面向组播的动态虚拟网络功能放置算法

邢焕来[1),2)] 王心汉[1)] 宋富洪[1)] 赵博文[1)] 罗寿西[1),2)] 戴朋林[1),2)] 李 可[1),2)]

[1)](西南交通大学计算机与人工智能学院 成都 611756)

[2)](可持续城市交通智能化教育部工程研究中心 成都 611756)

摘 要 组播在支持日益增长的多媒体应用方面具有广阔的应用前景,面向组播的虚拟网络功能放置是网络功能虚拟化中不可避免的研究趋势. 然而，对于该问题的大多数研究都聚焦于静态网络环境，难以应对网络中的各种资源随着时间动态变化，组播服务功能链（Service Function Chaining，SFC）请求动态到达的真实场景. 本文提出一种基于组播 SFC 请求预测的足球联赛竞争算法，以 Informer 模型为基础，预测即将到达的组播 SFC 请求. 基于足球联赛竞争的组播虚拟网络功能放置算法，设计多维个体编码策略，一次性求解所有活动组播组的 SFC 映射方案，提前部署预测的请求. 针对预测结果与真实结果不一致的情况，提出一种由正向搜索与反向搜索组成的快速修复策略以完成对请求的快速响应. 仿真结果表明，对比其它两种预测模型，Informer 在组播 SFC 请求预测上取得了更低的均方误差与平均绝对误差. 此外，与七种经典的启发式算法和深度强化学习算法相比，提出的算法在端到端时延和计算资源消耗方面达到更优性能的同时，取得了更低的组播 SFC 请求响应时间.

关键词 网络功能虚拟化；虚拟网络功能放置；组播 SFC 请求预测；足球联赛竞争算法；Informer

中图法分类号 TP18 **DOI 号** 10.11897/SP.J.1016.2023.02322

Multicast-Oriented Dynamic Virtual Network Function Placement Algorithm

XING Huan-Lai[1),2)] WANG Xin-Han[1)] SONG Fu-Hong[1)] ZHAO Bo-Wen[1)]
LUO Shou-Xi[1),2)] DAI Peng-Lin[1),2)] LI Ke[1),2)]

[1)](*School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu* 611756)

[2)](*Engineering Research Center of Sustainable Urban Intelligent Transportation, MoE, Chengdu* 611756)

Abstract Multicast, a potential technique to support ever-increasing multimedia applications, makes point-to-multipoint-oriented virtual network function (VNF) placement a promising research trend in network function virtualization (NFV). However, most existing research is hardly adapted to time-varying resources and dynamic multicast service function chaining (SFC) requests in the real-world network. This paper proposes a soccer league competition algorithm with multicast SFC request prediction (SLC-MSRP), which can foresee the incoming multicast SFC requests (MSRs) based on the Informer model. Based on multi-dimensional individual coding strategy, SLC-MSRP handles the SFC mapping of all active multicast groups simultaneously and deploys the predicted MSRs in advance. A fast repair strategy consisting of forward and backward search phases is developed to handle the differences between the predicted results and the actual arrival MSRs. Simulation results show that Informer achieves lower mean square error (MSE)

收稿日期：2022-10-10；在线发布日期：2023-04-03. 本课题得到国家自然科学基金（No. 62172342, No.62202392）、四川省自然科学基金（No. 2022NSFSC0568, No. 2022NSFSC0944, No. 2023NSFSC0459）、中央高校基本科研业务费资助. **邢焕来**（通信作者），博士，副教授，博士生导师，中国计算机学会（CCF）会员，主要研究领域为网络功能虚拟化、软件定义网络、人工智能、进化计算等. E-mail: hxx@home.swjtu.edu.cn. **王心汉**，博士研究生，主要研究领域为网络功能虚拟化、进化计算和深度学习. **宋富洪**，博士研究生，主要研究领域为边缘计算和多目标优化. **赵博文**，博士研究生，主要研究领域为深度学习和时间序列分类. **罗寿西**，博士，硕士生导师，主要研究领域为数据中心网络和网络化系统. **戴朋林**，博士，硕士生导师，主要研究领域为智能交通系统和车辆信息物理系统. **李 可**，博士，硕士生导师，主要研究领域为车联网.

and mean absolute error (MAE) values in the MSR prediction than two existing prediction models. Furthermore, the proposed algorithm gains lower MSR response time while achieving better performance in terms of end-to-end delay and computational resource consumption compared with seven state-of-the-art heuristic and deep reinforcement learning algorithms.

Keywords　network function virtualization; virtual network function placement; multicast SFC request prediction; soccer league competition algorithm; informer

1　引　言

随着网络通信技术的不断创新，以第五代移动通信技术（5th Generation Mobile Communication Technology，5G）和物联网（Internet of Things，IoT）为代表的新兴技术迅猛发展，已融入到人们日常生活和工作的方方面面，用户对通信服务的请求变得越来越多样化和动态化. 在传统电信网络中，企业和电信运营商为了满足用户的多种服务需求，需要在网络中部署大量的防火墙、入侵检测系统等专有网络设备，通常被称为中间件（Middle-box）. 通过增减和开发专用硬件设备扩展网络的方式会消耗大量的资金并增加维护成本. 网络功能虚拟化（Network Functions Virtualization，NFV）作为 5G 的重要支撑技术，就在这个大背景下诞生，为上述问题的解决带来了曙光[1]. NFV 引入虚拟化技术，将网络功能从传统网络硬件中解耦出来. 在 NFV 中，传统的中间件被管理为单个的软件模块，NFV 对这些软件模块进行编程，使其承担特定的虚拟网络功能（Virtual Network Function，VNF）. 通过将每个网络功能模块化，NFV 可以在通用服务器上安装和部署 VNF，且允许 VNF 在服务器间进行动态迁移. 这使得 NFV 技术能够大幅提升运营商灵活管理网络基础设施与快速部署网络服务的能力，显著降低其资本支出和运营成本. 在 NFV 中，网络服务以服务功能链（Service Function Chaining，SFC）的形式实现[2]. SFC 是一系列有序的带约束的 VNF 集合，数据流在到达用户之前必须按照指定的顺序依次通过 VNF 集合中的各个 VNF. 如何将 SFC 中的 VNF 合理地部署在网络中，既是响应服务请求的关键，也是 NFV 最重要的研究领域之一. 该问题被称为虚拟网络功能放置（VNF Placement，VNFP）或 SFC 映射（SFC Mapping）问题.

组播作为一种高效的一对多数据传输方式，在 IPTV、在线互动游戏、远程教育和视频会议等日益增长的多媒体应用中具有广阔的应用前景. 对于组播会话的 SFC 请求称为组播 SFC 请求（Multicast SFC Request，MSR）. 相比于面向单播的 VNF 放置（Unicast-oriented VNFP，UVNFP），在 NFV 环境实现组播存在更多挑战. 从 VNF 放置的角度来看，单播考虑的是单一路径，而组播构造的是多条路径. 在组播中，每条路径都源自同一个源节点，并最终到达各自的目的节点. 面向组播的 VNF 放置（Multicast-oriented VNFP，MVNFP）是在组播树的每条路径上映射一个相同的 SFC[3]. 将 VNF 放置在这些路径的重叠节点上时，组播的不同路径可以共享具有同种功能的 VNF[4]，该问题已被证明是 NP-难的[5]. 图 1 展示了一个面向组播的 VNF 放置问题的例子. 所请求的 SFC 包含 3 个 VNF，经过映射后，VNF1 被放置在节点 $v1$ 和 $v3$ 上，VNF2 被放置在节点 $v2$ 和 $v3$ 上，VNF3 被放置在节点 $v2$、$v4$ 和 $v6$ 上. 数据流在组播中从源节点出发到目的节点的每条路径上，都能完整地被 VNF1 → VNF2 → VNF3 依次处理. 放置在节点 $v3$ 上的 VNF1 与 VNF2 能够被两条路径所共享，即路径“源节点 → $v3$ → $v4$ → 目的节点 2”与路径“源节点 → $v3$ → $v6$ → 目的节点 3”.

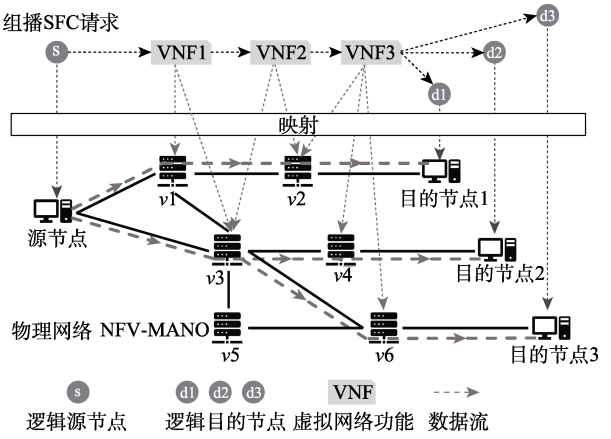

图 1　面向组播的 VNF 放置问题示例

当前，大多数 MVNFP 问题的研究都聚焦于静态网络环境[6−10]，网络中的计算资源、带宽资源等不会随着时间而改变. 然而，在真实网络环境中，MSR 可能随着时间的推移与用户需求的改变而实时变化. 针对静态网络环境的 MVNFP 解决方案难

以满足不断变化的网络服务需求. 在为数不多关注于动态网络环境下的 MVNFP 问题的研究中，MSR 动态到达，网络中各种资源会随着时间动态变化，完成服务请求后被占用的资源会实时释放. 例如，Asgarian 等人[11]研究了组播 VNF 服务链的多阶段嵌入中效率与复杂度的权衡问题，将组播服务链嵌入问题表述为一个整数线性规划（Integer Linear Programming，ILP），提出了一种两阶段方法分别求解 VNF 放置与路由选择. Xu 等人[3]提出了伪组播树的概念，在伪组播树中，相同的网络功能只能部署一次. 在之后的研究中，该团队利用近似算法[12]解决了支持 NFV 的 MEC 网络中组播资源共享问题. Yi 等人[13]提出了一种同时支持 VNF 和网络功能设备的混合基础设施网络多级解决方案，采用最小生成树的方法构造流量转发拓扑，并应用回溯法以交付相应的网络功能. Ma 等人[14]研究了静态和动态环境下 NFV 赋能的 MEC 网络中的组播问题，通过近似算法最大限度地提高网络吞吐量. Cai 等人[15]围绕分布式计算网络中组播 SFC 的控制问题展开研究，设计了一个完全分布式的组播流量管理策略. 综上所述，在动态网络环境下，现有研究集中在当 MSR 到达后，调用设计的在线放置算法求解 MVNFP 方案，以期望能快速响应该请求，所采用的方法均为启发式算法. 然而，启发式算法追求的是在尽可能短的时间内搜索到待求问题的一个可行解，解的差异性通常很大，往往难以保证优化质量[16]. 进化算法是一系列受自然启发的全局搜索和优化方法，具有强大的鲁棒性、快速收敛性和广泛的应用，被认为是处理 NP 难问题的理想候选技术之一. 足球联赛竞争（Soccer League Competition，SLC）算法[17]是根据足球联赛中球队和球员间的竞争机制而提出的一种群体优化进化算法. 与其它群体进化算法相比，SLC 算法具有收敛速度快、结果更准确等优点[17]，目前已成功应用于城市电网设计[18]、无线传感通信[19]和资源部署问题[20]等多个领域. 因此，本文将 SLC 算法应用到求解 MVNFP 问题中，尝试利用球员之间、球队之间以及联赛之间的竞争机制，帮助探索组播路径中各节点、路径间重叠节点以及多个组播组共同经过的节点间的 VNF 放置关系，形成面向组播的 VNF 放置方案.

由于 VNF 放置与路由决策过程的计算会消耗大量时间，上述方法可能仍需数秒或更长时间来处理到达的 MSR. 研究显示，如果视频服务启动时间超过 2 秒，终端用户就会开始放弃该视频，且每增加 1 秒时延，放弃率会额外增加 5.8%[21]. 因此，现有针对动态网络环境下的 MVNFP 问题解决方案仍难以快速响应动态到达的 MSR，无法保证用户体验质量（Quality of Experience，QoE）. 已有研究者在研究 UVNFP 问题时，采用基于深度神经网络等预测模型来辅助解决 VNF 放置问题. Kim 等人[22]分析了 SFC 数据对预测 VNF 资源使用的影响，提出了 VNF 资源预测的机器学习模型. Cai 等人[23]提出了一种基于计算负载和资源需求的 SFC 主动重配置机制，通过预测节点的计算负载和 SFC 的资源需求，实现服务质量和重配置成本两方面的权衡. Eramo 等人[24]在研究 NFV 网络中的资源分配时，采用非对称长短期记忆网络（Long Short-Term Memory，LSTM）来预测网络流量. 文献[25]提出了一种 VNF 资源容量自适应调整的方法，利用 LSTM 预测平台流量的变化趋势. Huang 等人[26]在研究 NFV 系统中服务链映射和资源调度问题时，提出了一种预测在线服务映射和资源调度策略. 总的来说，当前对于 VNF 预测的相关研究集中在对 VNF 资源[22,23]和流量[24,25]的预测上，仅有少量研究关注于预测 SFC 请求[26]. 然而，现有工作未涉及组播，针对 UVNFP 问题的解决方案也难以应用到 MVNFP 问题中. 相比于 UVNFP 问题中关于 VNF 的预测，在 MVNFP 问题中不仅需要考虑 SFC 和 VNF 的相关信息，还需要考虑组播组与 SFC 请求的对应关系，增加了预测内容的复杂度. 同时，现有研究所采用的预测模型以 LSTM 及其变种[22–25]为主. 然而，尽管 LSTM 模型相比传统统计学模型能够更好地解决序列预测问题，但研究人员已经证明，LSTM 仍然难以胜任对长序列时间序列的处理[27]. Informer[28]是基于 Transformer 架构[29]提出的时间序列预测模型，通过编码器—解码器（Encoder-Decoder）和概率稀疏自注意力（ProbSparse Self-attention）等机制提取时间序列特征，使得其在复杂时序预测问题中展现出优异的性能[30]. 因此，采用 Informer 作为对组播 SFC 请求的预测模型具有很高的潜力.

本文提出一种基于组播 SFC 请求预测的足球联赛竞争（Soccer League Competition Algorithm with Multicast SFC Request Prediction，SLC-MSRP）算法解决动态网络环境下的 MVNFP 问题. 表 1 展示了本文工作与现有研究的区别. 可以看出，虽然文献[3,11–15]关注于动态网络环境下的 MVNFP 问题，然而它们均未涉及通过预测辅助以快速响应用户的组播 SFC 请求. 文献[22–26]采用深度学习等技术预测辅助进行 VNF 放置，但它们仅研究了面向单播的 VNF 放置问题. 此外，仅有文献[26]考虑了对预测

表 1　本文工作与现有研究的区别

参考文献	[3]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[22]	[23]	[24]	[25]	[26]	本文
面向组播	√	√	√	√	√	√	√	√	√	√	√						√
动态网络环境	√						√	√	√	√	√	√	√	√	√	√	√
预测辅助												√	√	√	√	√	√
预测错误处理																√	√

错误的情况，通过调节预测决策与实时决策在总体决策中的占比来处理预测错误对用户单播 SFC 请求响应时间造成的影响，但这需要很强的先验知识来动态调整预测决策与实时决策的比例. 本文提出的算法与上述研究不同的是，利用深度神经网络模型基于历史 MSR 到达记录，预测未来的 MSR，使得系统能够在 MSR 实际到达前，提前映射相关 SFC，以提升用户体验质量. 针对预测错误的情况，本文提出的算法采用基于启发式的修复策略，对实际到达的 MSR 进行处理. 针对已有研究存在的问题，本文的主要贡献如下：

（1）以最小化端到端时延、计算资源消耗与 MSR 响应时间的总开销为目标，将动态网络环境下的 MVNFP 问题划分为 VNF 预放置与修复两个子问题. 在 VNF 预放置子问题中预测即将到达的 MSR，并提前映射对应的 SFC 及路由策略；在修复子问题中，通过与预测结果的差异对比，处理预测错误的 MSR，放置实际的 VNF 和重新路由.

（2）为处理上述两个子问题，本文提出了 SLC-MSRP 算法，包含两阶段和三部分，即 VNF 预放置和修复两个阶段，以及基于 Informer 的预测模型、基于足球联赛竞争的组播虚拟网络功能放置算法与快速修复策略三个部分. 通过两个阶段、三个部分的分工协作，该算法提供了一个解决动态网络环境下的 MVNFP 问题“预测——预放置——修复”的解决方案，减少了 MSR 响应时间.

（3）在 SLC-MSRP 算法 VNF 预放置阶段，本文提出了基于足球联赛竞争的组播虚拟网络功能放置算法，设计了基于多维个体编码（Multi-dimensional Individual Coding，MDIC）策略，使得单个球员可以完整地表示动态网络环境下的 MVNFP 问题中同一时隙多个活动组播组的 MSR 映射方案，通过足球联赛中球队和球员间的相互竞争，最小化端到端时延与计算资源消耗.

（4）在 SLC-MSRP 算法修复阶段，提出的快速修复策略通过限制候选节点的选择范围，充分利用已经放置的 VNF 以节约计算资源并减少 VNF 初始化时延. 同时，采用正向搜索与反向搜索两个过程，为每个 MSR 构造一棵生成树（Spanning Tree，SPT）并对该生成树进行优化，以最小化端到端时延.

本文接下来组织如下：第 2 节与第 3 节分别介绍了系统模型与问题描述；第 4 节给出 SLC-MSRP 算法的实现细节；第 5 节给出了仿真结果与对比分析；第 6 节总结全文并对未来工作进行展望.

2　系统模型

在面向组播的 VNF 放置场景中，网络表示为无向图 $G=(V,E)$，V 和 E 分别表示节点集和链路集. 网络工作在一个离散时间系统下，记 T 为系统的时隙集，每个时隙 $t\in T$ 为固定单位长度，$|T|$ 为 T 中的时隙数量. 每个节点 $v\in V$ 连接着一台 VNF 服务器，能够运行 VNF 和转发数据流，记 $v\in V$ 在时隙 t 的可用计算资源为 R_v^t. 数据在任意两个节点之间的链路中传输时会产生时延，称为传播时延. 记 $L^t(e)$ 与 $B^t(e)$ 分别为链路 $e\in E$ 在时隙 t 的传播时延和可用带宽. 网络可提供多种网络服务，每种服务 $F\in SFCs$ 都表示某个特定的 SFC，其中 $SFCs$ 是该网络系统可提供所有类型的 SFC 集合，SFC F 的最大端到端时延表示为 $L_{\max}(F)$. 每种 VNF $f\in F$ 在节点上实例化会消耗一定量的计算资源 R_f. 记 $L_{\text{init}}(f)$ 表示 f 的初始化时延，即 f 在节点上首次实例化需消耗的时间. 对于组播会话，为了提供网络服务，需要在数据传输之前将相应的 SFC 映射到组播树上. 记网络中可提供的组播会话集合为 $M=\{m_1,\cdots,\ m_{|M|}\}$，$m_i=(s_i,D_i)$ 为 M 中的一个组播组，$|M|$ 为 M 中组播组的数量，$i=1,\cdots,|M|$. s_i 和 $D_i=\{d_{i,1},\cdots,\ d_{i,|D_i|}\}$ 分别为 m_i 的源节点和目的节点集，在 T 期间内保持不变，其中 $d_{i,j}$ 为 D_i 的第 j 个目的节点，$|D_i|$ 为 D_i 中的节点数量，$j=1,\cdots,|D_i|$. 对于时隙 t，任意组播组 $m_i\in M$ 要么是活动的，要么是非活动的. 对于一个活动组，其组播树与相应的 SFC 映射将被构建；对于一个非活动组，不需要维护其组播树与相应的 SFC 映射. 在节点计算资源允许的情况下，可在节点 v 上放置任意数目的 VNF，但为了最大限度地利用每个 VNF 所提供的服务，本文不允许在一个节点上放置两个或多个具有相同功

能的 VNF[4]. 这表明同一节点上的任意两个 VNF 将提供不同的功能，放置在这些节点上的 VNF 可以被相应的路径共享. 如图 1 所示，本文假定网络中存在一台性能强大的 VNF 服务器作为中心管理节点，运行 NFV 管理与编排（Management and Orchestration，MANO）框架，以远小于系统时隙的监控周期 T_M 对各节点的计算资源和链路带宽等网络中的信息进行监控，同时负责 VNF 的编排与生命周期管理，包括 VNF 放置与路由转发等[31,32].

记 $MSR^t=\{msr_i^t|i=1,\cdots,|M|\}$ 为 t 时隙实际到达的 MSR 集合. $msr_i^t=(m_i,F_i^t,B_i^t)$ 为组播组 m_i 的一个 MSR，其中 $F_i^t=\{f_{i,1}^t,\cdots,f_{i,|F_i^t|}^t\}$ 表示请求的 SFC，$f_{i,k}^t$ 是 F_i^t 中第 k 个 VNF，$|F_i^t|$ 为 F_i^t 中的 VNF 数量，$F_i^t\subset SFCs$，$k=1,\cdots,|F_i^t|$；B_i^t 是 msr_i^t 的带宽要求. 对于解决动态网络环境下的 MVNFP 问题来说，在 $t-1$ 时隙，基于 t 时隙前实际到达的 MSR，预测 MSR^t，在物理网络中预放置相应的 VNF；在 t 时隙，迅速处理未成功预测的 MSR，以快速响应用户请求. 系统模型部分的主要符号见表 2.

3 问题描述

本文将解决动态网络环境下的 MVNFP 问题拆分为两个子问题，即 VNF 预放置与修复. VNF 预放置与修复子问题的关系如图 2 所示. 在 VNF 预放置子问题中，将在 $t-1$ 时隙，根据历史数据预测 MSR^t，并预放置相应的 VNF，确定对应的路由策略. 在修复子问题中，错误的 MSR 预测将被处理，当 t 时隙实际的 MSR 到达后，通过与预测结果的差异对比，快速放置实际的 VNF 和重新路由. 问题描述部分的主要符号见表 3.

表 2 系统模型部分的主要符号

符号	定义
$B^t(e)$	$e\in E$ 在时隙 t 的可用带宽
B_i^t	msr_i^t 的带宽要求
D_i	m_i 的目的节点集
$d_{i,j}$	D_i 的第 j 个目的节点
E	链路集
F	一组有序的 VNF 集合
F_i^t	msr_i^t 请求的 SFC
$f_{i,k}^t$	$f_{i,k}^t\in F_i^t$ 是 F_i^t 中第 k 个 VNF
$L^t(e)$	$e\in E$ 在时隙 t 的传播时延
$L_{\text{init}}(f)$	f 在节点上首次实例化需消耗的时间
$L_{\max}(F)$	F 的最大端到端时延
M	网络中可提供的组播会话集合
m_i	M 中的一个组播组
MSR^t	t 时隙实际到达的 MSR 集合
msr_i^t	组播组 $m_i\in M$ 的一个 MSR
R_f	$f\in F$ 在节点上实例化消耗的计算资源
s_i	m_i 的源节点
$SFCs$	网络系统可提供所有类型的 SFC 集合
T	时隙集
V	节点集

3.1 VNF预放置子问题

为区分预测结果与实际结果，使用上标“(t)”标

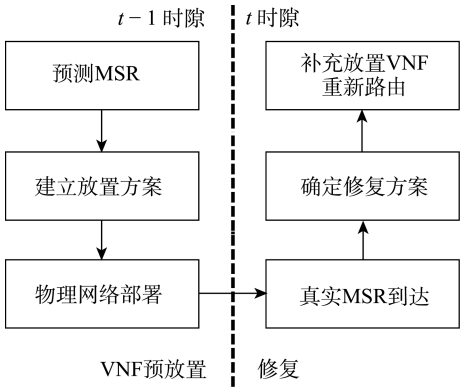

图 2 VNF 预放置与修复子问题的关系

表 3 问题描述部分的主要符号

符号	定义
C^t	端到端时延和计算资源消耗的总开销
C_{L}^t	MSR^t 中所有活动组播组的平均端到端时延
C_{R}^t	MSR^t 中所有 MSR 的计算资源消耗
C_{S}^t	系统接收到 MSR^t 后开始服务的 MSR 响应时间
$E_{\|F\|\to d}^i$	$P_{i,j}^{(t)}$ 上从放置了 $f_{i,\|F_i^{(t)}\|}^{(t)}$ 的节点出发到 $d_{i,j}$ 所经过的链路集
$E_{k\to k+1}^i$	$P_{i,j}^{(t)}$ 上从放置了 $f_{i,k}^{(t)}$ 的节点出发到达放置了 $f_{i,k+1}^{(t)}$ 的节点所经过的链路集
$E_{s\to 1}^i$	$P_{i,j}^{(t)}$ 上从 s_i 出发到达放置了 $f_{i,1}^{(t)}$ 的节点所经过的链路集
$L(P_{i,j}^t)$	路径 $P_{i,j}^{(t)}$ 上的端到端时延
$L^t(m_i)$	映射 $F_i^{(t)}$ 后，m_i 的端到端时延
$L_{\text{proc}}(f)$	f 的处理时延
L_{proc}^t	系统在处理 MSR_{re}^t 时所产生的处理时延
MSR_{re}^t	$MSR^{(t)}$ 与 MSR^t 之间的差异构成的集合
$N_{v,f}^t$	t 时隙节点 v 上 VNF f 被共享使用的次数
$P_{i,j}^t$	映射 F_i^t 后，从源节点 s_i 到目的节点 $d_{i,j}$ 的路径
$\rho_{i,j,k}^t$	映射 F_i^t 后，$P_{i,j}^t$ 中 VNF $f_{i,k}^t$ 放置的节点
R_v^t	$v\in V$ 在时隙 t 的可用计算资源
U_v^t	t 时隙节点 $v\in V$ 上所放置的 VNF 集合
$(\cdot)$	预测结果相关的时变符号
$\langle\cdot\rangle$	修复子问题相关的符号

记 MSR 中与预测结果相关的时变符号，使用上标“t”标记实际到达相关的时变符号. 记 $MSR^{(t)}=\{msr_i^{(t)}\,|\,i=1,\cdots,|M|\}$ 为 $t-1$ 时隙所预测到的 MSR^t，$msr_i^{(t)}=(m_i,F_i^{(t)},B_i^{(t)})$ 为 t 时隙 m_i 相关的预测 MSR. 其中，若 $F_i^{(t)}\in SFCs$，则表示在预测结果中 m_i 在 t 时隙是活动的；若 $F_i^{(t)}=\varnothing$，则表示 m_i 是非活动的. 得到 $MSR^{(t)}$ 后，将在 $t-1$ 时隙处理所有的预测 MSR. 对于组播组 m_i，记 $P_{i,j}^{(t)}$ 表示映射 $F_i^{(t)}$ 后，从源节点 s_i 到目的节点 $d_{i,j}$ 的路径，$j=1,\cdots,|D_i|$. 记 $\rho_{i,j,k}^{(t)}$ 表示映射 $F_i^{(t)}$ 后，$P_{i,j}^{(t)}$ 中 VNF $f_{i,k}^t$ 放置的节点，$k=1,\cdots,|F_i^{(t)}|$.

路径 $P_{i,j}^{(t)}$ 上的端到端时延主要包含 SFC 的处理时延与传播时延两部分. 记 VNF f 的处理时延为 $L_{\text{proc}}(f)$，则路径 $P_{i,j}^{(t)}$ 上的端到端时延 $L(P_{i,j}^{(t)})$ 表示为

$$L(P_{i,j}^{(t)})=\sum_{e\in E_{s\to 1}^i}L(e)+\sum_{e\in E_{|F|\to d}^i}L(e)+\sum_{k=1}^{|F_i^{(t)}|-1}\sum_{e\in E_{k\to k+1}^i}L(e)+\sum_{f\in F_i^{(t)}}L_{\text{proc}}(f)\tag{1}$$

其中，$|F_i^{(t)}|$ 表示 $F_i^{(t)}$ 中的 VNF 数量. $E_{s\to 1}^i$ 表示在 $P_{i,j}^{(t)}$ 上从 s_i 出发到达放置了 $f_{i,1}^{(t)}$ 的节点所经过的链路集，$E_{|F|\to d}^i$ 表示在 $P_{i,j}^{(t)}$ 上从放置了 $f_{i,|F_i^{(t)}|}^{(t)}$ 的节点出发到 $d_{i,j}$ 所经过的链路集，$E_{k\to k+1}^i$ 表示在 $P_{i,j}^{(t)}$ 上从放置了 $f_{i,k}^{(t)}$ 的节点出发到达放置了 $f_{i,k+1}^{(t)}$ 的节点所经过的链路集，$k=1,\cdots,|F_i^{(t)}|-1$. 记 $L^{(t)}(m_i)$ 为映射 $F_i^{(t)}$ 后，m_i 的端到端时延，定义为 m_i 所有路径中端到端时延的最大值，即

$$L^{(t)}(m_i)=\max_{1\leqslant j\leqslant|D_i|}L(P_{i,j}^{(t)})\tag{2}$$

记 $C_{\text{L}}^{(t)}$ 为 $MSR^{(t)}$ 中所有活动组播组的平均端到端时延，定义为

$$C_{\text{L}}^{(t)}=\operatorname*{avg}_{m_i\in M\text{ is active}}L^{(t)}(m_i)\tag{3}$$

记 U_v^t 为 t 时隙节点 $v\in V$ 上所放置的 VNF 集合. 令 $C_{\text{R}}^{(t)}$ 表示 $MSR^{(t)}$ 中所有 MSR 的计算资源消耗，定义为

$$C_{\text{R}}^{(t)}=\sum_{v\in V}\sum_{f\in U_v^{(t)}}[R_f\cdot(1+N_{v,f}^t\cdot\delta)]\tag{4}$$

其中，$N_{v,f}^t$ 为 t 时隙节点 v 上 VNF f 被共享使用的次数，$\delta\in(0,1)$ 为负载因子表示 f 被多次共享使用时的计算资源占用增量.

由于 VNF 预放置发生在 t 时隙之前，所以本文在 VNF 预放置子问题中不考虑 VNF 初始化时延. 参照于当前研究中的常用做法[8,10,13]，本文将总开销 $C_{\text{pre}}^{(t)}$ 定义为端到端时延和计算资源消耗的权重和. 那么，对于 t 时隙，VNF 预放置子问题可以表述为基于预测结果 $MSR^{(t)}$，最小化 $C_{\text{pre}}^{(t)}$，即

$$\min_{\rho_{i,j,k}^{(t)}}C_{\text{pre}}^{(t)}=\omega_1\cdot C_{\text{L}}^{(t)}+\omega_2\cdot C_{\text{R}}^{(t)}\tag{5}$$

s.t.　$C1$:　$\rho_{i,j,k}^t\in V$，$i=1,\cdots,|M|$，$j=1,\cdots,|D_i|$，$k=1,\cdots,|F_i^t|$

$C2$:　$L^t(m_i)\leqslant L_{\max}(F_i^t)$，$i=1,\cdots,|M|$,

$C3$:　$R_v^t\geqslant\sum_{f\in U_v^t}R_f$，$\forall v\in V$,

$C4$:　$B^t(e)\geqslant\sum_{i=1}^{|M|}\lambda_{i,e}^t\cdot B_i^t$，$\forall e\in E$,

$C5$:　$F_i^t\subseteq\bigcup_{v\in V_{i,j}^t}U_v^t$，$i=1,\cdots,|M|$，$j=1,\cdots,|D_i|$.

其中，ω_1 和 ω_2 为两个权重因子，用以平衡端到端时延 $C_{\text{L}}^{(t)}$ 与计算资源消耗 $C_{\text{R}}^{(t)}$，使得二者保持相同的量纲，$\omega_1+\omega_2=1$. $C1$ 表示 VNF 可以选择 V 中的任意节点进行放置. $C2$ 表示 SFC 的端到端时延限制，即完成 SFC 映射后，组播 m_i 的端到端时延不得超过 SFC 的最大端到端时延. $C3$ 表示计算资源限制，即完成 SFC 映射后，对于任意节点 v，其可用计算资源必须足以支持放置在 v 上的所有 VNF 实例运行. $C4$ 确保了任何流经 SFC 中各个 VNF 实例间链路 e 的流所占用带宽不得超过 e 在该时隙的可用带宽，其中 $\lambda_{i,e}^t\in\{0,1\}$ 是一个二进制变量，表示链路 e 被 m_i 使用的情况，当链路 e 被 m_i 使用时 $\lambda_{i,e}^t=1$，否则 $\lambda_{i,e}^t=0$. $C5$ 保证了 SFC F_i^t 被完整地映射在路径 $P_{i,j}^t$ 的节点上，使得从源节点 s_i 出发的数据流能够在到达目的节点 $d_{i,j}$ 前被 F_i^t 中的每个 VNF 按照 $f_{i,1}^t\to\cdots\to f_{i,|F_i^t|}^t$ 的顺序依次处理，其中 $V_{i,j}^t$ 表示路径 $P_{i,j}^t$ 上节点集合.

3.2　修复子问题

受限于当前的技术手段，预测结果 $MSR^{(t)}$ 与实际结果 MSR^t 之间难免存在一定的差异. 记二者之间的差异为修复集 $MSR_{\text{re}}^t=\{msr_i^t|i=1,\cdots,N_{\text{re}}\}$，其包含所有实际上在 t 时隙到达，但未成功在 $t-1$ 时隙预测到的 MSR. 这里使用下标“$\langle i\rangle$”标记修复子问题相关的符号，msr_i^t 表示 MSR_{re}^t 中第 i 个 MSR，N_{re} 表示在 MSR_{re}^t 中的 MSR 数量，显然 $N_{\text{re}}\leqslant|M|$. 对于 MSR_{re}^t 中的每个 MSR，需要在 t 时隙立即处理. 此外，基于以下两个原因，本文没有立即释放 t 时隙中错误预测的 MSR 所占用的带宽和计算资源. 首先，超量提供 VNF 服务不会影响网络对已成功预测

的 MSR 的响应. 其次，如果 MSR_{re}^{t} 中的某些 MSR 需要错误预测的 MSR 中相同的 VNF，则可以使用上述超量提供的 VNF 来处理，以避免产生对应的 VNF 初始化时延.

记 L_{proc}^{t} 为系统在处理 MSR_{re}^{t} 时所产生的处理时延. 当 $MSR_{\mathrm{re}}^{t}=\varnothing$ 时，$L_{\mathrm{proc}}^{t}=0$；否则，L_{proc}^{t} 将根据系统的处理速度，在处理完 MSR_{re}^{t} 后，实际记录获得. 令 C_{S}^{t} 为系统接收到 MSR^{t} 后开始服务的 MSR 响应时间，表示为

$$C_{\mathrm{S}}^{t}=L_{\mathrm{proc}}^{t}+\max\{L_{\mathrm{init}}(f)|\forall f\in F_{i}^{t},i=1,\cdots,N_{\mathrm{re}}\} \quad (6)$$

需要注意的是，C_{S}^{t} 是一个重要的性能指标，因为其在一定程度上反映了系统对用户请求的响应速度.

那么基于修复子问题的处理结果，对于 t 时隙，本文将解决动态网络环境下 MVNFP 问题的优化目标定义为：最小化端到端时延、计算资源消耗与 MSR 响应时间的总开销 C^{t}，即

$$\min_{\rho_{i,j,k}^{t}} C^{t}=\eta_{1}\cdot(\omega_{1}\cdot C_{\mathrm{L}}^{t}+\omega_{2}\cdot C_{R}^{t})+\eta_{2}\cdot C_{S}^{t} \quad (7)$$

其中，C_{L}^{t} 和 C_{R}^{t} 的计算方式与 $C_{\mathrm{L}}^{(t)}$ 和 $C_{\mathrm{R}}^{(t)}$ 相同，见式(3)和式(4)；η_{1} 与 η_{2} 为权衡 $\omega_{1}\cdot C_{\mathrm{L}}^{t}+\omega_{2}\cdot C_{\mathrm{R}}^{t}$ 与 C_{S}^{t} 重要程度的两个权重因子，$\eta_{1}+\eta_{2}=1$. 显然，当 $MSR_{\mathrm{re}}^{t}=\varnothing$ 时，$C^{t}=\eta_{1}\cdot C_{\mathrm{pre}}^{(t)}$.

4 面向组播的虚拟网络功能放置算法

为了解决动态网络环境下的 MVNFP 问题，本文提出了 SLC-MSRP 算法，包含两阶段和三部分，即 VNF 预放置和修复两个阶段，与基于 Informer 的预测模型、基于足球联赛竞争的组播虚拟网络功能放置算法与快速修复策略三个部分，如图 3 所示. 其中，基于 Informer 的预测模型和基于足球联赛竞争的组播虚拟网络功能放置算法属于第一个阶段，快速修复策略属于第二个阶段. 在 VNF 预放置阶段中，SLC-MSRP 首先采用基于 Informer 的预测模型预测 $MSR^{(t)}$，之后应用基于足球联赛竞争的组播虚拟网络功能放置算法对预测的 MSR 进行预放置和路由. 同时，针对修复子问题中提到的预测结果 $MSR^{(t)}$ 与实际结果 MSR^{t} 不一致的问题，SLC-MSRP 在修复阶段提出了由正向搜索和反向搜索过程构成的快速修复策略，实现 VNF 补充放置和路由，以迅速完成对用户 MSR 的响应. 面向组播的虚拟网络功能放置算法部分的主要符号见表 4.

4.1 基于Informer的预测模型

Informer 是基于 Transformer 架构[29]提出的时间序列预测模型，用以解决长序列时间序列预测(Long Sequence Time-Series Forecasting，LSTF) 问题，由编码器与解码器两部分构成，如图 4 所示. 其中，编码器与解码器使用了多头注意力机制，“多头”即提供多个表征子空间，使模型在不同位置上共同关注来自不同表征子空间的信息. 通过采用“多头”机制，Informer 可以捕捉到更加丰富的特征信息. 相比于传统的 Transformer 在解决长序列时间序列预测问题时，存在的诸如平方时间复杂度、高内存使用量以及预测长输出时速度骤降等问题[28]，Informer 具有三个显著特点：(1) 提出了概率稀疏自注意力替代标准的自注意力机制，将自注意力机制的内存和计算开销从 $O(n^{2})$ 减小到 $O(n\log n)$；(2) 采用了蒸馏 (Distilling) 操作缩短每层的输入序列

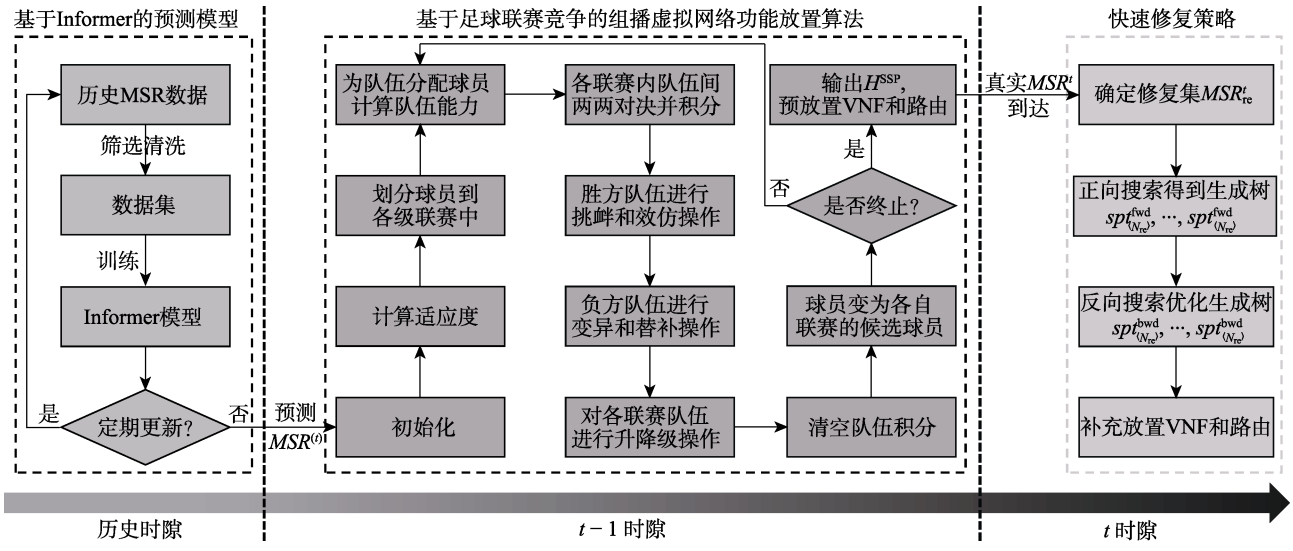

图 3 SLC-MSRP 算法流程图

表 4　面向组播的虚拟网络功能放置算法部分的主要符号

符号	定义	符号	定义
AVG_a^{φ}	TM_a^{φ} 中固定球员的平均水平	$P_{k\to k+1}^{\mathrm{fwd}\langle i\rangle}$	$spt_{\langle i\rangle}^{\mathrm{fwd}}$ 中从 $f_{\langle i\rangle,k}^t$ 到 $f_{\langle i\rangle,k+1}^t$ 的路径
$Fit(H_{\varphi_{m,n}})$	$H_{\varphi_{m,n}}$ 的适应度值	$P_{k\to v\to k-2}^{\langle i\rangle}$	从放置了 $f_{\langle i\rangle,k}^t$ 的节点出发，经过节点 v 后，到达放置了 $f_{\langle i\rangle,k-2}^t$ 的节点的路径，$\forall v\in V_{\mathrm{pla}}^t(f_{\langle i\rangle,k-1}^t)$
$H_{\varphi_{m,n}}$	φ 级联赛中第 m 支队伍的第 n 个球员	$P_{k\to k-1\to k-2}^{\mathrm{bwd}\langle i\rangle}$	spt_i^{bwd} 中从 $f_{i,k}^t$ 经过 $f_{i,k-1}^t$ 到 $f_{i,k-2}^t$ 的路径
$H_{\varphi_m}^{\mathrm{SP}}$	TM_m^{φ} 的明星球员	$PV(TM_a^{\varphi})$	比赛中 TM_a^{φ} 获胜的概率
H^{SSP}	联赛的超级明星球员	$spt_{\langle i\rangle}^{\mathrm{fwd}}$	正向搜索过程确定的生成树
N_{φ}	足球联赛中的级别数量	$spt_{\langle i\rangle}^{\mathrm{bwd}}$	反向搜索过程确定的生成树
N_{FP}	每支队伍固定球员数	TM_m^{φ}	φ 级联赛的第 m 支队伍
N_{pop}	联赛中的球员总数	TP_m^{φ}	TM_m^{φ} 的队伍能力
N_{SB}	每支队伍替补球员数	$V_{\mathrm{pla}}^t(f)$	在 t 时隙中放置了 VNF f 的节点集合
N_{TM}	每个联赛中的球队数	X^t	预测模型在 t 时隙的输入
$P_{k\to v}^{\langle i\rangle}$	从放置了 $f_{\langle i\rangle,k}^t$ 的节点出发到节点 v 的路径 $P_{k\to v}^{\langle i\rangle}$，$\forall v\in V_{\mathrm{pla}}^t(f_{\langle i\rangle,k+1}^t)$	Y^t	预测模型在 t 时隙的输出

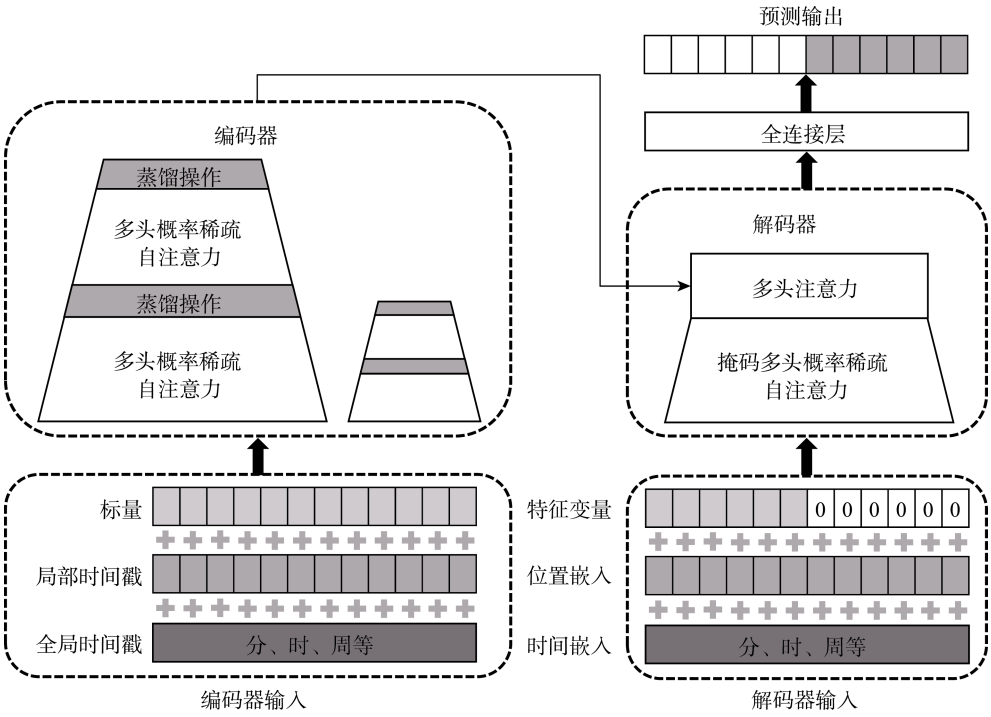

图 4　Informer 模型结构

长度来减少维度和网络参数量，突出主要注意力，以有效地处理极长的输入序列；（3）提出了生成式解码器（Generative Style Decoder），一次性地预测输出整个序列，在提高长序列时间序列预测的推理速度的同时，避免了在推理阶段的误差积累. 长序列时间序列预测问题将预测模型在 t 时隙的输入定义为 $X^t=\{x_1^t,\cdots,x_{|X^t|}^t\}$，输出定义为 $Y^t=\{y_1^t,\cdots,y_{|Y^t|}^t\}$，其中 $|X^t|$ 与 $|Y^t|$ 分别表示 X^t 与 Y^t 的长度. 在进行 MSR 预测时，Informer 根据历史 MSR，对下个时隙的 MSR 进行实时预测. 因此，令 $|X^t|=|Y^t|=|M|\cdot|SFCs|$ 且 $x_k^t,y_k^t\in\{0,1\}$，$k=1,\cdots,|X^t|$. 那么，$x_k^t=1$ 且 $k=(i-1)\cdot|SFCs|+j$，表示在 t 时隙中，组播组 m_i 是活动的，且其请求 $SFCs$ 中的第 j 个 SFC，$|SFCs|$ 为 $SFCs$ 中的 SFC 的数量，$i=1,\cdots,|M|$；$j=1,\cdots,|SFCs|$. 在长序列时间序列预测问题中，数据间的顺序关系十分重要，Transformer 中使用位置嵌入保存数据在序列中的相对位置信息（局部时间戳）. Informer 在保存局部时间戳的基础上，引入全局时间戳的概念，

对分、时、周等时间信息进行编码，通过增加时间嵌入，进一步提高预测模型捕获长期依赖的能力. 最后，Informer 将对位置嵌入信息、时间嵌入信息与特征变量进行融合，作为模型的输入.

4.1.1 概率稀疏自注意力

Transformer 中的自注意力机制存在“长尾（Long Tail）”现象，只有少部分的点积对（Dot-product Pair）贡献了大部分注意力分数，大多数点积对是可以忽略不计的. 在 Informer 中，概率稀疏自注意力机制的公式表示为

$$A(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})=\mathrm{Softmax}\left(\frac{\bar{\boldsymbol{Q}}\cdot\boldsymbol{K}^{\mathrm{T}}}{\sqrt{N_{\mathrm{dim}}}}\right)\cdot\boldsymbol{V} \tag{8}$$

其中，$\boldsymbol{Q}$，$\boldsymbol{K}$，$\boldsymbol{V}$ 分别为由输入特征变量线性变换得到的三个同尺寸的矩阵，N_{dim} 为输入维度，$\boldsymbol{K}^T$ 为 $\boldsymbol{K}$ 的转置，$\bar{\boldsymbol{Q}}$ 是由 $\boldsymbol{Q}$ 稀疏化之后得到的一个稀疏矩阵.

4.1.2 自注意力蒸馏

经过概率稀疏自注意力机制后，编码器的特征映射会产生很多值的冗余组合. 因此，Informer 提出了自注意力蒸馏的操作，来压缩特征维度并突出主要特征. 通过在相邻的注意力模块（Attention Block）之间加入卷积池化操作，来对特征进行降级采样，从第 j 层到第 $j+1$ 层的蒸馏过程表示为

$$X_{j+1}^{t}=\mathrm{MaxPool}(\mathrm{ELU}(\mathrm{Conv1d}([X_j^t]_{\mathrm{AB}}))) \tag{9}$$

其中，$[\cdot]_{\mathrm{AB}}$ 表示注意力模块. 每个模块的输出都会依次经过一个 Conv1d 卷积层，一个 ELU 激活层与一个步长为 2 的 Maxpooling 层.

4.1.3 生成式解码器

Informer 模型的解码器由掩码多头概率稀疏自注意力和多头注意力组成. 其中，掩码多头概率稀疏自注意力在多头概率稀疏自注意力机制上，采用“掩码”的方式将未来位置的信息遮挡起来，以防止解码器提前看到未来的信息. Informer 使用的解码器摈弃了 Transformer 中 step-to-step 的形式，在解码器的最后采用一个全连接层一次输出得到所有的预测结果，解码器的输入 X_{de}^{t} 表示为

$$X_{\mathrm{de}}^{t}=\mathrm{Concat}(X_{\mathrm{token}}^{t},X_0^t) \tag{10}$$

其中，X_{token}^{t} 表示开始字符，包含历史的 MSR 数据. X_0^t 表示目标序列的占位符，其标量用 0 填充. 将需要预测的目标序列标量填充为零，并作为解码器输入的一部分. Informer 未使用 Transformer 中耗时的动态解码方式，而是通过一个前向过程一次性地预测输出所有预测值. 因此，相比于传统编码器-解码器架构，Informer 对长序列时间序列的预测速度有大幅提升.

4.1.4 模型的训练和预测

对于组播 SFC 请求预测问题来说，预测结果多了会导致 VNF 超量提供，造成计算资源的浪费；预测结果少了则会导致实际请求到达时重新求解放置方案，造成更长的 MSR 响应时间. 因此，本文在设计损失函数时，兼顾考虑了这两种情况并为它们设置了权重值. 在训练 Informer 模型时，本文定义如式（11）所示的损失函数以衡量组播 SFC 请求的实际结果与预测结果间的损失.

$$\begin{aligned}Loss=\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}[&\kappa_1\cdot\mathcal{F}(MSR^t-MSR^{(t)})+\\&\kappa_2\cdot\mathcal{F}(MSR^{(t)}-MSR^t)]\end{aligned} \tag{11}$$

其中，$\mathcal{T}$ 为预测的时隙集合，$|\mathcal{T}|$ 为 $\mathcal{T}$ 中的时隙数量. MSR^t 与 $MSR^{(t)}$ 分别为组播 SFC 请求的实际结果与预测结果. 若 $MSR^t-MSR^{(t)}>0$，则说明存在实际结果中 MSR 到达，而预测结果中并未成功进行预测的错误情况，即需要对修复集中的 MSR 进行处理；若 $MSR^t-MSR^{(t)}<0$，则说明存在预测结果中 MSR 到达，而实际结果中 MSR 并未到达的错误情况，即造成 VNF 超量提供. 当门函数 $\mathcal{F}(\cdot)$ 的输入大于 0 时，输出为正值；否则，输出为 0. κ_1 与 κ_2 为权重因子，用以权衡 MSR 实际到达但未成功预测与错误预测造成 VNF 超量提供的重要程度，$\kappa_1+\kappa_2=1$.

在训练时，将历史组播 SFC 请求作为训练样本，通过学习速率（Learning Rate）ε，迭代训练更新模型参数，以最小化损失 $Loss$，直到模型收敛为止. 训练完毕后，在 VNF 预放置阶段对历史组播 SFC 请求进行预处理，并将其输入 Informer 模型以获得下一个组播 SFC 请求. 之后，将预测结果提交给组播虚拟网络功能放置算法，以完成 VNF 预放置子问题的求解.

4.2 基于足球联赛竞争的组播虚拟网络功能放置算法

足球联赛竞争算法为受足球联赛中球队和球员间的竞争机制启发，而提出的一种群体优化算法[17]. SLC 将球员视为种群中的个体，即每名球员代表问题的一个解决方案. 每个联赛中的球队由固定球员（Fixed Players，FP）和替补球员（Substitutes，SB）组成，采取积分排名方式，利用球队之间、球员之间相互竞争，实现种群的整体高效进化. 本文结合动态网络环境下的 MVNFP 问题特点，将整个足球联赛划分为 N_φ 个级别的联赛，每个联赛中各有 N_{TM}

支球队. 一轮赛季开始时，一个联赛中的队伍两两对决，胜方积 3 分，负方不得分，共进行 $(N_{\mathrm{TM}}\cdot(N_{\mathrm{TM}}-1))/2$ 场比赛. 一轮赛季结束时，将根据本轮赛季的积分对联赛中的队伍进行升降级操作，其中 φ 级联赛积分排名靠后的两支队伍将降级到 $\varphi+1$ 级联赛中，同时 $\varphi+1$ 级联赛积分排名靠前的两支队伍将升级到 φ 级联赛中，$\varphi=1,\cdots,N_{\varphi}$.

每支队伍有 N_{FP} 名固定球员和 N_{SB} 名替补球员，则所有联赛中的球员总数 $N_{\mathrm{pop}}=N_{\varphi}\cdot[N_{\mathrm{TM}}\cdot(N_{\mathrm{FP}}+N_{\mathrm{SB}})]$. 记 $TM_m^{\varphi}=\{H_{\varphi_{m,1}},\cdots,H_{\varphi_{m,N_{\mathrm{FP}}+N_{\mathrm{SB}}}}\}$ 为 φ 级联赛的第 m 支队伍，$\varphi=1,\cdots,N_{\varphi}$，$m=1,\cdots,N_{\mathrm{TM}}$. $H_{\varphi_{m,n}}$ 表示 φ 级联赛中第 m 支队伍的第 n 个球员，$n=1,\cdots,N_{\mathrm{FP}}+N_{\mathrm{SB}}$. 这里，为了方便区分一支队伍的固定球员和替补球员，本文将队伍中球员按照先固定球员再替补球员的顺序编号，如对于 TM_m^{φ} 中的球员 $H_{\varphi_{m,n}}$，当 $n\leqslant N_{\mathrm{FP}}$ 时，表明 $H_{\varphi_{m,n}}$ 是该支队伍的固定球员；当 $N_{\mathrm{FP}}+1\leqslant n\leqslant N_{\mathrm{FP}}+N_{\mathrm{SB}}$ 时，表明 $H_{\varphi_{m,n}}$ 是该支队伍的替补球员. 每名球员表示动态网络环境下的 MVNFP 问题的一个解，由于解决动态网络环境下的 MVNFP 问题中存在多个活动的组播组，单个球员需要表示 $MSR^{(t)}$ 中所有 MSR 的映射方案，本文在文献[4]针对 MVNFP 问题的二维个体编码基础上，提出了多维个体编码策略. 具体来说，将一名球员 $H_{\varphi_{m,n}}$ 表示为当前时隙中所有组播组 VNF 放置方案的集合，即 $H_{\varphi_{m,n}}=\{h_{\varphi_{m,n}}^{i}\,|i=1,\cdots,|M|\}$. 其中，$h_{\varphi_{m,n}}^{i}$ 是针对 m_i 的 VNF 放置方案，表示为

$$h_{\varphi_{m,n}}^{i}=\begin{bmatrix}\hbar_{\varphi_{m,n}}^{i,1,1} & \cdots & \hbar_{\varphi_{m,n}}^{i,1,|F_i|}\\ \vdots & \ddots & \vdots\\ \hbar_{\varphi_{m,n}}^{i,|D_i|,1} & \cdots & \hbar_{\varphi_{m,n}}^{i,|D_i|,|F_i|}\end{bmatrix} \tag{12}$$

其中，$\hbar_{\varphi_{m,n}}^{i,j,k}$ 表示 TM_m^{φ} 的第 n 名球员，对于组播 SFC 请求 msr_i 中针对 m_i 的第 j 条路径上第 k 个 VNF 的放置位置，$\hbar_{\varphi_{m,n}}^{i,j,k}\in V$，$j=1,\cdots,|D_i|$；$k=1,\cdots,|F_i|$. 当 m_i 非活动时，$h_{\varphi_{m,n}}^{i}=\varnothing$. 记 $Fit(H_{\varphi_{m,n}})$ 为 $H_{\varphi_{m,n}}$ 的适应度值，$Fit(H_{\varphi_{m,n}})$ 由式（5）计算得出. 每支队伍存在一名关键球员，称为该支队伍的明星球员（Star Player，SP），记 $H_{\varphi_m}^{\mathrm{SP}}$ 表示 TM_m^{φ} 的明星球员，则

$$Fit(H_{\varphi_m}^{\mathrm{SP}})=\min\{Fit(H_{\varphi_{m,n}})|\forall H_{\varphi_{m,n}}\in TM_m^{\varphi}\} \tag{13}$$

此外，在所有联赛中存在一名特殊的球员，称为超级明星球员(Super Star Player，SSP)，记 H^{SSP} 表示超级明星球员，则

$$\begin{aligned}Fit(H^{\mathrm{SSP}})=\min\{&Fit(H_{\varphi_{m,n}})\,|\,\varphi=1,\cdots,N_{\varphi},\\ &m=1,\cdots,N_{\mathrm{TM}};n=1,\cdots,N_{\mathrm{FP}}+N_{\mathrm{SB}}\}\end{aligned} \tag{14}$$

每支队伍的实力由该支队伍中的球员共同决定，记 TP_m^{φ} 表示 TM_m^{φ} 的队伍能力，定义为

$$TP_m^{\varphi}=\frac{\sum_{H_{\varphi_{m,n}}\in TM_m^{\varphi}}Fit(H_{\varphi_{m,n}})}{N_{\mathrm{FP}}+N_{\mathrm{SB}}} \tag{15}$$

在每场比赛中，实力更强的队伍获胜的概率更高. 由于解决动态网络环境下的 MVNFP 问题是最小化问题，队伍能力值更小的队伍，实力反而更强，因此这里定义 TM_a^{φ} 与 TM_b^{φ}，$a,b\in\{1,\cdots,N_{\mathrm{TM}}\}$，$a\neq b$，在一场比赛中各自获胜的概率 $PV(TM_a^{\varphi})$ 与 $PV(TM_b^{\varphi})$ 为

$$PV(TM_a^{\varphi})=\frac{1/TP_a^{\varphi}}{1/TP_a^{\varphi}+1/TP_b^{\varphi}} \tag{16}$$

$$PV(TM_b^{\varphi})=\frac{1/TP_b^{\varphi}}{1/TP_a^{\varphi}+1/TP_b^{\varphi}} \tag{17}$$

这里假设比赛结束后，TM_a^{φ} 获胜，TM_b^{φ} 落败. 那么，TM_a^{φ} 的各名球员将分别进行效仿（Imitation）与挑衅（Provocation）操作，TM_b^{φ} 的各名球员将分别进行变异（Mutation）与替补（Substitution）操作.

4.2.1　效仿操作

对于胜方队伍 TM_a^{φ} 的固定球员，他们将同时效仿本队的明星球员和超级明星球员以提高自身的实力，定义为

$$\begin{aligned}H_{\varphi_{a,n}}=&\mu_1\cdot H_{\varphi_{a,n}}+\tau_1\cdot(H^{\mathrm{SSP}}-H_{\varphi_{a,n}})+\\ &\tau_2\cdot(H_{\varphi_a}^{\mathrm{SP}}-H_{\varphi_{a,n}}),\quad n=1,\cdots,N_{\mathrm{FP}}\end{aligned} \tag{18}$$

其中，μ_1 被用以确定 $H_{\varphi_{m,n}}$ 对 H^{SSP} 与 $H_{\varphi_a}^{\mathrm{SP}}$ 进行效仿的程度，$\mu_1\in(\theta,\delta)$，$1\leqslant\delta\leqslant 2$，$0\leqslant\theta\leqslant 1$. τ_1 与 τ_2 是两个随机值，$\tau_1,\tau_2\in(0,2)$. 若更新后该固定球员的适应度值比之前更优（更低），则更新完毕，否则使用式（19）对其进行更新，其中 μ_2 与 μ_1 的作用相似，$\mu_2\in(0,\theta)$.

$$\begin{aligned}H_{\varphi_{a,n}}=&\mu_2\cdot H_{\varphi_{a,n}}+\tau_1\cdot(H^{\mathrm{SSP}}-H_{\varphi_{a,n}})+\tau_2\cdot(H_{\varphi_a}^{\mathrm{SP}}-H_{\varphi_{a,n}}),\\ &n=1,\cdots,N_{\mathrm{FP}}\end{aligned} \tag{19}$$

需要注意的是，由于 $\hbar_{\varphi_{m,n}}^{i,j,k}$ 为 VNF 的放置位置，其值是整数表示的节点编号，当更新操作后，$\hbar_{\varphi_{m,n}}^{i,j,k}$ 的取值不为整数时，本文将对其采取四舍五入的操作使得 $\hbar_{\varphi_{m,n}}^{i,j,k}\in V$，$j=1,\cdots,|D_i|$；$k=1,\cdots,|F_i|$.

4.2.2　挑衅操作

在挑衅操作中，胜方队伍 TM_a^{φ} 中的替补球员将向着 TM_a^{φ} 中固定球员的平均水平靠近. 令 AVG_a^{φ} 代表 TM_a^{φ} 中固定球员的平均水平，表示为

$$AVG_a^{\varphi}=\frac{1}{N_{\mathrm{FP}}}\cdot\sum_{n=1}^{N_{\mathrm{FP}}}H_{\varphi_{a,n}} \tag{20}$$

对于替补球员的挑衅操作定义为

$$H_{\varphi_{a,n}}=AVG_a^{\varphi}+\chi_1\cdot(AVG_a^{\varphi}-H_{\varphi_{a,n}}),\quad n=N_{\mathrm{FP}}+1,\cdots,N_{\mathrm{FP}}+N_{\mathrm{SB}} \tag{21}$$

其中，χ_1为随机值，$\chi_1\in(0.9,1)$. 若更新后该替补球员的适应度值比之前更优（更低），则更新完毕，否则使用式（22）对其进行更新. 其中，χ_2为随机值，$\chi_2\in(0.4,0.6)$.

$$H_{\varphi_{a,n}}=AVG_a^{\varphi}+\chi_2\cdot(H_{\varphi_{a,n}}-AVG_a^{\varphi}),\quad n=N_{\mathrm{FP}}+1,\cdots,N_{\mathrm{FP}}+N_{\mathrm{SB}} \tag{22}$$

若使用式(22)更新后，该替补球员的适应度值仍然比之前更差，则重新随机生成该名球员.

4.2.3 变异操作

在完成胜方队伍的相关操作后，负方队伍TM_b^{φ}中的固定球员将按照变异概率P_{mu}进行变异操作. 对于固定球员$H_{\varphi_{a,n}}$，$n=1,\cdots,N_{\mathrm{FP}}$，的变异操作与遗传算法中的变异操作类似[33]，即对于$h_{\varphi_{m,n}}^{i}$中的每一位编码$\hbar_{\varphi_{m,n}}^{i,j,k}$按照变异概率进行变异操作，$i=1,\cdots,|M|$，$j=1,\cdots,|D_i|$，$k=1,\cdots,|F_i|$.

4.2.4 替补操作

在替补操作中，负方队伍TM_b^{φ}中的替补球员将尝试两两配对组合以期产生新的个体. 对于任意一对替补球员$H_{\varphi_{b,n}}$与$H_{\varphi_{b,l}}$，$n,l=N_{\mathrm{FP}}+1,\cdots,N_{\mathrm{FP}}+N_{\mathrm{SB}}$，$n\neq l$，的替补操作定义为

$$H_{\varphi_{b,n}}=\varpi\cdot H_{\varphi_{b,n}}+(1-\varpi)\cdot H_{\varphi_{b,l}} \tag{23}$$

$$H_{\varphi_{b,l}}=\varpi\cdot H_{\varphi_{b,l}}+(1-\varpi)\cdot H_{\varphi_{b,n}} \tag{24}$$

其中，ϖ为一个随机数，$\varpi\in(0,1)$. 进行替换操作时，从替补球员中随机选择N_{SB}对球员依次进行式（23）和式（24）的操作，若产生的新球员满足约束 C1~C5，则此次替补操作成功，否则放弃此次替补操作.

4.2.5 基于 SLC 的 MVNFP 算法

收到 Informer 的预测结果$MSR^{(t)}$之后，SLC 首先随机生成N_{pop}名球员，计算球员的适应度，确定超级明星球员，将球员们划分到各级联赛中. 接着，各级联赛划分球员到各支队伍中，计算各支队伍的能力，确定队伍明星球员. 之后，各级联赛内队伍两两对决，并根据结果积分对胜方队伍的球员分别进行效仿与挑衅操作，对负方队伍的球员分别进行变异与替补操作. 最后，对各级联赛中的队伍进行升级与降级操作，确定最新的超级明星球员. 通过开展新的赛季，球队之间、球员之间相互竞争，算法不断迭代，最终产生针对$MSR^{(t)}$的解H^{SSP}. 本文所提基于 SLC 的 MVNFP 算法的伪代码如算法 1 所示.

算法 1. 基于 SLC 的 MVNFP 算法

输入：预测结果$MSR^{(t)}$，以及算法最大运行时间N_{time}、联赛数量N_{φ}、联赛球队数N_{TM}、固定球员数N_{FP}、替补球员数N_{SB}和变异概率P_{mu}等算法参数.

输出：超级明星球员H^{SSP}.

```
随机生成 N_pop 名球员，计算他们的适应度，确定 H^SSP，按照球员的适应度从小到大排名，将第 (φ−1)·N_pop/N_φ+1 到第 φ·N_pop/N_φ 名球员划分到 φ 级联赛中，φ=1,⋯,N_φ；
WHILE 当前算法运行时间 < N_time DO
    FOR φ=1,⋯,N_φ DO
        对 φ 级联赛中的球员按照适应度的大小从小到大进行排名；
        FOR m=1,⋯,N_TM DO
            按照排名，将第 (m−1)·(N_FP+N_SB)+1 到第 m·(N_FP+N_SB) 名球员划分到队伍 TM_m^φ 中；
            计算队伍能力 TP_m^φ，确定 H_{φ_m}^SP；
        FOR a=1,⋯,N_TM DO
        // φ 级联赛内队伍两两对决
            FOR b=a+1,⋯,N_TM DO
                TM_a^φ 与 TM_b^φ 对决，并根据对决结果进行积分；
                胜方队伍的球员进行效仿、挑衅操作；
                负方队伍的球员进行变异、替补操作；
    对 φ 级联赛积分排名最差的两支队伍降级，即把它们加入到 φ+1 级联赛中；
    对 φ+1 级联赛积分排名最差的两支队伍升级，即把它们加入到 φ 级联赛中；
    确定最新的 H^SSP；
```

4.3 快速修复策略

当t时隙到来，通过比较预测结果$MSR^{(t)}$与实际结果MSR^t之间的差异，得到修复集MSR_{re}^t. 快速修复策略需要迅速处理MSR_{re}^t中的每个 MSR，以降低 MSR 处理时延L_{proc}^t，减少用户等待时间. 令$V_{\mathrm{pla}}^t(f)\subseteq V$表示在$t$时隙中放置了 VNF f 的节点集合. 若在t时隙中没有节点放置f，则设$V_{\mathrm{pla}}^t(f)=V$. 为了充分利用已经放置的 VNF 并减少 VNF 初始化时延，在快速修复策略中，仅将每个$v\in V_{\mathrm{pla}}^t(f)$作为放置$f$的候选节点. 本文提出的快速修复策略由正向搜索与反向搜索两个过程组成，在正向搜索过程中为每个 MSR 构造一棵生成树，在反向搜索过

程中对该生成树进行改进. 注意，这里采用 Dijkstra 算法寻找网络中两个节点之间的路径.

4.3.1　正向搜索

对于在 t 时隙到达的 MSR $msr^t_{\langle i\rangle}$，正向搜索过程会针对组播 m_i 建立一个生成树，$i=1,\cdots,N_{\mathrm{re}}$. 首先，将所有放置了 VNF $f^t_{\langle i\rangle,1}$ 的节点作为候选节点，然后分别建立从 $s_{\langle i\rangle}$ 出发到所有候选节点 v 的路径 $P^{\langle i\rangle}_{s\to v}$. 之后，从这些路径中找出端到端时延最小的路径确定为从 $s_{\langle i\rangle}$ 出发到 VNF $f^t_{\langle i\rangle,1}$ 的路径，记为 $P^{\mathrm{fwd}\langle i\rangle}_{s\to 1}$. 记 $L(P^{\langle i\rangle}_{s\to v})$ 为路径 $P^{\langle i\rangle}_{s\to v}$ 的端到端时延，则

$$L(P^{\mathrm{fwd}\langle i\rangle}_{s\to 1})=\min\{L(P^{\langle i\rangle}_{s\to v})|\forall v\in V^t_{\mathrm{pla}}(f^t_{\langle i\rangle,1})\} \quad (25)$$

接着，按照同样的方式，建立从放置了 $f^t_{\langle i\rangle,k}$ 的节点出发到节点 v 的路径 $P^i_{k\to v}$，从中选出端到端时延最小的路径，确定为从 $f^t_{\langle i\rangle,k}$ 到 $f^t_{\langle i\rangle,k+1}$ 的路径 $P^{\mathrm{fwd}\langle i\rangle}_{k\to k+1}$，$k=1,\cdots,|F^t_{\langle i\rangle}|-1$. 记 $L(P^{\langle i\rangle}_{k\to v})$ 为路径 $P^{\langle i\rangle}_{k\to v}$ 的端到端时延，则

$$L(P^{\mathrm{fwd}\langle i\rangle}_{k\to k+1})=\min\{L(P^{\langle i\rangle}_{k\to v})|\forall v\in V^t_{\mathrm{pla}}(f^t_{\langle i\rangle,k+1})\} \quad (26)$$

然后，建立从放置了 $f^t_{\langle i\rangle,|F^t_{\langle i\rangle}|}$ 的节点出发到目的节点 $d_{\langle i\rangle,j}$ 的路径 $P^{\mathrm{fwd}\langle i\rangle}_{|F|\to d}$，并将其确定为从 $f^t_{\langle i\rangle,|F^t_{\langle i\rangle}|}$ 到 $d_{\langle i\rangle,j}$ 的路径，$j=1,\cdots,|D_{\langle i\rangle}|$. 最后，通过合并之前确定的路径来构造 $m_{\langle i\rangle}$ 的生成树 $spt^{\mathrm{fwd}}_{\langle i\rangle}$. 通过以上步骤，在建立生成树时，会优先考虑已经放置过相同功能 VNF 的节点，以节点计算资源. 同时，确定 VNF $f\in F^t_{\langle i\rangle}$ 放置的节点 $v\in V^t_{\mathrm{pla}}(f)$ 后，若 v 已存在相同功能的 VNF 实例，则会直接共享使用该实例，此时不会产生 VNF 初始化时延 $L_{\mathrm{init}}(f)$；否则，将会在 v 上实例化 f. 通过充分利用已放置的 VNF，达到节约节点计算资源与减少 VNF 初始化时延的目的. 本文所提快速修复策略中正向搜索过程的伪代码如算法 2 所示.

算法 2.　快速修复策略中正向搜索过程

输入：修 1 复集 MSR^t_{re}.

输出：正向搜索确定的生成树，$spt^{\mathrm{fwd}}_{\langle 1\rangle},\cdots,spt^{\mathrm{fwd}}_{\langle N_{\mathrm{re}}\rangle}$.

1. FOR　$i=1,\cdots,N_{\mathrm{re}}$　DO
2.　　FOR　$v\in V^t_{\mathrm{pla}}(f^t_{\langle i\rangle,1})$　DO
3.　　建立路径 $P^{\langle i\rangle}_{s\to v}$，计算该路径的端到端时延；
4.　　根据公式(25)，确定从 $s_{\langle i\rangle}$ 到 $f^t_{\langle i\rangle,1}$ 的路径 $P^{\mathrm{fwd}\langle i\rangle}_{s\to 1}$；
5.　　FOR　$k=1,\cdots,|F^t_{\langle i\rangle}|-1$　DO
6.　　FOR　$v\in V^t_{\mathrm{pla}}(f^t_{\langle i\rangle,k+1})$　DO
7.　　建立路径 $P^{\langle i\rangle}_{k\to v}$，计算该路径的端到端时延；
8.　　根据公式(26)，确定从 $f^t_{\langle i\rangle,k}$ 到 $f^t_{\langle i\rangle,k+1}$ 的路径 $P^{\mathrm{fwd}\langle i\rangle}_{k\to k+1}$；
9.　　FOR　$j=1,\cdots,|D_i|$　DO
10.　　建立路径 $P^{\mathrm{fwd}\langle i\rangle}_{|F|\to d}$，并将其确定为从 $f^t_{\langle i\rangle,|F^t_{\langle i\rangle}|}$ 到 $d_{\langle i\rangle,j}$ 的路径；
11.　　合并确定的路径，构造 $m_{\langle i\rangle}$ 的生成树 $spt^{\mathrm{fwd}}_{\langle i\rangle}$；

4.3.2　反向搜索

通过正向搜索过程为每个 MSR 构造生成树后，快速修复策略在反向搜索过程中对正向搜索过程建立的生成树进行改进. 反向搜索过程基于文献[6]的思想，通过反向检查从目的节点到源节点的每一条路径，寻找一个具有更小端到端时延的路径作为替代方案. 具体来说，将在 $spt^{\mathrm{fwd}}_{\langle i\rangle}$ 中，对于组播 $m_{\langle i\rangle}$ 的路径 $P^t_{\langle i\rangle,j}$，一个接一个地检查放置了 $f^t_{\langle i\rangle,|F^t_i|},\cdots,f^t_{\langle i\rangle,1}$ 的节点，$i=1,\cdots,N_{\mathrm{re}}$. 首先，建立从 $d_{\langle i\rangle,j}$ 出发，经过节点 v 后，到达放置了 $f^t_{\langle i\rangle,|F^t_i|-1}$ 的节点的路径 $P^{\langle i\rangle}_{d\to v\to|F|-1}$，$\forall v\in V^t_{\mathrm{pla}}(f^t_{\langle i\rangle,|F^t_{\langle i\rangle}|})$，从中选出端到端时延最小的路径，确定为从 $d_{\langle i\rangle,j}$ 经过 $f^t_{\langle i\rangle,|F^t_i|}$ 到 $f^t_{\langle i\rangle,|F^t_i|-1}$ 的路径 $P^{\mathrm{bwd}\langle i\rangle}_{d\to|F|\to|F|-1}$，$j=1,\cdots,|D_i|$. 记 $L(P^{\langle i\rangle}_{d\to v\to|F|-1})$ 为路径 $P^{\langle i\rangle}_{d\to v\to|F|-1}$ 的端到端时延，则

$$L(P^{\mathrm{bwd}\langle i\rangle}_{d\to|F|\to|F|-1})=\min\{L(P^{\langle i\rangle}_{d\to v\to|F|-1})|\forall v\in V^t_{\mathrm{pla}}(f^t_{\langle i\rangle,|F^t_i|})\} \quad (27)$$

接着，按照同样的方式，建立从放置了 $f^t_{i,k}$ 的节点出发，经过节点 v 后，到达放置了 $f^t_{\langle i\rangle,k-2}$ 的节点的路径 $P^{\langle i\rangle}_{k\to v\to k-2}$，$\forall v\in V^t_{\mathrm{pla}}(f^t_{\langle i\rangle,k-1})$，从中选出端到端时延最小的路径，确定为从 $f^t_{\langle i\rangle,k}$ 经过 $f^t_{\langle i\rangle,k-1}$ 到 $f^t_{\langle i\rangle,k-2}$ 的路径 $P^{\mathrm{bwd}\langle i\rangle}_{k\to k-1\to k-2}$，$k=|F^t_{\langle i\rangle}|,\cdots,3$. 记 $L(P^{\langle i\rangle}_{k\to v\to k-2})$ 为路径 $P^{\langle i\rangle}_{k\to v\to k-2}$ 的端到端时延，则

$$L(P^{\mathrm{bwd}\langle i\rangle}_{k\to k-1\to k-2})=\min\{L(P^{\langle i\rangle}_{k\to v\to k-2})|\forall v\in V^t_{\mathrm{pla}}(f^t_{\langle i\rangle,k-1}))\} \quad (28)$$

然后，建立从放置了 $f^t_{\langle i\rangle,2}$ 的节点出发，经过节点 v 后，到达 $s_{\langle i\rangle}$ 的路径 $P^{\langle i\rangle}_{2\to v\to s}$，$\forall v\in V^t_{\mathrm{pla}}(f^t_{\langle i\rangle,1})$，从中选出端到端时延最小的路径，确定为从 $f^t_{\langle i\rangle,2}$ 经过 $f^t_{\langle i\rangle,1}$ 到 $s_{\langle i\rangle}$ 的路径 $P^{\mathrm{bwd}\langle i\rangle}_{2\to 1\to s}$. 记 $L(P^{\langle i\rangle}_{2\to v\to s})$ 为路径 $P^{\langle i\rangle}_{2\to v\to s}$ 的端到端时延，则

$$L(P^{\mathrm{bwd}\langle i\rangle}_{2\to 1\to s})=\min\{L(P^{\langle i\rangle}_{2\to v\to s})|\forall v\in V^t_{\mathrm{pla}}(f^t_{\langle i\rangle,1}))\} \quad (29)$$

最后，用反向搜索中确定的路径替换 $spt^{\mathrm{fwd}}_{\langle i\rangle}$ 中对应路径，得到优化后的生成树 $spt^{\mathrm{bwd}}_{\langle i\rangle}$. 本文所提快速修复策略中反向搜索过程的伪代码如算法 3 所示.

算法 3.　快速修复策略中反向搜索过程

输入：正向搜索确定的生成树，$spt^{\mathrm{fwd}}_{\langle 1\rangle},\cdots,spt^{\mathrm{fwd}}_{\langle N_{\mathrm{re}}\rangle}$.

输出：反向搜索优化后的生成树，$spt^{\mathrm{bwd}}_{\langle 1\rangle},\cdots,spt^{\mathrm{bwd}}_{\langle N_{\mathrm{re}}\rangle}$.

1. FOR　$i=1,\cdots,N_{\mathrm{re}}$　DO
2.　　FOR　$j=1,\cdots,|D_i|$　DO
3.　　FOR　$v\in V^t_{\mathrm{pla}}(f^t_{\langle i\rangle,|F^t_{\langle i\rangle}|})$　DO
4.　　建立路径 $P^{\langle i\rangle}_{d\to v\to|F|-1}$，计算路径的端到端时延；

5. 根据公式(27)，确定从 $d_{\langle i\rangle,j}$ 经过 $f^t_{\langle i\rangle,|F_i^t|}$ 到 $f^t_{\langle i\rangle,|F_i^t|-1}$ 的路径 $P^{\text{bwd}\langle i\rangle}_{d\to|F|\to|F|-1}$；
6. FOR $k=|F^t_{\langle i\rangle}|,\cdots,3$ DO
7. FOR $v\in V^t_{\text{pla}}(f^t_{\langle i\rangle,k-1})$ DO
8. 建立路径 $P^{\langle i\rangle}_{k\to v\to k-2}$，计算该路径的端到端时延；
9. 根据公式(28)，确定从 $f^t_{\langle i\rangle,k}$ 经过 $f^t_{\langle i\rangle,k-1}$ 到 $f^t_{\langle i\rangle,k-2}$ 的路径 $P^{\text{bwd}\langle i\rangle}_{k\to k-1\to k-2}$；
10. FOR $v\in V^t_{\text{pla}}(f^t_{\langle i\rangle,1})$ DO
11. 建立路径 $P^{\langle i\rangle}_{2\to v\to s}$，计算该路径的端到端时延；
12. 根据公式(29)，确定从 $f^t_{\langle i\rangle,2}$ 经过 $f^t_{\langle i\rangle,1}$ 到 s_i 的路径 $P^{\text{bwd}\langle i\rangle}_{2\to1\to s}$；
13. 用确定的路径替换 $spt^{\text{fwd}}_{\langle i\rangle}$ 中对应的路径，得到 $spt^{\text{bwd}}_{\langle i\rangle}$；

4.4 算法时间复杂度分析

本文提出的 SLC-MSRP 算法，通过采用历史组播 SFC 请求对 Informer 预测模型进行训练，以实现对 MSR 进行预测. 在 VNF 预放置阶段，对即将到达的 MSR 进行预测，基于预测的 MSR，使用基于的足球联赛竞争的 MVNFP 算法，进行预放置和路由. 在修复阶段,针对预测 MSR 与实际到达的 MSR 不一致的情况，通过快速修复策略，以迅速响应实际到达的 MSR. 在对预测模型进行训练时，训练 Informer 模型的时间复杂度为 $O(\mathcal{L}\log\mathcal{L})$[28]，其中 $\mathcal{L}=|M|\cdot|SFCs|$ 为序列长度.

在 VNF 预放置阶段,采用 Informer 预测模型对即将到达的 MSR 进行预测的时间复杂度为 $O(1)$[28]. 基于足球联赛竞争的 MVNFP 算法（算法 1）可以被进一步划分为初始化和主循环两个部分. 令 $O(\mathcal{N})$ 表示评估球员适应度的时间复杂度. 在初始化部分，算法 1 中，步骤 1 生成并评估 N_{pop} 名球员的时间复杂度为 $O(\mathcal{N}\cdot N_{\text{pop}})$，对球员们的适应度进行快速排序的时间复杂度为 $O(N_{\text{pop}}\cdot\log N_{\text{pop}})$，将他们划分到各级联赛的时间复杂度为 $O(N_{\text{pop}})$. 事实上适应度评估的时间复杂度远高于排序和联赛划分，因此初始化部分的时间复杂度为 $O(\mathcal{N}\cdot N_{\text{pop}})$. 在主循环部分，步骤 4 中，对 φ 级联赛中的球员排名的时间复杂度为 $O(N_{\text{pop}}/N_\varphi\cdot\log(N_{\text{pop}}/N_\varphi))$. 步骤 6~7 进行队伍划分和计算队伍能力时，队伍能力的计算可在常数时间内完成，队伍划分的时间复杂度为 $O(N_{\text{pop}}/N_\varphi)$. 因此，步骤 3~7 的时间复杂度为 $O(N_{\text{pop}}\cdot\log N_{\text{pop}}/N_\varphi+N_{\text{pop}}\cdot N_{\text{TM}})$. 步骤 10 中，两支队伍的对决可在常数时间内完成. 步骤 11~12 中，效仿、挑衅、变异与替补操作需要对球员的个体编码进行遍历，其时间复杂度为 $O(|M|\cdot|F|\cdot|D|)$，其中 $|M|$、$|F|$ 和 $|D|$ 分别为组播组数量、SFC 长度和目的节点数量. 那么，步骤 8~12 的时间复杂度为 $O(N_{\text{pop}}\cdot N_{\text{TM}}\cdot(\mathcal{J}+\mathcal{N}))$，其中 $\mathcal{J}=|M|\cdot|F|\cdot|D|$. 假设算法 1 的迭代次数为 N_{iter}，那么主循环部分的时间复杂度为 $O(N_{\text{iter}}\cdot N_{\text{pop}}\cdot N_{\text{TM}}\cdot(\mathcal{J}+\mathcal{N})+N_{\text{iter}}\cdot N_{\text{pop}}\cdot\log N_{\text{pop}}/N_\varphi)$. 由于 Informer 模型进行预测的时间复杂度为 $O(1)$，且算法 1 中主循环部分的时间复杂度远高于初始化部分. 因此，VNF 预放置阶段的时间复杂度接近于基于足球联赛竞争的 MVNFP 算法的时间复杂度，即 $O(N_{\text{iter}}\cdot N_{\text{pop}}\cdot N_{\text{TM}}\cdot(\mathcal{J}+\mathcal{N})+N_{\text{iter}}\cdot N_{\text{pop}}\cdot\log N_{\text{pop}}/N_\varphi)$.

在修复阶段，SLC-MSRP 算法中的快速修复策略分为正向搜索（算法 2）与反向搜索（算法 3）两个过程. 令 N_{pla} 表示 $V^t_{\text{pla}}(f)$ 中的节点个数. 在算法 2 中，步骤 2~3 采用 Dijkstra 算法建立 N_{pla} 条路径的时间复杂度为 $O(N_{\text{pla}}\cdot|V|^2)$，步骤 4 中选择端到端时延最短路径的时间复杂度为 $O(N_{\text{pla}})$. 那么，算法 2 步骤 2~4 的时间复杂度为 $O(N_{\text{pla}}\cdot|V|^2)$. 类似地，算法 2 步骤 5~8 的时间复杂度为 $O(N_{\text{pla}}\cdot|F|\cdot|V|^2)$. 算法 2 步骤 9~10 建立从 SFC 中放置最后一个 VNF 的节点到目的节点的路径的时间复杂度为 $O(|F|\cdot|D|)$，步骤 11 对多条路径进行合并以构成生成树的时间复杂度为 $O(|D|)$. 因此，算法 2 的时间复杂度为 $O(N_{\text{re}}\cdot N_{\text{pla}}\cdot|V|^2+N_{\text{re}}\cdot|F|\cdot|D|)$. 在算法 3 中，步骤 3~4 建立从目的节点到 SFC 中放置倒数第二个 VNF 的节点的时间复杂度为 $O(N_{\text{pla}}\cdot|V|^2)$，步骤 5 选择端到端时延最短路径的时间复杂度为 $O(N_{\text{pla}})$. 那么算法 3 步骤 2~5 的时间复杂度为 $O(N_{\text{pla}}\cdot|D|\cdot|V|^2)$. 同理可得，算法 3 步骤 6~9 的时间复杂度为 $O(N_{\text{pla}}\cdot|F|\cdot|V|^2)$，步骤 10~12 的时间复杂度为 $O(N_{\text{pla}}\cdot|V|^2)$. 因此，算法 3 的时间复杂度为 $O(N_{\text{re}}\cdot N_{\text{pla}}\cdot|V|^2\cdot(|F|+|D|))$. 综上所述，修复阶段的时间复杂度为 $O(N_{\text{re}}\cdot N_{\text{pla}}\cdot|V|^2\cdot(|F|+|D|)+N_{\text{re}}\cdot|F|\cdot|D|)$.

5 仿真结果与分析

5.1 仿真设置

为了验证所提 SLC-MSRP 算法的性能，本文假定系统中存在 10 种类型的 VNF，每条 SFC 中的 VNF 数量 $|F_i^t|$ 为[2,6]个之间的随机数[9]. 每个节点的可用计算资源 R_v^t 设置为[50,200] unit 之间的随机数，VNF 实例化计算资源需求 R_f 设置为[10,25] unit 之

间的随机数[34]. 每条链路的可用带宽 $B^t(e)$ 与每个 MSR 的带宽需求分别设置为[1000,10000] Mbps 与 [50, 200] Mbps 之间的随机数[3]. 权重因子 ω_1、ω_2、η_1 与 η_2 的取值分别为 0.92、0.08、0.5 与 0.5. Informer 预测模型中，权重因子设置为 $\kappa_1=\kappa_2=0.5$，采用修正线性单元（Rectified Linear Unit，ReLU）作为 $\mathcal{F}(\cdot)$ 函数，编码器与解码器的层数分别设置为 2 层与 1 层，学习速率 ε 设置为 0.0001，小批量集规模（Batch Size）设置为 32，采用高斯误差线性单元（Gaussian Error Linear Units，GELU）作为激活函数，Adam 作为优化器. 在基于 SLC 的 MVNFP 算法中，联赛数量 N_φ 设置为 2 个，每个联赛中的球队数 N_{TM} 设置为 16 支，每支球队有 11 名固定球员与 11 名替补球员. 表 5 给出了系统部分与算法部分的参数配置. 采用 Python 3.7 来实施所有算法，且仿真结果由每种算法独立运行 10 次后取均值获得.

表 5　仿真参数配置

参数	值
系统部分参数取值	
VNF 种类	10 种
SFC 中 VNF 的数量（$\lvert F_i^t\rvert$）	[2,6]个
SFC 的最大端到端时延（$L_{\max}(F)$）	[50,100] ms
节点可用计算资源（R_v^t）	[50,200] unit
VNF 实例化计算资源需求（R_f）	[10,25] unit
链路可用带宽（$B^t(e)$）	[1000,10000] Mbps
MSR 带宽需求（B_i^t）	[50,200] Mbps
算法部分参数取值	
算法最大运行时间（N_{time}）	6 min
联赛数量（N_φ）	2 个
联赛球队数（N_{TM}）	16 支
固定球员数（N_{FP}）	11 名
替补球员数（N_{SB}）	11 名
变异概率（P_{mu}）	0.05

为了模拟组播 SFC 请求（即 MSR）的多样性，参照现有相关 VNF 放置问题研究中的常用做法[13,23,35]，本文引入了八个真实的网络拓扑作为测试场景，包括 SNDlib（http://sndlib.zib.de/）中的 Sun、Arnes 和 Ta2，以及互联网拓扑动物园（http://www. topology-zoo.org/）中的 Tinet、Dfn、Tata、UsCarrier 和 Kentucky，对应的具体参数如表 6 所示. 与文献[11，36]类似，每个场景中存在着多个组播组请求组播 SFC 服务，数量设置为 2 到 5 个. 这类场景是组播的常见应用，例如在 IPTV 中同时向多个用户提供直播电视服务[37]. 每个组播组的目的节点数量设置为 2~10 个[8]，场景中 SFC 的数量设置为 3~10 种[38]. 从源节点出发的数据流在到达目的节点前均需要被 SFC 中的每个 VNF 按照正确的顺序依次处理. 针对每个场景的特点，依据表 6 所示参数，本文为每个场景随机生成了 60000 条组播 SFC 请求（即 $\{MSR^t|t=1,\cdots,60000\}$）作为 MSR 的历史数据集，其中每个时隙的持续时间设置为 15 min. 每个场景中，MSR 历史数据集的前 80%被划分为训练集，剩余的 20%被划分为测试集.

表 6　测试场景

场景名称	节点数	链路数	组播组数量	目的节点数量	SFC 数量
Sun	27	51	2	[2,3]	3
Arnes	34	46	2	[2,4]	4
Tinet	53	89	3	[2,4]	4
Dfn	58	87	3	[2,6]	6
Ta2	65	108	4	[2,6]	6
Tata	145	186	4	[4,8]	8
UsCarrier	158	189	5	[4,10]	10
Kentucky	745	895	5	[4,10]	10

5.2　预测模型的有效性

为了验证预测模型的有效性，本文采用 Loss、均方误差（Mean Square Error，MSE）与平均绝对误差（Mean Absolute Error，MAE）三个指标进行评估，三者数值越小代表预测效果越好. MSE 与 MAE 的计算公式如式（30）与式（31）所示.

$$MSE=\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}(MSR^t-MSR^{(t)})^2 \tag{30}$$

$$MAE=\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}|MSR^t-MSR^{(t)}| \tag{31}$$

其中，$\mathcal{T}$ 为预测的时隙集合，$|\mathcal{T}|$ 为 $\mathcal{T}$ 中的时隙数量. 这里将基于 Informer 的预测模型与 LSTM[23]和双向长短期记忆人工神经网络（Bi-directional LSTM，BiLSTM）[22]进行对比. 图 5 展示了 Informer、LSTM 与 BiLSTM 在所有场景下的 Loss 曲线. 从图 5 中可以看出，本文所采用的基于 Informer 的预测模型能够快速收敛. 并且，相比于 LSTM 与 BiLSTM，Informer 在所有场景下均得到了最低的 Loss 值. 这是由于 LSTM 和 BiLSTM 在处理 MSR 预测这种长序列时间序列预测问题时仍难以避免出现梯度消失的情况. 并且，随着网络规模变大、预测序列长度增加，梯度求解变得更加困难，LSTM 和 BiLSTM 的预测性能也变得更差. 而 Informer 采

用了自注意蒸馏机制，减少了维度和网络参数量，突出了主要注意力，以充分发掘 MSR 序列特征. 同时，生成式解码器的运用也令其避免了推理阶段的误差积累，使得 Informer 具有较高的长序列预测精度. 此外，表 7 展示了三种预测模型在八个场景下的均方误差与平均绝对误差，其中最优的结果以粗体突出显示. 毫无疑问，基于 Informer 的预测模型拥有最优的预测性能，其在所有场景下均取得了最低的 MSE 与 MAE 结果. 图 6 展示了 SLC-MSRP 算法和未采用预测模型当实际组播 SFC 请求到达后直接调用足球联赛竞争（SLC）算法在八个场景下的平均总开销，其中 SLC 算法最大运行时间设置为 1 min. 可以看出，SLC-MSRP 算法在所有场景下均明显优于 SLC 算法. 综上所述，本文提出的预测模型在求解动态环境下面向组播的 VNF 放置问题是有效的.

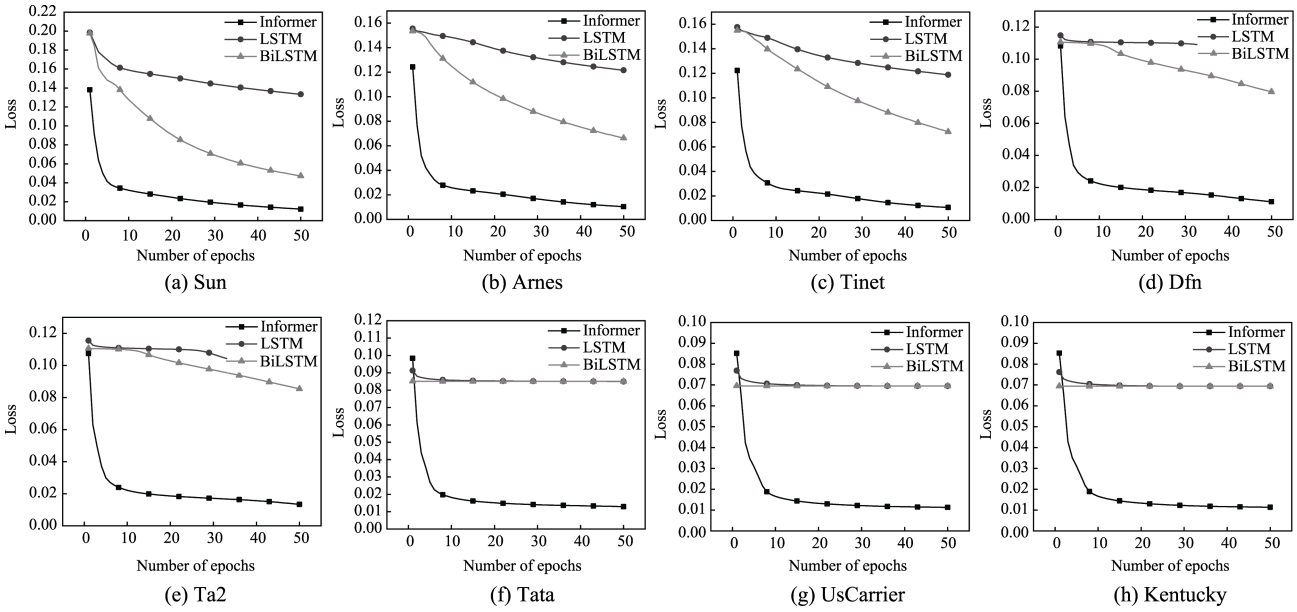

图 5 不同场景下三种预测模型的 Loss 曲线

表 7 不同场景下三种预测模型的预测结果

场景名称	Sun		Arnes		Tinet		Dfn	
指标	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Informer	**0.0475**	**0.0797**	**0.0361**	**0.0692**	**0.0401**	**0.0778**	**0.0278**	**0.0564**
LSTM	0.3820	0.5285	0.2041	0.3568	0.2094	0.3674	0.1133	0.2019
BiLSTM	0.0836	0.1879	0.0973	0.2355	0.1125	0.2590	0.1004	0.2116
场景名称	Ta2		Tata		UsCarrier		Kentucky	
指标	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Informer	**0.0281**	**0.0530**	**0.0211**	**0.0335**	**0.0169**	**0.0269**	**0.0171**	**0.0273**
LSTM	0.1116	0.2141	0.1040	0.1095	0.0869	0.0870	0.0868	0.0868
BiLSTM	0.1056	0.2121	0.1064	0.1064	0.0869	0.0870	0.0868	0.0868

5.3 整体性能评估

为了综合评估提出算法的整体性能，将 SLC-MSRP 与七种经典的算法进行比较，即五种启发式算法 ONMP、TSA、MMTC、SFMP、OTMP 和两种深度强化学习算法 DDPG-SFCO、DQL-SFCM，具体为：

- ONMP：用于 NFV 赋能的在线组播问题（Online NFV-enabled Multicasting Problem，ONMP）的启发式算法[3]，旨在最大化网络吞吐量的同时，最小化组播请求的实现开销.
- TSA：解决最优服务功能树嵌入问题的二阶段算法（Two-stage Algorithm，TSA）[6]，旨在最小化组播任务的传输开销.
- MMTC：针对多源组播路由问题而提出的多

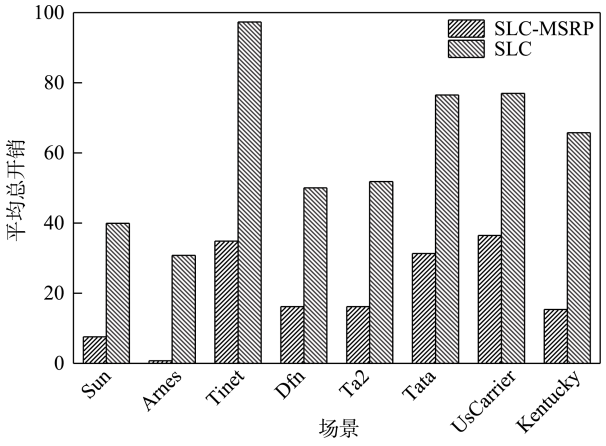

图 6　不同场景下的平均总开销

源组播树构建（Multi-source Multicast Tree Construction，MMTC）启发式算法[7].

- SFMP：应对服务功能组播问题（Service Function Multicast Problem，SFMP）的启发式算法[13]，为同时支持 VNF 和网络功能设备的混合基础设施网络提供解决方案.
- OTMP：在支持组播的移动边缘云网络中，针对在线吞吐量最大化问题（Online Throughput Maximization Problem，OTMP）的一种近似算法[14].
- DDPG-SFCO：针对 SFC 动态编排问题所提出的基于深度确定性策略梯度的 SFC 编排（Deep Deterministic Policy Gradient-based SFC Orchestration，DDPG-SFCO）算法[39].
- DQL-SFCM：为平衡工业物联网服务的质量以及计算和通信资源消耗，而提出的一种基于深度 Q 学习的 SFC 映射（Deep Q-Learning based SFC Mapping，DQL-SFCM）算法[40].

其中，原始的 DDPG-SFCO 与 DQL-SFCM 算法是针对 UVNFP 问题而提出的. 本文在进行 MVNFP 问题求解时先分别对组播 m_i 的每条路径 $P_{i,j}$ 求解出 VNF 放置方案，最后再将各条路径的 VNF 放置方案进行组合，得到组播 m_i 的 SFC 映射方案. 仿真中，两种深度强化学习算法的学习速率设置为 0.001，小批量集规模设置为 64，训练回合数设置为 5000[39,40].

这里，在每个场景中，预测 20 个连续的时隙，用于八种算法的性能评估. 图 7 展示的是八种算法在所有场景下的平均端到端时延. 可以看出，SLC-MSRP 在场景 Sun、Arnes、Tinet、Dfn、UsCarrier 与 Kentucky 中取得了最低的平均端到端时延，在场景 Ta2 与 Tata 中分别取得了第二与第三低的端到端时延. 这是由于 SLC-MSRP 通过 MDIC 策略准确地表示了网络中所有组播的 VNF 放置方案，使得明星球员与超级明星球员在进化的过程中可以有效代表最优解的方向，引导联赛中的所有球员朝着最优的 MVNFP 方案进化. 同时，在快速修复策略中，采用反向搜索过程对正向搜索过程得到的解进一步优化，为每个生成树寻找更低的端到端时延方案.

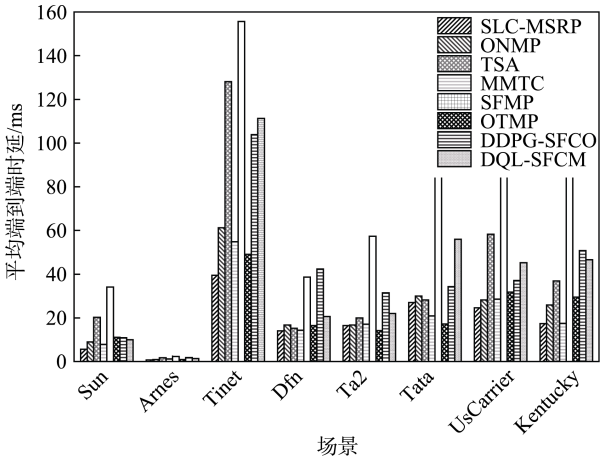

图 7　八种算法的平均端到端时延

图 8 展示的是八种算法在所有场景下的平均计算资源消耗. 从图 8 中可以看出，除场景 Tata 与 UsCarrier 外，SLC-MSRP 在其余所有场景下均获得了最低的平均计算资源消耗. 主要原因是，应用 MDIC 策略后，SLC-MSRP 不仅可以在进化过程中探索同一组播组内多条路径的重叠节点与 VNF 放置位置的关系，还能够挖掘该时隙各组播组间多条路径共同经过的节点与不同 MSR 中同种 VNF 放置位置的关系，充分利用这些重叠位置的节点，合理放置 VNF，达到节约节点计算资源的目的. 同时，在快速修复策略中，通过将节点选取范围控制在已放置对应 VNF 的节点内，充分利用已经放置的 VNF，以避免放置冗余的 VNF 造成的节点计算资源浪费.

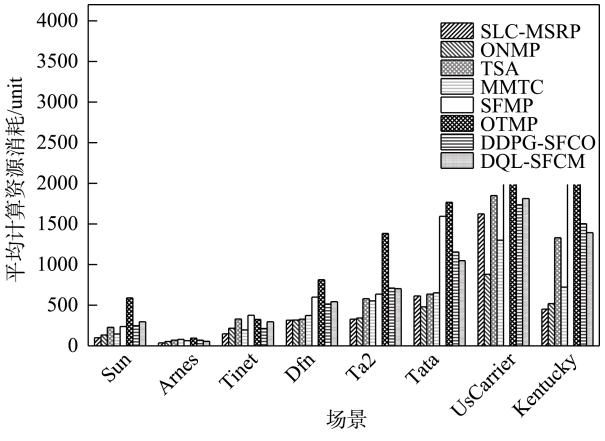

图 8　八种算法的平均计算资源消耗

表 8 展示的是八种算法在所有场景下的平均 MSR 响应时间，其中最优的结果以粗体显示. 更小的 MSR 响应时间意味着更快的用户请求响应速度. DDPG-SFCO 与 DQL-SFCM 等深度强化学习算法结合了深度学习的特征提取能力和强化学习的决策能力,可根据输入的 MSR 直接做出 VNF 放置决策. 因此，DDPG-SFCO 与 DQL-SFCM 在处理实时决策问题时，通常具有比启发式算法和进化算法更低的响应时间. 然而，SLC-MSRP 仍然在 Sun、UsCarrier 与 Kentucky 三个场景下，取得了最低的平均 MSR 响应时间. 并且，在剩余五个场景下，SLC-MSRP 略逊于 DDPG-SFCO，取得了第二低的平均 MSR 响应时间. 这是由于 SLC-MSRP 采用了基于 Informer 的预测模型，对即将到达的 MSR 提前预测，以节约求解 VNF 放置方案的时间. 同时，在真实 MSR 到达后，通过应用快速修复策略，SLC-MSRP 可以在短时间内求解对应的 MVNFP 方案，完成对用户请求的快速响应.

表 9 展示了八种算法在所有场景下的平均总开销. 通过 VNF 预放置和修复两个阶段，以及基于 Informer 的预测模型、基于足球联赛竞争的组播虚拟网络功能放置算法与快速修复策略三个部分的分工协作，提出的“预测——预放置——修复”解决方案，使得 SLC-MSRP 能够在真实 MSR 到达前取得更低端到端时延和计算资源消耗. 并且，在真实 MSR 到达后，仅需对预测与真实结果不一致的情况做出部分调整,以获得更快的 MSR 响应时间. 因此，从表 9 中可以看出，除场景 Tata 与 UsCarrier 外，SLC-MSRP 在其余大多数场景下取得了最低的平均总开销值. 此外，本文还采用了弗里德曼检验[41]对八种算法进行性能比较. 根据平均端到端时延、平均计算资源消耗、平均 MSR 响应时间和平均总开销，八种算法在所有测试场景下的平均排名如表 10 所示. 显然，同其它七种启发式算法与深度强化学习算法相比，本文所提出的 SLC-MSRP 算法具有最好的性能.

表 8 八种算法的平均 MSR 响应时间（s）

场景名称	Sun	Arnes	Tinet	Dfn	Ta2	Tata	UsCarrier	Kentucky
SLC-MSRP	**0.0005**	0.0478	0.1280	0.2395	0.3243	0.4546	**0.4358**	**0.5357**
ONMP	40.3204	24.1071	81.5488	88.4307	112.7606	162.1446	223.9217	552.0189
TSA	0.2258	0.1574	1.8412	2.8338	4.9359	75.8339	144.6292	20480.8153
MMTC	0.1643	0.0966	0.2343	0.2922	0.3743	0.7252	1.1479	9.5236
SFMP	0.9177	0.6433	1.5728	2.2059	3.7664	22.8606	53.0896	2091.0252
OTMP	0.1690	0.1177	1.4479	2.1352	3.6974	51.0221	95.6321	12022.2121
DDPG-SFCO	0.0353	**0.0132**	**0.0865**	**0.1188**	**0.1126**	**0.2363**	0.4728	0.8042
DQL-SFCM	0.3336	0.1623	0.4043	0.7482	0.9402	1.8418	4.3505	3.2370

表 9 八种算法的平均总开销

场景名称	Sun	Arnes	Tinet	Dfn	Ta2	Tata	UsCarrier	Kentucky
SLC-MSRP	**7.5629**	**0.7701**	**34.8758**	**16.2081**	**16.1799**	31.3392	36.4987	**15.4046**
ONMP	30.8238	13.0303	92.9582	61.6253	72.9193	109.5522	138.8879	298.8584
TSA	20.6830	1.6180	93.2032	18.4270	26.4673	70.2549	128.3508	10284.7224
MMTC	10.8575	1.4462	47.2928	18.0956	22.1604	**29.7666**	**33.9420**	27.6424
SFMP	27.6621	2.0987	85.2506	37.2606	45.0418	109.6795	125.9054	1144.8423
OTMP	34.6185	1.4633	63.7770	33.1920	43.1747	86.6055	125.1165	6093.8759
DDPG-SFCO	17.3272	1.6188	67.4572	35.4044	32.7704	34.1176	44.2855	54.7938
DQL-SFCM	19.4743	1.3224	81.8049	26.4188	28.5093	58.6644	51.3173	51.8743

6 结论与将来工作

针对动态网络环境下面向组播的虚拟网络功能放置问题，本文将该问题拆分为 VNF 预放置和修复两个子问题，提出了一种基于组播 SFC 请求预测的足球联赛竞争（SLC-MSRP）算法，利用历史组播 SFC 请求（MSR）记录，预测即将到来的 MSR，提前进行映射，并快速修复未能成功预测的 MSR，在优化端到端时延和计算资源消耗的同时，显著降低系统对 MSR 的响应时间. SLC-MSRP 算法以 Informer

表 10　八种算法的弗里德曼检验

算法名称	平均端到端时延		平均计算资源消耗		平均 MSR 响应时间		平均总开销	
	平均序值	排名	平均序值	排名	平均序值	排名	平均序值	排名
SLC-MSRP	1.3750	1	1.3750	1	1.6250	2	1.2500	1
ONMP	3.3750	4	2.0000	2	7.6250	8	7.2500	8
TSA	5.5000	5	4.5000	4	6.6250	7	5.5000	5
MMTC	2.7500	2	3.5000	3	3.1250	3	1.8750	2
SFMP	7.8750	8	6.2500	7	6.0000	6	6.6250	7
OTMP	3.1250	3	7.7500	8	5.2500	5	5.5000	5
DDPG-SFCO	6.2500	7	5.3750	6	1.2500	1	4.2500	4
DQL-SFCM	5.7500	6	5.2500	5	4.2500	4	3.7500	3

预测模型为基础，通过对历史 MSR 请求的训练，预测即将到达的 MSR. 提出了基于 SLC 的 MVNFP 算法，采用多维个体编码（MDIC）策略，一次性求解所有活动组播组的 SFC 映射方案，对预测的 MSR 进行提前部署，在实际 MSR 到达前预先放置 VNF 和路由. 针对预测 MSR 与实际 MSR 不一致的情况，本文设计了一种快速修复策略，以迅速响应实际到达的 MSR. 仿真结果表明，基于 Informer 的预测模型具有优异的 MSR 预测性能. 同七种经典的启发式算法与深度强化学习算法相比，SLC-MSRP 算法能够获得最优的端到端时延和计算资源消耗. 同时，SLC-MSRP 采用的 VNF 预放置与修复相结合的二阶段方式可显著减少系统在服务到达 MSR 时的响应时间.

移动边缘计算（MEC）通过在靠近移动用户的位置上提供服务环境和云计算能力，被认为是 5G 以及未来 6G 网络的核心技术之一. 欧洲电信标准化协会（European Telecommunications Standards Institute，ETSI）于 2018 年提出了 MEC 参考体系结构[42]，其中 MEC 被部署为 NFV 环境的一部分. 因此，在 NFV 赋能的 MEC 网络中开展面向组播的 VNF 放置问题研究是我们的未来工作计划之一. 此外，本文提出的 SLC-MSRP 算法通过采用深度学习技术预测未来的 MSR，基于预测结果提前生成 MVNFP 解决方案. 然而，基于现有技术，预测结果与实际结果产生不一致的情况是难以避免的. 虽然 SLC-MSRP 采用快速修复策略以迅速应对预测产生的差错，但仍可能产生一定的处理时延，造成用户组播 SFC 请求响应不及时的情况. 深度强化学习技术将强化学习与深度神经网络相结合，可以处理动态场景中的实时决策问题. 因此，在未来的工作中，我们计划将基于深度强化学习的算法应用于动态网络环境下面向组播的虚拟网络功能放置问题，对动态到达的 MSR 做出实时决策.

参　考　文　献

[1] Manias D M, Shami A. The need for advanced intelligence in NFV management and orchestration. IEEE Network, 2021, 35(1): 365-371

[2] Schardong F, Nunes I, Schaeffer-Filho A. NFV resource allocation: a systematic review and taxonomy of VNF forwarding graph embedding. Computer Networks, 2021, 185: 107726

[3] Xu Z, Liang W, Huang M, et al. Efficient NFV-Enabled Multicasting in SDNs. IEEE Transactions on Communications, 2019, 67(3): 2052-2070

[4] Wang X, Xing H, Zhan D, et al. A Two-stage Approach for multicast-oriented virtual network function placement. Applied Soft Computing, 2021, 112: 107798

[5] Cohen R, Lewin-Eytan L, Naor J S, et al. Near optimal placement of virtual network functions// IEEE Conference on Computer Communications (INFOCOM). Hong Kong, China, 2015: 1346-1354

[6] Ren B, Guo D, Shen Y, et al. Embedding service function tree with minimum cost for NFV-enabled multicast. IEEE Journal on Selected Areas in Communications, 2019, 37(5): 1085-1097

[7] Xie K, Zhou X, Semong T, et al. Multi-Source multicast routing with QoS constraints in network function virtualization//IEEE International Conference on Communications (ICC). Shanghai, China, 2019: 1-6

[8] Mirjalily G, Asgarian M, Luo Z-Q. Interference-Aware NFV- enabled multicast service in resource-constrained wireless mesh networks. IEEE Transactions on Network and Service Management, 2022, 19(1): 424-436

[9] Muhammad A, Qu L, Assi C. Delay-Aware multi-Source multicast resource optimization in NFV-enabled network//IEEE International Conference on Communications (ICC). Dublin, Ireland, 2020: 1-7

[10] Wang K, Zhao N, Li J, et al. Service function chain embedding algorithm with wireless multicast in mobile edge computing network. Journal on Communications, 2020, 41(10): 37-47 (in Chinese)
(王侃, 赵楠, 李军怀等. 移动边缘计算网络中联合无线多播的服务功能链部署算法. 通信学报, 2020, 41(10): 37-47)

[11] Asgarian M, Mirjalily G, Luo Z-Q. Trade-Off between efficiency and complexity in multi-stage embedding of multicast VNF ser-

vice chains. IEEE Communications Letters, 2022, 26(2): 429-433
[12] Xu Z, Zhang Y, Liang W, et al. NFV-enabled multicasting in Mobile edge clouds with resource sharing//Proceedings of the 48th International Conference on Parallel Processing (ICPP). Kyoto, Japan, 2019: 1-10
[13] Yi B, Wang X, Huang M, et al. SDN and NFV enabled service function multicast mechanisms over hybrid infrastructure. Peer-To-Peer Networking and Applications, 2019, 12(2): 405-421
[14] Ma Y, Liang W, Wu J, et al. Throughput maximization of NFV-enabled multicasting in mobile edge cloud networks. IEEE Transactions on Parallel and Distributed Systems, 2020, 31(2): 393-407
[15] Cai Y, Llorca J, Tulino A M, et al. Optimal multicast service chain control: packet processing, routing, and duplication//IEEE International Conference on Communications (ICC). Virtual Conference, 2021: 1-7
[16] Xing H, Li S, Dai P, et al. A PBIL for delay constrained virtual network function placement with load balancing//Proceedings of the ACM Turing Celebration Conference-China (ACM TURC'19). Chengdu, China, 2019, 7: 1-6
[17] Moosavian N, Roodsari B K. Soccer league competition algorithm, a new method for solving systems of nonlinear equations. International Journal of Intelligence Science, 2014, 04(01): 7-16
[18] Dogan A. Optimum sitting and sizing of WTs, PVs, ESSs and EVCSs using hybrid soccer league competition-pattern search algorithm. Engineering Science and Technology, an International Journal, 2021, 24(3): 795-805
[19] Ebrahimi S, Tabatabaei S. Using Clustering via soccer league competition algorithm for optimizing power consumption in WSNs (Wireless Sensor Networks). Wireless Personal Communications, 2020, 113(4): 2387-2402
[20] Qiao Y, Dao T-K, Pan J-S, et al. Diversity teams in Soccer league competition algorithm for wireless sensor network deployment problem. Symmetry, 2020, 12(3): 445
[21] Krishnan S S, Sitaraman R K. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. IEEE/ACM Transactions on Networking, 2013, 21(6): 2001-2014
[22] Kim H-G, Jeong S-Y, Lee D-Y, et al. A deep learning approach to VNF resource prediction using correlation between VNFs//2019 IEEE Conference on Network Softwarization (NetSoft), 2019: 444-449
[23] Cai J, Qian K, Luo J, et al. SARM: Service function chain active reconfiguration mechanism based on load and demand prediction. International Journal of Intelligent Systems, Wiley Online Library, 2022, 37(9): 6388-6414
[24] Eramo V, Lavacca F G, Catena T, et al. Application of a long short term memory neural predictor with asymmetric loss function for the resource allocation in NFV network architectures. Computer Networks, 2021, 193: 108104
[25] Yuan Q, You W, Ji X, et al. Adaptive Scaling of Virtualized network function resource capacity. Journal of Electronics & Information Technology, 2021, 43(7): 1841-1848. (in Chinese)
(袁泉, 游伟, 季新生等. 虚拟网络功能资源容量自适应调整方法. 电子与信息学报, 2021, 43(7): 1841-1848)
[26] Huang X, Bian S, Gao X, et al. Online VNF chaining and predictive scheduling: optimality and Trade-Offs. IEEE/ACM Transactions on Networking, 2021, 29(4): 1867-1880
[27] Zhao J, Huang F, Lv J, et al. Do RNN and LSTM have long memory?// Proceedings of the 37th International Conference on Machine Learning. 2020, 119: 11365-11375
[28] Zhou H, Zhang S, Peng J, et al. Informer: beyond efficient transformer for long sequence Time-Series forecasting// Proceedings of the AAAI Conference on Artificial Intelligence. Virtual Conference: AAAI Press, 2021, 35(12): 11106-11115
[29] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need//Advances in Neural Information Processing Systems. Long Beach, USA: Curran Associates, Inc., 2017, 30: 1-11
[30] Gong M, Zhao Y, Sun J, et al. Load forecasting of district heating system based on Informer. Energy, 2022, 253: 124179
[31] Bunyakitanon M, da Silva A P, Vasilakos X, et al. Auto-3P: An autonomous VNF performance prediction & placement framework based on machine learning. Computer Networks, 2020, 181: 107433
[32] Fu X, Yu F R, Wang J, et al. Dynamic service function chain embedding for NFV-Enabled IoT: a deep reinforcement learning approach. IEEE Transactions on Wireless Communications, 2020, 19(1): 507-519
[33] Wang X, Xing H, Yang H. On Multicast-Oriented virtual network function placement: a modified genetic algorithm//International Conference on Signal and Information Processing, Networking and Computers (ICSINC). Yuzhou, China, 2019: 420-428
[34] Alhussein O, Do P T, Li J, et al. Joint VNF Placement and Multicast traffic routing in 5G core networks//2018 IEEE Global Communications Conference (GLOBECOM). Abu Dhabi, United Arab Emirates, 2018: 1-6
[35] Asgarian M, Mirjalily G, Luo Z-Q. Embedding multicast service function chains in NFV-Enabled networks. IEEE Communications Letters, 2021, 25(4): 1264-1268
[36] Soni H, Dabbous W, Turletti T, et al. NFV-Based Scalable Guaranteed-Bandwidth multicast service for software defined ISP networks. IEEE Transactions on Network and Service Management, 2017, 14(4): 1157-1170
[37] Joo H, Kwak K, Kim I, et al. Fast/On-time channel zapping scheme using a cache server over IPTV multicast system//2021 IEEE Industrial Electronics and Applications Conference (IEACon). 2021: 276-279
[38] Muhammad A, Sorkhoh I, Qu L, et al. Delay-Sensitive Multi-Source multicast resource optimization in NFV-Enabled Networks: a Column generation approach. IEEE Transactions on Network and Service Management, 2021, 18(1): 286-300
[39] Liu Y, Lu H, Li X, et al. Dynamic service function chain orchestration for NFV/MEC-Enabled IoT networks: a deep reinforcement learning approach. IEEE Internet of Things Journal, 2021, 8(9): 7450-7465
[40] Xu S, Li Y, Guo S, et al. Cloud-Edge collaborative SFC mapping for industrial IoT using deep reinforcement learning. IEEE Transactions on Industrial Informatics, 2022, 18(6): 4158-4168
[41] Sheldon M Ross. Introduction to probability and statistics for engineers and scientists. 5th Edition. London, UK: Academic Press, 2020
[42] ETSI. Mobile Edge Computing (MEC); Deployment of Mobile Edge Computing in an NFV environment. ETSI GR MEC 017 V1.1.1, 2018

XING Huan-Lai, Ph.D., associate professor, Ph.D. supervisor. His research interests include network function virtualization, software defined networks, artificial intelligence, and evolutionary computation.

WANG Xin-Han, Ph.D. candidate. His research interests include network function virtualization, evolutionary computation, and deep learning.

SONG Fu-Hong, Ph.D. candidate. His research interests include edge computing and multi-objective optimization.

ZHAO Bo-Wen, Ph.D. candidate . His research interests include deep learning and time series classification.

LUO Shou-Xi, Ph.D., master supervisor. His research interests include data center networks and networked systems.

DAI Peng-Lin, Ph.D., master supervisor. His research interests include intelligent transportation systems and vehicular cyber-physical systems.

LI Ke, Ph.D., master supervisor. Her research interests focus on internet of vehicles.

Background

Multicast-oriented dynamic virtual network function placement problem under dynamic network environment is a promising research trend in network function virtualization (NFV). Current research on multicast-oriented dynamic virtual network function placement is insufficient, and several problems need to be addressed urgently. In terms of problem-solving, approximation and heuristic algorithms have been the mainstream focus. Most studies model multicast-oriented dynamic virtual network function placement as integer linear programming (ILP) or mixed ILP and address it by using approximation and heuristic algorithms. However, these algorithms usually lead to unstable performance with respect to the solution quality. In terms of problem-modeling, for dynamic scenarios, the existing studies focus on proposing online placement algorithms that handle the multicast SFC requests (MSRs) after their actual arrival. As a result, these online algorithms cannot respond to the requests quickly enough. When studying unicast-oriented dynamic VNFP problems, some researchers have adopted unicast SFC request prediction based on deep learning. Nevertheless, the existing work does not consider multicast, which is not conducive to the fast response of MSR in dynamic scenarios.

This paper divides the multicast-oriented dynamic virtual network function placement problem into two sub-problems, i.e., VNF pre-placement and repair. For the VNF pre-placement sub-problem, we predict the incoming MSRs based on historical data and map the corresponding SFCs in advance. In the repair sub-problem, when MSRs arrive, the actual arrival MSRs that is not successfully predicted can be quickly handled. Generally, a full-process solution (prediction→pre-placement→repair) to the multicast-oriented dynamic virtual network function placement problem is presented in this paper. To be specific, we propose a soccer league competition algorithm with multicast SFC request prediction (SLC-MSRP) to tackle the multicast-oriented dynamic virtual network function placement problem. In the VNF pre- placement stage, the Informer model is applied to predict the incoming MSRs, and an SLC algorithm with multi- dimensional individual coding (MDIC) strategy is devised to address the multicast-oriented virtual network function placement (MVNFP) problem for the first time. In the repair stage, a fast repair strategy consisting of forward and backward search phases is developed to deal with the actual arrival MSRs that is not successfully predicted. The fast repair strategy adopts a forward search phase to make full use of the over-provisioning VNFs so as to save the computational resources of nodes and reduce the initialization delay of VNFs. In the backward search phase, the spanning trees (SPTs) established in the forward search phase are further optimized. Simulation results show that, SLC-MSRP gains the best performance in terms of end-to-end delay and computational resource consumption, and it achieves lower MSR response time compared with seven state-of-the-art algorithms.

This work was supported in part by the National Natural Science Foundation of China (No. 62172342, No. 62202392), the Natural Science Foundation of Sichuan Province (No. 2022NSFSC0568, No. 2022NSFSC0944, No. 2023NSFSC0459), and the Fundamental Research Funds for the Central Universities, P. R. China.