文章编号:1007-5321(2022)02-0009-07 **DOI**:10.13190/j.jbupt.2021-165

面向多样化需求的网络切片业务链部署

李 航, 温向明, 孔紫璇, 向 万, 王鲁晗

(1. 北京邮电大学 信息与通信工程学院, 北京 100876;
2. 北京邮电大学 网络体系构建与融合北京市重点实验室, 北京 100876;
3. 北京邮电大学 先进信息网络北京实验室, 北京 100876)

摘要: 网络切片通过业务链部署来实现切片的创建和编排。针对网络切片中的业务链部署, 考虑了业务链的多样化需求,并引入了虚拟网络功能共享和准入控制,以降低部署成本并提高业务链的接受率。将上述问题建模成一个最大化网络净收益的优化模型,提出了部署算法。仿真结果表明,所提的部署算法优于已有的基准算法,能达到接近最优的性能。

关 键 词: 网络切片; 业务链部署; 虚拟网络功能; 启发式算法

中图分类号: TP393.0 **文献标志码**: A

Service Function Chain Embedding for Network Slicing with Diversified Requirements

LI Hang, WEN Xiangming, KONG Zixuan, XIANG Wan, WANG Luhan

(1. School of Information and Communications Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China;
2. Beijing Key Laboratory of Network System Architecture and Convergence, Beijing University of Posts and Telecommunications, Beijing 100876, China;
3. Beijing Laboratory of Advanced Information Networks, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: Network slicing realizes the creation and orchestration of slices through service function chain embedding. However, the service function chain embedding in network slicing should consider diversified requirements of service function chains, reduce deployment costs and increase the acceptance ratio by taking virtual network function sharing and admission control into consideration. The service chain embedding for network slicing with diversified requirements is studied, where virtual network function sharing and admission control are introduced. First, the above problem is formulated as an optimization model that maximizes the network net revenue, and then an embedding algorithm is devised. The simulation results show that the proposed algorithm outperforms the existing benchmark algorithm, and achieves near-optimal performance.

Key words: network slicing; service function chain embedding; virtual network function; heuristic algorithm

网络切片技术[1]旨在通过单一物理网络基础设施来满足多样化的服务。在网络切片中,底层物理网络被划分为多个逻辑虚拟网络,每一个虚拟网络被称为切片。不同的虚拟化网络功能(VNF,

收稿日期: 2021-08-06
基金项目: 国家重点研发计划项目(2019YFB1803301); 北京市自然科学基金项目(L202002); 国家自然科学基金项目(61801036)
作者简介: 李 航(1994—), 男, 博士生。
通信作者: 王鲁晗(1989—), 男, 讲师, 硕士生导师, 邮箱: wluhan@bupt.edu.cn。

virtual network function)按照特定的顺序依次处理用户数据流,形成业务链(SFC, service function chain)[2],实现相应的服务。基于网络功能虚拟化和软件定义网络技术,网络运营者需要实施SFC部署[3]来实现切片的创建和编排,如图1所示。具体来讲,SFC部署主要是为SFC确定合适的VNF实例化位置和数据流路由路径,满足相应需求的同时降低部署成本。

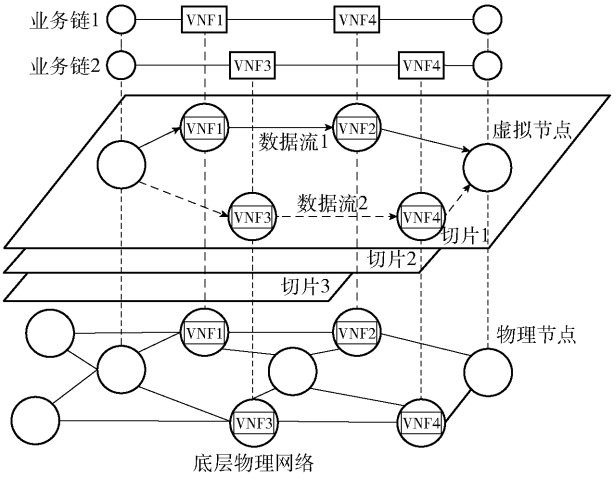


图1 网络切片的业务链部署

对于网络切片场景下的SFC部署(以下简称切片SFC部署),相比传统的SFC部署,应重点考虑以下3个因素:网络切片中SFC的多样化需求;VNF共享;准入控制[4]。首先,网络切片中每一个切片承载不同类型的服务,因此属于不同切片的SFC需求也不同。如增强型移动宽带(eMBB, enhanced mobile broadband)服务下的SFC要求较高的带宽;超可靠和低时延通信(uRLLC, ultra-reliable and low-latency communications)服务下的SFC需要严格的时延保障;而大规模机器类型通信(mMTC, massive machine type communications)服务下的SFC对带宽和时延都没有太严格的要求[5]。因此,切片SFC部署应综合多种因素,如带宽消耗、端到端时延等,以应对这些多样化的SFC需求。其次,网络中包含很多切片和相应的SFC,为SFC中的每个VNF都部署VNF实例会增加部署成本和管理成本[6],而通过SFC间VNF共享可以降低成本。由于需要考虑切片维度,网络切片中的VNF共享比传统SFC部署中的VNF共享更加复杂。根据VNF的功能特性,有些VNF(网络地址转换)可在所属不同切片的SFC之间共享(以下简称切片间共享),而有些VNF(防火墙)则因为要保障切片间的隔离性而不能进行共享[7]。最后,在网络切片中,有限的底层网络资源不一定能够满足所有SFC部署请求。此时需要通过准入控制[8]来决定需部署哪些SFC,使得最大化总SFC接受率的同时最小化总部署成本。与传统SFC部署中的准入控制不同,网络切片中的准入控制在提高总SFC接受率的同时也要尽可能提高各切片上的SFC接受率。

尽管已有较多学者研究了传统的SFC部署和切片部署,但仅有少数人[6-7,9-10]研究了结合两者的切片SFC部署。Papagianni等[6]考虑了切片内VNF共享的切片SFC部署,并没有考虑切片间VNF共享。Truong-Huu等[7]考虑了SFC多样化需求和切片间VNF共享的网络切片SFC部署,并提出了启发式算法进行部署。然而在上述工作中,没有考虑准入控制。Zhang等[9]研究了最小化带宽消耗的网络切片SFC部署,但未考虑时延,忽略了SFC的多样化需求。Chen等[10]考虑了端到端时延的切片SFC部署,然而均未考虑VNF共享,同时也忽略了准入控制。

鉴于以上工作的不足,笔者研究了面向多样化需求的网络切片SFC部署,并考虑了VNF共享和准入控制。首先把上述问题建模成一个最大化网络净收益的整数线性规划模型,然后设计了切片SFC部署JSA(joint sharing and admission)算法。仿真结果显示,JSA算法相比已有的切片SFC部署算法能提高1倍的总净收益和10%以上的总接受率,并达到接近最优的性能。

1 系统模型和建模

为建立面向多样化需求的网络切片SFC部署模型,应对多样化需求的SFC,综合考虑了链路带宽容量、SFC带宽需求、网络带宽消耗、链路时延、SFC端到端时延等多种因素,同时引入了切片间VNF共享和准入控制机制。

1.1 系统模型

1.1.1 底层物理网络

底层物理网络定义为图$G(N,E)$,其中N和E分别为节点集合和链路集合。节点分为服务节点和转发节点,服务节点上可布置VNF实例和转发流量,而转发节点只负责流量的转发。每个节点n的节点计算容量表示为$c_{\text{node}}(n)$(转发节点的资源容量为0),同时设n的相邻节点集合为ε_n。$e_{m,n} \in E$

表示从点 m 到点 n 的有向链路，e_{mn} 表示由点 m 和点 n 组成的无向链路。链路 e_{mn} 的链路带宽容量和传播时延分别为 $c_{\text{link}}(e_{mn})$，$d_{\text{tr}}(e_{mn})$。

1.1.2　VNF

设网络中支持的 VNF 类型集合为 $V = V_{\text{sh}} \cup V_{\text{ush}}$，其中 V_{sh} 为可以在切片间共享的 VNF 集合，V_{ush} 为不可以在切片间共享的 VNF 集合。同文献[7]，假设每个 VNF 实例所需的计算资源正比于其处理的数据流量大小。设 δ_v 为 VNF$v \in V$ 的一个实例处理单位数据流量时需要的计算资源。为避免排队时延，限定每个 v 实例所能处理的最大数据流量大小为 $c_{\text{VNF}}(v)$。由于每种 VNF 的软件许可证允许的实例副本数量可能是有限的，所以设 v 在节点 n 最多可布置 ζ_n^v 个实例。部署一个 VNF 实例需要一定的开销，如待机耗电、软件授权等，将这部分开销认为是 VNF 部署成本，并设部署 v 的一个实例需要 α_v 的费用，同时假设不可切片间共享的 VNF 在同一个切片内可被属于该切片的 SFC 之间共享。

1.1.3　切片与 SFC 部署请求

设网络的切片集合为 S，$s \in S$ 是其中一个切片。使用 F_s 来表示属于切片 s 的 SFC 部署请求（以下简称 SFC 请求）集合，则总的 SFC 请求集合为 $F = \{F_s \mid s \in S\}$。假设 F 到来时，整个网络中没有已部署的 SFC 和 VNF（本文的模型易扩展到此场景）。使用链状拓扑来表示每一个 SFC 请求，也就是 $f = \{U_f = n_{\text{str}} \cup V_f \cup n_{\text{des}}, H_f\} \in F_s$，其中 $n_{\text{str}}, n_{\text{des}}$ 分别为 f 的数据流的起始节点和目的节点；V_f 为 f 的数据流需要遍历的 VNF 集合。$H_f = \{h_1, h_2, \cdots, h_{|H_f|}\}$ 为 f 的逻辑链路集合，其中 h_i 表示其第 i 个逻辑链路。定义 SFC 请求 f 的长度为 $|U_f|$，则有 $|U_f| = |H_f| + 1$。使用布尔值 μ_f^v 来表示 $v \in V$ 是否用于 f，用整数值 o_f^v 表示 $v \in V_f$ 是 f 的第几个 VNF。设 f 的数据流量大小（带宽需求）为 $r_{\text{SFC}}(f)$，同时用 β 表示单位数据流量产生的带宽消耗费用。同文献[11]，假设成功部署 f 可获得 $m_{\text{SFC}}(f)$ 的收益，拒绝部署将需要承担 $p_{\text{SFC}}(f)$ 的损失。最后，设 SFC 请求 f 的最大容忍端到端时延为 $t_{\text{SFC}}(f)$。

1.2　建模

1.2.1　决策变量

定义布尔决策变量 x_f，$y_n^{f,v}(y_n^{f,v} \leqslant x_f)$，$z_{e_{m,n}}^{f,h}(z_{e_{m,n}}^{f,h} \leqslant x_f)$，其中 $x_f = 1$ 表示 SFC 请求 f 部署成功；$y_n^{f,v} = 1$ 表示服务 f 的 VNFv 部署在节点 n；$z_{e_{m,n}}^{f,h} = 1$ 表示链路 $e_{m,n}$ 用于承载 f 的逻辑链路 $h \in H_f$。定义整数决策变量 w_n^v 表示节点 n 上部署的 VNFv 实例个数，满足 $0 \leqslant w_n^v \leqslant \zeta_n^v$。而对于 VNF$v \in V_{\text{ush}}$，定义 $w_n^{v,s}$，表示节点 n 上部署的用于切片 s 的 v 实例数量，则有

$$\sum_{s \in S} w_n^{v,s} = w_n^v$$

1.2.2　约束条件

1）需确保已部署的 VNF 节点和相应物理链路之间的关系，表达式如下：

$$y_n^{f,v} \leqslant \sum_{m \in \varepsilon_n} z_{e_{n,m}}^{f,h_{o_f^v}}, \ \forall f \in F, v \in V_f, n \in N \tag{1}$$

$$y_n^{f,v} \leqslant \sum_{m \in \varepsilon_n} z_{e_{n,m}}^{f,h_{o_f^v+1}}, \ \forall f \in F, v \in V_f, n \in N \tag{2}$$

同文献[9]，假设一个 SFC 请求中的任意 2 个 VNF 都不能部署在相同节点上，表达式为

$$\sum_{v \in V_f} y_n^{f,v} \leqslant 1, \ \forall n \in N, f \in F \tag{3}$$

2）需确保数据流守恒，有如下表达式：

$$\sum_{m \in \varepsilon_{n_{\text{str}}}} z_{e_{n_{\text{str}},m}}^{f,h_1} = x_f, \ \forall f \in F \tag{4}$$

$$\sum_{m \in \varepsilon_{n_{\text{des}}}} z_{e_{m,n_{\text{des}}}}^{f,h_{|H_f|}} = x_f, \ \forall f \in F \tag{5}$$

$$\sum_{m \in \varepsilon_n} \sum_{h \in H_f} z_{e_{m,n}}^{f,h} = \sum_{o \in \varepsilon_n \backslash m} \sum_{h \in H_f} z_{e_{n,o}}^{f,h}, \quad \forall f \in F; n \in N \backslash \{n_{\text{str}}, n_{\text{des}}\} \tag{6}$$

3）需确保每个 VNF 实例能处理的数据量满足约束条件，有如下表达式：

$$\sum_{f \in F} \mu_f^v y_n^{f,v} r_{\text{SFC}}(f) \leqslant c_{\text{VNF}}(v) w_n^v, \quad \forall v \in V_{\text{sh}}, n \in N \tag{7}$$

$$\sum_{f \in F_s} \mu_f^v y_n^{f,v} r_{\text{SFC}}(f) \leqslant c_{\text{VNF}}(v) w_n^{v,s}, \quad \forall v \in V_{\text{ush}}, n \in N, s \in S \tag{8}$$

4）需分别确保节点计算资源和链路带宽资源满足约束条件，有如下表达式：

$$\sum_{v \in V} \sum_{f \in F} \mu_f^v y_n^{f,v} \delta_v r_{\text{SFC}}(f) \leqslant c_{\text{node}}(n), \ \forall n \in N \tag{9}$$

$$\sum_{f \in F} \sum_{h \in H_f} r_{\text{SFC}}(f) z_{e_{m,n}}^{f,h} \leqslant c_{\text{link}}(e_{mn}), \ \forall e_{m,n} \in E \tag{10}$$

5）需确保每个 SFC 请求的端到端时延满足约束条件，表达式为

$$\sum_{h \in H_f} \sum_{e_{m,n} \in E} d_{\text{tr}}(e_{mn}) z_{e_{m,n}}^{f,h} \leqslant t_{\text{SFC}}(f), \ \forall f \in F \tag{11}$$

6）需确保 SFC 的数据流路由路径不会产生回流[7]，有如下表达式：

$$\sum_{h \in H_f} (z_{e_{m,n}}^{f,h} + z_{e_{n,m}}^{f,h}) \leqslant 1, \ \forall m \in N, n \in \varepsilon_m, f \in F \tag{12}$$

$$\sum_{\forall m\in\varepsilon_n}\sum_{h\in H_f} z^{f,h}_{e_{m,n}} \leqslant 1,\ \forall n\in N,\ f\in F \tag{13}$$

$$\sum_{\forall m\in\varepsilon_n}\sum_{h\in H_f} z^{f,h}_{e_{n,m}} \leqslant 1,\ \forall n\in N,\ f\in F \tag{14}$$

1.2.3 优化目标

网络的总净收益可表示为

$$R=\left(\sum_{f\in F} x_f m_{\mathrm{SFC}}(f)-\sum_{f\in F}(1-x_f)p_{\mathrm{SFC}}(f)\right)-\sum_{v\in V}\alpha_v w_n^v-\sum_{f\in F}\sum_{h\in H_f}\sum_{e_{m,n}\in E}\beta r_{\mathrm{SFC}}(f)z^{f,h}_{e_{m,n}} \tag{15}$$

其中:第1项表示部署SFC请求所获得的收益,第2项为部署VNF实例的成本,第3项为带宽成本。总优化目标是最大化总净收益。由于每个SFC请求f所需的总节点计算资源为固定值$\sum_{v\in V_f}\delta_v r_{\mathrm{SFC}}(f)$,可不考虑节点计算资源成本(该成本可融入$m_{\mathrm{SFC}}(f)$)。因此,切片SFC部署问题可描述为以下整数线性规划模型:

$$\max\ R \tag{16}$$

$$\text{s.t. 式(1)~式(14)}$$

上述问题的一个特殊情况是只存在一个切片,即此时问题转换为传统SFC部署问题,为NP(non-deterministic polynomial)困难问题[12]。因此,式(16)也是NP困难问题。考虑到NP的困难特性,设计了一个启发式算法进行求解。

2 业务链部署算法JSA

2.1 算法步骤

步骤1 排序所有SFC请求。首先将请求集合F按照SFC的长度$|U_f|$进行排序,$|U_f|$越小,该SFC的优先级越高。相同长度的SFC请求按照其数据流量大小进行排序,数据流量需求越小,优先级越高。JSA算法按照优先级从高到低的顺序,尝试对SFC进行依次部署。

步骤2 选取当前需要部署的SFC请求f,进行带宽验证。验证方法如下:首先,对底层网络中每一条链路的剩余带宽都减去$r_{\mathrm{SFC}}(f)$,结果为负数时断开该链路;其次,检查此时n_{str}到n_{des}的连通性,如果不连通,则f部署失败,转到步骤7;如果连通,继续步骤3。

步骤3 时延验证。利用Dijkstra算法[13]计算n_{str}到n_{des}的最短时延,如果超过$t_{\mathrm{SFC}}(f)$,则f部署失败,转到步骤7;否则,继续步骤4。

步骤4 尝试部署当前f,提取当前网络的剩余节点计算资源容量数组A和剩余链路带宽容量矩阵$\boldsymbol{B}$。设服务节点集合为N_s,则对于每一个服务节点$n\in N_s$,初始化VNF部署节点记录数组q^n,路由路径记录数组p^n,累计虚拟成本记录量l^n,计算资源容量数组$A^n=\{a_u^n\}_{1\times|N_s|}$,链路带宽容量矩阵$\boldsymbol{B}^n=\{b_{jk}^n\}_{|N|\times|N|}$,虚拟链路成本矩阵$\boldsymbol{G}^n=\{g_{jk}^n\}_{|N|\times|N|}$。下面从$f$的第1个逻辑链路开始进行步骤5中的计算。

步骤5 设当前逻辑链路为h_i,判断当前逻辑链路是第几个逻辑链路。

1)h_i是第1个逻辑链路时,此时链路包含n_{str}和v_1。对于每个潜在可部署的服务节点$n\in N_s$进行赋值$A^n=A$,$\boldsymbol{B}^n=\boldsymbol{B}$。如果$v_1$可以部署到当前节点$n$(通过验证$A^n$中$n$的剩余节点资源、$v_1$的共享性、$v_1$的最大可布置实例数、$n$中已有的$v_1$实例数以及所有已有$v_1$实例承载的总流量大小),计算此时的部署成本$\vartheta$,若能共享到已有实例,则$\vartheta=0$;然后按式(17)生成$\boldsymbol{G}^n=\{g_{jk}^n\}_{|N|\times|N|}$,其中$\gamma$为虚拟单位时延成本(为考虑链路时延的影响,把链路时延也看作一种虚拟的成本开销,并入虚拟链路成本)。

$$g_{jk}^n=\begin{cases}\beta r_{\mathrm{SFC}}(f)+\gamma d_{\mathrm{tr}}(e_{jk}), & b_{jk}^n\geqslant r_{\mathrm{SFC}}(f)\\ \infty, & b_{jk}^n<r_{\mathrm{SFC}}(f)\end{cases} \tag{17}$$

使用Dijkstra算法计算n_{str}到n在$\boldsymbol{G}^n$下的最短路径以及最短路径累计虚拟链路成本,分别记作ρ和τ。如果$\tau=\infty$,只赋值$l^n=\infty$;否则记录$q^n=n$,$p^n=\rho$,$l^n=\tau+\vartheta$。更新A^n上节点n的剩余资源;更新$\boldsymbol{B}^n$上所属p^n链路上的剩余资源;更新A^n和$\boldsymbol{B}^n$后,根据p^n和式(17)再次更新$\boldsymbol{G}^n$。如果v_1不可以部署在节点n,只赋值$l^n=\infty$。设当前逻辑链路为下一个逻辑链路,并返回步骤5。

2)h_i不是第1个和最后一个逻辑链路时,链路包含v_{i-1}和v_i。对于每个服务节点$n\in N_s$,如果v_i可以部署到当前节点n,计算此时的部署成本ϑ。使用Dijkstra算法计算所有$m\in N_s$($l^m\neq\infty$)到n在$\boldsymbol{G}^m$下的最短路径以及最短路径累计虚拟链路成本,分别记作ρ^m和τ^m。令$m_{\min}=\arg\min_{m\in N_s}\{l^m+\tau^m+\vartheta\}$,如果$\tau^{m_{\min}}=\infty$,只进行赋值$l^n=\infty$;否则,记录$n^n=(n^{m_{\min}},n)$,$p^n=(p^{m_{\min}},\rho^{m_{\min}})$,$l^n=l^{m_{\min}}+\tau^{m_{\min}}+\vartheta$,并更新$A^n$,$\boldsymbol{B}^n$以及$\boldsymbol{G}^n$。如果$v_i$不可以部署在节点$n$,只进行赋值$l^n=\infty$。设当前逻辑链路为下一个逻辑链路,并返回步骤5。

3)h_i是最后一个逻辑链路时,此时链路包含$v_{|V_f|}$和n_{des}。使用Dijkstra算法计算所有$n\in N_s$($l^n\neq$

∞)到 n_{des} 在 $\boldsymbol{G}^n$ 下的最短路径以及最短路径累计虚拟链路成本，分别记作 ρ^n 和 τ^n。令 $n_{\min}=\arg\min_{n\in N_s}\{l^n+\tau^n\}$，$l=l^{n_{\min}}+\tau^{n_{\min}}$，$q=q^{n_{\min}}$，$p=(p^{n_{\min}},\rho^{n_{\min}})$，继续步骤 6。

步骤 6　若 $l=\infty$，则 f 部署失败；否则，检查路径 p 的时延是否小于等于 $t_{\text{SFC}}(f)$。若不满足，则 f 部署失败；若满足，f 部署成功，数组 n 中的各元素依次为 f 的 $v_1,v_2,\cdots,v_{|V_f|}$ 部署位置，路径 p 为 f 的路由路径。根据部署信息更新 A 和 B，转到步骤 7。

步骤 7　若还未部署完所有的 SFC 请求，则选取下一个请求以后返回步骤 2；若所有的 SFC 请求部署完毕，则算法结束。

2.2　算法的时间复杂度

步骤 1 中，假设使用排序算法的排序时间复杂度为 $\varphi(n)$，则步骤 1 的时间复杂度为 $\varphi(|F|)$。步骤 2 和步骤 3 的时间复杂度均为 $O(|N|^2)$。步骤 5 的时间复杂度为 $O(|H_f||N_s|^2|N|^2)$。结合步骤 2 ~ 步骤 5 可知，在最坏情况下部署单个 f 的时间复杂度为 $O(|H_f||N_s|^2|N|^2)$，因此，JSA 算法的时间复杂度为

$$\varphi(|F|)+O(|F|\max_{f\in F}|H_f||N_s|^2|N|^2)$$

需要注意，上述的时间复杂度是在最坏情况下所计算的时间复杂度上限。在绝大多数情况下，JSA 算法的运行时间要远小于这个上限。

3　JSA 算法性能

3.1　仿真参数设置与方式

在仿真实验中，底层物理网络的生成采用文献[14]中的随机网络生成方式。网络中节点计算资源和链路带宽容量分别以均匀分布 $U(500,600)$ 和 $U(1\,000,1\,200)$ 生成，并随机选取 30% 的节点设为转发节点。链路的时延正比于节点间距离。假设网络支持 5 种类型的 VNF，每种 VNF v 的 δ_v，$c_{\text{VNF}}(v)$ 和 α_v 分别以均匀分布 $U(1,1.3)$，$U(100,200)$ 及 $U(10,20)$ 生成，同时随机设置其切片间的共享性，每种 VNF 在各节点上可部署的最大实例数为{4,5,6}中的随机数。实验中，考虑 eMBB 切片、uRLLC 切片和 mMTC 切片 3 种类型的切片[5,7]。eMBB 切片上的 SFC 具有高带宽需求；uRLLC 切片上的 SFC 具有严格的时延要求；mMTC 切片上具有较多数量的 SFC，对带宽和时延的要求都比较宽松。属于 3 种切片的 SFC 请求数量比设置为1∶2∶3，每个 SFC 请求 f 的数据流量大小以均匀分布生成，最大容忍端到端时延根据式(18)生成。

$$t_{\text{SFC}}(f)=\xi D_{n_{\text{str}},n_{\text{des}}}\frac{|U_f|+1}{\min\limits_{f'\in F}|U_{f'}|+1}\tag{18}$$

其中：ξ 为时延因子，$D_{n_{\text{str}},n_{\text{des}}}$ 为 n_{str} 和 n_{des} 之间的最短时延。SFC 请求相关参数如表 1 所示。

表 1　SFC 请求相关参数

所属切片	数据流量	时延因子
eMBB	$U(80,90)$	$U(1.5,1.6)$
uRLLC	$U(40,50)$	$U(1.1,1.2)$
mMTC	$U(20,30)$	∞

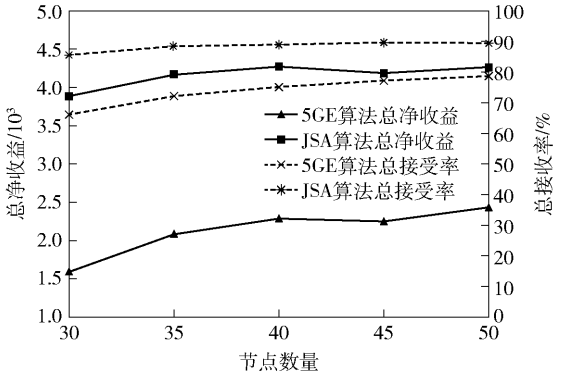
每个 SFC 请求 f 的长度 $|U_f|$ 是{3,4,5}中的随机数，同时 $m_{\text{SFC}}(f)$ 和 $p_{\text{SFC}}(f)$ 分别以均匀分布 $U(100,200)$ 和 $U(50,60)$ 生成。最后，β 设置为 0.25，JSA 算法中的 γ 设置适应 α_v 和 β，使 VNF 部署成本、带宽消耗成本以及虚拟时延成本在同一个数量级上。所有仿真在配置为 2.38 GHz AMD Ryzen 5 4500U CPU 和 16 GB 内存的笔记本电脑中进行。

仿真实验中，首先将 JSA 算法与网络切片 SFC 部署算法 5GE[7]（goodness-factor based embedding algorithm for the 5G network slices）进行对比。5GE 算法的思想是贪婪地将 VNF 嵌到节点上，从而使节点计算资源和链路带宽资源最小化；比较了 JSA 算法和 5GE 算法在不同节点规模和总 SFC 请求数下的总净收益、总 SFC 请求接受率以及各切片上的 SFC 请求接受率；最后，为进一步评价 JSA 算法求得解的质量，比较了 JSA 算法求得的解与 Cplex 优化器求解式(16)。Cplex 优化器是用于求解优化问题的求解器，其求得的解可认为是最优解。

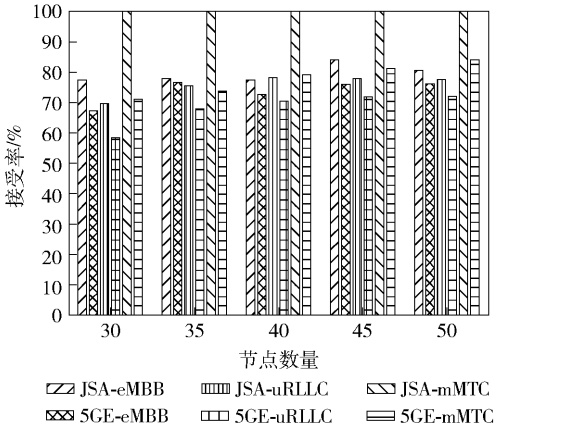
3.2　仿真结果分析

JSA 算法和 5GE 算法在相同 SFC 请求数($|F|=50$)，不同拓扑节点规模大小情况下的总净收益、总 SFC 请求接受率以及各切片上的 SFC 请求接受率如图 2 所示。每组实验均重复 50 次并进行了平均。图 2(a)显示，随着节点规模的增大，2 种算法的总净收益和总接受率也随着增大。这是因为更大的拓扑规模有更多可选择的路径，从而提高了总接受率，增加了总净收益。然而，从总净收益来看，JSA 算法的总净收益大约是 5GE 算法的 2 倍，从总接受率来看，JSA 算法的接受率在 86% ~90%，而 5GE 算法的接受率仅在 66% ~79%。图 2(b)示出

了2种算法在不同切片上的接受率,可以看出,在3种切片上SFC接受率的排序为mMTC > eMBB > uRLLC。mMTC切片上的SFC对网络的要求不高,故接受率最高。而uRLLC切片上的SFC具有较为严格的时延要求,故接受率最低。JSA算法在每种切片上的SFC接受率均高于5GE算法,这是因为在考虑共享性的同时JSA算法优先嵌入了长度较短的链路,嵌入的链路越长占用的带宽资源越多,而5GE算法相比带宽消耗更加考虑共享性。因此,在资源较为紧张的情况下,JSA算法的接受率更有保障。



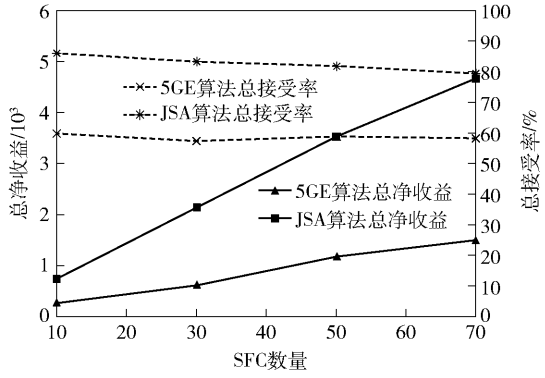
(a) 不同节点规模下的总净收益和总接受率



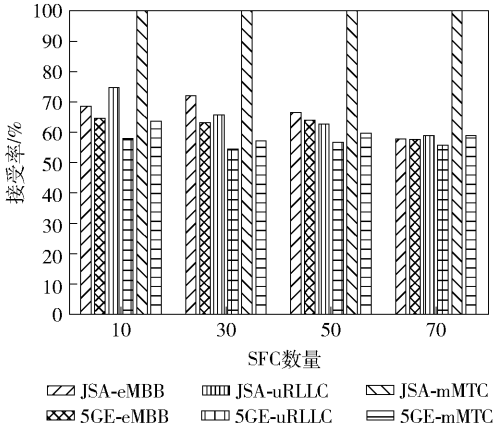
(b) 不同节点规模下各切片上的接受率

图2 JSA和5GE算法在不同节点规模下的比较

JSA算法和5GE算法在相同拓扑节点规模($|N|=20$),不同SFC请求数情况下的总净收益、总SFC请求接受率以及各切片上的SFC请求接受率如图3所示。每组实验均重复50次并进行了平均。从图3(a)可见,随着SFC请求数的增加,2种算法下的总接受率下降,而总净收益则逐渐增长。这是由于SFC请求数较小时,网络中的资源充足,而随着SFC请求数增大,网络中的资源会越来越紧张,总接受率会下降。但是由于接受一个SFC请求f的收益$m_{\mathrm{SFC}}(f)$大约是拒绝f所承担损失$p_{\mathrm{SFC}}(f)$的2倍,故在其接受率大于50%时,随着链路的增加其收益仍然会持续上升。JSA算法的总净收益和总接受率都要优于5GE算法。从总净收益来看,JSA算法大约是5GE算法的2倍;从总接受率来看,JSA算法的接受率为79% ~86%,而5GE算法的接受率为57% ~60%。同时,由图3(b)可知,JSA算法在每种切片上的SFC接受率均高于5GE算法。



(a) 不同SFC请求总数下的总净收益和总接受率

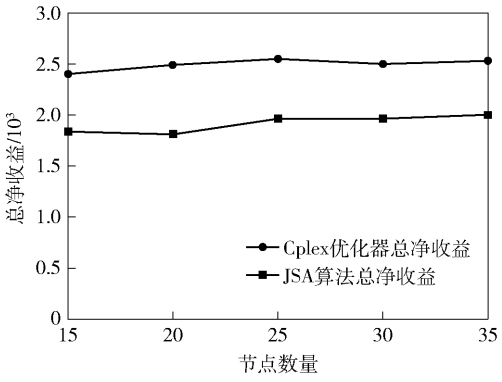


(b) 不同SFC请求总数下各切片上的接受率

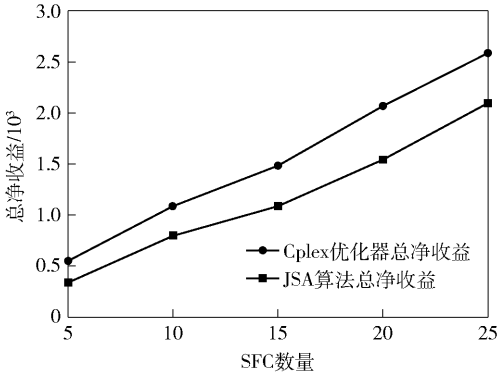
图3 JSA和5GE算法在不同SFC请求总数下的比较

最后,比较了JSA算法和Cplex优化器,以评估JSA算法所求得的解逼近最优解的程度,结果如图4所示。由于Cplex优化器求解需要花费较长的时间,选取了较小规模的拓扑和较小数量的SFC请求数。每组实验均重复20次并进行了平均。在SFC请求数为25,不同拓扑大小的情况下,JSA算法和Cplex优化器的总净收益如图4(a)所示。在拓扑节点数为30,不同SFC请求数的情况下,2种方案的总净收益如图4(b)所示。可以看出,JSA算法的总净收益已经非常接近Cplex优化器求出的最优总净收益,Cplex优化器最优总净收益是JSA算法的总净

收益 1.25 倍左右。



(a) 不同节点规模下的总净收益



(b) 不同SFC请求总数下的总净收益

图 4　JSA 算法和 Cplex 优化器的总净收益比较

4　结束语

研究了面向多样化需求的网络切片 SFC 部署，同时考虑了 VNF 共享和准入控制。建立了相关的数学模型，并提出了相应的网络切片 SFC 部署算法 JSA。仿真结果表明，所设计的 JSA 算法明显优于已有的算法并达到接近最优的性能。

参考文献：

[1] ZHANG H, LIU N, CHU X, et al. Network slicing based 5G and future mobile networks: mobility, resource management, and challenges[J]. IEEE Communications Magazine, 2017, 55(8): 138-145.

[2] QUINN P, NADEAU T. Problem statement for service function chaining: IETF technique report: RFC 7498 [R/OL]. (2018-12-20)[2021-07-01]. https://datatracker.ietf.org/doc/rfc7498/.

[3] LAGHRISSI A, TALEB T. A survey on the placement of virtual resources and virtual network functions[J]. IEEE Communications Surveys and Tutorials, 2018, 21(2): 1409-1434.

[4] LI H, KONG Z, XIANG W, et al. Slice-based service function chain embedding with multiple factors[C]// 2021 IEEE Globecom Workshops (GC Wkshps). Piscataway, NJ: IEEE Press, 2021: 1-6.

[5] SERIES M. Minimum requirements related to technical performance for IMT-2020 radio interface (s): ITU-R report: M. 2410-0 [R/OL]. (2017-11-16)[2021-07-01]. https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2410-2017-PDF-E.pdf.

[6] PAPAGIANNI C, PAPADIMITRIOU P, BARAS J S. Rethinking service chain embedding for cellular network slicing[C]// 2018 IFIP Networking Conference (IFIP Networking) and Workshops. Piscataway, NJ: IEEE Press, 2018: 1-9.

[7] TRUONG-HUU T, MOHAN P M, GURUSAMY M. Service chain embedding for diversified 5G slices with virtual network function sharing[J]. IEEE Communications Letters, 2019, 23(5): 826-829.

[8] NEJAD M A T, PARSAEEFARD S, MADDAH-ALI M A, et al. vSPACE: VNF simultaneous placement, admission control and embedding[J]. IEEE Journal on Selected Areas in Communications, 2018, 36(3): 542-557.

[9] ZHANG N, LIU Y F, FARMANBAR H, et al. Network slicing for service-oriented networks under resource constraints[J]. IEEE Journal on Selected Areas in Communications, 2017, 35(11): 2512-2521.

[10] CHEN W K, LIU Y F, DOMENICO A D, et al. Optimal network slicing for service-oriented networks with flexible routing and guaranteed E2E latency[J]. IEEE Transactions on Network and Service Management, 2021, 18(4): 4337-4352.

[11] XIE Y, WANG S, DAI Y. Revenue-maximizing virtualized network function chain placement in dynamic environment[J]. Future Generation Computer Systems, 2020, 108: 650-661.

[12] ROST M, SCHMID S. Charting the complexity landscape of virtual network embeddings[C]// 2018 IFIP Networking Conference (IFIP Networking) and Workshops. Piscataway, NJ: IEEE Press, 2018: 1-9.

[13] DIJKSTRA E W. A note on two problems in connexion with graphs[J]. Numerische Mathematik, 1959, 1(1): 269-271.

[14] WAXMAN B M. Routing of multipoint connections[J]. IEEE Journal on Selected Areas in Communications, 1988, 1(3): 286-292.