设置下，网关节点仅在其他网段的边界节点中选择少量低代价邻居，从而形成跨网段的入口跳转边；边界节点则在其他网段的边界节点中选择最小代价邻居，补充跨网段的直接跳跃通道。基于这些逻辑边，控制器能够在 eMDT 图上快速得到一条少跳逻辑骨架，例如 $c \rightarrow g \rightarrow i$。随后通过映射将逻辑跳转转化为物理可行段，例如逻辑跳转 $c \rightarrow g$ 可由物理段 $c \rightarrow e \rightarrow g$ 实现，与 $g \rightarrow i$ 的物理段拼接后即可形成可下发的端到端候选路径，从而在不显著增加在线计算开销的前提下，为骨架网关转发与 PSI 复用候选提供低跳数替代路径补充，提升候选池的覆盖率与绕行冗余。

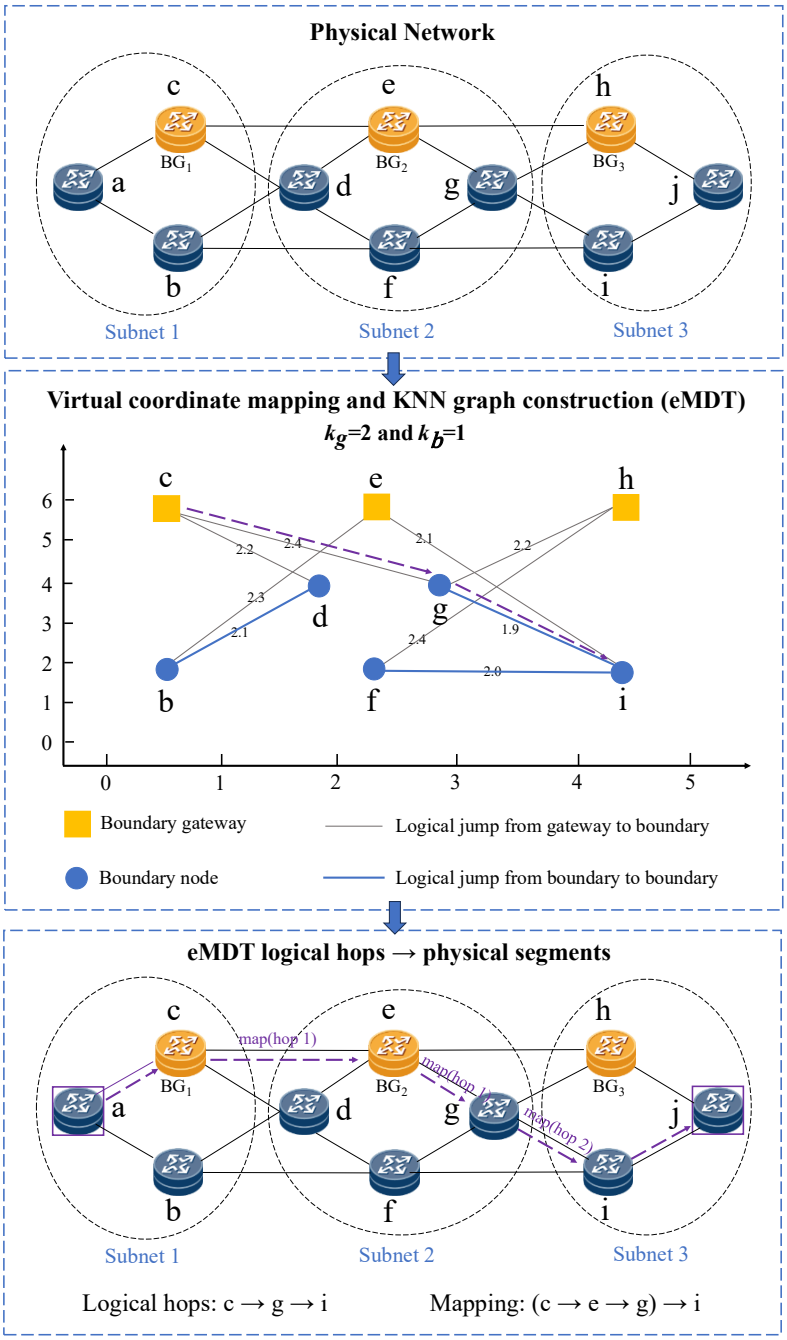


图 4-3 eMDT 构图与逻辑跳转映射示意

算法 1 给出了候选路径池构建与融合生成的执行流程。该算法首先根据源/目的地址是否位于同一子网生成基线候选：同一子网内仅在对应子图中产生子网内候选，跨子网则在骨架层确定网关转发序列并补全两端接入段（见第 2-6 行）。随后以模式键κ_m在 PSI 中快速检索，优先返回与当前Partition_version一致的 Top-K 复用路径；若无同版本命中，则从旧版本中抽取少量记录作为弱候选，并经连通性与剩余容量快检后再纳入候选集合（见第 7-16 行）。最后在关键节点集$\tilde{V} = \mathcal{G} \cup \mathcal{B}$上构建延迟加权的 eMDT 逻辑图，按约束 kNN 生成跨网段跳转边并在跳数上限内搜索少跳逻辑骨架，将其逐跳映射并拼接为物理可行路径，通过快检

后加入$\mathcal{K}_m$（见第 17-28 行）。通过三类来源的并行补充与融合筛选，最终形成完整的候选路径池。

算法 1：候选路径池构建与融合生成

输入：包含交换机节点与链路边的网络图$G(V,E)$，网关集合$\mathcal{G}$，边界集合$\mathcal{B}$，到达业务流f_m，PSI 表，Partition_version，参数K、k_g、k_b、s_m、t_m

输出：统一候选路径池$\mathcal{K}_m$

1: $\mathcal{K}_m \leftarrow \emptyset$
2: if $\sigma(s_m) = \sigma(t_m)$ then
3: $\mathcal{K}_m \leftarrow \mathcal{K}_m \cup$ 子网内候选生成(s_m, t_m)
4: else
5: $\mathcal{K}_m \leftarrow \mathcal{K}_m \cup$ 骨架候选生成$(s_m, t_m, \mathcal{G})$
6: end if
7: $\kappa_m \leftarrow (\sigma(s_m), \sigma(t_m), \rho(f_m))$
8: $\mathcal{P} \leftarrow$PSI.查询$(\kappa_m,$ Partition_version$, K)$
9: if $\mathcal{P} = \emptyset$ then
10: $\mathcal{P} \leftarrow$PSI.弱查询(κ_m, K)
11: end if
12: for 每一条$P \in \mathcal{P}$ do
13: if 快速一致性检查(P, b_m) then
14: $\mathcal{K}_m \leftarrow \mathcal{K}_m \cup P$
15: end if
16: end for
17: $\tilde{V} \leftarrow \mathcal{G} \cup \mathcal{B}$
18: for 每个$u \in \tilde{V}$ do
19: $z_u \leftarrow \Phi(u)$
20: end for
21: 构建$G_{\mathrm{eMDT}} = (\tilde{V}, \tilde{E}, w)$，采用 kNN，且$w(u,v) = \widehat{D_r}(u,v)$
22: $\Omega \leftarrow$搜索少跳逻辑路径$(G_{\mathrm{eMDT}}, s_m, t_m)$
23: for 每个逻辑路径$l \in \Omega$ do
24: $P_l \leftarrow$拼接$\big(\mathrm{map}(l)\big)$
25: if 快速一致性检查(P_l, b_m) then
26: $\mathcal{K}_m \leftarrow \mathcal{K}_m \cup P_l$
27: end if
28: end for
29: return $\mathcal{K}_m$

(2) QVFER 算法主流程描述

在算法 1 已生成候选路径池$\mathcal{K}_m$的基础上，QVFER 接下来的核心任务是从多源候选中快速选出可下发路径，并在负载波动时进行轻量调节。算法 2 给出了 QVFER 算法的执行流程。具体而言，控制器首先读取算法 1 输出的候选池$\mathcal{K}_m$，并依据式(4-23)与式(4-24)对每条候选路径进行可行性判别，统一检查链路容量约束与 SLA 约束，筛得可行候选集合（见第 1-3 行）。若可行集合为空，则该流在当前周期被拒绝（见第 4-7 行）；否则，控制器采用式(4-21)对可行候选计算综合评分并选择最优路径下发（见第 8-9 行）。为增强对异常状态的鲁棒性，若候选路径存在对式(4-13)或式(4-25)的违反（例如出现不允许的结构条件），则在式(4-21)的基础上叠加惩罚项，使其在择优过程中被显著抑制（见第 10-12 行）。路径下发后，控制器同步更新资源占用状态，并将成功路径写回 PSI 对应桶（携带当前划分版本号），以提升后续同类通信模式到达时的复用效率（见第 14-16 行）。在周期末，控制器基于链路波动率 VS 识别热点链路集合；当热点集合非空时，触发 VS 在线调节过程（见算法 3）。

算法 2：QVFER 算法主流程

输入：包含交换机节点与链路边的网络图$G(V,E)$，到达流集合$\mathcal{F}(\tau)$，链路状态（容量与VS），PSI 表与 Partition_version

输出：最终下发路径P^*与资源更新结果

1: for 每一条$f_m \in \mathcal{F}(\tau)$ do
2: $\mathcal{K}_m$←执行算法 1
3: $\mathcal{K}_m^f$←在$\mathcal{K}_m$上按式(4-23)、(4-24)进行可行性判别
4: if $\mathcal{K}_m^f = \emptyset$ then
5: 拒绝该流
6: continue
7: end if
8: for 每一条$P \in \mathcal{K}_m^f$ do
9: *Score*(P)←按照式(4-21)计算综合得分
10: if P违反式(4-13)或(4-25) then
11: *Score*(P)←*Score*(P) - *Penalty*
12: end if
13: end for
14: P^*←从 $\mathcal{K}_m^{fea}$中选取 *Score*(P)最大的路径
15: 下发P^*，更新链路资源占用与状态
16: 将P^*写回 PSI（携带当前 Partition_version）
17: end for

18: 刷新链路状态并计算VS，得到热点链路集合$\mathcal{E}(\tau)$
19: if $\mathcal{E}(\tau) \neq \emptyset$ then
20: 调用算法 3
21: end if
22: return

算法 3 给出了 VS 反馈调节的执行流程。该过程不重新生成候选池，而是直接在既有候选集合上进行轻量调整：首先冻结少量核心高优业务的既有路径以保持稳定（见第 2 行），随后对其余受影响流在原候选池中屏蔽经过热点链路的候选（见第 5 行），并按式(4-23)(4-24)重新判别可行性、按式(4-21)重新计算评分择优，从而快速下发替代路径以释放热点压力（见第 6-16 行）。

算法 3：VS 在线调节机制

输入：包含交换机节点与链路边的网络图$G(V,E)$，热点链路集合$\mathcal{E}(\tau)$，周期内已下发路径与流集合
输出：更新后的路径与资源状态

1: 选出路径经过$\mathcal{E}(\tau)$的受影响流集合$\mathcal{F}_{hot}$，并按优先级排序
2: 冻结少量核心高优流的既有路径不变
3: for 每一条受影响流$f_m{}' \in \mathcal{F}_{hot}$ do
4: $\mathcal{K}_m$←读取该流候选池
5: $\mathcal{K}_m{}'$←从$\mathcal{K}_m$中屏蔽经过$\mathcal{E}(\tau)$的候选
6: $\mathcal{K}_m^{f}{}'$←对$\mathcal{K}_m{}'$按式(4-23)、(4-24)进行可行判别
7: if $\mathcal{K}_m^{f}{}' = \emptyset$ then
8: continue
9: end if
10: for 每一条$P \in \mathcal{K}_m^{f}{}'$ do
11: *Score*(P)←按照式(4-21)计算综合得分
12: if P违反式(4-13)或(4-25) then
13: *Score*(P)←*Score*(P) - *Penalty*
14: end if
15: end for
16: 选择 *Score*(P)最大的候选作为替代路径下发，更新资源状态
17: end for
18: return

4.6.2 算法复杂度分析

在子网与网关骨架约束下，三段算法的时间开销主要由子网内局部搜索、子网间骨架搜索以及候选池扫描共同决定，而不再直接受全网物理拓扑规模主导。设源节点所在子网的边数和节点数分别为m_i和n_i，目标节点所在子网的边数和节点数分别为m_j和n_j，子网间骨架图的边数和节点数分别为m_g和n_g。对于算法 1，其主要任务是生成候选路径集。当源节点与目标节点位于同一子网内时，最短路径仅需在该子网局部子图内计算，其时间复杂度为$O(m_i + n_i \log n_i)$；当二者分属不同子网时，则需依次完成源子网内部搜索、子网间骨架搜索和目标子网内部搜索，其复杂度为$O(m_i + m_g + m_j + n_i \log n_i + n_g \log n_g + n_j \log n_j)$。此外，设$K_p$表示 PSI 每个桶最多返回的候选路径条数，$l$表示单条候选路径的平均长度，$|\Omega|$表示 eMDT 返回的逻辑候选路径条数，$h$表示单条逻辑路径映射为物理路径时的平均长度，则 PSI 快检代价为$O(K_p l)$，eMDT 的映射与拼接代价为$O(|\Omega| hl)$。因此，算法 1 的总体时间复杂度可写为$T_1 = O(m_i + m_g + m_j + n_i \log n_i + n_g \log n_g + n_j \log n_j + K_p l + |\Omega| hl)$。

对于算法 2，其主要任务是在算法 1 生成的候选集基础上完成可行性判别与评分选择。设控制周期τ内到达的业务流数量为$|\mathcal{F}(\tau)|$，单条业务流对应的候选路径数为$|\mathcal{K}_m|$，由于可行性判别与评分计算均需沿候选路径线性扫描，因此单条业务流的处理复杂度为$O(T_1 + |\mathcal{K}_m| l)$，从而算法 2 在整个控制周期内的总体复杂度为$T_2 = O(|\mathcal{F}(\tau)| (T_1 + |\mathcal{K}_m| l))$。对于算法 3，其仅在检测到热点链路或波动异常后触发，不再重新生成候选路径，而是在既有候选池上对受影响流集合$\mathcal{F}_{hot}$执行局部重判别与重评分，因此其总体复杂度为$T_3 = O(|\mathcal{F}_{hot}||\mathcal{K}_m| l)$。

4.7 仿真设计结果及分析

本节将详细介绍 QVFER 算法的实验环境设置与结果评估。首先说明仿真实验平台与实现配置，包括操作系统、控制器与仿真工具链等基础环境；随后给出实验所采用的网络拓扑与请求流量设置，并对实验中涉及的关键参数与资源约束进行统一定义与说明。最后，通过与其他三类对比算法在多种拓扑场景下的对照实验，系统展示 QVFER 算法的综合性能与优势。

4.7.1 实验方法及仿真参数设置

本文仍在虚拟机 Ubuntu 18.04 环境下，基于 Ryu 4.34 控制器、Mininet 2.3.1b4 与 Python 3.11.7 搭建仿真平台，对 QVFER 算法的性能进行验证。实验拓扑来自 The Internet Zoo，选取 BICS 拓扑（33 个节点、48 条边）、DFN 拓扑（58 个节点、

87 条边）以及 Colt Telecom 拓扑（选取 100 个节点、122 条边）作为测试场景。流量到达过程采用泊松模型建模，平均到达率λ设置为 0.5~5.0 flow/s，步长为 0.5 flow/s。流请求带宽在 1~80 Mbps 范围内取值，业务规模设置为 100~1500 MB 以表征请求流的传输数据量，端到端 SLA 以多维上限阈值约束刻画，其中延迟上限为 5~20 ms，抖动上限为 0.5~5 ms，丢包率上限为 0.5%~1%[62]。实验中使用的交换机为千兆交换机，每条链路的带宽容量设定为 1000 Mbps。链路底噪 QoS 参数设置为基础传播延迟 0.1~0.4 ms、基础抖动 0.02~0.1 ms、基础丢包率 0.01%~0.5%[63]。实验的具体参数设置详见表 4-2 相关说明。

表 4-2 实验具体参数设置

参数	值
BICS 拓扑	33 个节点、48 条边
DFN 拓扑	58 个节点、87 条边
Colt Telecom 部分拓扑	100 个节点、122 条边
平均到达率	0.5~5 flow/s
流请求带宽	1~80 Mbps
业务规模	100~1500 MB
流请求延迟上限	5~20 ms
流请求抖动上限	0.5~5 ms
流请求丢包上限	0.5%~1%
交换机带宽容量	1000 Mbps
链路带宽容量	1000 Mbps
链路基础传播延迟	0.1~0.4 ms
链路基础抖动	0.02~0.1 ms
链路基础丢包率	0.01%~0.5%

4.7.2 对比算法介绍

为确保比较的有效性与公正性，本文在统一的仿真环境与评估标准下，选取相关工作中具有代表性的三种算法作为对比基线：

(1) PEDL[11]：该方法以优先级保障与负载均衡为目标，先对到达流按优先级排序，并为每条流生成满足能耗约束的候选能效路径集合；随后计算候选路径的负载离散度，选择离散度最小且容量可行的路径下发并更新资源。

(2) DFRDRL[23]：该方法融合模糊逻辑与深度强化学习进行在线路由决策，通过模糊规则综合延迟、带宽等多维状态以处理不确定性，并由 DRL 持续学习更新策略；当检测到拥塞时对高资源需求请求实施延后处理，待网络状态改善后重新评估与调度。

(3) SRDRL[25]：该方法基于深度强化学习构建包含网络状态、业务请求与

反馈信息的状态空间，学习 QoS 感知的路径选择策略以适应链路状态变化；同时对网络模型进行处理以降低决策复杂度，并结合交互反馈迭代更新路由策略。

4.7.3 评价指标

为全面评估 QVFER 算法在子网网关骨架上的综合性能，本文在验证上一章的端到端平均延迟、平均跳数与请求流执行时间等指标的基础上，进一步补充并定义如下关键评估指标：

(1) **接受率**：衡量算法的业务接纳能力，表示到达请求中成功满足约束、获得可行路径并完成资源分配的比例。接受率越高，说明同等负载下可服务的请求越多。

(2) **平均抖动**：表示已接纳请求在传输过程中端到端延迟随时间波动的平均水平，抖动越大说明链路排队与拥塞变化越剧烈、业务体验越不稳定。

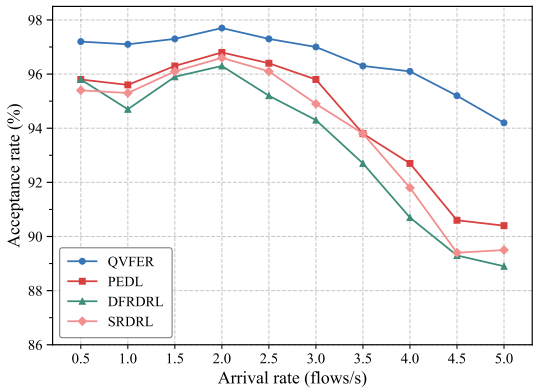
(3) **丢包率**：表示端到端传输过程中丢失的数据包数量占发送数据包数量的比例，直接反映链路拥塞、缓存溢出等因素对可靠传输的影响。

(4) **SLA 违反率**：衡量 QoS 保障能力，表示已接纳请求中在传输期间出现任一 SLA 指标超限的占比，通常包含延迟超限、抖动超限或丢包率超限等情形，反映算法在动态负载下的约束满足程度。

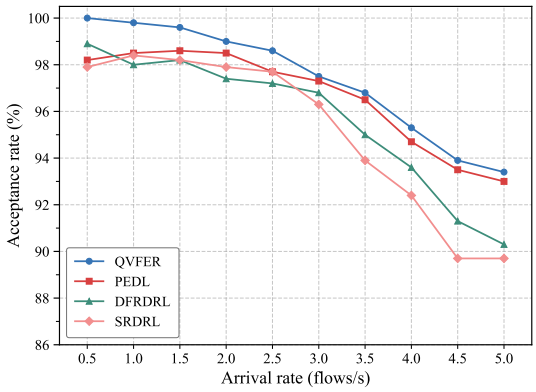
4.7.4 实验结果与分析

(1) 实验 1：接受率评估

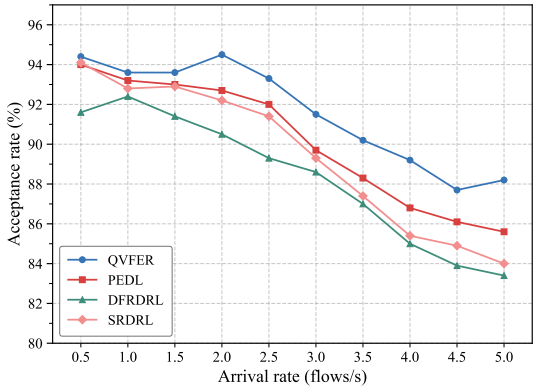
如图 4-4 所示，随着到达率由 0.5 提升至 5.0 flow/s，各拓扑下接受率均随负载增强而下降，但 QVFER 始终优于次优算法且在高负载端更稳定。以λ=5.0 为例，QVFER 在 BICS、DFN 与 Colt Telecom 下的接受率分别为 94.2%、93.4%与 88.2%，相较对应次优算法分别提升了 3.8%、0.4%与 2.6%。从 0.5~5.0 flow/s 全区间平均看，QVFER 在三种拓扑下相较次优算法的平均接受率分别提升 2.1%、0.7%与 1.5%，三拓扑汇总后的总体平均接受率提升约 1.43%。这一优势主要来自 QVFER 以可接纳总流量最大化为牵引进行路径择优分配，并结合链路波动率 VS 反馈对热点链路进行抑制与在线微调，使资源占用在路径与时域上更均衡，从而在负载升高时仍能维持更高的接纳水平。



a) BICS 拓扑下的接受率评估



b) DFN 拓扑下的接受率评估

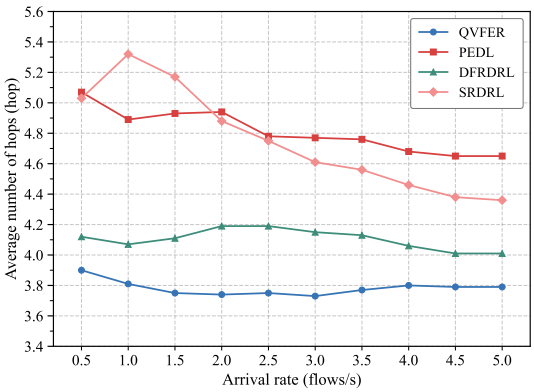


c) Colt Telecom 拓扑下的接受率评估

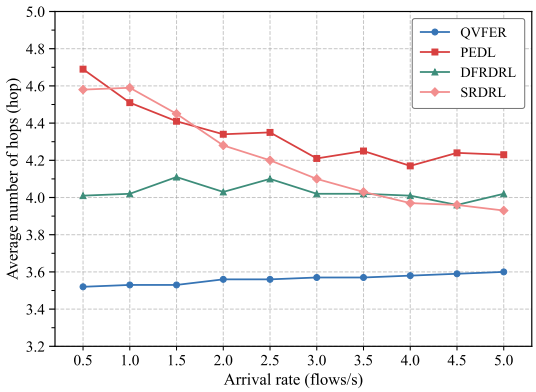
图 4-4 不同拓扑下的到达率变化对接受率的影响

(2) 实验 2：平均跳数评估

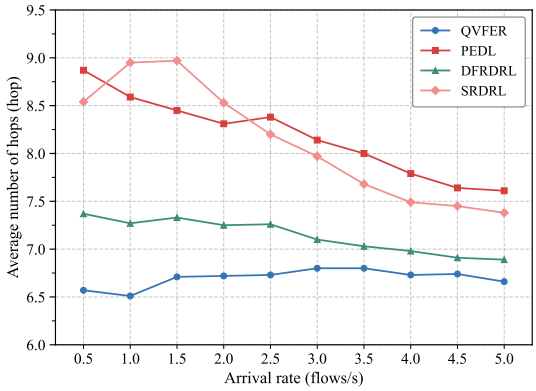
如图 4-5 所示，在到达率从 0.5 提升至 5.0 flow/s 的过程中，各算法的端到端平均跳数整体保持平稳或缓慢下降，但 QVFER 在三种拓扑下始终维持最低跳数水平。具体而言，BICS 拓扑下 QVFER 的平均跳数稳定在 3.73–3.90 之间，λ=5.0 时为 3.79，相比次优算法 DFRDRL 的 4.01 减少 0.22 跳，降幅约 5.49%，全区间平均减少 0.32 跳；DFN 拓扑下 QVFER 为 3.52–3.60，λ=5.0 时为 3.60，相比次优 DFRDRL 的 4.02 减少 0.42 跳，降幅约 10.45%，全区间平均减少 0.47 跳；Colt Telecom 拓扑下 QVFER 为 6.51–6.80，λ=5.0 时为 6.66，相比次优 DFRDRL 的 6.89 减少 0.23 跳，降幅约 3.34%，全区间平均减少 0.45 跳。将三种拓扑与全部到达率样本汇总后，QVFER 相较次优算法的总体平均跳数绝对减少约 0.41 跳，整体降幅约 8.58%。该优势主要源于 QVFER 在子网网关骨架上进行候选构造与择优，跨域转发优先沿骨架确定的紧凑连接关系完成拼接，减少传统全局搜索带来的冗余跳数，同时 eMDT 在逻辑层补充少跳候选并通过映射拼接将其落到局部最短段，使候选池中更易出现低跳可行路径。



a) BICS 拓扑下的平均跳数评估



b) DFN 拓扑下的平均跳数评估

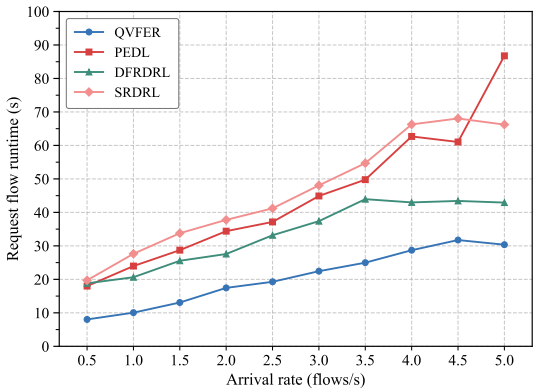


c) Colt Telecom 拓扑下的平均跳数评估

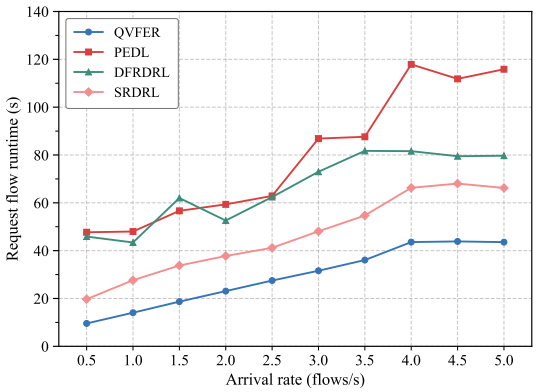
图 4-5 不同拓扑下的到达率变化对平均跳数的影响

(3) 实验 3：请求流执行时间评估

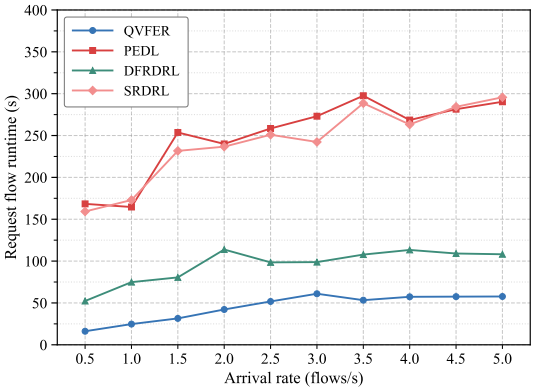
从请求流执行时间的变化趋势可以看出（图 4-6 所示），随着到达率持续增大，各算法的执行时间均有所上升，但 QVFER 在三种拓扑下始终保持最低水平，且曲线增长更为平缓，说明其在高负载条件下对请求完成时长的控制更稳定。以 λ=5.0 为例，QVFER 在 BICS、DFN 与 Colt Telecom 拓扑下的请求流执行时间分别为 30.35 s、43.56 s 和 57.71 s，相较对应次优算法分别缩短 29.30%、34.22%和 46.64%；从 0.5 至 5.0 flow/s 全区间平均水平看，时间分别缩短 38.74%、37.08%和 52.64%，三种拓扑汇总后的总体平均缩短约 42.82%。这一优势主要在于 QVFER 利用了子网划分的结构特性，使子网内业务可直接在局部子网范围内完成路径查询与候选生成，减少了全网搜索带来的额外决策开销。与此同时，PSI 能够复用历史可行路径，减少重复计算，eMDT 进一步补充少跳候选路径，使得请求在路径构造与下发阶段均具有更高效率。



a) BICS 拓扑下的请求流执行时间评估



b) DFN 拓扑下的请求流执行时间评估

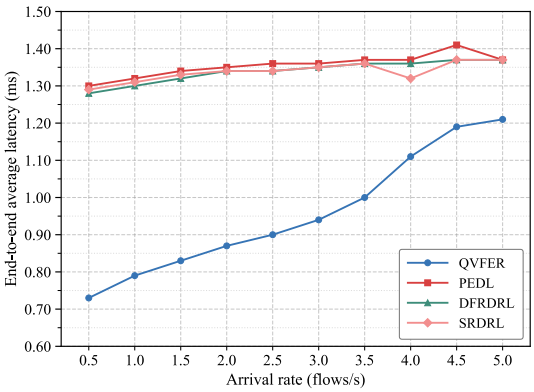


c) Colt Telecom 拓扑下的请求流执行时间评估

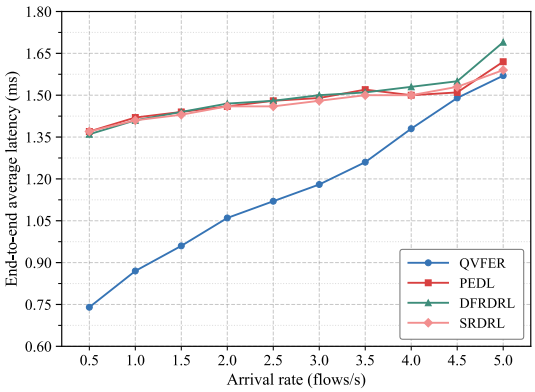
图 4-6 不同拓扑下的到达率变化对请求流执行时间的影响

(4) 实验 4：端到端平均延迟评估

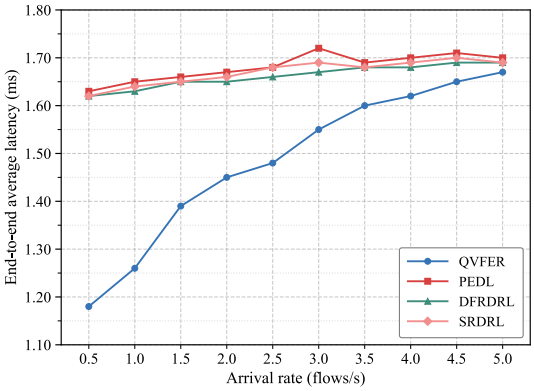
从图 4-7 中可以看出，QVFER 在端到端平均延迟上的优势主要体现在低到中等负载阶段的显著低延迟起点，而不是简单表现为延迟增长最缓。在到达率较低时，QVFER 三种拓扑下的平均延迟均明显低于其余算法，例如在λ=0.5 时，BICS、DFN 与 Colt Telecom 中的平均延迟分别仅为 0.73 ms、0.74 ms 和 1.18 ms，较对应次优算法分别降低 42.97%、45.59%和 27.16%；随着到达率增大，QVFER 的延迟上升幅度相对更明显，但这是因为其接受率始终处于较高水平，在相同负载下承载了更多请求，因而平均延迟会随资源占用增加而更快抬升。即便如此，在高负载端λ=5.0 时，QVFER 仍分别以 1.21 ms、1.57 ms 和 1.67 ms 保持最优，较次优算法仍降低 11.68%、1.26%和 1.18%；从全区间平均来看，三种拓扑下分别降低 28.48%、21.05%和 10.65%，总体平均降幅约为 20.06%。这一结果说明，QVFER 的关键优势在于其优化目标中显式刻画了包含延迟项的路径成本最小化，使算法在负载初期能够优先选取低延迟路径。



a) BICS 拓扑下的端到端平均延迟评估



b) DFN 拓扑下的端到端平均延迟评估

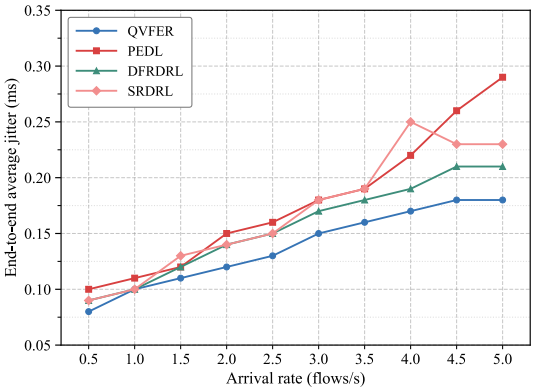


c) Colt Telecom 拓扑下的端到端平均延迟评估

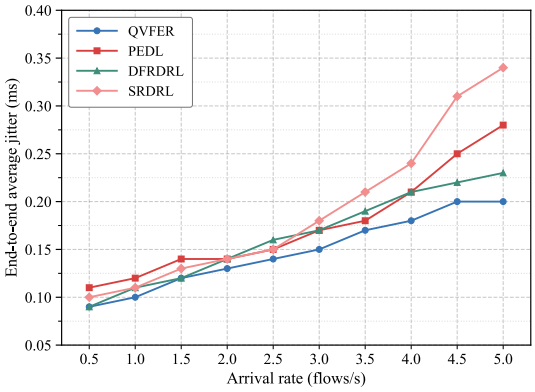
图 4-7 不同拓扑下的到达率变化对端到端平均延迟的影响

(5) 实验 5：端到端平均抖动评估

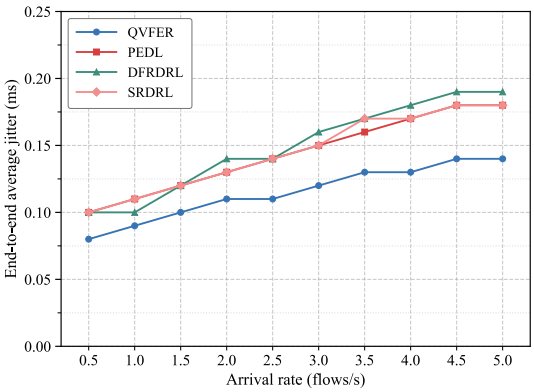
如图4-8所示，随着到达率增大，各算法的端到端平均抖动整体呈上升趋势，但 QVFER 在三种拓扑下始终保持最低或并列最低水平，说明其对延迟波动具有更好的抑制能力。以λ=5.0 为例，QVFER 在 BICS、DFN 与 Colt Telecom 拓扑下的平均抖动分别为 0.18 ms、0.20 ms 和 0.14 ms，相较次优算法分别降低 14.29%、13.04%和 22.22%；从全区间平均水平看，分别降低 11.54%、9.76%和 20.14%，三种拓扑汇总后的总体平均降幅约为 13.81%。



a) BICS 拓扑下的端到端平均抖动评估



b) DFN 拓扑下的端到端平均抖动评估

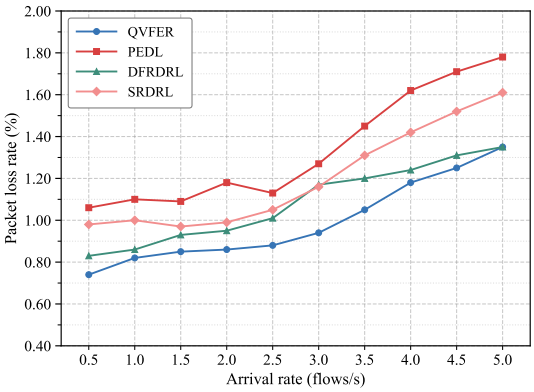


c) Colt Telecom 拓扑下的端到端平均抖动评估

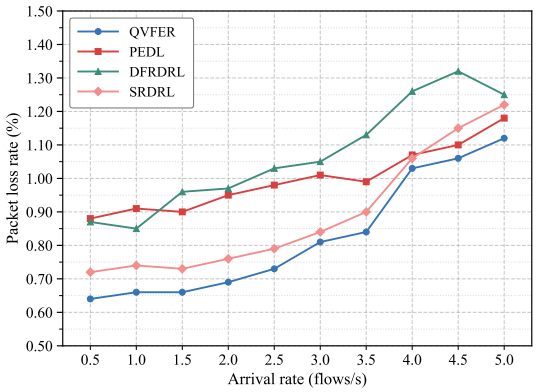
图 4-8 不同拓扑下的到达率变化对端到端平均抖动的影响

(6) 实验 6：丢包率评估

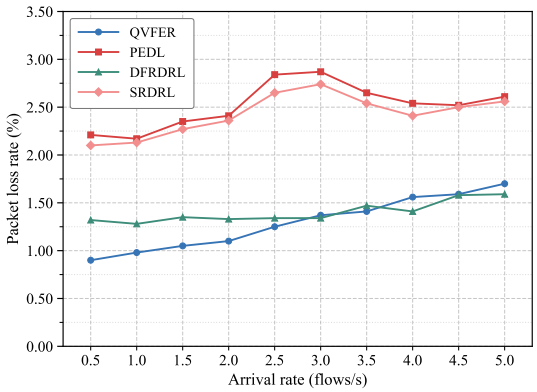
如图 4-9 所示，随着到达率升高，各算法的丢包率总体均呈上升趋势，但 QVFER 在多数场景下保持最优，优势较为稳定。具体而言，在 BICS 拓扑下，QVFER 在全到达率区间内始终最低或并列最低；在 DFN 拓扑下，QVFER 在大部分负载区间内同样保持最优，至高负载端仍较次优算法低 5.08%；在 Colt Telecom 拓扑下，整体丢包率水平明显高于前两种拓扑，说明随着网络规模扩大、路径变长以及跨域转发增多，链路排队与缓存竞争更易累积并放大为端到端丢包。在该拓扑中，QVFER 在中低负载阶段依然保持最优，但在高负载端略高于 DFRDRL。尽管如此，从 0.5 至 5.0 flow/s 全区间平均水平看，QVFER 在 BICS、DFN 与 Colt Telecom 三种拓扑下相较次优算法的平均丢包率仍分别降低 8.57%、7.52%与 7.85%，总体平均降幅达到 7.98%。这表明，QVFER 在绝大多数负载区间和拓扑场景下都能更有效地抑制丢包，其原因在于算法通过 VS 反馈机制及时规避热点链路、缓解局部拥塞，从而降低链路状态恶化向端到端丢包的传递。



a) BICS 拓扑下的丢包率评估



b) DFN 拓扑下的丢包率评估

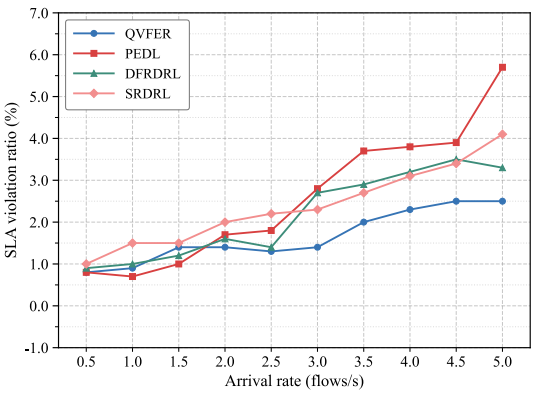


c) Colt Telecom 拓扑下的丢包率评估

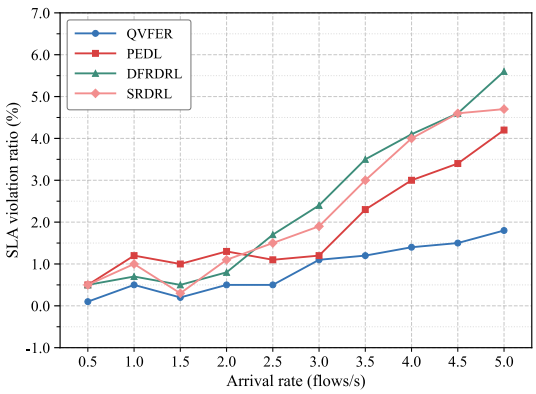
图 4-9 不同拓扑下的到达率变化对丢包率的影响

(7) 实验 7：SLA 违反率评估

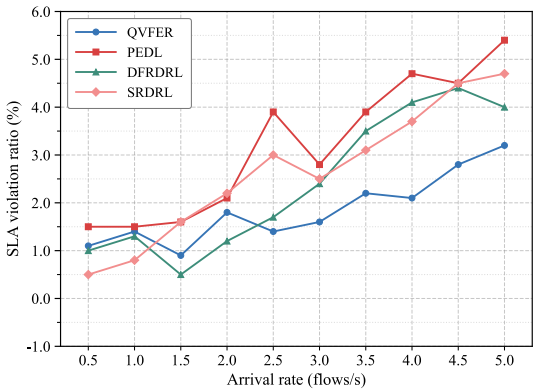
如图 4-10 所示，随着到达率增大，各算法的 SLA 违反率整体均呈上升趋势。总体来看，QVFER 在 BICS 和 DFN 两种拓扑下优势较为明显，在 Colt Telecom 拓扑下则体现出一定的性能权衡。具体而言，在 BICS 拓扑中，QVFER 在低负载阶段并非最优，但当到达率增至 2.0 flow/s 后，其 SLA 违反率整体保持最低，从 0.5 至 5.0 flow/s 全区间平均看，QVFER 的平均 SLA 违反率为 1.65%，较次优算法的 2.17%降低 0.52%；在 DFN 拓扑中，QVFER 的优势更为突出，其全区间平均 SLA 违反率为 0.88%，较次优算法 1.92%降低 1.04%。在 Colt Telecom 拓扑下，QVFER 也并非在所有到达率点上都最优，但从全区间平均看仍保持最低水平，其平均 SLA 违反率为 1.85%，较次优算法 2.41%降低 0.56%。综合三种拓扑的全部结果，QVFER 的总体平均 SLA 违反率为 1.46%，相较次优算法 2.17%下降 0.71%。出现该现象的主要原因在于，QVFER 并非单纯以最小化违反率为目标，而是在提高流接受率的同时兼顾路径代价与 QoS 约束。因此，算法可能接纳部分接近 SLA 边界的请求，导致个别到达率下违反率并非最低；但从整体结果看，QVFER 仍能更有效抑制延迟、抖动与丢包的联合超限，使总体 SLA 违反率保持最低。



a) BICS 拓扑下的 SLA 违反率评估



b) DFN 拓扑下的 SLA 违反率评估



c) Colt Telecom 拓扑下的 SLA 违反率评估

图 4-10 不同拓扑下的到达率变化对 SLA 违反率的影响

4.8 本章小结

本章在上一章动态子网划分与弹性网关部署结果的基础上，进一步面向多业务并发场景提出了 QoS 感知的波动反馈增强路由算法 QVFER。围绕在线业务调度需求，本章首先引入 WPCA 进行流优先级划分，并建立了以可接纳流量最大化和综合路径代价最小化为目标的优化模型；在此基础上，进一步设计了链路波动率 VS 反馈机制，以增强路由决策对链路状态变化和局部热点的在线响应能力。在算法设计方面，本章给出了 QVFER 的候选路径池构建、可行性筛选、综合评分选择与 VS 在线调节等流程。实验结果表明，QVFER 在接受率、平均跳数、请求流执行时间、端到端平均延迟、平均抖动、丢包率和 SLA 违反率等指标上总体优于对比算法，验证了该方法在上一章结构优化基础上进一步提升 QoS 保障能力与动态路由性能的有效性。

5 总结和展望

5.1 主要工作总结

为了应对数据中心网络在规模扩展与业务动态叠加条件下面临的路由复杂度高、多维 QoS 保障困难等问题，本文围绕数据中心网络中的协同路由优化机制进行了系统研究与实验验证，提出了一套兼顾结构降维、跨域转发组织与运行期反馈增强的路由优化方法，以提升网络的在线决策能力、资源利用效率与服务质量保障水平。具体工作内容如下：

1. 本文首先对数据中心网络中多约束路由优化以及结构降维路由优化等相关研究进行了系统梳理，分析了现有方法在全网尺度求解复杂度高、子网划分与网关部署协同不足、路径下发后缺乏运行态反馈调节等方面的局限，明确了本文的研究目标与技术路线。同时，对后续研究所涉及的相关理论与关键方法进行了整理，为模型建立与算法设计奠定了基础。

2. 针对传统路由在大规模拓扑和动态负载条件下难以在保障 QoS 的同时兼顾负载均衡、能耗控制与路由效率的问题，本文提出了基于动态子网划分与网关部署协同的路由优化框架 ROCDSG。该框架通过动态子网划分与弹性网关部署的联动设计，构建了面向跨域通信的子网网关骨架，并在此基础上提出了延迟感知的分层路由方法。最后通过仿真实验验证了所提框架在端到端延迟、平均跳数、路由效率以及负载与能耗协同控制等方面的有效性。

3. 在既有子网网关骨架基础上，针对多业务场景下多维 QoS 约束缺乏细粒度保障、跨子网路径候选构造受限以及路径下发后共享链路状态波动难以及时调节等问题，本文进一步提出了 QoS 感知的波动反馈增强路由算法 QVFER。该方法通过优先级感知、统一优化建模、候选路径增强与运行期反馈调节相结合，实现了面向动态负载环境的在线路由优化。最后通过仿真实验验证了所提算法在接受率、丢包率、SLA 违反率以及多维 QoS 保障等方面的有效性。

5.2 创新点

本文的创新点包括但不限于以下几个方面：

1. 本文的第一个主要创新点在于提出了一种基于动态子网划分与网关部署协同的路由优化框架 ROCDSG。该框架在流量、负载、延迟与能耗等多维约束驱动下，联合设计动态子网划分、弹性网关部署与延迟感知分层路由，实现了网络结构组织与路径决策的协同优化。其中，利用基于 Louvain 算法的动态子网划分机制实现网络结构压缩，通过网关部署优化模型与解空间优化萤火虫算法缓解跨

子网通信中的边界汇聚问题，并结合延迟感知分层路由提升跨域路径组织效率。与已有研究中结构划分与网关优化相对独立的方式不同，ROCDSG 实现了子网结构、网关布局与路径生成之间的协同联动，使 QoS、负载与能耗等多维目标能够在统一框架下进行综合权衡与优化。仿真结果表明，ROCDSG 的端到端平均延迟较次优算法降低 25.2%，平均跳数与执行时间分别降低 14.4%与 25.6%，同时能够有效保持负载与能耗均衡。

2. 本文的第二个主要创新点在于提出了一种 QoS 感知的波动反馈增强路由算法 QVFER。该算法基于上述子网网关骨架，围绕业务优先级感知、统一优化建模、候选路径扩展与运行期反馈调节展开设计。其中，利用 WPCA 实现 QoS 需求分级，并构建兼顾业务流接纳收益与路径综合代价的在线路由优化模型；同时，结合 PSI 与 eMDT 增强跨子网候选路径构造能力，并引入链路波动率 VS 对热点链路进行动态感知与轻量化调节。与已有方法相比，QVFER 进一步提升了骨架路由在多约束 QoS 场景下的动态适应能力与在线调节能力。仿真结果表明，QVFER 在接受率、平均跳数、请求流执行时间以及端到端 QoS 指标等方面均优于对比算法，其中总体平均接受率提升约 1.43%，平均跳数下降约 8.58%，请求流执行时间缩短约 42.82%，并有效降低了延迟、抖动、丢包率和 SLA 违反率，体现出较好的综合性能优势。

5.3 不足与展望

尽管本文围绕数据中心网络中的协同路由优化问题开展了较为系统的研究，并验证了 ROCDSG 框架与 QVFER 算法在提升路由效率、多维 QoS 保障以及运行稳定性方面的有效性，但在研究过程中仍存在一些不足，并有待在后续工作中进一步完善。具体而言，主要体现在以下三个方面：

1. 本文提出的 ROCDSG 框架虽然能够通过动态子网划分与网关部署协同机制有效压缩搜索空间，并在结构层面提升跨域转发的可控性，但当前网关部署仍以单一部署结果为主，即使通过优化模型进行了综合设计，在复杂流量变化条件下仍可能出现局部过载等现象。另一方面，当网络状态发生明显变化并触发子网重划分时，子网结构更新与路由调整之间的衔接仍可能带来一定波动，进而影响网络运行的连续性与稳定性。

2. 本文提出的 QVFER 算法在多业务场景下实现了基于优先级划分、候选路径增强与波动反馈调节的在线路由优化，并取得了较好的实验效果，但当前研究仍基于一定程度的抽象与简化，尚未充分覆盖更加复杂的真实运行环境。例如，在多控制器协同以及更长时间尺度的连续反馈调节场景下，现有模型与机制仍有进一步扩展空间。

3. 当前实验主要基于 BICS、DFN、Colt Telecom 等公开拓扑开展验证，与

实际数据中心广泛采用的 Fat-Tree、Clos 等生产级网络结构仍存在一定差异。在流量建模方面，本文主要采用泊松到达过程及正态分布生成业务流，尚未充分刻画真实数据中心中“大象流”与“老鼠流”共存、长期时变等复杂特征，因此实验结果在现实场景中的泛化能力仍需继续验证。此外，当前研究主要基于 Ryu 控制器与 Mininet 仿真平台开展验证，对于实际工程部署过程中可能面临的协议兼容性、跨控制器状态同步等问题，仍有待进一步深入研究。

针对上述不足，未来研究可进一步面向大规模动态数据中心网络环境，探索更加智能化与协同化的路由优化机制。例如，可结合多网关协同与动态子网演化过程，研究子网结构调整与路由更新之间的自适应联动机制，从而提升复杂流量环境下的网络稳定性与资源协同能力。同时，可进一步引入更贴近真实生产环境的数据中心拓扑与业务运行数据，包括 Fat-Tree、Clos 等典型架构，以及具有“大象流—老鼠流”混合特征与长期时变行为的真实流量模式，以增强模型对复杂业务场景的刻画能力与实验结果的现实参考价值。

此外，未来还可面向云边协同与大规模分布式控制场景，进一步研究多控制器协同决策、长时间尺度闭环反馈调节以及跨域状态感知等机制，并结合实际部署需求，逐步探索与现有网络协议栈的兼容互通、跨控制器状态同步等关键问题，从而推动协同路由优化方法向更强适应性与更高工程可落地性的方向发展。

参考文献

[1] Chen T, Gao X, Chen G. The features, hardware, and architectures of data center networks: A survey[J]. Journal of Parallel and Distributed Computing, 2016, 96: 45-74.

[2] 伍乙生. 大规模 SDN 的自适应实时调度优化方法研究[J]. 自动化仪表, 2025, 46(07): 97-102.

[3] Yang H, Zhang J, Zhao Y, et al. Software defined clustered-optical access networking for ubiquitous data center optical interconnection[C]//2016 International Conference on Computing, Networking and Communications (ICNC). IEEE, 2016: 1-5.

[4] Zhang L, Deng Q, Su Y, et al. A box-covering-based routing algorithm for large-scale SDNs[J]. IEEE Access, 2017, 5: 4048-4056.

[5] Zhang S, Xue X, Yan F, et al. Feasibility study of optical wireless technology in data center network[J]. IEEE Photonics Technology Letters, 2021, 33(15): 773-776.

[6] Zhao Y, Cao J, Wang X, et al. Power optimization for low transmission delay in software defined data center networks[J], IEEE Transactions on Mobile Computing, 2025, 24(7): 5716-5730.

[7] Hohlfeld O, Kempf J, Reisslein M, et al. Guest editorial scalability issues and solutions for software defined networks[J]. IEEE Journal on selected areas in communications, 2018, 36(12): 2595-2602.

[8] Wang Y, Wang X, Li H, et al. A multi-service differentiation traffic management strategy in SDN cloud data center[J]. Computer Networks, 2020, 171: 107143.

[9] Łepek M, Makulski K, Fronczak A, et al. Beyond traditional box-covering: Determining the fractal dimension of complex networks using a fixed number of boxes of flexible diameter[J]. Chaos, Solitons & Fractals, 2025, 199: 116908.

[10] He Y, Lu Z, Lei J, et al. Joint optimization of energy saving and load balancing for data center networks based on software defined networks[J]. Concurrency and Computation: Practice and Experience, 2021, 33(9): e6134.

[11] Pathan M N, Muntaha M, Sharmin S, et al. Priority based energy and load aware routing algorithms for SDN enabled data center network[J]. Computer Networks, 2024, 240: 110166.

[12] Lu Z, Lei J, He Y, et al. Energy optimization for Software-Defined data center networks based on flow allocation strategies[J]. Electronics, 2019, 8(9): 1014.

[13] Naeem F, Tariq M, Poor H V. SDN-enabled energy-efficient routing optimization framework for industrial Internet of Things[J]. IEEE Transactions on Industrial Informatics, 2020, 17(8): 5660-5667.

[14] 肖慧, 游嘉俊. 基于多条件约束的网络端到端流量 QoS 路由优化[J]. 长江信息通信, 2025, 38(11): 117-119.

[15] Wu J, Qiao X, Chen J. PDMR: priority-based dynamic multi-path routing algorithm for a software defined network. IET Communications 13.2 (2019): 179-185.

[16] 尹凤杰, 马雪莉. 软件定义无线 Mesh 网络多目标路由优化算法[J]. 辽宁大学学报(自然科学版), 2024, 51 (01): 88-96.

[17] Niu Y, Shan D. Research on multi-constraint QoS routing based on improved whale algorithm[J]. Applied Sciences, 2025, 15(21): 11592.

[18] BinSahaq A, Sheltami T, Mahmoud A, et al. Fast and efficient algorithm for delay-sensitive QoS provisioning in SDN networks[J]. Wireless Networks, 2024, 30(5): 4607-4628.
[19] Sanchez L P A, Shen Y, Guo M. DQS: A QoS-driven routing optimization approach in SDN using deep reinforcement learning[J]. Journal of Parallel and Distributed Computing, 2024, 188: 104851.
[20] Wang Y, Li Y, Wang T, et al. Towards an energy-efficient Data Center Network based on deep reinforcement learning[J]. Computer networks, 2022, 210: 108939.
[21] Kim G, Kim Y, Lim H. Deep reinforcement learning-based routing on software-defined networks[J]. IEEE Access, 2022, 10: 18121-18133.
[22] Chen J, Huang X, Wang Y, et al. ASTPPO: A proximal policy optimization algorithm based on the attention mechanism and spatio–temporal correlation for routing optimization in software-defined networking[J]. Peer-to-Peer Networking and Applications, 2023, 16(5): 2039-2057.
[23] Wang Y, Othman M, Choo W O, et al. DFRDRL: a dynamic fuzzy routing algorithm based on deep reinforcement learning with guaranteed latency and bandwidth for software-defined networks[J]. Journal of Big Data, 2024, 11(1): 150.
[24] Wu J, Zhu Z. Intelligent routing optimization for SDN based on PPO and GNN[J]. Journal of Network and Computer Applications, 2025, 242: 104249.
[25] She H, Yan L, Mao C, et al. Service-driven dynamic QoS on-demand routing algorithm[J]. Future Generation Computer Systems, 2025, 166: 107685.
[26] 杨宋亮. SDN 架构下智能路由算法研究[D]. 西南科技大学, 2023.
[27] 刘佩鑫. 基于 SDN 的数据中心网络智能路由优化研究[D]. 内蒙古大学, 2025.
[28] El-Hefnawy A N, Raouf A O, Askr H. Dynamic routing optimization algorithm for software defined networking[J]. Computers, Materials & Continua, 2022, 70(1): 1349-1362.
[29] Tiwari N K, Bajpai A, Maurya S, et al. Efficient forwarding rule management in software defined network via subnet-based pattern matching[J]. SN Computer Science, 2024, 5(6): 796.
[30] Wang X, Hu J, Lin H, et al. QoS and privacy-aware routing for 5G-enabled industrial Internet of Things: A federated reinforcement learning approach[J]. IEEE Transactions on Industrial Informatics, 2021, 18(6): 4189-4197.
[31] Dev J, Mishra J. Energy efficient routing in cluster based heterogeneous wireless sensor network using hybrid GWO and firefly algorithm[J]. Wireless Personal Communications, 2024, 137(2): 997-1028.
[32] Zou Y, Li H. Study on power grid partition and attack strategies based on complex networks[J]. Frontiers in Physics, 2022, 9: 790218.
[33] Wang S, Li W, Lin Z. Resilience enhancement study of renewable energy power networks by integrating subnet division and priority enhanced: Q-learning[J]. Reliability Engineering & System Safety, 2025: 111652.
[34] Torkzaban N, Gholami A, Baras J S, et al. Joint satellite gateway placement and routing for integrated satellite-terrestrial networks[C]//ICC 2020-2020 IEEE international conference on communications (ICC). IEEE, 2020: 1-6.
[35] Matni N, Moraes J, Oliveira H, et al. Lorawan gateway placement model for dynamic internet of things scenarios[J]. Sensors, 2020, 20(15): 4336.
[36] Cao Y, Guo H, Liu J, et al. Optimal satellite gateway placement in space-ground integrated networks[J]. IEEE Network, 2018, 32(5): 32-37.
[37] Rafique W, Hafid A S, Cherkaoui S. Softcaching: A framework for caching node selection and routing in software-defined information centric internet of things[J]. Computer Networks, 2023, 235: 109966.

[38] Choi S H, Jang Y, Seo H, et al. An effective algorithm to find a cost minimizing gateway deployment for node-replaceable wireless sensor networks[J]. Sensors, 2021, 21(5): 1732.
[39] Huang S, Tao M. Competitive swarm optimizer based gateway deployment algorithm in cyber-physical systems[J]. Sensors, 2017, 17(1): 209.
[40] Scott-Hayward S, O'Callaghan G, Sezer S. SDN security: A survey[C]//2013 IEEE SDN For Future Networks and Services (SDN4FNS). IEEE, 2013: 1-7.
[41] Lara A, Kolasani A, Ramamurthy B. Network innovation using openflow: A survey[J]. IEEE communications surveys & tutorials, 2013, 16(1): 493-512.
[42] Schönwälder J, Björklund M, Shafer P. Network configuration management using NETCONF and YANG[J]. IEEE communications magazine, 2010, 48(9): 166-173.
[43] Lee Y, Le Roux J L, King D, et al. Path computation element communication protocol (PCEP) requirements and protocol extensions in support of global concurrent optimization: RFC 5557[R]. 2009.
[44] 左青云, 陈鸣, 赵广松, 等. 基于 OpenFlow 的 SDN 技术研究[J]. 软件学报, 2013, 24(05): 1078-1097.
[45] Bhowmick A K, Meneni K, Danisch M, et al. Louvainne: Hierarchical louvain method for high quality and scalable network embedding[C]//Proceedings of the 13th international conference on web search and data mining. 2020: 43-51.
[46] Fister I, Fister Jr I, Yang X S, et al. A comprehensive review of firefly algorithms[J]. Swarm and evolutionary computation, 2013, 13: 34-46.
[47] 王文杰, 姜念祖, 林帅男, 等. CR-WPCA:一种针对高维小样本数据集的加权主成分分析方法[J]. 白城师范学院学报, 2024, 38 (05): 48-56.
[48] Amer A A, Ravana S D, Habeeb R A A. Effective k-nearest neighbor models for data classification enhancement[J]. Journal of Big Data, 2025, 12(1): 86.
[49] Hirahara S. NP-hardness of learning programs and partial MCSP[C]//2022 IEEE 63rd annual symposium on foundations of computer science (FOCS). IEEE, 2022: 968-979.
[50] Wang Y, Kang R, Guo L, et al. Deadline-constrained multi-commodity flow routing and scheduling optimization with consideration of edge lengths and capacities[J]. Computers & Industrial Engineering, 2024, 192: 110193.
[51] Li H, Liu T, Wu X, et al. Correlated SVD and its application in bearing fault diagnosis[J]. IEEE transactions on neural networks and learning systems, 2021, 34(1): 355-365.
[52] Lozano-Rizk J E, Nieto-Hipolito J I, Rivera-Rodriguez R, et al. QOSCOMM: A data flow allocation strategy among SDN-based data centers for IoT big data analytics[J]. Applied Sciences, 2020, 10(21): 7586.
[53] Wu J, Wang Y G, Burrage K, et al. An improved firefly algorithm for global continuous optimization problems[J]. Expert Systems with Applications, 2020, 149: 113340.
[54] 刘阳, 鲁圆圆, 郭成城. 基于优先级的数据中心任务优化调度算法设计[J]. 计算机仿真, 2025, 42(01): 497-500+507.
[55] Fu C, Wang B, Liu H, et al. Software-defined virtual private network for SD-WAN[J]. Electronics, 2024, 13(13): 2674.
[56] Koulougli D, Nguyen K K, Cheriet M. Joint optimization of routing and flexible ethernet assignment in multi-layer multi-domain networks[C]//2020 29th International Conference on Computer Communications and Networks (ICCCN). IEEE, 2020: 1-9.
[57] Alghamdi S A. Stable zone-based 5G clustered MANET using interest-region-based routing and gateway selection[J]. Peer-to-Peer Networking and Applications, 2021, 14(6): 3559-3577.

[58] Kumar M, Raw R S. RC-LAHR: Road-side-unit-assisted cloud-based location-aware hybrid routing for software-defined vehicular ad hoc networks[J]. Sensors, 2024, 24(4): 1045.

[59] Guo D, Xie J, Shi X, et al. HDS: A fast hybrid data location service for hierarchical mobile edge computing[J]. IEEE/ACM Transactions on Networking, 2021, 29(3): 1308-1320.

[60] Jayasumana A P, Paffenroth R, Mahindre G, et al. Network topology mapping from partial virtual coordinates and graph geodesics[J]. IEEE/ACM Transactions on Networking, 2019, 27(6): 2405-2417.

[61] Guo Z, Dou S, Wang Y, et al. HybridFlow: Achieving load balancing in software-defined WANs with scalable routing[J]. IEEE Transactions on Communications, 2021, 69(8): 5255-5268.

[62] Zhang J, Ye M, Guo Z, et al. CFR-RL: Traffic engineering with reinforcement learning in SDN[J]. IEEE Journal on Selected Areas in Communications, 2020, 38(10): 2249-2259.

[63] Padhy S, Chou J. Reconfiguration aware orchestration for network function virtualization with time-varied workload in virtualized datacenters[J]. IEEE Access, 2021, 9: 48413-48428.

攻读学位期间的学术论文与研究成果

■ 已发表学术论文

[1] Lin Q, Chen S, Xie H, et al. ROCDSG: a routing optimization framework for DCN[J]. Computer Networks, 2026: 112207.（CCF 推荐 B 类期刊，第一作者，已发表）

[2] Luo Y, Shu Z, Chen S, et al. Multi-objective optimization algorithm for VNF migration with priority awareness in dynamic networks[J]. Computer Networks, 2025: 111666.（CCF 推荐 B 类期刊，第六作者，已发表）

■ 科研项目

[1] 福建省科技厅福厦泉协同创新平台项目，“Research and industrial application of high reliability intelligent gateway devices and systems based on LLM and SD-WAN”，No. 2025E3012，2025.09-2027.09.（项目主要成员，负责 SD-WAN 场景下路由优化算法的相关设计）

[2] 来自网络通信公司的产学研项目，“Research on intelligent monitoring and optimization of power information network and its industrial application”，No. KH240131A，2024.04-2027.04.（项目主要成员，负责电力信息网络的路由优化相关工作）

■ 学科竞赛

[1] “华为杯”第二十一届中国研究生数学建模竞赛三等奖（国 A 级，第一作者）

[2] “华为杯”第二十二届中国研究生数学建模竞赛一等奖（国 A 级，第一作者）

[3] 2025 年第十五届 MathorCup 数学应用挑战赛研究生组全国二等奖（国 C 级，第一作者）

[4] 第十八届“认证杯”数学建模网络挑战赛第一阶段三等奖（省 C 级，第一作者）

[5] 第十八届“认证杯”数学建模网络挑战赛第二阶段特等奖（省 C 级，第一作者）

[6] 第九届华为 ICT 大赛实践赛网络赛道福建赛区三等奖（省 A 级，第一作者）

[7] 中国高校计算机大赛 2025 网络技术挑战赛华东赛区三等奖（省 A 级，第一作者）

[8] 2025 年 APMCM 亚太地区大学生数学建模竞赛中文赛项二等奖（省 C 级，第一作者）

致谢

岁月不居，时节如流。行文至此，方觉三年研究生时光已近尾声。回望来路，从本科毕业时的依依不舍，到此间求学的沉淀与打磨，再到此刻写下论文的最后一页，心中百感交集。一路走来，有迷茫，有压力，也有坚持与收获。幸得诸多师长、亲友相伴左右，让我在反复与迟疑中慢慢坚定方向，走到今天。谨以此文，致谢这段岁月，也致谢一路相逢的人。

首先，我要衷心感谢我的导师舒兆港老师。三年求学，从方向初定到课题推进，再到论文数易其稿，老师始终耐心指导、细致把关。无论是思路受阻时的点拨，还是文字细节中的反复推敲，皆凝聚着老师的心力。老师不仅教会我如何做研究，更让我懂得做学问应有的分寸与态度。言传身教之间，润物无声，却影响深远。这些看似平常的教诲，终将在今后的道路上不断回响。在此，谨致以最真挚的敬意与感谢。

其次，我要感谢我的家人。求学多年，你们始终在我身后，沉默而坚定。或许少有言语，却在每一个需要支撑的时刻给予我力量，让我得以安心读书、专心做事。你们的付出无需多言，却始终沉甸甸地放在心上。愿家人平安顺遂，岁岁安宁，烟火日常里皆是温暖与欢喜。

我还要感谢在研究生阶段一路同行的实验室伙伴、舍友和朋友。科研之路并不轻松，论文写作亦常伴随反复与自我怀疑。所幸这段旅程从不孤单。那些一起讨论问题的时刻、一起赶进度的夜晚、彼此鼓励的瞬间，都让原本紧张的时光多了几分温度与光亮。山高路远，或许终将各赴前程，但这段并肩走过的岁月，会长久留在记忆之中。愿此去繁花似锦，前路皆明，所到之处皆为坦途。

最后，我也想感谢一路走来的自己。感谢那个在平凡日子里默默坚持、在不确定中仍愿意继续向前的自己。论文至此暂告一段落，而人生仍在书写！