网络边缘资源高效复用的延迟感知 VNF 链部署

摘要

随着对低延迟网络服务的需求日益增长，移动边缘计算 (MEC) 作为一种新兴范式，展示了其在为用户提供接近的服务器资源和处理能力方面的优势。这种方法依赖于网络功能虚拟化（NFV）技术，允许网络服务以灵活的虚拟网络功能（VNF）链的形式部署在边缘服务器上。然而，在资源有限的网络边缘环境中，如何有效地部署延迟敏感的 VNF 链，同时确保服务质量和资源效率，依然是一个充满挑战的问题。

在本研究中，文章专注于边缘服务器资源和物理链路的共同优化，以满足延迟限制的需求。具体而言，文章将 VNF 链的部署问题形式化为一个混合整数线性规划（MILP）模型，旨在最小化总体资源消耗。为此，文章设计了一种创新的两阶段延迟感知 VNF 部署方案。首先，文章应用了一个基于约束的深度优先搜索算法（CDFSA）来选择最优路径。接着，通过一个基于路径的贪婪算法（PGA），文章实现了 VNF 的高效分配，尽可能地重用现有 VNF 实例。

关键词： VNF；边缘服务器

1 引言

近年来，随着移动设备的普及，来自移动用户的服务需求急剧增加。特别是对于虚拟现实、物联网和可穿戴设备等应用，用户对高速、低延迟的网络服务需求越来越强烈。为了应对这一趋势，移动边缘计算 (MEC) 作为一种新兴技术，通过在网络边缘提供 IT 服务环境和云计算能力，已经成为降低端到端延迟、提升用户体验的关键技术。此外，基于网络功能虚拟化 (NFV) 的概念，网络服务可以灵活地通过在边缘服务器部署的虚拟网络功能 (VNF) 链来提供。然而，由于网络边缘资源的有限性，如何在保证延迟和资源效率的前提下有效地部署 VNF 链仍然是一个挑战。

考虑到移动边缘计算的兴起以及对低延迟网络服务的日益增长的需求，研究如何在资源有限的网络边缘环境中有效部署 VNF 链，具有很高的实用价值和研究意义。特别是，对于如何在满足延迟限制的同时最大化资源利用率的问题，成为这一领域的关键课题。

选题的意义在于，它不仅有助于理解和解决网络边缘计算中的实际问题，即如何有效利用有限的资源来满足日益增长的服务需求，同时也对实现更加高效和可靠的网络服务具有重要价值。此外，这一研究可以为未来的网络设计和优化提供理论指导和技术参考，特别是在快速发展的 5G 和物联网技术背景下。

2 相关工作

2.1 VNF 放置和路由概述

多数现有研究都集中于 VNF 放置和路由问题，目标不同并提出了相应的解决方案。例如，Fei 等人 [1] 研究了 VNF 提供商如何动态扩展 VNF 部署并重新路由流量需求。他们首先使用在线学习方法预测流量流，并随后提出了一个新的 VNF 实例分配和服务链重定向的联合在线算法。Sang 等人 [2] 则专注于解决最小化网络中 VNF 实例总数的问题，他们设计了两种贪婪算法来解决这个问题，这两种算法都被证明是渐进最优的。

还有一些工作旨在同时优化计算资源和带宽资源的消耗。Cohen 等人 [3] 处理了物理网络中的 VNF 放置问题，旨在最小化功能的总设置成本和客户端与服务器之间的距离成本。他们提供了一个接近最优的近似算法，其性能经过理论证明。然而，他们假设每个服务需求只需要单个 VNF 而非服务链。在 [4] 中，作者探讨了链路和服务器资源之间的关系，并研究了 VNF 放置和路径选择的联合问题，以更好地利用网络和服务器资源，其目标是最大化被接受需求的总量而非优化资源利用，并忽略了服务延迟要求。

2.2 网络边缘的 VNF 放置和路由

以上工作都集中在云场景中，由于网络边缘的特点，如资源短缺和多用户接入点，这些方法不能直接应用于网络边缘。目前也有一些工作着手解决网络边缘的 VNF 放置和路由问题。例如，[5] 研究了为移动边缘计算（MEC）联合优化接入网络选择和服务放置的问题。他们设计了一个在线框架来提高服务的 QoS，但每项服务仅包含单个 VNF。张等人 [6] 专注于 5G 服务定制网络切片中的 VNF 放置问题，他们提出了一个考虑 VNF 干扰的通用 5G 网络切片框架，并提出了一种自适应干扰感知启发式（AIA）方法来自动在每个 5G 网络切片中放置 VNF。然而，他们并没有考虑资源优化。Cziva 等人 [7] 延续 [8] 的工作，研究了单 VNF 放置和路由问题的延迟优化，但没有考虑 VNF 实例的可重用性。此外，他们强调了网络边缘的有限资源供应，但没有考虑资源优化。

据文章所知，尚未有研究调查网络边缘在资源优化和延迟限制条件下的高效 VNF 链部署问题。因此，在本文中，文章的目标是在提供延迟保证的同时，同时重用服务器资源并减少带宽消耗。

3 本文方法

3.1 网络模型

文章将网络建模为一个无向图 $G = (U, D, E)$，其中 U, D 和 E 分别表示用户集、硬件服务器集（包括所有配备了计算能力的网络边缘设备，如接入点、交换机、路由器等）以及服务器之间的物理链接集合。每个服务器 $d_j \in D$ 都有一个资源容量 C_j，并且每个链路 $e(u, v) \in E$ 都有一个带宽容量 $B(u, v)$。

假设不同的用户有不同的服务需求，每个需求都需要一个由不同 VNFs 组成的服务链，文章用 N 和 S 来表示不同类型的 VNFs 和不同服务链集合，对于每个用户 $u_i \in U$，文章用 $S_i \in S$ 来表示请求的服务链，并且 $|S_i|$ 是服务链的长度（即 VNFs 的数量）。接下

表 1. 主要符号定义。

符号	定义
$G=(U,D,E)$	网络图谱
S	VNF 服务链集合
$\epsilon(u,v)$	链接 (u,v) 上可用的带宽资源
$B_{(u,v)}$	链接 $\epsilon(u,v)\in E$ 的带宽容量
$l_i(u,v)$	使用 S_i 时链接 $\epsilon(u,v)$ 的延迟
C_j	硬件设备 $d_j\in D$ 的资源容量
R_n	类型 n VNF 实例的资源需求
W_n	类型 n VNF 实例的处理能力
β_i	$S_i\in S$ 的最大延迟约束
(ϕ_i,t_i,r_i)	服务链 S_i 的三元组: $\phi_i\subset D$ 是 S_i 的可用接入点集合; $t_i\in d_j$ 表示 S_i 的目的地, r_i 是 S
$X^l_{i,j}$	服务链 S_i 的第 i 个 VNF 是否由服务器 d_j 提供服务
$Z^{U'}_{i,(u,v)}$	S_i 的第 l 个和第 $l+1$ 个 VNF 之间的子链是否在链接 $\epsilon(u,v)$ 上路由
$T^l_{i,n}$	服务链 $S_i\in S$ 的第 l 个 VNF 是否需要类型 n 的 VNF
$Y_{j,n}$	服务器 d_j 中类型 n VNF 实例的数量

来，文章假设 $s^l_i\in S_i$ 是服务链 S_i 的第 l 个 VNF，而 $\Phi^{l,l'}_i$ 是 S_i 中第 l 和 l' 之间的子链 $(1\le l<l'\le|S_i|,\Phi^{l,l'}_i\subseteq S_i)$。具体来说，文章引入 $T^l_{i,n}$ 来指示服务链 S_i 的第 l 个 VNF 是否为类型 n。在现实中，用户可能需要不止一个服务链，这可以被视为同一物理位置的不同用户需要不同的服务链。

文章进一步定义了一个三元组 (ϕ_i,t_i,r_i) 来表示服务链 S_i 的来源、目的地和流量率。具体来说，ϕ_i 是 S_i 的一组可用接入点集合，这些接入点归属于不同的边缘云，它们与用户 u_i 的位置非常接近，这意味着 S_i 可以通过接入点集合 ϕ_i 中的任何一个 $d_{i,k}\in\phi_i$ 访问网络。由于每个 $d_{i,k}\in\phi_i$ 对应一个服务器 $d_j\in D$，文章简单地用 α_j 表示接入点 $d_j\in D$ 的处理能力，如图 1 所示。最后，假设 P 是服务链 S_i 可以路由的物理路径集合，每个路径 $P_i\in P$ 由一系列服务器之间的物理链接组成。

3.2 问题描述

文章假设所有 VNF 可以被不同的服务链共享，这样它们的剩余处理能力可以得到最大限度的重用。文章用 $Y_{j,n}$ 表示服务器 $d_j\in D$ 中类型为 n 的 VNF 实例的数量。每当一个服务链需要来自 d_j 的类型 n 的 VNF 时，如果有可重用的类型 n VNF 实例，$Y_{j,n}$ 保持不变；否则文章将在 d_j 中启动一个新的类型 n 实例，并且 $Y_{j,n}$ 变为 $Y_{j,n}+1$。启动一个类型 n VNF 实例时，它需要 R_n 设备资源。为了保证每个服务器 d_j 的总资源消耗不超过资源容量 C_j，文章有:

$$\sum_{n\in N}Y_{j,n}R_n\le C_j,\quad\forall d_j\in D. \tag{1}$$

文章使用 $Z'_{i,l}(u,v)$ 来表示服务链 S_i 的子链 $\Phi'_{i,l}$ 是否穿越链路 $e(u,v)$。同样的方式，所

有沿着链路 $e(u,v) \in E$ 的流量之和不应超过 $B(u,v)$。这可以表示为：

$$\sum_{\substack{s_i^l \in S_i, \\ 1 \le l < l' \le |S_i|}} Z'_{i,l}(u,v) \cdot r_i \le B(u,v), \quad \forall e(u,v) \in E, 1 \le l < l' \le |S_i|. \tag{2}$$

设 W_n 是每个类型-n VNF 实例的处理能力。为了确保任何服务器 $d_j \in D$ 上的所有 VNF 实例都能处理流量，文章有：

$$\sum_{s_i^l \in S_i} T_{i,n}^l \cdot X_{i,j}^l \cdot r_i \le Y_{j,n} \cdot W_n, \quad \forall n \in N, d_j \in D, \tag{3}$$

其中 $X_{i,j}^l$ 表示服务链 S_i 的第 l 个服务是否由服务器 d_j 提供。

如上所述，文章假设每个服务链 S_i 携带单一流量，即每个 $s_i^l \in S_i$ 只由一个服务器服务。那么文章对 $X_{i,j}^l$ 有如下约束：

$$\sum_{d_j \in D} X_{i,j}^l = 1, \quad \forall S_i \in S, \forall l \in [1, |S_i|]. \tag{4}$$

接下来，对于每个服务链 S_i，如果它可以被成功接受，文章需要确保总延迟不会超过其延迟限制 β_i。这里，文章考虑了两种类型的延迟，一种是服务链在 AP $d_j \in \phi_i$ 中的排队延迟，另一种是传输延迟和服务链选择的链路 (u,v) 的传播延迟。为了分析 AP 中用户的排队延迟，文章将每个 AP 建模为一个 M/M/1 队列。通过应用 Little 定律，文章可以计算 AP d_j 中的平均排队延迟如下：

$$q_j = \frac{1}{\alpha_j - \sum_{S_i \in S} A_{i,j} \cdot r_i} \tag{5}$$

其中 $A_{i,j}$ 表示服务链 S_i 是否选择 AP $d_j \in \phi_i$ 来访问边缘网络。然后文章有：

$$\sum_{d_j \in D} A_{i,j} \cdot q_j + \sum_{(l,l'):(l,l') \in \Phi_i} Z'_{i,l}(u,v) \cdot r_i \le \beta_i, \quad \forall e(u,v) \in E, S_i \in S \tag{6}$$

由于 S_i 每次只能选择一个 AP，因此文章对 $A_{i,j}$ 有以下约束：

$$\sum_{d_j \in D} A_{i,j} = 1, \quad \forall S_i \in S \tag{7}$$

最后，文章需要保证所有在 S_i 中的 VNF 都被表示如下：

$$\sum_{n \in N} \sum_{d_j \in D} X_{i,j}^l = |S_i|, \quad \forall S_i \in S \tag{8}$$

文章的目标是共同最小化服务器和链路上的资源消耗。由于服务器 d_j 上的设备资源消耗为 $\sum_{n \in N} Y_{j,n} R_n$，链路 (u,v) 上的总带宽消耗为 $\sum_{S_i \in S} \sum_{(l,l') \in \Phi_i} Z'_{i,l}(u,v) r_i$。那么联合资源消耗优化问题可以表述如下：

$$\min \sum_{d_j \in D} \sum_{n \in N} Y_{j,n} \cdot R_n + \sum_{(u,v) \in E} \sum_{S_i \in S} \sum_{(l,l') \in \Phi_i} Z'_{i,l}(u,v) \cdot r_i \tag{9}$$

3.3 解决方案概述

由于 VNF 部署问题被证明是 NP-hard，即使在计算上求解最优解的成本是昂贵的，它也是服务链部署的关键步骤。通过假设一个服务链的部署没有未来到达的知识，文章的问题侧重于如何选择一条服务链，并以最小化资源消耗的方式进行路由。为了解决这个问题，文章将原始的单服务链问题分解为两个子问题，即首先选择路径然后分配 VNFs。具体来说，在第一个子问题中，对于服务链 $S_i = (\phi_i, t_i, r_i)$，文章找到所有从 ϕ_i 到目的地 t_i 的可行路径，这些路径称为预路由路径，遵循链路的带宽限制和服务链的延迟约束。之后，为了解决第二个子问题，文章将路径放入一个被称为 $\mathcal{P}_i$ 的路径集合中。对于每个路径 $p \in \mathcal{P}_i$，文章选择沿着路径具有最大重用数量的 VNFs 来分配 VNFs。接下来，文章选择一个在所有路径中具有最小资源消耗的分配方案。通过这种方式，文章的两阶段策略获得的单服务链解决方案是近似最优的。最后，文章将所有到达服务链的总资源消耗加起来。

3.4 路径选择

对于第一个子问题，文章构造一个新的带权无向图。文章用 $l_{i,k}$ 和 $r_{i,k}$ 表示链路的延迟和剩余带宽，以及每个链路中服务链 S_i 可以容忍的延迟和剩余带宽。换言之，给定图 G' 和服务链 S_i 带有延迟限制，文章的目标是找到所有从 ϕ_i 到 t_i 的可行路径，且不超过服务链 S_i 在 G' 中的带宽容量限制。

为了描述网络图，文章构建一个新的加权混合多图 $H = (D', E'; w)$，其中加权值由 $l_{(u,v)}$ 来表示。下面，文章详细描述在三个步骤中构建图 H：

步骤 1：服务器派生。首先，文章添加所有可用的 APs $d_j \in \phi_i$ 到 H。然后文章假设 $s_{i,l} \in S_i$ 最小的资源消耗为 $R_{min,i}$。如果服务器 $d_j \in D$ 上的剩余资源多于 $R_{min,i}$，或者有任何可被服务链 S_i 的 VNF $s_{i,l} \in S_i$（$l \in \{1, \ldots, |S_i|\}$）重用的可用 VNF 实例，则文章添加服务器 d_j 到 H。

步骤 2：只要服务器有足够的剩余资源，链中的 VNF 就有可能由上游 VNF 已服务的服务器提供服务，这意味着路由路径中可能存在环路。同时，如果带宽容量足够，图的每个链接都可以被遍历多次。由于文章的目标是在网络边缘的延迟约束下最大限度地减少资源消耗，因此服务链一次又一次地经过某个链路是不合理的。因此，文章假设服务链最多遍历同一链路两次（在不同方向）。

步骤 3：分离服务器修剪。至此，如果 H 中存在没有链接的服务器，则将其删除即可。

约束深度优先搜索算法（CDFSA）。给定图 H，文章将继续使用 CDFSA 来计算服务链 S_i 的预路由路径集 $\mathcal{P}_i$，如算法 1 所示。首先，文章使用邻接矩阵 M 来存储有向和无向链接在 H 中的信息。具体来说，文章使用 0, 1 和 2 来表示无链接、无向链接和有向链接，分别对应。对于节点对 u 和 v，文章有：

$$M[u][v] = M[v][u] = \begin{cases} 0, & \text{如果没有链接,} \\ 1, & \text{如果有无向链接,} \\ 2, & \text{如果有两个有向链接.} \end{cases} \tag{10}$$

此外，文章还使用另一个加权值矩阵 E 来描述链路的延迟，其中每个元素 $E[u][v]$ 表示链路 $e(u, v)$ 的延迟为 $l_{i(u,v)}$。然后，文章使用一个叫做深度优先搜索（DFS）的递归遍历算法

来选择可行路径。具体来说，文章首先选择其中一个 AP $d_j \in \phi_i$，并将 AP d_j 上的排队延迟加到总延迟中。考虑当前服务器是 u，对于每个可用的链路 $e(u,v)$，只要总延迟仍在限制之内，文章就将其添加到路径中。如果链路 $e(u,v)$ 成功地被添加到路径中，文章就应该相应地更新邻接矩阵 M。在 $M[u][v]=1$ 的情况下，即链路 $e(u,v)$ 是无向的并且只能被遍历一次，文章更新 $M[u][v]=0$ 和 $M[v][u]=0$，这意味着链路 $e(u,v)$ 和 $e(v,u)$ 将从图 H 中被移除。在另一种情况下，如果 $M[u][v]=2$，通过设置 $M[u][v]=0$，文章从图中移除有向链路 $e(u,v)$，并保持有向链路 $e(v,u)$ 不变。通过递归调用算法 1 中的函数 Selection_Paths 直到到达节点 t_i，文章可以获得服务链 S_i 的预路由路径集 $\mathcal{P}_i^A$。最终，如果不存在从 $d_j \in \phi_i$ 到 t_i 的任何可行路径，服务链 S_i 将不被选择。

Algorithm 1 约束深度优先搜索算法（CDFSA）

初始化邻接矩阵 M 和权重值矩阵 E，初始化一个栈 cp 来存储路径的节点。
for 每个 $d_j \in \phi_i$ **do**
    $u \leftarrow d_j$;
    计算 d_j 的排队延迟 q_j 如等式 (5)
    $L_t \leftarrow q_j$;
    function SELECTION_PATHS(u, t_i, L_t, β_i)
        将当前节点 u 压入栈 cp。
        if 当前节点 $u ==$ 终点 t_i **then**
            从 cp 中提取路径并将其放入集合 P_i^A。
            return
        end if
        for 每个节点 v 在图 H 中 **do**
            if $M[u][v] > 0$ **and** $L_t + E[u][v] < \beta_i$ **then**
                if $M[u][v] == 2$ **then**
                    $M[u][v] \leftarrow 0$;
                else if $M[u][v] == 1$ **then**
                    $M[u][v] \leftarrow 0$; $M[v][u] \leftarrow 0$;
                end if
                Selection_Paths($v, t_i, L_t + E[u][v], \beta_i$);
                恢复站点;
            end if
        end for
    end function
end for

3.5 VNF 分配

给定一个单独的服务链 S_i 和一组预路由路径 $\mathcal{P}_i^A$，文章的下一个目标是选择一个分配方案，使资源消耗最小化。由于文章无法预见哪条路径将产生最优解，文章轮流挑选 $\mathcal{P}_i^A$ 中的

Algorithm 2 基于路径的贪婪算法 (PGA)

输入：服务链 S_i，流速 r_i，预路由路径 p（长度为 k），$W_{j,n}$，C_j
输出：路径向量 B，服务器消耗 C_s，重用数量 p_r
初始化：$l = 0$，$j = 0$，$C_s = 0$，$p_r = 0$，$B = \emptyset$
查找服务链 S_i 中每个 VNF 类型 n_l 及其资源消耗 $R_{l,i}$
查找路径 p 中每个服务器 d_j，剩余资源容量 C_j 和所有 VNF 实例的剩余处理能力
根据方程 (11) 计算矩阵 A
function ASSIGNMENT(l, j, C_s, p_r)
    for $y = j$ 到 k **do**
        标记是否有类型-N 服务器供 s_l 使用
        if d_y 是 s_l 的类型-N 且 $A[l][y] == 0$ 且 $C_y \geq R_{l,i}$ **then**
            标记 $\leftarrow 1$
            资源消耗：$s_l : C_s^1 \leftarrow C_s + R_{l,i}$
            更新重用数量：$p_r^1 \leftarrow p_r$
            if 存在尚未分配的 VNF **then**
                调用 assignment$(l + 1, y, C_s^1, p_r^1)$
            end if
            更新服务器 p 中的资源容量
            标记 $\leftarrow 1$
        end if
    end for
    记录 y 到部署向量 B
    选择具有最小 C_s 的分配方案
end function

预路由路径 p，计算每条路径的最优分配方案。最后，通过比较所有局部最优方案，文章选择所有预路由路径中全局最优的方案。

对于选定的预路由路径 $p \in \mathcal{P}_i^A$，设 k 为路径的长度（链接的数量），路径上相应的服务器数量为 $k+1$。因此，文章可以定义路径 p 为一系列服务器节点 $\{sp_0, sp_1, ..., sp_k\}$ （$sp_i \in D, i \in \{0, 1, ..., k\}$）。由于路径 p 中可能存在环路，不同的服务器节点 sp_k 可能对应同一个服务器 $d_j \in D$。对于每个服务器 $d_j \in D$，文章应确保 d_j 中所有 VNFs 的资源消耗不超过资源容量，且处理能力能够处理流量。此外，保持服务链中 VNFs 的顺序是必要的。不幸的是，分配子问题可以归约为一个装箱问题，这在 NP-hard 问题中是已知的。

基于路径的贪婪算法（PGA）。为了解决这个问题，文章提出了一个称为 PGA 的近最优算法。PGA 的关键思想是顺序分配服务链 S_i 的 VNFs 到路径 p 上的节点，同时尽可能多地重用可用的 VNF 实例，并考虑服务器资源和 VNF 处理能力的限制。

首先,文章解释如何计算最大数量的可重用 VNF 实例。给定一个服务链 $S_i = \{s_0, s_1, ..., s_{|S_i|-1}\}$ 和一个预路由路径 $p = \{sp_0, sp_1, ..., sp_k\}$ 作为输入，文章首先构建一个新的 $|S_i| \times (k+1)$ 矩阵 A 来描述 S_i 和 p 之间的关系。矩阵 A 的每个元素 $A[l][j]$ 表示 $s_l \in S_i$ 是否能重用服务器节点 sp_j 上的 VNF 实例。文章用 $W_{j,n}$ 表示设备 d_j 上剩余的类型-n VNF 实例处理能力。如果 s_l 是类型 n_l 并且服务器节点 sp_j 对应于服务器 d_j，那么 $A[l][j]$ 可以如下计算：

$$A[l][j] = \begin{cases} 1 & \text{如果} W(d_j, n_l) - r_i \geq 0 \\ 0 & \text{否则.} \end{cases}$$

此外，如果 VNF $s_l \in S_i$ 需要一个新实例，文章假设其资源消耗为 $\hat{R}_i$。

保持 VNFs 的顺序非常重要。文章在算法 2 中使用递归函数 assignment 来详细说明操作。对于 S_i 中的每个 s_l，有两种服务器可以被分配：服务器有可重用的 VNF 实例，被标记为类型-R 和服务器没有可重用的 VNF 实例但有足够资源至少启动一个新 VNF 实例，被标记为类型-V。在每个递归级别中，有四种情况：1) 文章首先找到类型-R 并终止 s_l 的循环，这将是 s_l 的最佳分配位置；2) 文章首先找到类型-V，然后是类型-R；3) 文章只找到类型-V；4) 文章找不到任何类型，即没有地方可以分配 s_l，因此服务链 S_i 将被拒绝。注意，在递归的每个级别，文章只需要为 s_l 的分配尝试最多两个可选节点。因此，对于每个 s_l，文章调用递归函数最多两次，这显著减少了 PGA 的计算复杂性。

对于每个预路由路径 $p \in \mathcal{P}_i^A$，文章只能得到一个局部最优分配解决方案。为了得到全球最优解，文章首先计算路径 p 的总资源消耗，方法是将最小的服务器资源消耗和带宽消耗（路径 p 的长度和 S_i 的流量率的乘积）加在一起。然后，文章比较每个预路由路径的总资源消耗，并选择全球最优解作为文章的最终结果。这样，通过两阶段策略获得的解决方案仍然是最优的。

4 复现细节

4.1 与已有开源代码对比

本文并没有开源的代码，因此未引用本文的相关代码。

4.2 实验设置

通过进行模拟，以评估复现算法在真实世界网络拓扑之上的性能。我在互联网拓扑动物园 [9] 中的几种不同规模的网络拓扑上进行了实验：例如 Bellsouth（51 个节点和 66 个链路）、Cogentco（197 个节点和 245 个链路）和 Kdl（754 个节点和 899 个链路）来模拟三种不同的网络规模。根据复现的论文，将网络中的节点配置为 [0, 200] 个单位的剩余可用资源容量。为了表征每个服务器的当前配置，文章随机生成一组可重用的 VNF，其中包含从 20 个 VNF 类型中挑选的 0 到 8 个 VNF。文章进一步分配一个随机数令服务器之间每条链路的带宽容量范围为 0 到 1000 Mbps。

服务链。文章为每个 AP 配置了 100 Mbps 到 200 Mbps 的处理能力。此外，文章通过从 20 种 VNF 类型中随机挑选 1 到 6 个 VNF 来生成服务链，每种类型的 VNF 实例需要 [20, 50] 单位的 CPU 和内存服务器资源。该链的流量范围从 30 Mbps 到 60 Mbps，截止时间随机分布在 [30, 80] ms [6] 中。

5 实验结果分析

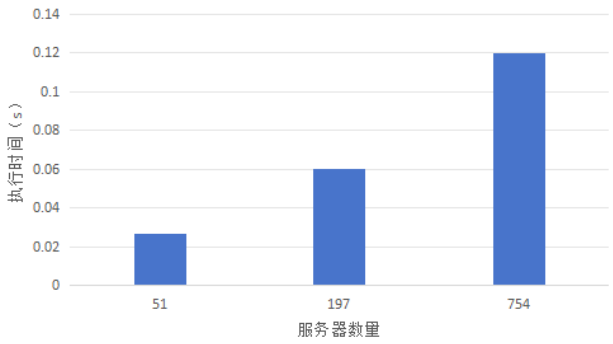


图 1. 不同服务器数量的执行时间

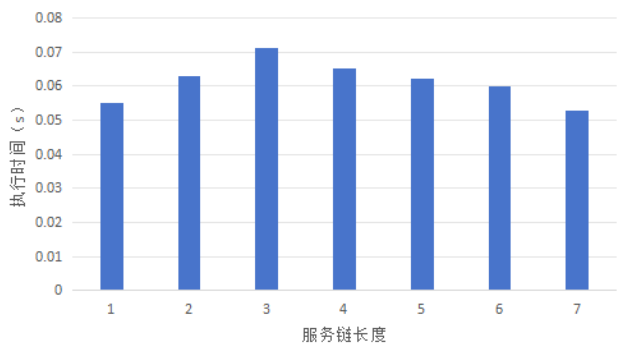


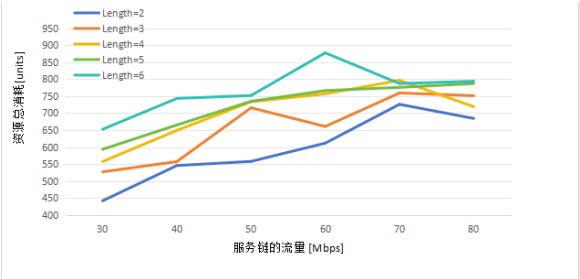
图 2. 不同服务链长度的执行时间

从前面的分析可知，文章提出的算法的时间复杂度与服务器的数量和服务链的长度相关。

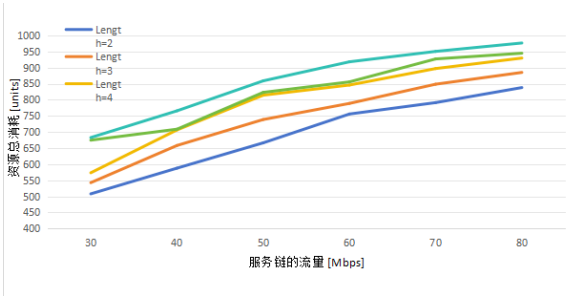
首先在六种不同尺度的现实网络拓扑中进行了实验。如图 1 所示，随着服务器数量从 51 到 754 不等，执行时间也相应增加。具体来说，复现出来的算法即使在 754 个节点中，执行时间也只需要 0.12 秒。与论文结论相比，仿真结果的趋势相似，但是执行时间有差异。这可能是因为论文并没有给出具体的数据集，数据集导致了执行时间的差异。

接下来探究不同服务链长度的执行时间。如图 2 所示，仿真结果表明，执行时间不随服务链长度线性变化，这可能是因为服务链越长，第一阶段可获得的可行路径越少。仿真的结果与论文结论相似。

下面对探究复现算法的总资源消耗量。主要考虑影响服务器资源消耗的服务链长度和影响带宽资源消耗的服务链流量。首先，在图 3 中，分别描绘了 Bellsouth 网络 (51 个节点) 和 Cogentco 网络 (197 个节点) 的服务链长度和服务链速率的影响关系。正如预期的那样，总资源消耗不仅随着服务链的长度而增加，而且随着服务链的流量而增加。由于数据集的不同，与论文相比，无论是 Bellsouth 还是 Cogentco 网络所消耗的总资源都要多。但是总的来说，仿真结果的趋势一致。

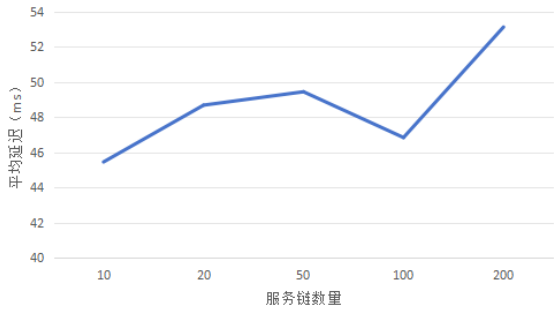


(a) Bellsouth（51 个节点）

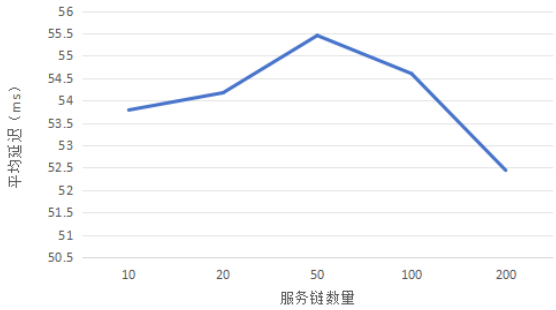


(b) Cogentco（197 个节点）

图 3. 服务链长度与服务链速率的影响关系



(a) Bellsouth（51 个节点）



(b) Cogentco（197 个节点）

图 4. 服务链长度与平均延迟的影响关系

最后探究复现算法的在不同服务链长度下平均延迟。如图 4 所示，服务链的平均延迟在 40 ms 到 55 ms 之间。这与原文的仿真结果相似。

6 总结与展望

这篇文研究了 VNF 链在网络边缘的部署问题，在提供延迟保证的同时复用服务器资源，减少带宽消耗。首先揭示了服务器和链接的资源利用之间的困境。然后文章将问题形式化为一个 MILP 模型，并设计了一个两阶段的解决方案来解决它。在第一阶段，文章使用 CDFSA 来选择所有的预路由路径。给定这些路径，在第二阶段，文章设计 PGA 以最小的资源消耗分配 vnf。最后，对真实的网络拓扑进行了仿真。总的来说，文章所提的算法对减少服务器资源消耗具有重要的意义。

在复现的过程中，由于数据集并未给出，难以复现出良好的结果，将来可以设置更好的数据集进行实验。

参考文献

[1] Xincai Fei, Fangming Liu, Hong Xu, and Hai Jin. Adaptive vnf scaling and flow routing with proactive demand prediction. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 486–494, 2018.

[2] Yu Sang, Bo Ji, Gagan R. Gupta, Xiaojiang Du, and Lin Ye. Provably efficient algorithms for joint placement and allocation of virtual network functions. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, 2017.

[3] Rami Cohen, Liane Lewin-Eytan, Joseph Seffi Naor, and Danny Raz. Near optimal placement of virtual network functions. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1346–1354, 2015.

[4] Tung-Wei Kuo, Bang-Heng Liou, Kate Ching-Ju Lin, and Ming-Jer Tsai. Deploying chains of virtual network functions: On the relation between link and server usage. *IEEE/ACM Transactions on Networking*, 26(4):1562–1576, 2018.

[5] Bin Gao, Zhi Zhou, Fangming Liu, and Fei Xu. Winning at the starting line: Joint network selection and service placement for mobile edge computing. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 1459–1467, 2019.

[6] Qixia Zhang, Fangming Liu, and Chaobing Zeng. Adaptive interference-aware vnf placement for service-customized 5g network slices. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 2449–2457, 2019.

[7] Richard Cziva, Christos Anagnostopoulos, and Dimitrios P. Pezaros. Dynamic, latency-optimal vnf placement at the network edge. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 693–701, 2018.

[8] Richard Cziva and Dimitrios P. Pezaros. Container network functions: Bringing nfv to the network edge. *IEEE Communications Magazine*, 55(6):24–31, 2017.

[9] Simon Knight, Hung X. Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.