研究与开发

基于微服务架构的粒度可变服务功能链映射算法

吴晓春，洪晨，张岳，张俊楠，周静静
（浙江工商大学信息与电子工程学院，浙江 杭州 310018）

摘　要：针对 5G 环境下服务功能链（SFC）端到端时延无法满足时延敏感型应用需求的问题，将传统虚拟网络功能（VNF）拆分成粒度不一的映射单元，提出了基于微服务架构的粒度可变服务功能链映射（VG-SFCM）算法。首先将传统粗粒度的 VNF 解耦成细粒度的微服务单元，随后通过 SFC 内冗余微服务单元的合并及 SFC 间微服务单元的复用，减少微服务单元的实例化，降低 SFC 的处理时间。仿真结果表明，所提算法在降低平均部署网络成本的同时，其 SFC 端到端时延相较于传统的映射算法降低了 14.81%。

关键词：网络功能虚拟化；微服务架构；服务功能链；时延
中图分类号：TN915，TP393
文献标志码：A
doi: 10.11959/j.issn.1000–0801.2022289

Variable granularity service function chain mapping algorithm based on microservice architecture

WU Xiaochun, HONG Chen, ZHANG Yue, ZHANG Junnan, ZHOU Jingjing
School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China

Abstract: For the problem that the end-to-end delay of the service function chain (SFC) cannot meet the demand of delay-sensitive applications in 5G environment, a variable granularity service function chain mapping (VG-SFCM) algorithm based on microservice architecture was proposed by splitting the traditional virtualized network function (VNF) into mapping units of varying granularity. Firstly, the traditional coarse-grained VNF was decoupled into fine-grained microservice units, and then the instantiation of microservice units was reduced through the consolidation of redundant microservice units within SFC and the reuse of microservice units between SFC, thus reducing the processing time of SFC. The simulation results show that the algorithm reduces the end-to-end delay of SFC by 14.81% compared to the traditional mapping algorithm while reducing the average deployment network cost.

Key words: network function virtualization, microservice architecture, service function chain, delay

收稿日期：2022-06-09；**修回日期**：2022-11-14
通信作者：洪晨，2281026509@qq.com
基金项目：浙江省自然科学基金资助项目（No. LY19F020002，No. Y19F020031），浙江省新型网络标准与应用技术重点实验室项目（No. 2013E10012）
Foundation Items: The Zhejiang Province Natural Science Foundation (No. LY19F020002, No. Y19F020031), Zhejiang Provincial Key Laboratory of New Network Standards and Application Technology (No. 2013E10012)

0 引言

随着互联网的不断普及和物联网相关技术的飞速发展，人们对网络的需求呈指数级增长，各式各样的服务请求日新月异。为了应对不断增长且纷繁复杂的服务请求，网络功能虚拟化（network function virtualization，NFV）技术应运而生。NFV 通过虚拟化的技术将网络功能以软件化的形式呈现，形成虚拟网络功能（virtualized network function，VNF），实现了网络功能与专用物理设备的解耦，使资源可以灵活共享，并大大降低了运营商的运营成本（operating expense，OPEX）及资本支出（capital expenditure，CAPEX）[1]。不同的 VNF 按照一定逻辑顺序进行链接并形成服务功能链（service function chain，SFC）[2]。通过 SFC，运营商可以更加快速且灵活地为用户提供所需的服务，当用户的服务需求或网络环境发生变化时也可快速进行 VNF 的迁移、创建、销毁等操作。

5G 等技术的发展使许多应用对时延的要求提高，因此，如何合理地部署 VNF 或对 SFC 本身进行一定优化，使得 SFC 端到端时延得到一定程度的降低是映射问题的一大挑战。目前，已有不少学者对 SFC 的映射及部署问题展开了研究。当前该问题通常被建模成整数线性规划（integer linear programming，ILP）模型或混合整数线性规划（mixed integer linear programming，MILP）模型等数学规划模型。文献[3]使用基于混合整数线性规划的可变邻域搜索方法对时延受限情况下的 SFC 映射问题进行求解。文献[4]在最小化时延的约束下，利用混合整数二次约束规划建模对 SFC 映射问题进行求解。但此类数学规划方法仅在面对小规模网络时拥有较快的求解速度，在大规模网络中求解速度较慢并且会带来较高的运行成本，因此各类启发式算法被相继提出。文献[5]采用可拓展的启发式算法，每次针对单个网络节点进行映射，若映射不成功则不断回溯当前的结果，直至完成映射。文献[6]提出了一种基于改进多阶段图的启发式算法，从而实现服务功能链的高效部署。然而，启发式算法虽然拥有较快的求解速度，但无法保证所得的解总是全局最优解。因此，元启发式算法在启发式算法的基础上进行改进，结合了随机算法以及局部搜索的策略以避免陷入局部最优解。文献[7]融合遗传算法与模拟退火算法进行求解，同时通过判断个体约束及纠正遗传的方法来规避局部最优解，从而达到降低服务功能链端到端时延的目的，但该算法需要反复迭代才能求出最优解，时间消耗较大，无法对 VNF 进行快速部署。文献[8]首先通过广度优先搜索完成服务功能链的构建，随后利用改进的遗传粒子群算法对映射问题进行求解，提高了服务功能链请求接受率，同时降低了资源开销，但在对映射问题求解时没有考虑映射顺序，产生了一定的时延。文献[9]针对服务功能链端到端时延优化以及服务可用性最大化问题，建立模型后通过遗传算法进行求解，但面对大规模物理拓扑时容易陷入局部最优解。而随着人工智能领域研究的不断深入，强化学习算法凭借其在求解组合优化问题方面的先天优势在多个领域得到应用。文献[10]提出了基于深度强化学习的服务功能链可靠部署算法，在保证可靠性的同时有效减少了服务功能链的损失。文献[11]使用强化学习算法寻找匹配的 VNF 调度策略，并提出基于 Q 学习的 VNF 部署算法，但在大规模网络环境中强化学习算法无法准确地描述网络资源的动态变化，会对 VNF 的部署产生不良影响。

上述研究大多将作为 SFC 构成单位的 VNF 视作一个整体，统一进行设计、开发和部署，尚未考虑进一步细化的优化方案。VNF 虽然实现了与专用硬件设备的解耦，但这种单体式的映射方法内部各逻辑功能间的耦合程度仍较高，且灵活性较差，难以被扩展或伸缩。因此提出将微服务架构运用到 SFC 映射及部署的问题中，对 VNF 进行微服务化的拆分及合并，以实现端到端时延的最优化。

微服务体系架构[12]的核心在于对传统应用进

行功能上的分解，将细粒度的微服务单元作为执行相关逻辑操作的最小单元，每个微服务单元均能被独立部署、运行。文献[13]提出的 OpenBox 框架首次探索了 SFC 中多个 VNF 的拆分优化，并完成了拆分后微服务单元的定义，但并未考虑 SFC 中出现不可拆分 VNF 的状况。文献[14]首次探讨了服务功能链的优化问题与微服务架构结合的可能性，随后通过实例将传统的粗粒度服务功能链进行微服务化的拆分及合并，绘制了新的有向服务图，但在拆分过程中并未考虑中间状态，容易产生过多的资源开销。

与上述文献不同，本文针对服务功能链中虚拟网络功能的逻辑功能重复执行问题，考虑时延优化，提出一种基于微服务架构的粒度可变服务功能链映射（variable granularity service function chain mapping，VG-SFCM）算法，对 VNF 进行微服务化的拆分时，进一步考虑映射时网络拓扑中的资源使用情况和具体的服务功能链请求，以确定粒度不一的映射单元，从而达到降低 SFC 端到端时延及提高资源利用率的目的。该算法通过基于典型 VNF 的快速拆分策略将相关 VNF 拆分为最细粒度的微服务单元，随后对单一 SFC 内的冗余微服务单元进行合并，以实现端到端时延的最优化。针对物理拓扑中已完成部分 SFC 部署的情况，通过 SFC 之间的微服务单元复用策略，根据可复用微服务单元所处的网络节点及其邻近节点的资源状况，判断复用及新到达 SFC 中剩余微服务单元的部署可能性，减少微服务单元的实例化。仿真结果表明，VG- SFCM 能够有效降低 SFC 的平均端到端时延及平均部署网络成本，提高网络资源利用率。

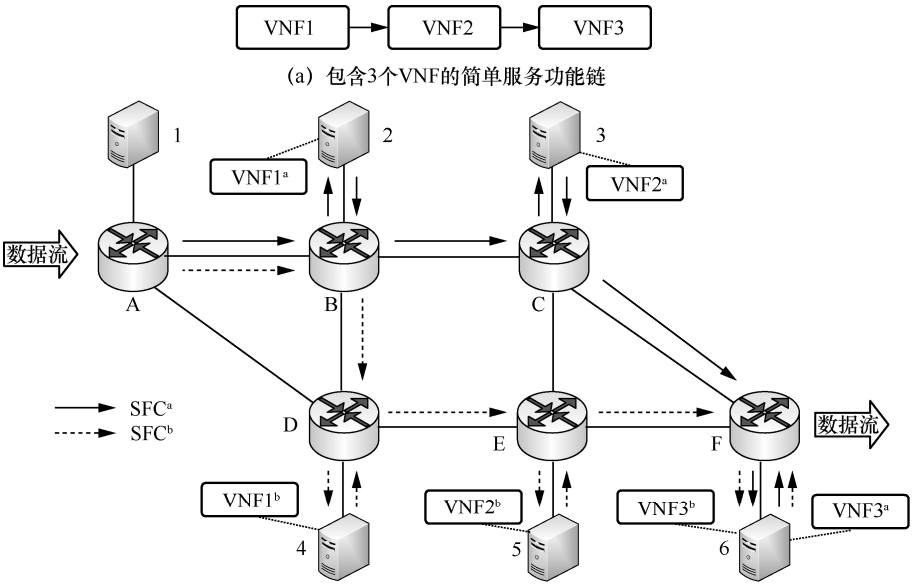
1　问题描述与系统模型

1.1　问题描述

在传统网络中，每个中间件都有相对应的专用服务器，网络功能服务设备多并且复杂，不利于按需扩展和调整。虚拟化技术的出现，使得通用服务器取代传统的专用物理设备成为可能，从而有效降低网络功能的平均部署网络成本。然而，单条 SFC 内的不同单体 VNF 间存在许多重复执行的逻辑功能，如数据包的输入输出、报文分类等。在 SFC 映射的过程中，考虑映射的最小粒度并进行微服务化的拆分及合并，可有效地去除这些冗余的逻辑功能，降低根据映射方案部署时产生的端到端时延及资源开销。

不同映射方案对比如图 1 所示，两种方案均将单体 VNF 作为 SFC 的最小映射单元。单 SFC 的拆分、合并及映射如图 2 所示，该方案首先对服务功能链中的各 VNF 进行逻辑功能分析，并根据分析结果执行相应的拆分工作，将映射的最小单元转化为粗粒度的微服务单元。完成 SFC 中所有 VNF 的拆分后，按照不改变初始逻辑顺序的原则进行相同微服务单元的合并并构建新服务图，合并后剩下的 AtomNF 个数相较于拆分初期减少了 44.4%。合并完成后按照原始的逻辑顺序进行新服务功能链的构建。此类合并操作的对象为单 SFC 内部的微服务单元，微服务单元的实例是按照映射方案创建并部署的，当新服务功能链请求（service function chain request，SFCR）到来时，相同的实例又会再次创建。因此通过合并及复用可有效地减少相关资源的浪费，并降低端到端时延。

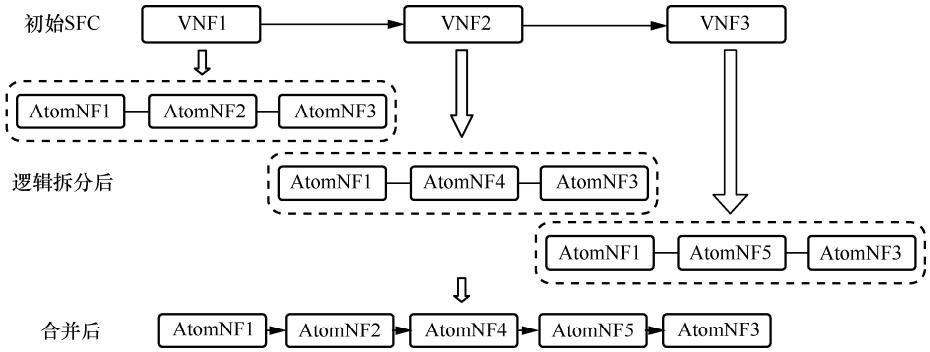
针对多个 SFCR 到来的情况，后续到达的 SFCR 将依据上一 SFCR 的映射方案进行可复用微服务单元的筛选及节点资源分析。SFC 间的微服务单元复用及映射如图 3 所示。多 SFC 映射方案中共有两个 SFC 需完成映射及后续的部署，SFC1 的映射方案与如图 2（b）一致。SFC2 包含 AtomNF1、AtomNF3 这两个与 SFC1 相同的可复用微服务单元，通过判断节点 B 中的计算资源是否能够支持 AtomNF6 和 AtomNF7 的映射来确定映射方案。该映射方案复用了 AtomNF1 和



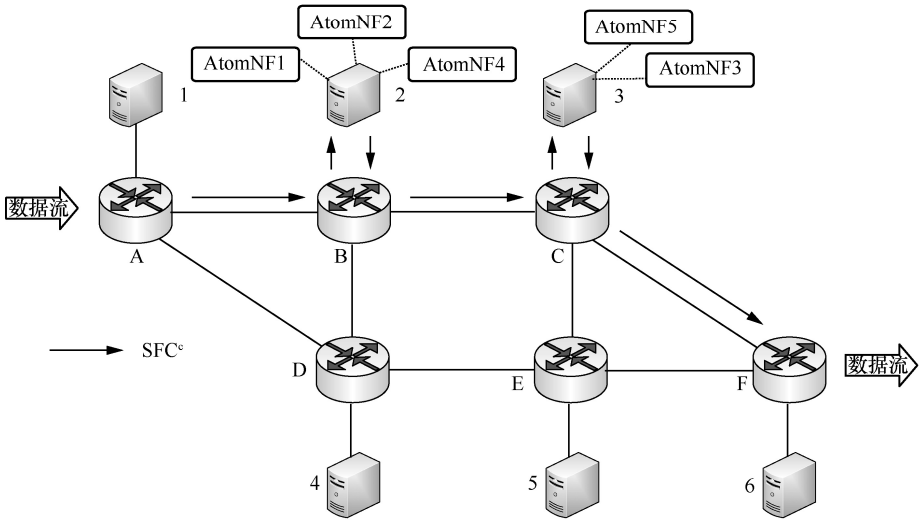
(a) 包含3个VNF的简单服务功能链

(b) 简单服务功能链的两种映射及部署方案

图 1　不同映射方案对比

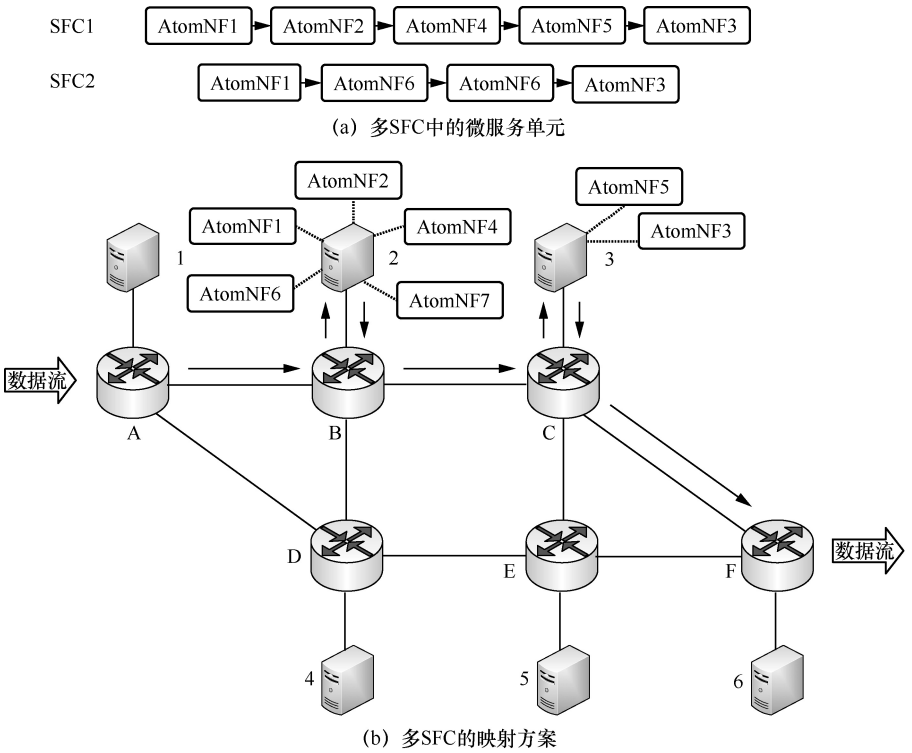


(a) 单SFC中VNF的拆分、合并



(b) 单SFC的映射

图 2　单 SFC 的拆分、合并及映射



(a) 多SFC中的微服务单元

(b) 多SFC的映射方案

图 3　SFC 间的微服务单元复用及映射

AtomNF3 两个微服务单元，并将剩余单元映射至已激活节点中，相较于传统映射方案减少了节点的激活数量，并通过减少微服务单元实例数降低了 SFC2 的端到端时延及资源开销。

1.2　网络模型

为了更好地形式化描述问题，列出了与底层网络、服务功能链请求相关的符号及其含义，相关符号及其含义见表 1。

1.3　约束分析及优化目标

考虑底层网络中节点资源约束和链路带宽资源约束，为了最小化 SFC 的端到端时延，SFC 与 VNF 的映射及部署问题可以建模为一个整数规划模型。相较于传统粗粒度的单体 VNF 映射方案，基于微服务架构的映射方案的主要创新在于映射粒度变为细粒度的微服务单元，因此在资源约束方面与传统映射方案的区别之处在于其中的映射单元，未拆分的 VNF 在建模过程中除开销方面的差异外，表现形式与 AtomNF 保持一致。

表 1　相关符号及其含义

符号	含义
G	底层网络拓扑
N	拓扑中的网络节点集合，$i \in N$ 表示设计具体节点
E	网络节点间链路集合，$(i,j) \in E$ 表示具体链路
E_{i}	与物理链路相对应的虚拟链路
R^i_{cpu}、R^i_{mem}	节点 i 的 CPU 资源及内存资源
c_i	节点 i 的激活代价
B_{ij}	链路带宽资源
F	网络中支持的 VNF 集合，$p \in F$ 表示具体类别
S	服务功能链请求集合，$f \in S$ 表示单个服务功能链请求
V	构成服务功能链的 VNF 集合，$v \in V$ 表示单个 VNF
R_{C}、R_{B}	服务功能链请求的计算资源需求及带宽需求
l	执行最细粒度拆分后得到的 AtomNF 个数
A^{fn}_m	构成服务功能链 f 的第 n 个 VNF 中的第 m 个 AtomNF
$D^{A^{fn}_m}_i$	AtomNF 单元是否映射在节点 i 上
$c^{A^{fn}_m}_{\mathrm{p}}$	p 类 AtomNF 单元的部署代价
$T_{\max}$	服务功能链请求的最大可接受时延

在服务功能链的部署中，计算资源可具体化为CPU资源以及内存资源，因此必须确保某个节点上部署的全部AtomNF的CPU开销及内存开销不超过该节点可提供的相关资源。节点的CPU资源约束如式（1）所示。

$$\sum_{f\in S}\sum_{A_m^{fn}\in V}\mathrm{cpu}^{A_m^{fn}}\cdot D_i^{A_m^{fn}}<R_{\mathrm{cpu}}^i,i\in N \tag{1}$$

其中，$\mathrm{cpu}^{A_m^{fn}}$ 表示处理服务功能链 f 的第 n 个VNF中第 m 个AtomNF带来的CPU开销，R_{cpu}^i 表示节点 i 能提供的最大CPU资源。与CPU资源约束相似，节点的内存资源约束如式（2）所示。

$$\sum_{f\in S}\sum_{A_m^{fn}\in V}\mathrm{mem}^{A_m^{fn}}\cdot D_i^{A_m^{fn}}<R_{\mathrm{mem}}^i,i\in N \tag{2}$$

完成对SFC的部署后，数据流量按照AtomNF部署后的各个节点顺序进行遍历。网络拓扑中每条物理链路的带宽资源有限，因此SFC产生的需求所占用的链路带宽资源需要小于带宽总容量，相关约束如式（3）所示。

$$\begin{aligned}&\sum_{f\in S}B_{(v,v+1)}^f\cdot\mathrm{map}_{(n,n+1)}^{f,(v,v+1)}<B_{(n,n+1)},\\&(n,n+1)\in E,(v,v+1)\in E_{\mathrm{i}}\end{aligned} \tag{3}$$

其中，$\mathrm{map}_{n,n+1}^{f,(v,v+1)}\in\{0,1\}$ 是一个二维决策变量，表示虚拟链路 $(v,v+1)$ 是否被映射到物理链路 $(n,n+1)$ 上，$B_{v,v+1}^f$ 表示服务功能链 f 中不同的AtomNF映射至两个物理节点后它们之间的带宽开销，$B_{n,n+1}$ 表示节点 n 和 n+1 之间的链路带宽容量。

基于优化SFC平均部署网络成本的目标，需要考虑服务器节点激活开销、VNF部署成本、计算资源开销及链路带宽资源开销，如式（4）所示，其中 C_{active} 表示节点激活开销，来源于对非活跃物理节点的激活；C_{depoly} 表示VNF部署成本，来源于特定节点中一个新的VNF实例化；C_{cpu} 表示计算资源开销中的CPU资源开销；C_{mem} 表示计算资源开销中的内存资源开销；C_{bw} 表示链路带宽资源开销；α、β、γ、δ、ε 表示平衡各部分成本的权重系数，取值范围均为[0,1]。

$$\begin{aligned}&\min\ \alpha\cdot C_{\mathrm{active}}+\beta\cdot C_{\mathrm{depoly}}+\\&\gamma\cdot C_{\mathrm{cpu}}+\delta\cdot C_{\mathrm{mem}}+\varepsilon\cdot C_{\mathrm{bw}}\end{aligned} \tag{4}$$

由于不同应用对时延的敏感程度有差异，用户对SFC端到端时延的需求也各有不同，因此需要进行时延约束的相关定义。SFC的整体端到端时延可根据时延产生的位置划分为链路时延和节点时延两大部分，其中链路时延包括传播时延、传输时延及排队时延，节点时延即处理时延。因此，SFC部署在物理链路上产生的链路时延如式（5）所示。

$$d_f^{(x,y)}=d_{\mathrm{prop}}^{(x,y)}+d_{\mathrm{trans}}^{(x,y)}+d_{\mathrm{que}}^{(x,y)},f\in S \tag{5}$$

其中，d_{prop} 表示传播时延，来源于数据包在从源节点到目的节点的物理链路上传播花费的时间，即端到端路径长度之和与电磁波的传输速度的商，此项时延无限趋近零；d_{trans} 表示传输时延，来源于数据包从第一个比特到最后一个比特传输完成所花费的时间，即数据包长度与链路传输速率的商；d_{que} 为排队时延，来源于数据包在物理节点处理完成后在节点出口排队等待的时间，可通过M/M/1排队模型对其进行计算[15]。其中，$b_{x,y}$ 表示链路 (x,y) 已被占用的链路带宽资源，其与 $B_{x,y}$ 的商表示当前链路的带宽资源利用率，上下同乘 $B_{x,y}$ 后得到排队时延，如式（6）所示。

$$d_{\mathrm{que}}^{(x,y)}=\frac{b_{x,y}}{B_{x,y}-b_{x,y}}d_{\mathrm{trans}}^{(x,y)},(x,y)\in N \tag{6}$$

处理时延来源于数据包在各物理节点中进行处理所花费的时间，其表现方法与排队时延类似，取决于节点计算资源利用率及处理速率Rate。r_{cpu}^{fi} 表示节点 i 中已使用的计算资源，R_{cpu}^i 表示节点 i 的计算资源总量，r_{cpu}^{fi} 与 R_{cpu}^i 的商表示计算资源利用率。节点 i 上的处理时延如式（7）所示。

$$d_f^i=\frac{r_{\mathrm{cpu}}^{fi}}{R_{\mathrm{cpu}}^i-r_{\mathrm{cpu}}^{fi}}\ \mathrm{Rate}\ ,f\in S \tag{7}$$

结合式（6）与式（8），单条服务功能链的端

到端时延构成如式（9）所示。

$$T=\sum_{\substack{(x,y)\in E,\\(v,v+1)\in E_{\mathrm{i}}}}\mathrm{map}_{(x,y)}^{f,(v,v+1)}\cdot d_f^{(x,y)}+\sum_{i\in N,v_m\in f}D_i^{f,v_m}\cdot d_f^i,f\in S \tag{8}$$

$$\sum_{\substack{(x,y)\in E,\\(v,v+1)\in E_{\mathrm{i}}}}\mathrm{map}_{(x,y)}^{f,(v,v+1)}\cdot(d_{\mathrm{que}}^{(x,y)}+d_{\mathrm{trans}})+\sum_{i\in N,v_m\in f}D_i^{f,v_m}\cdot d_f^i<T_{\max} \tag{9}$$

式（10）的优化目标是在满足式（1）~式（3）的情况下，最小化服务功能链端到端时延。

$$\min\left(T=\sum_{\substack{(x,y)\in E,\\(v,v+1)\in E_{\mathrm{i}}}}\mathrm{map}_{(x,y)}^{f,(v,v+1)}\cdot d_f^{(x,y)}+\sum_{\substack{i\in N,\\v_m\in f}}D_i^{f,v_m}\cdot d_f^i\right) \tag{10}$$

2　基于微服务架构的粒度可变服务功能链映射算法设计

基于微服务架构的粒度可变服务功能链构建问题的关键在于根据 SFC 进行合理的逻辑拆分，继而根据拆分所得微服务单元集合筛选重复执行部分，并根据 SFC 原始逻辑进行服务图的绘制。其中，若拆分所得单元的标准不一致，则会导致后续重复执行的操作无法正常合并及复用，因此需将对数据流执行操作的最小逻辑功能作为最小工作单元并定义为微服务单元。

2.1　基于典型 VNF 的快速拆分策略

VNF 的拆分主要根据构成 SFC 的 VNF 集合及其相应的逻辑功能，将 VNF 拆分成若干个细粒度的微服务单元，并定义为 AtomNF，一个微服务单元代表对数据进行操作的最小逻辑单元。

服务功能链构成关系如图 4 所示，表述了微服务单元、虚拟网络功能及服务功能链之间的构成关系。3 类个体之间的关系可用式（11）与式（12）表述。

$$\mathrm{SFC}=\mathrm{nVNF},n\in\boldsymbol{N} \tag{11}$$

$$\mathrm{VNF}=m\mathrm{AtomNF},m\in\boldsymbol{N} \tag{12}$$

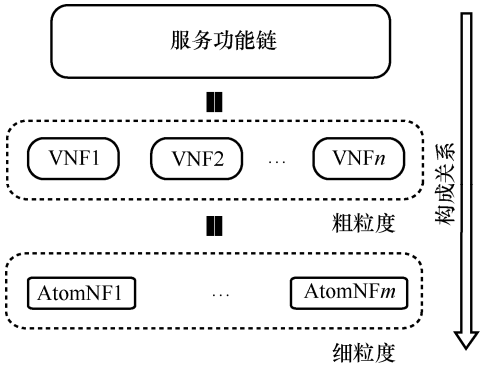


图 4　服务功能链构成关系

服务功能链拆分过程如图 5 所示，该 SFC 包含边缘防火墙、监视器、应用防火墙 3 个 VNF。通过逻辑分析，构成这 3 个 VNF 的微服务单元如图 5（b）所示。这些微服务单元经过拆分断开相互间原有的关系后，最终形成的微服务单元集合如图 5（c）所示。

为了实现 SFC 的快速拆分并降低拆分所带来的资源开销，基于典型 VNF 的快速拆分将逻辑分析与拆分工作前置，通过预先对出现频率较高的 VNF 进行逻辑分析，提前准备高频 VNF 所对应的微服务单元镜像文件并放入仓库。当 SFCR 到达时，即可快速根据相关的 VNF 拆分方案查找并直接拉取相关镜像。

此外用户的需求各有不同且无法预测，SFC 中出现未经前置拆分的 VNF 情况不可避免。针对此类情况，基于典型 NFV 的快速拆分将不对镜像仓库中未包含的 VNF 执行相关拆分操作，仍将其视作粗粒度的单一整体。

根据上述相关流程的描述，基于典型 NFV 的快速拆分算法流程如下。

算法 1：基于典型 NFV 的快速拆分算法

输入：服务功能链

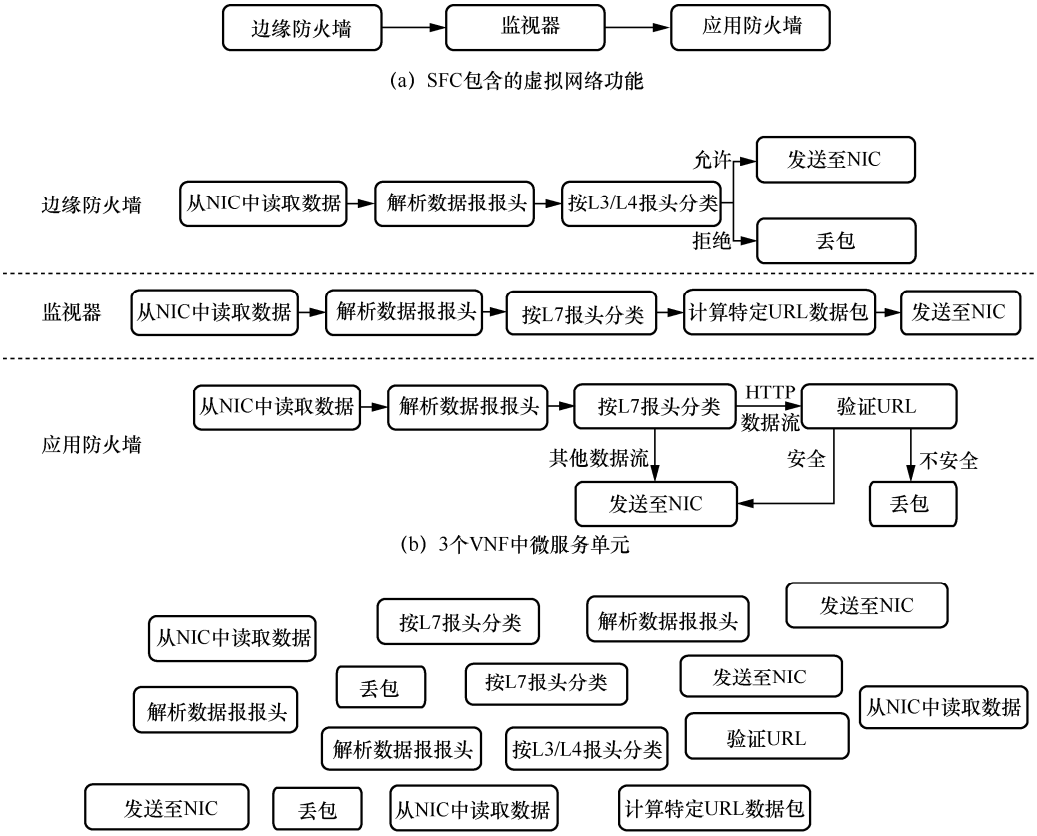
输出：AtomNF 集合

for i=1 to len(SFC) **do**

　if 当前 VNF 已前置拆分 **then**

　　确定拆分方案

　　从仓库中拉取细粒度镜像



(a) SFC包含的虚拟网络功能

(b) 3个VNF中微服务单元

(c) 拆分后的微服务单元集合

图 5　服务功能链拆分过程

加入 AtomNF 集合

else

当前 VNF 确定为整体粗粒度的映射单元

end

end

2.2　SFC 内的微服务单元合并策略

观察 SFC 拆分所得的微服务单元，可以发现 3 个 VNF 拆分后均存在部分共有的微服务单元。经过分析可以发现，重复执行相同的微服务单元并不会对数据流带来实际的改变，反而会增加所对应的映射节点的资源开销及端到端时延，因此可以对这些多次出现的微服务单元进行分析及合并。合并可以有效地缩短服务功能链的总长度，以降低端到端时延。此外，减少微服务单元的重复执行可以相应地降低相关计算资源的开销。

鉴于不同微服务单元对数据流的操作各有不同，参考 MicroNF[16]的分类规则，SFC 内的冗余微服务单元合并策略将微服务单元分为端点、区分、修改、重构、流量控制及静态 6 类，AtomNF 分类见表 2。

为了保证合并后的新 SFC 所提供的服务与合并前保持一致，同时最大限度地降低端到端时延，微服务单元的合并需要遵守以下原则：

（1）数据包须经历与原始服务功能链相同的处理步骤，但可减少部分冗余步骤的执行次数；

（2）每一个多次执行的微服务单元均需进行合并可能性分析，力求尽可能压缩服务功能链的端到端长度；

（3）与原始 SFC 相比，微服务单元在新 SFC 中的位置均可发生变化，但变化后不可使数据流发生改变。

合并后的服务功能链如图 6 所示，仅需 8 个微服务单元便可实现原始 SFC 的相关功能，最大

表 2　AtomNF 分类

AtomNF 类别	类别特征	典型例子
端点	开始或结束对数据包的操作	从网络接口控制器（network interface controller，NIC）读取数据，发送数据流至 NIC
区分	依据相关准则进行数据包的分类并输出至相关单元	协议解析、按 L7 报头分类
修改	对数据执行修改操作	IP 地址更改
重构	丢弃或增加一个具体的数据包	丢包
流量控制	修改数据流的传输速率，但不改变其传输内容	速率限制
静态	不修改该数据包本身及其特征，以及其他不属于前 5 类的单元	计算特定统一资源定位符（uniform resource locator，URL）数据包、记录器

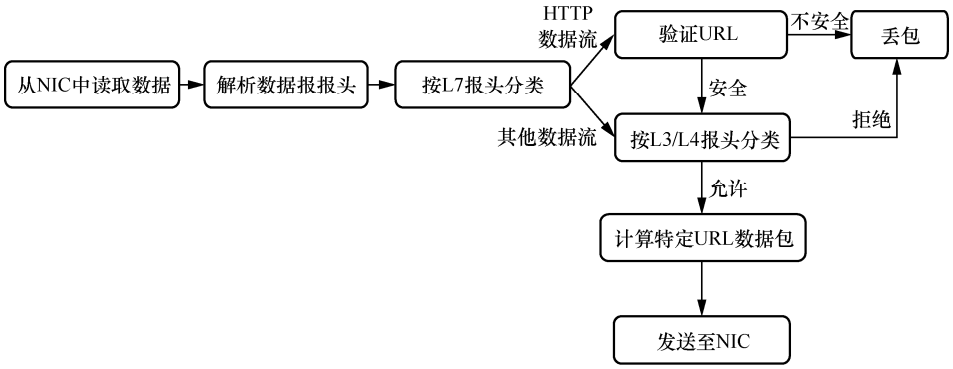


图 6　合并后的服务功能链

路径长度为 7，相较于原始 SFC 拆分后的路径长度 14 降低了 50%。微服务单元间的通信将采用网络服务分组头（network service header，NSH）进行数据的交互[17]，该方法可以有效地处理单一微服务单元，然后将其传递给多个微服务单元。

根据上述条件，SFC 内的微服务单元合并算法流程如下。

算法 2：SFC 内的微服务单元合并算法、

输入：AtomNF 集合

输出：服务图

for *i*-1 to len(AtomNF 集合) **do**

　遍历集合中的 AtomNF 并统计出现次数>1 的相关单元

　将相关单元加入冗余单元集合

　确认的单元类别

end

for *j*-1 to len(冗余单元集合) **do**

　单元类别划分

end

Function Select(AtomNF 集合)

　if SFC 长度== 0 **then**

　M[*i*][*n*++].append(映射方案)

　return

end

while *j*<链路长度+1 **do**

　Select(链路长度, SFC 长度−1)

　j=*j*+1

end

for *i*←1 to 链路数量 **do**

　for *j*←1 to 链路长度 **do**

　　Select(链路长度, SFC 长度−1)

　end

end

return *M*

2.3　SFC 间的微服务单元复用策略

在实际应用场景中，不同的 SFCR 内部往往

包含相同的 VNF，此外，从细粒度的角度进行观察，可进一步发现不同的 VNF 内存在大量相同的逻辑处理部分。从单一 SFC 内部的角度进行观察，相同逻辑处理部分可以通过合并的形式进行优化。若从多个 SFC 的角度进行观察，单一 SFC 内重复逻辑部分的合并形式可转化为复用的形式，以减少节点的激活，进一步降低服务功能链部署产生的网络成本。

VNF 或 AtomNF 实例的复用目前可通过多租户技术实现，多租户技术可以使得一个软件实例为多个租户提供相应服务[18-19]。相较于传统不可复用的映射方案，在 SFC 映射过程中，通过一个可复用的微服务单元实例为多个 SFCR 提供服务，可以有效提高节点资源利用率。

为了实现后续请求的复用，首次到达的 SFCR 需通过改进的灰狼优化（grey wolf optimizer，GWO）算法确定最佳映射方案。后续 SFCR 将以此映射方案为基准，通过对应映射节点的资源状况判断复用的可能性。

标准灰狼优化算法的数学描述如式（13）～式（16）所示。式（13）描述灰狼与猎物的距离，式（14）描述灰狼个体位置的更新，其中 t 表示当前的迭代次数，$\boldsymbol{X}_{\mathrm{p}}$ 与 $\boldsymbol{X}$ 分别表示猎物的位置向量和灰狼的位置向量，$\boldsymbol{A}$ 与 $\boldsymbol{C}$ 是系数向量，其中 $\boldsymbol{r}_1$ 和 $\boldsymbol{r}_2$ 是 [0,1] 内的随机数。

$$\boldsymbol{D}=\left|\boldsymbol{C}\cdot\boldsymbol{X}_{\mathrm{p}}(t)-\boldsymbol{X}(t)\right| \tag{13}$$

$$\boldsymbol{X}(t+1)=\boldsymbol{X}_{\mathrm{p}}(t)-\boldsymbol{A}\cdot\boldsymbol{D} \tag{14}$$

$$\boldsymbol{A}=2\boldsymbol{a}\cdot\boldsymbol{r}_1-\boldsymbol{a} \tag{15}$$

$$\boldsymbol{C}=2\boldsymbol{r}_2 \tag{16}$$

标准灰狼优化算法的全局搜索及局部搜索能力取决于参数 $\boldsymbol{A}$。当 $|\boldsymbol{A}|>1$ 时，灰狼远离潜在猎物，转而寻找更优的猎物，旨在保证算法的全局搜索能力；当 $|\boldsymbol{A}|<1$ 时，灰狼接近潜在猎物，并在其周围进行进一步搜索，旨在保证算法的局部搜索能力。而参数 $\boldsymbol{A}$ 的值取决于收敛因子 $\boldsymbol{a}$，当前 $\boldsymbol{a}$ 是随着迭代次数线性降低的。线性降低策略在面对复杂的、非线性的问题时容易陷入局部最优解。若要实现尽可能精确地搜索，则需要保证参数 $\boldsymbol{a}$ 的变化幅度为前缓后陡峭，因此本文对控制参数 $\boldsymbol{a}$ 执行非线性优化策略，如式（17）所示。

$$\boldsymbol{a}(t)=2\times\left[1-\frac{\mathrm{e}^{t/t_{\max}}-1}{(\mathrm{e}-1)^2}\right] \tag{17}$$

改进后的收敛因子 $\boldsymbol{a}$ 可以更好地平衡算法的全局搜索与局部搜索能力。在迭代前期，使参数 $|\boldsymbol{A}|$ 保持较大的值，有助于全局寻优，快速找到全局最优区域；在迭代后期，使参数 $|\boldsymbol{A}|$ 保持较小的值，从而提高算法的局部搜索能力。

后续到达的 SFCR 在完成前期的编排及拆分合并工作后，将自身的微服务单元集合与上一 SFCR 的微服务单元集合进行对比，重复出现的微服务单元即可复用的微服务单元，随后进行相关微服务单元映射节点的复用能力判断工作。判断节点的剩余资源能否承载该 SFC 其余不可复用的微服务单元。若不能，则将相关微服务单元部署至资源足够的邻近节点中。当出现邻近节点的可用资源均无法满足相关需求时，对该 SFC 再次通过改进的灰狼优化算法进行映射方案的确定。

根据上述工作流程，SFC 间的微服务单元复用算法流程如下。

算法 3：SFC 间的微服务单元复用算法

输入：服务功能链请求

输出：映射方案

SFC 内冗余微服务单元合并←算法 2

if 存在与上一到达 SFCR 相同的微服务单元 **then**

　分析可复用单元所部署节点的资源状况

　if 该节点可承载新到达服务功能链的余下微服务单元 **then**

　　确定映射方案

　else

　　if 相邻节点能够承载可复用微服务单

元外的各单元 **then**

确定映射方案

else

基于改进的灰狼优化算法的 SFC 映射

end

end

else

基于改进的灰狼优化算法的 SFC 映射

end

return 映射方案

2.4　映射算法设计

基于微服务架构的粒度可变服务功能链映射算法流程如下。首先与粗粒度单体 VNF 映射方案确定流程一致，根据用户请求的源节点和目的节点，通过 K 最短路径算法搜索可能的映射方案；然后通过基于典型 VNF 的快速拆分算法进行 SFC 的拆分，并得到相关微服务单元集合；最后执行 SFC 内的微服务单元合并操作，以得到端到端长度尽可能压缩的新服务功能链。

用户的 SFCR 往往是先后到达的，对于首个到来的 SFCR，网络中尚无已经部署的 SFC，因此无法执行相关微服务单元的复用，故需利用改进的灰狼优化算法进行映射方案的确定。针对后续到来的 SFCR，同样通过拆分及合并得到细粒度的服务功能链后，根据 SFC 间的微服务单元复用策略进行映射方案的确定。

算法 4：基于微服务架构的粒度可变服务功能链映射算法

输入：网络拓扑、服务功能链请求集合

输出：映射方案

遍历网络拓扑中各节点的资源状况

if 首个到达的 SFCR **then**

基于改进的灰狼优化算法的 SFC 映射

else

SFC 间微服务单元复用算法←算法 3

end

return 映射方案

为了有效衡量 VG-SFCM 算法寻找映射方案的效率，采用时间复杂度对其运行效率进行分析。首先该算法通过 K 最短路径算法搜索可能映射方案的时间复杂度为 $O\left(k|N|\left(|E|+|N|\lg|E|\right)\right)$；其次在利用改进的灰狼优化算法确定映射方案时，灰狼群体规模与物理节点集合 N 中的节点数量相等，待优化目标的维度与映射方案集合 M 中的个数相等，对灰狼种群初始化参数的时间复杂度为 $O(1)$，计算每个灰狼个体适应度值的时间复杂度为 $O\left(|N|\right)$，灰狼个体位置迭代更新所需的时间复杂度为 $O\left(|M||N|\right)$，则改进的灰狼优化算法的总时间复杂度为 $O(1)+O\left(|N|\right)+O\left(|M||N|\right)=O\left(|M||N|\right)$；而 SFC 间微服务单元复用算法的时间复杂度约为 $O\left(|N|\right)$。VG-SFCM 算法包含以上步骤，因此算法复杂度约为 $O\left(k|N|\left(|E|+|N|\lg|E|\right)\right)+O\left(|M||N|\right)+O\left(|N|\right)$，可以看出该算法时间复杂度主要由拓扑中的网络节点集合 N、节点间链路集合 E 及可能映射方案集合 M 决定。

3　仿真与结果分析

为了验证 VG-SFCM 算法的性能和有效性，本文将其与几种基线算法进行比较，分别是 OpenBox 解耦（OpenBox decoupling，OBD）算法、队列感知可靠嵌入（queue-aware reliable embedding，QARE）算法、基于动态规划的成本优化算法（dynamic programming based cost optimization algorithm，DP-COA）以及 SFC 动态编排（SFC dynamic orchestration，SFCDO）算法，并采用服务功能链请求的平均端到端时延、请求接受率以及平均部署网络成本作为评价指标。仿真使用 MATLAB 2020b 软件在配置为 AMD Ryzen 7 5800H CPU、16.0 GB RAM 的计算机上完成。OBD 算法利用 OpenBox 解耦的思想，将处理块作为映射的最小单元，映射时基于贪婪策略进行节点的选择[20]；

QARE 算法根据节点队列信息对 VNF 进行部署[21]；DP-COA 算法将 VNF 部署看成多阶段决策过程[22]，并基于动态规划的思想进行求解；SFCDO 算法采用广度优先搜索对 SFC 部署进行优化[23]。

3.1 仿真设计

为了便于进行仿真分析，本文采用典型物理网络拓扑——国家科学基金会网络（National Science Foundation Network，NSFNET）[24]，NSFNET 拓扑如图 7 所示。网络拓扑中某个时间段内到达的 SFCR 数量从 0 增加至 1 200。

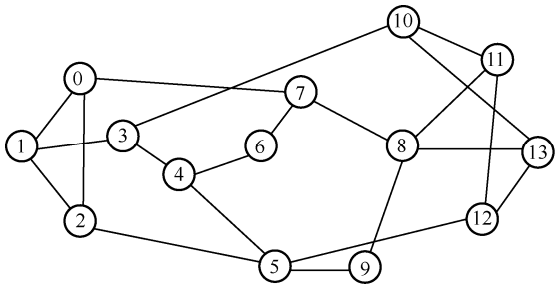


图 7 NSFNET 拓扑

该拓扑的具体参数及仿真过程中所需的其他参数见表 3。

仿真中产生的不同虚拟网络功能预设为 10 种，并对其中的三层路由器、二层交换机、IP 层防火墙、应用层防火墙、虚拟专用网设备、负载均衡及监视器共计 6 种 VNF 进行拆分方案的前置确定，而入侵检测系统、入侵防御系统、深度报文检测及虚拟专用网络则仍保持粗粒度的单体 VNF，以模拟实际情况中 VNF 未进行前置拆分的情况。各网络功能的具体 CPU 资源开销及内存资源开销各不相同，需进行前期的设定，AtomNF 级的微服务单元的 CPU 需求及内存需求均为 1 单位。

每条 SFC 的长度即拆分前的虚拟网络功能数量，为 4～6 间的随机整数，每个服务功能链中具体的 VNF 随机产生于前期预设的 10 种 VNF，由经过前置拆分方案确认的 VNF 及未经前置拆分方案确认的 VNF 构成。物理拓扑中网路节点间链路的传输时延设置为 1 ms，排队时延及处理时延通过式（6）和 式（7）结合当前时刻的资源利用率进行计算。

3.2 性能指标

为了验证 NFV 环境中 VG-SFCM 算法的可用性，本文使用以下 3 个性能指标作为仿真分析对象。

（1）所有服务功能链请求的平均端到端时延定义如式（18）所示，其中 Acc(S) 表示接受的请求数量。

表 3 该拓扑的具体参数及仿真过程中所需的其他参数

仿真对象	仿真参数	仿真值
NSFNET 拓扑	物理节点数/个	14
	物理链路数/条	21
	链路带宽资源	1 000 单位
	节点 CPU 资源	100 单位
	节点内存资源	100 单位
	节点激活成本	50 单位
服务功能链请求	单服务功能链 VNF 数/个	4、5、6
	入口节点及出口节点/个	1、13
	带宽需求/（Mbit·s^{-1}）	[40,60]
	最大可接受时延/ms	[50,100]
虚拟网络功能	已预设前置拆分 VNF 数/个	6
	未预设前置拆分 VNF 数/个	4
	CPU 开销及内存开销	每种类型前期设定
	未拆分 VNF 部署成本	1 单位
	单 AtomNF 部署成本	0.4 单位

$$T_{\text{avg}}=\frac{\sum_{f\in S}\sum_{\substack{(x,y)\in E,\\(v,v+1)\in E_{\text{i}}}}\text{map}_{(x,y)}^{f,(v,v+1)}\cdot d_f^{(x,y)}+\sum_{f\in S}\sum_{i\in N,v_m\in f}D_i^{f,v_m}\cdot d_f^i}{\text{Acc}(S)} \tag{18}$$

（2）网络拓扑中服务功能链请求接受率定义如式（19）所示，其中 $\text{Arr}(S)$ 表示到达的总请求数量。

$$\text{rate}=\frac{\text{Acc}(S)}{\text{Arr}(S)} \tag{19}$$

（3）服务功能链请求的平均部署网络成本定义如式（21）所示。

$$\text{COST}=\frac{\alpha\cdot C_{\text{active}}+\beta\cdot C_{\text{depoly}}+\gamma\cdot C_{\text{cpu}}+\delta\cdot C_{\text{mem}}+\varepsilon\cdot C_{\text{bw}}}{\text{Acc}(S)} \tag{20}$$

3.3　仿真结果分析

式（18）～式（20）描述的各项性能指标均将 SFCR 数量作为自变量。不同算法的服务功能链请求的平均端到端时延如图 8 所示，展示了部署映射方案后 SFC 的平均端到端时延随 SFCR 数量变化的趋势。通过观察，可得 VG-SFCM 算法始终保持较低的平均端到端时延。这是因为该算法在确定映射方案前总会进行 SFC 的微服务化拆分及合并，减少了冗余微服务单元的执行，缩短了处理时间。随着 SFCR 数量的增加，各算法的平均端到端时延均呈现上升趋势，这种现象主要归因于各节点的计算资源和节点间的链路带宽资源不断被消耗。

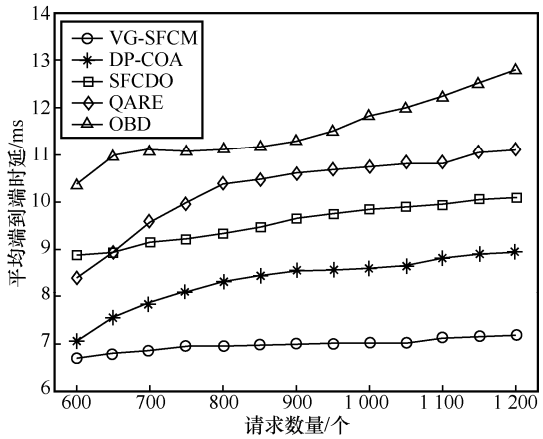


图 8　不同算法的服务功能链请求的平均端到端时延

不同算法的服务功能链请求接受率如图 9 所示，展示了不同算法在相同 SFCR 数量下的请求接受率。由图 9 可以得到，当服务功能链请求数量不断增加时，各算法的请求接受率均呈下降趋势，但 VG-SFCM 的性能优于其他算法。当请求数量超过 1 000 时，SFCDO 算法的接受率开始直线下降。VG-SFCM 算法的请求接受率最后比 SFCDO 算法高 5%左右，比 OBD 算法高 15%左右。这是因为 VG-SFCM 算法改变了最小的映射单元，有效避免了对比算法中物理节点剩余资源无法进行单体 VNF 的部署时产生资源碎片的情况。此外，通过微服务单元的合并及复用操作减少微服务单元的实例化，处理更多 SFC，进一步提高了 SFCR 接受率。

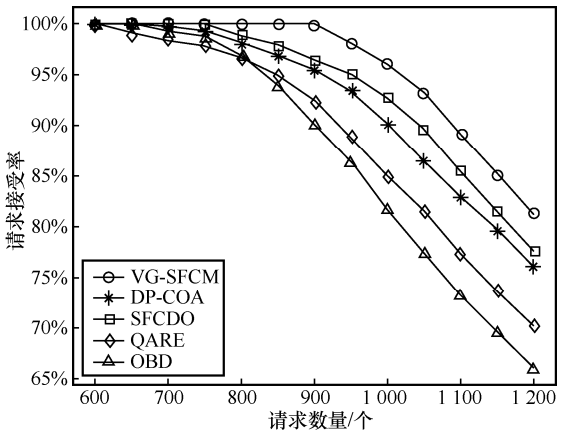


图 9　不同算法的服务功能链请求接受率

不同服务功能链请求数量下的平均部署网络成本如图 10 所示。平均部署网络成本的权重系数 α、β、γ 与 δ 均设置为 1，ε 设置为 0.1。从图 10 可以看出，相较于其他算法，VG-SFCM 算法的平均部署网络成本始终在保持较低水平。各个算法的平均部署网络成本在初期都随服务功能链请求数量的增加呈上升趋势，导致其上升的原因是后续到达的服务功能链请求增加了带宽资源开销并且使得节点使用数量增多，从而产生更多的节点激活成本。SFCDO 算法在部署过程中大概率会选择源节点与目的节点间路由跳数最少的路径，导致其他方面的成本增加。DP-COA 算法通过激活更多的节点达到

更好的负载均衡效果，造成节点运行代价变高。而 VG-SFCM 算法对平均部署网络成本的优化主要体现在两个方面，一是前期节点使用数量减少，使得网络拓扑中的节点激活成本降低；二是合并操作使得冗余微服务单元的计算资源开销减少。平均部署网络成本的变化趋势表明了基于 VG-SFCM 算法的映射方案的平均部署网络成本控制要优于其他算法。

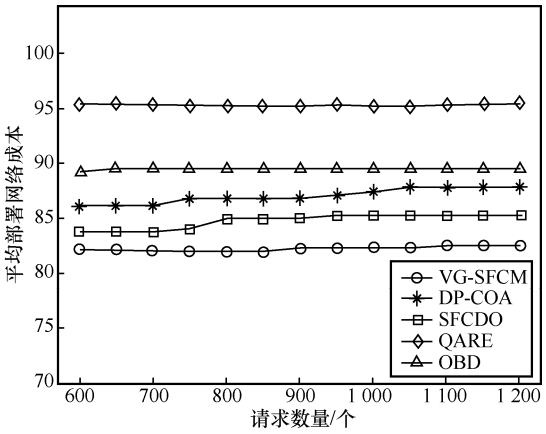


图 10　不同服务功能链请求数量下的平均部署网络成本

为了更加直观地体现出 VG-SFCM 算法的优势，对 VG-SFCM 和其他算法的整体性能进行总结，算法性能对比见表 4。通过观察可以看出，在同一实验环境下，与其他算法相比，VG-SFCM 算法确实有效降低了服务功能链的端到端时延，提高了服务功能链请求接受率并且减少了平均部署网络成本。

4　结束语

本文研究了 NFV 环境中考虑时延优化和可变粒度的服务功能链映射问题，旨在最小化服务功能链的端到端时延，并降低映射及部署产生的平均部署网络成本。针对这一优化目标，提出了基于微服务架构的粒度可变服务功能链映射算法，该算法首先对服务功能链进行快速拆分，并执行合并操作以减少不必要的微服务单元重复执行。随后通过 SFC 间的微服务单元复用策略，减少后续到达的 SFCR 相关实例的映射及部署，提高节点资源利用率。仿真结果表明，该算法在保证用户需求的情况下，有效降低了服务功能链的平均端到端时延，提高了请求接受率，减少了节点激活成本。但目前该算法对服务功能链的拆分均基于人工的逻辑分析及微服务化前置，无法实时地对未经相关前置操作的虚拟网络功能进行微服务化。在后续的研究中，笔者将对该算法进行相关

表 4　算法性能对比

算法	性能	SFCR 数量/个			
		600	800	1 000	1 200
OBD	平均端到端时延/ms	10.395	11.142 9	11.845 6	12.982 8
	请求接受率	99.33%	96.88%	81.80%	66.00%
	平均部署网络成本	89.284 1	89.502 4	89.456 2	89.483 3
QARE	平均端到端时延/ms	8.389 3	10.391 7	10.727 4	11.158 8
	请求接受率	99.17%	96.75%	84.90%	70.17%
	平均部署网络成本	95.284 7	95.104 5	95.024 1	95.331 1
DP-COA	平均端到端时延/ms	7.052 1	8.312 4	8.601 2	8.910 4
	请求接受率	99.50%	98.00%	90.20%	76.00%
	平均部署网络成本	86.083 8	86.692 3	87.375 8	87.767 4
SFCDO	平均端到端时延/ms	8.868 5	9.333 5	9.820 1	10.173 2
	请求接受率	99.50%	98.75%	92.60%	77.50%
	平均部署网络成本	83.683 7	84.885 7	85.155 6	85.152 6
VG-SFCM	平均端到端时延/ms	**6.669 2**	**6.944 3**	**7.006 7**	**7.201 1**
	请求接受率	**99.83%**	**99.75%**	**96.00%**	**81.25%**
	平均部署网络成本	**82.084 1**	**81.876 6**	**82.329 0**	**82.517 9**

优化，设计对应的自动化微服务拆分算法，以满足网络状态动态变化的特征。

参考文献：

[1] SCHARDONG F, NUNES I, SCHAEFFER-FILHO A. NFV resource allocation: a systematic review and taxonomy of VNF forwarding graph embedding[J]. Computer Networks, 2021, 185: 107726.

[2] 翟东, 孟相如, 康巧燕, 等. 面向时延与可靠性优化的服务功能链部署方法[J]. 电子与信息学报, 2020, 42(10): 2386-2393.
ZHAI D, MENG X R, KANG Q Y, et al. Service function chain deployment method for delay and reliability optimization[J]. Journal of Electronics & Information Technology, 2020, 42(10): 2386-2393.

[3] REDDY V S, BAUMGARTNER A, BAUSCHERT T. Robust embedding of VNF/service chains with delay bounds[C]//Proceedings of 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks. Piscataway: IEEE Press, 2016: 93-99.

[4] MEHRAGHDAM S, KELLER M, KARL H. Specifying and placing chains of virtual network functions[C]//Proceedings of 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet). Piscataway: IEEE Press, 2014: 7-13.

[5] MECHTRI M, GHRIBI C, ZEGHLACHE D. A scalable algorithm for the placement of service function chains[J]. IEEE Transactions on Network and Service Management, 2016, 13(3): 533-546.

[6] QI D D, SHEN S B, WANG G H. Towards an efficient VNF placement in network function virtualization[J]. Computer Communications, 2019, 138: 81-89.

[7] 陈卓, 冯钢, 刘怡静, 等. MEC 中基于改进遗传模拟退火算法的虚拟网络功能部署策略[J]. 通信学报, 2020, 41(4): 70-80.
CHEN Z, FENG G, LIU Y J, et al. Virtual network function deployment strategy based on improved genetic simulated annealing algorithm in MEC[J]. Journal on Communications, 2020, 41(4): 70-80.

[8] 孙士清, 彭建华, 游伟, 等. 5G 网络下资源感知的服务功能链协同构建和映射算法[J]. 西安交通大学学报, 2020, 54(8): 140-148.
SUN S Q, PENG J H, YOU W, et al. A coordinating composition and mapping algorithm for a service function chain with resource-aware[J]. Journal of Xi'an Jiaotong University, 2020, 54(8): 140-148.

[9] YALA L, FRANGOUDIS P A, KSENTINI A. Latency and availability driven VNF placement in a MEC-NFV environment[C]//Proceedings of 2018 IEEE Global Communications Conference. Piscataway: IEEE Press, 2018: 1-7.

[10] 唐伦, 曹睿, 廖皓, 等. 基于深度强化学习的服务功能链可靠部署算法[J]. 电子与信息学报, 2020, 42(12): 2931-2938.
TANG L, CAO R, LIAO H, et al. Reliable deployment algorithm of service function chain based on deep reinforcement learning[J]. Journal of Electronics & Information Technology, 2020, 42(12): 2931-2938.

[11] LI J L, SHI W S, ZHANG N, et al. Reinforcement learning based VNF scheduling with end-to-end delay guarantee[C]//Proceedings of 2019 IEEE/CIC International Conference on Communications in China (ICCC). Piscataway: IEEE Press, 2019: 572-577.

[12] 冯志勇, 徐砚伟, 薛霄, 等. 微服务技术发展的现状与展望[J]. 计算机研究与发展, 2020, 57(5): 1103-1122.
FENG Z Y, XU Y W, XUE X, et al. Review on the development of microservice architecture[J]. Journal of Computer Research and Development, 2020, 57(5): 1103-1122.

[13] BREMLER-BARR A, HARCHOL Y, HAY D. OpenBox: a software-defined framework for developing, deploying, and managing network functions[C]//Proceedings of the 2016 ACM SIGCOMM Conference. New York: ACM Press, 2016: 511-524.

[14] CHOWDHURY S R, SALAHUDDIN M A, LIMAM N, et al. Re-architecting NFV ecosystem with microservices: state of the art and research challenges[J]. IEEE Network, 2019, 33(3): 168-176.

[15] DWARAKI A, WOLF T. Adaptive service-chain routing for virtual network functions in software-defined networks[C]//Proceedings of the 2016 workshop on Hot Topics in Middleboxes and Network Function Virtualization. New York: ACM Press, 2016: 32-37.

[16] MENG Z L, BI J, WANG H P, et al. MicroNF: an efficient framework for enabling modularized service chains in NFV[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(8): 1851-1865.

[17] NEKOVEE M, SHARMA S, UNIYAL N, et al. Towards AI-enabled microservice architecture for network function virtualization[C]//Proceedings of 2020 IEEE Eighth International Conference on Communications and Networking (ComNet). Piscataway: IEEE Press, 2020: 1-8.

[18] MA H J, ZHANG J Y, WANG Z X. Implementation of multi-tenancy adaptive cache admission model[J]. Journal of Physics: Conference Series, 2021, 2026(1): 012045.

[19] LEIVADEAS A, FALKNER M. VNF placement problem: a multi-tenant intent-based networking approach[C]//Proceedings of 2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN). Piscataway: IEEE Press, 2021: 143-150.

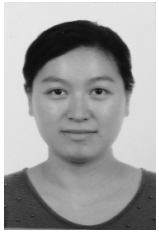
[20] LI D F, HONG P L, XUE K P, et al. Virtual network function placement considering resource optimization and SFC requests in cloud datacenter[J]. IEEE Transactions on Parallel and Distributed Systems, 2018, 29(7): 1664-1677.

[21] TANG L, ZHAO G F, WANG C M, et al. Queue-aware reliable embedding algorithm for 5G network slicing[J]. Computer Networks, 2018, 146: 138-150.

[22] 刘昀. 虚拟网络功能资源分配与服务功能链路由研究[D]. 合肥: 中国科学技术大学, 2020.
LIU Y. Virtual network function resource allocation and service function chain routing[D]. Hefei: University of Science and Technology of China, 2020.

[23] SUN G, XU Z, YU H F, et al. Low-latency and resource-efficient service function chaining orchestration in network function virtualization[J]. IEEE Internet of Things Journal, 2020, 7(7): 5760-5772.
[24] STRAWN G. Masterminds of the NSFnet: jennings, wolff, and van houweling[J]. IT Professional, 2021, 23(6): 67-69.

[作者简介]



吴晓春（1983- ），女，博士，浙江工商大学信息与电子工程学院高级实验师、硕士生导师，主要研究方向为新一代网络技术架构、软件定义网络、网络功能虚拟化、人工智能与网络安全的结合等。

洪晨（1998- ），女，浙江工商大学信息与电子工程学院硕士生，主要研究方向为知识图谱、新一代网络技术架构。

张岳（1998- ），女，浙江工商大学信息与电子工程学院硕士生，主要研究方向为知识图谱、新一代网络技术架构。

张俊楠（1997- ），男，浙江工商大学信息与电子工程学院硕士生，主要研究方向为新一代网络技术架构。

周静静（1980- ），女，博士，浙江工商大学信息与电子工程学院副教授、硕士生导师，主要研究方向为新一代网络技术架构、软件定义网络、网络流量建模与分析、大数据处理、深度学习等。