Joint Resource Management and Flow Scheduling for SFC Deployment in Hybrid Edge-and-Cloud Network

1st Yingling Mao
Dept. of Electrical and Computer Eng.
Stony Brook University
New York, USA
yingling.mao@stonybrook.edu

2nd Xiaojun Shang
Dept. of Electrical and Computer Eng.
Stony Brook University
New York, USA
xiaojun.shang@stonybrook.edu

3rd Yuanyuan Yang
Dept. of Electrical and Computer Eng.
Stony Brook University
New York, USA
yuanyuan.yang@stonybrook.edu

IEEE INFOCOM 2022 - IEEE Conference on Computer Communications | 978-1-6654-5822-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/INFOCOM48880.2022.9796884

***Abstract*—Network Function Virtualization (NFV) migrates network functions from proprietary hardware to commercial servers on the edge or cloud, making network services more cost-efficient, manage-convenient, and flexible. To facilitate these advantages, it is critical to find an optimal deployment of the chained virtual network functions, i.e. service function chains (SFCs), in hybrid edge-and-cloud environment, considering both resource and latency. It is an NP-hard problem. In this paper, we first limit the problem at the edge and design a constant approximation algorithm named chained next fit (CNF), where a sub-algorithm called double spanning tree (DST) is designed to deal with virtual network embedding. Then we take both cloud and edge resources into consideration and create a promotional algorithm called decreasing sorted, chained next fit (DCNF), which also has a provable constant approximation ratio. The simulation results demonstrate that the ratio between DCNF and the optimal solution is much smaller than the theoretical bound, approaching an average of 1.25. Moreover, DCNF always has a better performance than the benchmarks, which implies that it is a good candidate for joint resource and latency optimization in hybrid edge-and-cloud networks.**

I. INTRODUCTION

Compared with traditional middle-boxes, network function virtualization (NFV) [1] introduces new flexibility, scalability, and cost-efficiency to network services by migrating network functions, e.g., Proxies, Firewalls, Load Balancers, from dedicated hardware to commercial servers. There is a dependency among these virtual network functions (VNFs) and chaining the VNFs for a network appliance in a particular order forms a service function chain (SFC) [2]. With the fast development of low-latency edge computing [25]–[30], there is a growing motivation to deploy SFCs at the edge [5]–[11]. Apart from reducing communication delay, deploying SFCs on the network edges also has the potential of saving bandwidth and protecting data safety and privacy [3].

Meanwhile, the flexibility of VNFs and the expense of edge resources also pose challenges for the SFC deployment problem. On one hand, it is essential to implement efficient resource management to make full use of the insufficient, expensive edge resources. Specifically, in NFV, resource management is realized by placing VNFs on edge commercial servers and using as few servers as possible. On the other hand, the chaining links between adjacent VNFs need to be taken care of when deploying SFCs, called flow scheduling. A bad scheduling scheme may cause redundant flow paths, incurring high latency and network congestion, while a good scheme will further cut back communication latency. Thus, in our model, we targeting jointly resource and latency optimization, not only efficiently managing edge resources but also cutting back the communication latency.

Moreover, the insufficiency of edge resources may result in the case that there is no way to place all SFCs on the edge servers under the server capacity limitation. In such cases, the cloud can be an efficient supplementary due to its adequate computing resources. But *how to allocate the edge and cloud computing resources for these VNFs* is a challenging problem. Most existing work either neglects the special case when the edge resource is not enough or simply places all remaining VNFs onto the remote cloud after all edge servers are occupied. Since latency between cloud and edge is often higher than that between edge servers, choosing different data flow to transmit between edge and cloud will make a big difference in the total communication latency. Thus, it is significant to solve the SFC deployment problem in the hybrid edge-and-cloud network rather than a single edge network.

In all, it is desirable to jointly consider resource cost and communication latency in the SFC deployment problem in hybrid edge-and-cloud networks. In our paper, we envision to *design a constant approximation algorithm, minimizing the total resource cost and all corresponding communication latency when deploying SFCs on edge commercial servers and the remote cloud.* The work is hard since we need to balance the resource cost and communication latency. Considering both edge and cloud resources makes the problem more complicated. Additionally, in the resource-related part, the indivisible VNF and limited edge server capacity produce an integer-variable constraint, which results in the NP-hardness of the problem. In the latency-aware part, the complex network topology brings trouble to virtual network embedding, specifically the routing problem of multi-hops. Last but not least, achieving an excellent theoretical bound is also a big challenge.

In our paper, we first simplify the problem by limiting it at the edge and design a constant approximation algorithm called chained next fit (CNF). CNF is mainly based on the Next Fit strategy (NF) [4] since it can guarantee efficient resource utilization and meanwhile have a positive influence on the latency-aware part by avoiding the redundant data traffic. Besides, an algorithm called double spanning tree (DST) is proposed to match the virtual network onto the physical network. DST also reduces latency and avoids network congestion by ensuring no more than two data flows passing on each network connection. Furthermore, we put forward a promotional algorithm called decreasing sorted, chained next fit (DCNF), dealing with a general case when both edge and cloud resources are taken into consideration. DCNF is designed based on CNF and also has a provable constant approximation ratio.

Our main contributions are summarized as follows.

- We formulate the SFC placement problem (SFCP), jointly considering resource and latency, as an Integer Linear Programming (ILP) problem and prove its NP-hardness through a reduction from Bin Packing (BP) Problem.
- We first focus on the SFCP limited at the edge. A constant approximation algorithm called chained next fit algorithm (CNF) is created with two main parts: the NF strategy and a designed sub-algorithm to deal with virtual network embedding, called double spanning tree (DST).
- We produce a promotional algorithm to solve the complete SFCP with both edge and cloud resources considered. We call it decreasing sorted, chained next fit (DCNF), and prove its constant approximation ratio.
- We perform extensive simulations on 5 randomly generated small network topologies and 4 larger real network topologies. The simulation results demonstrate that the practical ratios of DCNF to the optimal solutions are among interval $[1, 2.375]$, far smaller than the proved theoretical bound. What's more, DCNF always has a better performance than the benchmarks.

The remainder of this paper is organized as follows. Section II briefly reviews the related work. In Section III, we give the formulation of the jointly resource-and-latency-aware SFC deployment problem (SFCP) and prove its NP-hardness. Section IV demonstrates our CNF algorithm, composed of the DST algorithm and the NF strategy, which is designed to solve the SFCP limited at the edge. Furthermore, we prove its constant approximation ratio. In Section V, a general algorithm for the complete SFCP called DCNF is introduced and its approximation ratio is proved to be constant. Additionally, Section VI is the performance evaluation of DCNF. Finally, we conclude the paper in Section VII.

II. Related Work

With the emerging of NFV, researchers have devoted much effort to SFC placement problem like [5]–[23]. Among them, work [12]–[16] takes both resource and latency into consideration. [12]–[14] formulate the SFC placement problem as a mixed-integer linear programming (MILP) model and propose various heuristic approaches while [15], [16] put forward provable approximation algorithms based on the rounding algorithm.

Due to the low latency of edge computing, there is a trend of deploying SFCs on commercial servers at the edge. See [5]–[11] as examples. However, the resource of edge computing is limited. The insufficiency of edge resources may result in the case that there is no way to place all SFCs on the edge servers under the server capacity limitation. Most of the existing work neglects such situations like [5]–[8]. Work [9], [10] introduces cloud resources into the model for supplementary. But they simply place all remaining VNFs onto the remote cloud after all edge servers are occupied. To the best of our knowledge, there is only one work [11] solving the SFC deployment problem in the edge-and-cloud network as a whole. But [11] only optimizes the network latency, ignoring the resource cost.

Besides, most of the work considering deploying SFCs on network edges, e.g. [6]–[10] is limited to the design of heuristic algorithms and does not have a provable performance guarantee. As far as we are concerned, there are only below two works giving the performance bounds. [5] designs a two-stage VNF deployment scheme with a constrained depth-first search algorithm (CDFSA) and a path-based greedy algorithm (PGA), giving a theoretically-proved worst-case performance bound by an implicit constant factor. In [11], Song Yang et al. propose an efficient randomized rounding approximation algorithm to solve the delay-aware virtual network function placement and routing in edge clouds.

III. Problem Formulation

A. System Model

The notations used in this model are shown in Table I.

Consider the edge network as a connected graph $G = (V, E)$, where each commercial server i is a vertex in the graph, noted as V_i. If servers V_p and V_q are directly connected in the edge network, then link $(p, q) \in E$. Otherwise $(p, q) \notin E$. Assume there are M servers at the edge and each server i has its processing capacity, noted as C_i.

Suppose there are m SFCs, where the VNFs are placed in a particular order. And in SFC i, there are n_i VNFs, noted as $F_{i,1}, F_{i,2}, \cdots, F_{i,n_i}$ in the chaining order, and the throughput of data flow is b_i. Note that VNF $F_{i,j}$ needs $f_{i,j}$ computing resource, i.e., the size of $F_{i,j}$ is $f_{i,j}$. The total number of VNFs is $N = \sum_{i=1}^{m} n_i$.

In each SFC i, there is a data flow between $F_{i,j}$ and $F_{i,j+1}$ for any $1 \leq j \leq n_i - 1$. If $F_{i,j}$ and $F_{i,j+1}$ are placed on different edge servers, data transmission from $F_{i,j}$ to $F_{i,j+1}$ will cause communication latency. We define the communication latency that data flow of SFC i passes through one link in G as l_i. Typically, G is NOT a complete graph. So data transmitting from $F_{i,j}$ on one server to $F_{i,j+1}$ on another server may need multiple hops, i.e., pass through several server-nodes or links in G. The number of hops here,

m	number of Service Function Chains (SFCs)
M	number of physical servers in the edge network
n_i	number of VNFs in SFC i
N	number of all VNFs
$F_{i,j}$	VNF j in SFC i
$f_{i,j}$	the size of VNF $F_{i,j}$
l_i	communication latency that data flow of SFC i passes through one link (or to say, hop) in the edge network
L_i	E&C communication latency of SFC i
b_i	data flow of SFC i
V_k	edge server k, if $k > 0$; the cloud, if $k = 0$
C_k	capacity of edge server k
$B_{p,q}$	bandwidth limit of network connection (p, q)
$w_{i,j}^{p,q}$	if data between $F_{i,j}$ and $F_{i,j+1}$ pass through link (p, q) or not
$x_{i,j}^k$	if VNF $F_{i,j}$ is placed on V_k or not
y_k	if server k is occupied or not
$z_{i,j}$	number of hops at the edge when data flows from $F_{i,j}$ to $F_{i,j+1}$
$Z_{i,j}$	if there is E&C trans. latency between $F_{i,j}$ and $F_{i,j+1}$ or not
M'	number of used servers in the results of CNF
M^*	number of used servers in OPT
V_j^*	used server j in OPT
$C(\cdot)$	(total) capacity of a server
$c(\cdot)$	occupied capacity of a server by VNFs.

TABLE I: Notations

noted as $z_{i,j}$, depends on the solutions of the multi-hop routing problem on G under the bandwidth limit of each link (p, q), noted as $B_{p,q}$. Note that if $(p, q) \notin E$, $B_{p,q} = 0$.

Above we talk about the latency at the edge. If some VNFs are put on the cloud, there are communication latency between edge and cloud, short for E&C communication latency. We now consider such a situation when $F_{i,j}$ and $F_{i,j+1}$ are placed one at the edge and one on the cloud. Then E&C communication latency appears. It is determined mainly by the data flow volume of SFC, the multiple hops outside the edge network and the bandwidths between edge and cloud, etc. The relationship is very complicated and there is work professionally targeting the related topic [24]. For simplicity, here we just assume the total E&C communication latency between $F_{i,j}$ and $F_{i,j+1}$ is L_i and $L_i \geq l_i$. Denote the cloud as V_0. Also, we assume the cloud is "connected" to all edge server-nodes and give the estimated bandwidth limit of the connection between an edge server V_k and the cloud V_0, noted as $B_{k,0} = B_{0,k}$.

B. Problem Formulation

In the SFC placement problem (SFCP), our task is to place the m SFCs onto the M edge commercial servers or the cloud without exceeding the constraints of edge server capacities and the bandwidth limits of network connections. Our goal is to minimize the total resource consumption and the whole communication latency when placing SFCs in the hybrid edge-and-cloud network. Specifically, the total resource consumption contains the total capacities of occupied edge servers and the total sizes of VNFs put on the cloud while the whole communication latency includes the total communication latency at the edge and that between edge and cloud.

In order to formulate SFCP, we first define three *Boolean* variables: (1) $x_{i,j}^k = 1$ if and only if VNF $F_{i,j}$ is placed on server k; (2) $y_k = 1$ if and only if server k is occupied; (3) $w_{i,j}^{p,q} = 1$ if and only if data flow between VNF $F_{i,j}$ and $F_{i,j+1}$ passes through link (p, q). Among them, $x_{i,j}^k$ and $w_{i,j}^{p,q}$ are the decision variables in our model while y_k is used for clear expression. Note that, here, server k $(1 \leq k \leq M)$ represents **a edge server** while server 0 is on behalf of **the cloud**.

The capacity constraint of each edge server asks

$$\sum_{i=1}^{m}\sum_{j=1}^{n_i} x_{i,j}^k \cdot f_{i,j} \leq y_k \cdot C_k, \quad \forall 1 \leq k \leq M. \tag{1}$$

Under the limitation of bandwidth, it satisfies

$$\sum_{i=1}^{m}\sum_{j=1}^{n_i-1} (w_{i,j}^{p,q} + w_{i,j}^{q,p}) \cdot b_i \leq B_{p,q}, \quad \forall\ 0 \leq p < q \leq M. \tag{2}$$

Since each VNF can not be split, which implies it is exactly placed on an edge server or the cloud,

$$\sum_{k=0}^{M} x_{i,j}^k = 1, \qquad \forall 1 \leq i \leq m, 1 \leq j \leq n_i. \tag{3}$$

According to *Flow Conservation Law*, as for any data flow between VNF $F_{i,j}$ and $F_{i,j+1}$, we have $\forall\ 0 \leq k \leq M$,

$$\sum_{p=0}^{M} w_{i,j}^{p,k} - \sum_{q=0}^{M} w_{i,j}^{k,q} = x_{i,j+1}^k - x_{i,j}^k. \tag{4}$$

Then the number of hops that data flows at the edge network, from $F_{i,j}$ to $F_{i,j+1}$, is

$$z_{i,j} = \sum_{p=1}^{M}\sum_{q=1}^{M} w_{i,j}^{p,q}, \quad \forall 1 \leq i \leq m, 1 \leq j \leq n_i - 1. \tag{5}$$

Note that we define $z_{i,0} = 1$ and $z_{i,n_i} = 1$, showing as for SFC i, there is input data flowing into the first VNF and output data flowing out of the last VNF at the edge.

Besides we create another *Boolean* variable, presenting whether there is E&C communication latency between $F_{i,j}$ and $F_{i,j+1}$. As for any $1 \leq i \leq m, 1 \leq j \leq n_i - 1$,

$$Z_{i,j} = \sum_{q=1}^{M} w_{i,j}^{0,q} + \sum_{p=1}^{M} w_{i,j}^{p,0}. \tag{6}$$

And we define $Z_{i,0} = x_{i,1}^0$ and $Z_{i,n_i} = x_{i,n_i}^0$, which means if $F_{i,1}$ is placed on the cloud, there is input data flowing from the edge to the cloud, producing E&C communication latency, while if F_{i,n_i} is placed on the cloud, there is output data flowing from the cloud to the edge, also producing E&C communication latency.

In our model, there are four optimization objectives, respectively

- Resource cost at the edge: $\mathbb{R}^{\mathbb{E}}$,
- Communication Latency between edge servers: $\mathbb{L}^{\mathbb{E}}$,
- Resource cost on the cloud: $\mathbb{R}^{\mathbb{C}}$,
- Communication Latency between edge and cloud: $\mathbb{L}^{\mathbb{C}}$,

$$\mathbb{R}^{\mathbb{E}} = \sum_{k=1}^{M} y_k \cdot C_k, \qquad \mathbb{L}^{\mathbb{E}} = \sum_{i=1}^{m}\sum_{j=0}^{n_i} l_i \cdot z_{i,j},$$

$$\mathbb{R}^{\mathbb{C}} = \sum_{i=1}^{m}\sum_{j=1}^{n_i} x_{i,j}^0, \cdot f_{i,j}, \qquad \mathbb{L}^{\mathbb{C}} = \sum_{i=1}^{m}\sum_{j=0}^{n_i} L_i \cdot Z_{i,j}.$$

Then SFCP can be formulated as the below ILP.

$$\begin{aligned} \min \quad & \alpha\mathbb{R}^{\mathbb{E}} + \beta\mathbb{L}^{\mathbb{E}} + \gamma\mathbb{R}^{\mathbb{C}} + \zeta\mathbb{L}^{\mathbb{C}} \\ \text{s.t.} \quad & (1) - (6), \end{aligned}$$

where $\alpha, \beta, \gamma, \zeta$ are weighting factors that are used to adjust the relative importance of the cost component.

C. Proof of NP-hardness

As for a classical Bin Packing Problem (BP) with any parameter setting, the problem can be reduced to an SFCP with $\alpha = 1, \beta = \gamma = \zeta = 0, C_k = 1, B_{p,q} = \infty,\ \forall 1 \leq k, p, q \leq M$, where each server can be seen as a bin while each VNF can be seen as an item that needs to be packed. Since BP is an NP-complete problem, SFCP is NP-hard.

D. Problem Complexity

In the above section, we show the NP-hardness of SFCP based on the resource cost alone by reducing from BP. Similarly, SFCP can also be proved to be NP-hard by the reduction from the Travelling Salesman Problem (TSP) if only the network latency is considered. It implies the complexity of the SFCP problem comes from both the resource cost part and the communication latency part. Besides, considering the hybrid edge and cloud network as a whole also bring troubles to this problem. We should take care of the four optimization objectives at the same time, which is dramatically complicated.

Thus, in the below section, we start from a simplified SFCP limited at the edge with two optimization objectives and then promote the scheme to deal with the complete SFCP with four objectives.

IV. THE SFCP LIMITED AT THE EDGE

A. Basic Ideas and Challenges

As for the SFCP limited at the edge, there exist two optimization objectives, which is still difficult. Thus we take a look at a simple version of it to find some intuitions. If ignoring the communication latency of data flows between VNFs in SFCs, specifically $\alpha = 1, \beta = 0, B_{p,q} = \infty\ \forall 1 \leq k, p, q \leq M$, the simplified SFCP can be seen as a BP problem with servers being viewed as bins and VNFs as items. There are plenty of approximation algorithms for BP. In all, the Next Fit strategy (NF) is best suitable for SFCP, since it not only guarantees efficient resource utilization, but also has a positive influence on communication latency by avoiding the redundant data traffic. What's more, it has no rules on sequences of bins and items, which gives us space to make rules for virtual network embedding.

We intend to design an approximation algorithm based on NF, satisfying both server capacity constraints and connection bandwidth limits. Then we prove the upper bounds of the resource consumption and communication latency separately and finally get the approximation ratio of our algorithm by integrating these two parts.

However, challenges still exist. The first challenge comes from the complicated network topology. The disconnection of some servers and the resulted various multi-hop paths between them make it difficult to match the virtual network to the physical edge network, meanwhile satisfying the bandwidth limit. Thus before applying the NF strategy, we need to add a preparing part for virtual network embedding, specifically sorting servers by minimizing the total number of multiple hops. Additionally, different server capacities make it hard to prove the bound of communication latency because it leads to a gap between total used resources and the number of used servers.

B. Chained Next Fit (CNF) Algorithm Procedures

We design an approximation algorithm called Chained Next Fit algorithm (CNF). In the beginning of CNF, we devise a preparing algorithm to sort servers called double spanning tree (DST), as shown in Algo. 1. Specifically, we first choose the maximal-capacity server. Then use the depth-first search (DFS) algorithm to obtain a spanning tree with this server node as the root. In DFS, let the node with larger capacity have higher priority for the neighbor search. Double this spanning tree and delete a link between the root and its child, we will get a path. Finally, starting from the root, along the path, we traverse all nodes. We sort the servers by the traversing order. Note that if some node has been reached before, hop it and continue to the next node.

Algorithm 1: Double Spanning Tree Algorithm (DST)

Input: G and the servers $V_1, V_2, \cdots, V_M$.
Output: Sorted servers $V_{k_1}, V_{k_2}, \cdots, V_{k_M}$ and multi-hop paths T_j between each two adjacent sorted servers V_{k_j} and $V_{k_{j+1}}$.

Find the server V_p with maximal capacity and mark it.
Use DFS to obtain a spanning tree T with V_p as its root, where node with larger capacity has higher priority for the neighbor search.
Double T and delete a link between V_p and its child with the smallest capacity to get a path T'.
$j \leftarrow 1,\ k_j \leftarrow p$.
while $j < M$ **do**
  Find the next unmarked node of V_{k_j} on T', noted as V_q, and mark it.
  $j \leftarrow j + 1,\ k_j \leftarrow q$.
for $j = 1 \rightarrow M - 1$ **do**
  Find the shortest path between V_{k_j} and $V_{k_{j+1}}$ in G, noted as the multi-hop path T_j from V_{k_j} to $V_{k_{j+1}}$.

After DST, we sort SFCs in decreasing order of their latency parameters l_i and sort VNFs in their chaining order. Next, we employ the NF strategy to do SFC deployment. Specifically, if a VNF fits inside the currently considered edge server, place it on this server. Otherwise, the placement on the current server ends, a new edge server is opened and the VNF is placed on this new server. Repeat the same procedures on the next VNF until all VNFs are placed at the edge.

The detailed CNF algorithm with **time complexity** $O(N + mlog(m) + M^2 \log(M))$ is shown in Alg. 2.

Then we demonstrate the solution obtained by CNF is feasible under the assumption that each edge server has enough capacity for any VNF and each edge network connection has enough bandwidth for double of any data flow. Based on this premise, the NF strategy (line 5-13 in Algo. 2) maintains the server capacity constraints in Ineq. 1. Combined with chained sorting, it also ensures there is at most one data flow between any two edge servers. And as shown in Fig. 1b, DST and the property of tree structure guarantee that there are at most two hops (or data flows) passing through any edge network connection, which will be explained in detail in Section IV-D. So the connection bandwidth constraints in Ineq. 2 are satisfied.

Algorithm 2: CNF algorithm

Input: The list of VNFs $\{F_{i,j}\}$ and the capacities of servers $C_1, C_2, \cdots, C_M$.
Output: The placement scheme $x_{i,j}^k$ and the number of used servers M'.

```
1  Sort and reindex servers by DST.
2  Sort and reindex SFCs so that l_1 ≥ l_2 ≥ ··· ≥ l_m;
3  k ← 1;
4  for i = 1 → m do
5      for j = 1 → n_i do
6          if V_k has enough rest capacity for F_{i,j} then
7              x^k_{i,j} = 1; (Place F_{i,j} on V_k)
8          else
9              if k < M then
10                 For all links (p,q) in the multi-hop path
                   T_k, let w^{p,q}_{i,j} = 1;
11                 x^{k+1}_{i,j} = 1,  k ← k + 1;
12             else
13                 Return 0; (No enough edge servers)
14 M' ← k.
```

C. Bound of Resource Part

We now demonstrate CNF has an asymptotic approximation ratio of 2 to the optimal solution (OPT) on $\mathbb{R}^{\mathbb{E}}$. For proof of the ratio, we use functions of $C(\cdot)$ and $c(\cdot)$ to respectively represent the capacity of an edge server and the occupied part. Obviously, for any $k \in [1, M]$, $c(V_k) \leq C(V_k)$.

Besides we assume $V_1, \cdots, V_{M'}$ are used servers by CNF while $V_1^*, \cdots, V_{M^*}^*$ are those by OPT. Since they both deal with the same VNF set $\{F_{i,j}\}$,

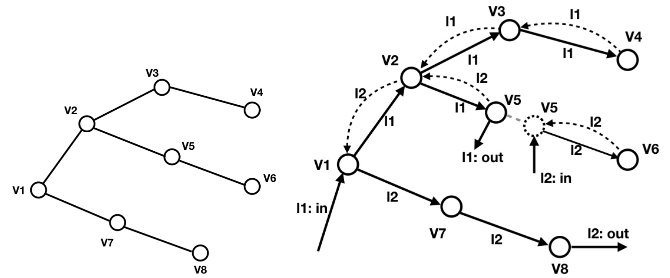
$$\sum_{k=1}^{M'} c(V_k) = \sum_{i=1}^{m} \sum_{j=1}^{n_i} f_{i,j} = \sum_{k=1}^{M^*} c(V_k^*).$$

Theorem 1. *$\mathbb{R}^{\mathbb{E}}{}_{CNF} < 2 \cdot \mathbb{R}^{\mathbb{E}}{}_{OPT} + C$, where $C = \max C_k$.*

Proof. Referring to the NF strategy, the first VNF placed on the next server can NOT be placed on the current server. Thus all VNFs of the current server plus the first VNF on the next server are larger than the capacity of the current server, i.e. $\forall\ 1 \leq k < M',\ c(V_k) + c(V_{k+1}) > C(V_k)$. In sum,

$$\begin{aligned} \mathbb{R}^{\mathbb{E}}{}_{CNF} &= \sum_{k=1}^{M'} C(V_k) < \sum_{k=1}^{M'-1} (c(V_k) + c(V_{k+1})) + C, \\ &< 2 \sum_{k=1}^{M} c(V_k) + C = 2 \sum_{k=1}^{M^*} c(V_k^*) + C, \\ &\leq 2 \sum_{k=1}^{M^*} C(V_k^*) + C = 2 \cdot \mathbb{R}^{\mathbb{E}}{}_{OPT} + C. \end{aligned} \tag{7}$$

□



(a) A Tree (b) Multi-hop Paths between nodes occupied by two SFCs

Fig. 1: A simple example of DST algorithm

D. Bound of Communication Part

Here we show CNF also has a constant ratio to OPT on $\mathbb{L}^{\mathbb{E}}$. We now denote $z_i = \sum_{j=0}^{n_i} z_{i,j}$ and z_i^* respectively as the total number of hops in SFC i by CNF and OPT.

Theorem 2. $\mathbb{L}^{\mathbb{E}}{}_{CNF} \leq 4 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot \mathbb{L}^{\mathbb{E}}{}_{OPT}$.

Proof. In the path T' produced by Algo. 1, each network link appears at most twice. Line 2 in Algo. 2 ensures for each network link, the data flow of SFC later flowing through must be less than or equal to that former flowing through before. Thus

$$\sum_{i=1}^{m} z_i \cdot l_i \leq 2 \sum_{i=1}^{m} M_i \cdot l_i, \tag{8}$$

where M_i is the number of used servers when placing SFC i by CNF.

Take a simple case with 8 servers and 2 SFCs for example. Fig. 1a shows the spanning tree T of the graph G. In this case, the path T' is $V_1 - V_2 - V_3 - V_4 - V_3 - V_2 - V_5 - V_6 - V_5 - V_2 - V_1 - V_7 - V_8 - V_7 - V_1$. Each network link appears at most twice. SFC 1 occupied V_1, V_2, V_3, V_4, V_5 while SFC 2 occupied V_5, V_6, V_7, V_8. Fig. 1b shows the multi-hop paths in the worst case and the passing data flow, where dotted lines represent the link second used while lines show the link first used. In SFC 1 from V_4 to V_5, the multi-hop path $V_4 - V_3 - V_2 - V_5$ is feasible. Line 9 in Algo. 1 may find a shorter multi-hop path. For instance, if link $(V_2, V_4) \in E$, then $V_4 - V_2 - V_5$ is a shorter multi-hop path. Anyway $V_4 - V_3 - V_2 - V_5$ is the worst case. Here we only consider such multi-hop paths on T'. In Fig. 1b, the data flow on lines is no less than that on dotted lines. For SFC 1, data flows through $V_1 - V_2 - V_3 - V_4 - V_3 - V_2 - V_5$

and the number of hops is $6+2 < 2*5$. The total number of hops for SFC 1, 2 is $|T'|+4 = 18 \le 2*(5+4)$. And $l_1 \ge l_2$, thus Ineq. 8 holds.

As for SFC i, suppose the first used server by CNF is V_{p_i} and the last is V_{q_i}. Denote the number of used servers when placing SFC i by OPT as M_i^*. By definitions, we have

$$M_i^* \ge \left\lceil \frac{\sum_{j=1}^{n_i} f_{i,j}}{\max\limits_{1\le k\le M} C_k} \right\rceil \quad \text{and} \quad M_i \le \left\lceil \frac{\sum_{k=p_i}^{q_i} C(V_k)}{\min\limits_{1\le k\le M} C_k} \right\rceil. \quad (9)$$

In OPT, there is at least one hop between two used servers in SFC i, as well as one hop for data flowing in, one for data flowing out. Thus

$$z_i^* \ge M_i^* + 1. \quad (10)$$

In sum,

$$\begin{aligned} M_i &\le \left\lceil \frac{\sum_{k=p_i+1}^{q_i-1} C(V_k) + C(V_{p_i}) + C(V_{q_i})}{\min C_k} \right\rceil, && (11)\\ &\le \left\lceil \frac{2\sum_{j=1}^{n_i} f_{i,j} + 2\max C_k}{\min C_k} \right\rceil, && (12)\\ &\le 2\cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot M_i^* + 2\cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil, \\ &\le 2\cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot z_i^*, \end{aligned}$$

where $\lceil\cdot\rceil$ is the ceiling function, Format 11 $\le$ Format 12 stems from Ineq. 7.

Referring to Ineq. 8, we have

$$\begin{aligned} \mathbb{L}^{\mathbb{E}}{}_{CNF} &= \sum_{i=1}^{m} z_i \cdot l_i \le 2\cdot \sum_{i=1}^{m} 2\cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot z_i^* \cdot l_i, \quad (13)\\ &= 4\cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot \sum_{i=1}^{m} z_i^* \cdot l_i, \\ &= 4\cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot \mathbb{L}^{\mathbb{E}}{}_{OPT}. \end{aligned}$$

□

E. Approximation Ratio of CNF in SFC Placement Problem

Now we can combine upper bounds of $\mathbb{R}^{\mathbb{E}}{}_{CNF}$ and $\mathbb{L}^{\mathbb{E}}{}_{CNF}$ to prove the constant approximation ratio of CNF in SFCP. Denote CNF and OPT as the total cost respectively by CNF and OPT. Combined with Theorem 1, 2, we obtain the below theorem.

Theorem 3. *$CNF \le 4\cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot OPT + C$, where $C = \alpha \cdot \max C_k$.*

Proof.

$$\begin{aligned} CNF &= \alpha \cdot \mathbb{R}^{\mathbb{E}}{}_{CNF} + \beta \cdot \mathbb{L}^{\mathbb{E}}{}_{CNF}, \\ &\le 2\alpha \cdot \mathbb{R}^{\mathbb{E}}{}_{OPT} + C + 4\beta \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot \mathbb{L}^{\mathbb{E}}{}_{OPT}, \\ &< 4\cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot \left[\alpha \cdot \mathbb{R}^{\mathbb{E}}{}_{OPT} + \beta \cdot \mathbb{L}^{\mathbb{E}}{}_{OPT}\right] + C, \\ &\le 4\cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot OPT + C. \end{aligned}$$

□

V. THE COMPLETED SFCP (EDGE & CLOUD)

A. Basic Ideas and Challenges

Typically, the communication latency between edge and cloud is much larger than that among edge servers, i.e. $L_i \ge l_i$, owing to the longer distance. Thus we prioritize the use of edge resources.

In order to take full use of edge resources, the natural idea is to put as much VNFs at the edge as possible and then put the rest VNFs on the cloud. However, such direct operations may cost high communication latency between edge and cloud since what VNFs and SFCs are left is uncontrolled. The key challenge here is how to choose VNFs or SFCs, putting at the edge, so that the left VNFs need as small cloud resources as possible and meanwhile produce as low E&C communication latency as possible.

B. Algorithm Procedures

Here we design an approximation algorithm called Decreasing Sorted, Chained Next Fit algorithm (DCNF). Based on Theorem 1, CNF makes the left VNFs as few as possible, which helps minimizes the needed cloud resources. But it can not ensure the left SFCs produce as low E&C communication latency as possible.

In order to minimize the E&C communication latency, the basic rule is to put the whole SFC i or a continuously chaining segment of SFC i on the cloud, where the communication latency is merely $2L_i$. Otherwise, if there is k intermittent parts of SFC i placed on the cloud, the produced communication latency between edge and cloud is $2k \cdot L_i$. Luckily, the left VNFs of a SFC by CNF must be continuously chaining. Since the total resource consumption on the cloud is roughly fixed, noted as R_c, depending on the fixed total edge resources, the key problem is a 0-1 min-knapsack problem as below.

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} 2L_i \cdot X_i, \\ \text{s.t.} \quad & \sum_{i=1}^{m} F_i \cdot X_i \ge R_c, \end{aligned}$$

where X_i is a Boolean variable, representing if SFC i is placed on the cloud or not; $F_i = \sum_{j=1}^{n_i} f_{i,j}$ is the total size of all VNFs in SFC i, also mentioned as the size of SFC i.

This 0-1 min-knapsack problem can be solved by a greedy approach. Sort and reindex SFCs by $\frac{L_i}{F_i}$ in increasing order so that

$$\frac{L_1}{F_1} \le \frac{L_2}{F_2} \le \cdots \le \frac{L_m}{F_m}.$$

Then keep choosing SFCs with the smallest $\frac{L_i}{F_i}$ until the total sizes of choice SFCs reaching R_c. Since R_c is not exactly known, we sort SFCs in the inverse order and put as more SFCs with large $\frac{L_i}{F_i}$ at the edge as possible.

But the reindexing step of SFCs in line 2 of Algo. 2 makes the left SFCs uncontrolled, not necessarily with small $\frac{L_i}{F_i}$. Thus when employ CNF, we must ensure enough resources at the edge. Otherwise, there is a possibility that some VNFs in a SFC with large $\frac{L_i}{F_i}$ are left to be placed on the cloud. If no definitely adequate edge resources, we had better call CNF with the input of merely one SFC. Referring to Eq. 7, we can see there is always enough edge servers for VNFs whose total sizes are less than or equal to $\frac{1}{2}\sum_{k=1}^{M} C(V_k) \overset{\text{def}}{=} \frac{1}{2}\widetilde{C}$.

Above all, we first sort and reindex SFCs by $\frac{L_i}{F_i}$ in decreasing order and find the largest m' s.t. $\sum_{i=1}^{m'} F_i \le \frac{1}{2}\widetilde{C}$. Then employ CNF to place SFC 1 to m' at the edge. Keep calling CNF to deal with the next SFC, notes as i_0, until CNF breaks due to no enough edge resources. Finally, we put the rest VNFs of SFC i_0, as well as the whole SFCs i_0+1 to m on the cloud. The detailed DCNF algorithm with **time complexity** $O(N + mlog(m) + M^2\log(M))$ is shown in Alg. 3.

Algorithm 3: DCNF algorithm

Input: The list of VNFs $\{F_{i,j}\}$ and the capacities of servers $C_1, C_2, \cdots, C_M$.
Output: The placement scheme $x_{i,j}^k$.

1 Sort and reindex SFCs so that $\frac{L_1}{F_1} \ge \frac{L_2}{F_2} \ge \cdots \ge \frac{L_m}{F_m}$;
2 $\widetilde{C} \leftarrow \sum_{k=1}^{M} C_k$, $\;Sum \leftarrow F_1$, $\;i \leftarrow 1$;
3 **while** $Sum \le \frac{1}{2}\widetilde{C}$ **do**
4 $\quad i \leftarrow i+1$, $\;Sum \leftarrow Sum + F_i$;
5 Employ CNF to place SFC 1 to $i-1$ at the edge;
6 **while** *there exist edge resources left* **do**
7 $\quad$ Run CNF to deal with SFC i;
8 $\quad i_0 \leftarrow i$, $\;i \leftarrow i+1$;
9 Put the rest VNFs of SFC i_0, as well as the whole SFCs i_0+1 to m, on the cloud. ($x_{i,j}^0 = 1$)

Then we illustrate the solution obtained by DCNF is feasible. First CNF maintains all constraints at the edge, specifically the edge server capacity constraints in Ineq. 1, and the edge network connection bandwidth constraints in Ineq. 2. Besides, as for the connection bandwidth constraints between edge and cloud, i.e. $p = 0$ or $q = 0$, in Ineq. 2, the NF strategy and chaining sorting also ensure they are satisfied.

C. Approximation Ratio

We now demonstrate DCNF has provable constant ratios to OPT for two different cases. For proof of the ratios, we denote $DCNF$ and OPT as the total cost respectively by DCNF and OPT. Denote the total size of all VNFs as $\widetilde{F} = \sum_{i=1}^{m} F_i$.

Theorem 4. *$DCNF \le r \cdot OPT + C$, where if $\widetilde{F} \le \frac{1}{2}\widetilde{C}$,*

$$C = \alpha \cdot \max C_k \text{ and } r = 4 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil;$$

If $\widetilde{F} > \frac{1}{2}\widetilde{C}$,

$$C = 2\zeta \cdot \max L_i \text{ and } r = \max\left\{ 4\left\lceil \frac{\max C_k}{\min C_k}\right\rceil, \frac{2\alpha\widetilde{C} + (2\widetilde{F} - \widetilde{C})(\gamma + 2\zeta \max_i \frac{L_i}{F_i})}{2\alpha \min\left\{\widetilde{F}, \widetilde{C}\right\} + 2\max\left\{\widetilde{F} - \widetilde{C}, 0\right\}(\gamma + 2\zeta \min_i \frac{L_i}{F_i})} \right\}. \tag{14}$$

And when $\widetilde{F} \to \infty$, $r = \max\left\{4\left\lceil\frac{\max C_k}{\min C_k}\right\rceil, \frac{\gamma + 2\zeta \max_i \frac{L_i}{F_i}}{\gamma + 2\zeta \min_i \frac{L_i}{F_i}}\right\}$ is constant. When $\alpha = 0$, $\frac{1}{2}\widetilde{C} < \widetilde{F} \le \widetilde{C}$, r has another format, $r = 4 \cdot \left\lceil\frac{\max C_k}{\min C_k}\right\rceil + \frac{\max C_k}{\min l_i} \cdot \frac{(2\widetilde{F} - \widetilde{C})(\gamma + 2\zeta \max_i \frac{L_i}{F_i})}{\beta(\widetilde{F} + \widetilde{C})}$.

Proof. When $\widetilde{F} \le \frac{1}{2}\widetilde{C}$, DCNF is reduced to CNF and SFCP is limited at the edge. By Theorem 3, we know

$$r = 4 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil \text{ and } C = \alpha \cdot \max C_k.$$

When $\widetilde{F} > \frac{1}{2}\widetilde{C}$,

$$\mathbb{R}^{\mathbb{E}}{}_{DCNF} = \widetilde{C}, \quad \mathbb{R}^{\mathbb{E}}{}_{OPT} \ge \min\left\{\widetilde{F}, \widetilde{C}\right\}.$$

And referring to Theorem 2, $\mathbb{L}^{\mathbb{E}}{}_{DCNF} \le 4\left\lceil\frac{\max C_k}{\min C_k}\right\rceil \cdot OPT_c$, where OPT_c represents the minimal latency when placing these VNFs at the edge. In OPT, more VNFs are placed at edge servers due to its low latency, thus $\mathbb{L}^{\mathbb{E}}{}_{OPT} \ge OPT_c$.

Line 6-8 in Algo. 3 ensures all edge servers are occupied, then by Eq. 7, the total sizes of VNFs placed at the edge $\ge \frac{1}{2}\widetilde{C}$, which implies $\mathbb{R}^{\mathbb{C}}{}_{DCNF} \le \widetilde{F} - \frac{1}{2}\widetilde{C}$.

In OPT, the total sizes of VNFs placed at the edge $\le \widetilde{C}$, thus when $\widetilde{F} \ge \widetilde{C}$, the resource cost on the cloud $\ge \widetilde{F} - \widetilde{C}$. When $\widetilde{F} < \widetilde{C}$, there is a possibility that all SFCs are placed at the edge, so in this case, we can only get the resource cost on the cloud ≥ 0. In all, $\mathbb{R}^{\mathbb{C}}{}_{OPT} \ge \max\left\{\widetilde{F} - \widetilde{C}, 0\right\}$.

In DCNF, SFCs i_0+1 to m are totally put on the cloud, thus SFC i ($i_0 + 1 \le i \le m$) has exactly two data flows between cloud and edge and the total E&C latency is $2L_i$. VNFs of SFC i_0 are partly put on the cloud. But the NF strategy ensures it is a continuous part, thus SFC i_0 also has at most two data flows between cloud and edge, producing $2L_{i_0}$ delay. In sum, $\mathbb{L}^{\mathbb{C}}{}_{DCNF} \le \sum_{i=i_0}^{m} 2L_i \le 2L_{i_0} + 2(\max\frac{L_i}{F_i})\sum_{i=i_0+1}^{m} F_i$. Eq. 7 demonstrates $\sum_{i=1}^{i_0} F_i \ge \frac{1}{2}\widetilde{C}$, which implies $\sum_{i=i_0+1}^{m} F_i = F - \sum_{i=1}^{i_0} F_i \le F - \frac{1}{2}C$. Thus, $\mathbb{L}^{\mathbb{C}}{}_{DCNF} \le 2\max L_i + \max\frac{L_i}{F_i} \cdot (2\widetilde{F} - \widetilde{C})$.

And if some VNFs in SFC i are placed on the cloud, the E&C communication latency of this chain is at least $2L_i$. And the total sizes of SFCs that contain VNFs on the cloud is larger than or equal to the used cloud resources. Thus $\mathbb{L}^{\mathbb{C}}{}_{OPT} \ge \min\frac{2L_i}{F_i} \cdot \max\left\{\widetilde{F} - \widetilde{C}, 0\right\}$.

Above all,

$$\begin{aligned} DCNF =& \alpha\mathbb{R}^{\mathbb{E}}{}_{DCNF} + \beta\mathbb{L}^{\mathbb{E}}{}_{DCNF} + \gamma\mathbb{R}^{\mathbb{C}}{}_{DCNF} + \zeta\mathbb{L}^{\mathbb{C}}{}_{DCNF}, \\ \le& \alpha\widetilde{C} + 4\left\lceil\frac{\max C_k}{\min C_k}\right\rceil \cdot (\beta \cdot OPT_c) + 2\zeta \cdot \max L_i, \end{aligned}$$

$$
\begin{aligned}
&+ (\frac{1}{2}\gamma + \zeta \cdot \max \frac{L_i}{f_i}) \cdot (2\widetilde{F} - \widetilde{C}), \\
OPT =& \alpha \mathbb{R}^{\mathbb{E}}{}_{OPT} + \beta \mathbb{L}^{\mathbb{E}}{}_{OPT} + \gamma \mathbb{R}^{\mathbb{C}}{}_{OPT} + \beta \mathbb{L}^{\mathbb{C}}{}_{OPT}, \\
\geq& \alpha \min \left\{ \widetilde{F}, \widetilde{C} \right\} + \beta \cdot OPT_c \\
&+ (\gamma + 2\zeta \cdot \min \frac{L_i}{f_i}) \cdot \max \left\{ \widetilde{F} - \widetilde{C}, 0 \right\}.
\end{aligned}
$$

Since $\frac{a+b}{c+d} \leq \max\left\{\frac{a}{c}, \frac{b}{d}\right\}$, letting $b = 4\left\lceil \frac{\max C_k}{\min C_k} \right\rceil \cdot (\beta \cdot OPT_c)$ and $d = \beta \cdot OPT_c$, we can get Eq. 14.

And by Eq. 10, $OPT_c \geq \sum_{i=1}^{m} \left(\frac{F_i}{\max C_k} + 1 \right) \cdot l_i \geq (\frac{\widetilde{F}}{\max C_k} + m) \cdot \min l_i \geq \frac{\min l_i}{\max C_k}(\widetilde{F} + \widetilde{C})$. Thus, when $\alpha = 0$, $\frac{1}{2}\widetilde{C} < \widetilde{F} \leq \widetilde{C}$, r has another format. $r = 2 \cdot \left\lceil \frac{\max C_k}{\min C_k} \right\rceil + \frac{\max C_k}{\min l_i} \cdot \frac{(4\widetilde{F} - \widetilde{C})(\gamma + 2\zeta \max_i \frac{L_i}{F_i})}{\beta(\widetilde{F} + \widetilde{C})}$. □

VI. PERFORMANCE EVALUATION

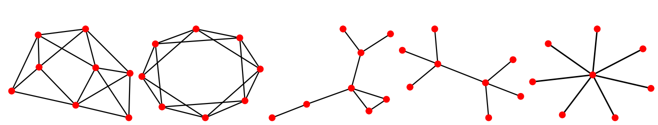
In this section, we perform extensive simulations on different network topologies to evaluate and compare the performance of DCNF with those of benchmarks. For each group of outcomes, we use the average value from 100 groups of simulations to reduce accidental errors. And the errors shown on all the plots below are determined by the standard variances of the corresponding 100 groups of simulations.

A. Algorithm Benchmarks

In the simulations, we compare our proposed DCNF with three other heuristic algorithms as below.

- **Rounding Algorithm (Rd)**: After formulating SFCP as an ILP, the rounding algorithm can be used here. See [15], [16] as example.
- **First Fit Algorithm + Shortest Path Algorithm (FFSP)**: since our DCNF is based on the NF strategy of BP, here we try another popularly-used approximation algorithm for BP, first-fit algorithm (FF). Specifically, we use FF to assign VNFs on edge servers, and if there are no enough edge resources for a VNF, we put it on the cloud. Then we use the shortest path algorithm, e.g. Floyd algorithm [31], to find the routing of multiple hops between two servers to compute the communication latency.
- **Best Fit Algorithm + Shortest Path Algorithm (BFSP)**: Here we try another different popularly-used approximation algorithm for BP, best-fit algorithm (BF). Other steps are the same as FFSP.

B. Simulations on 5 Small-Scale Random Network Topologies

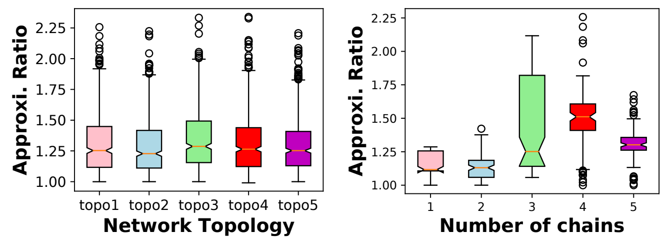


(a) Topo 1 (b) Topo 2 (c) Topo 3 (d) Topo 4 (e) Topo 5

Fig. 2: 5 different Network topologies with 8 Nodes

1) Network Topology and Simulation Setup: According to Section III-C, SFCP is NP-hard, which implies the time cost of finding OPT grows exponentially with the increase of problem scale, specifically the number of VNFs N and the number of edge servers M. Thus we first evaluate the performance of DCNF on the small-scale network topologies with 8 nodes, where we can use the MIP solver to find OPT for SFCP. We randomly generate 5 different such network topologies as shown in Fig. 2.

The capacities of 8 edge servers are set as respectively $4, 4, 4, 4, 4, 6, 6, 8$. Besides, we set $\alpha = 1, \beta = 1, \gamma = 2, \zeta = 1, f_{i,j} \sim \mathbb{N}(2, 0.5), l_i \sim \mathbb{N}(\overline{f_i}, 0.25), L_i \sim \mathbb{N}(\overline{f_i}, 0.25)$, where $\overline{f_i}$ is the mean size of all VNFs $F_{i,j}$ of a SFC i.



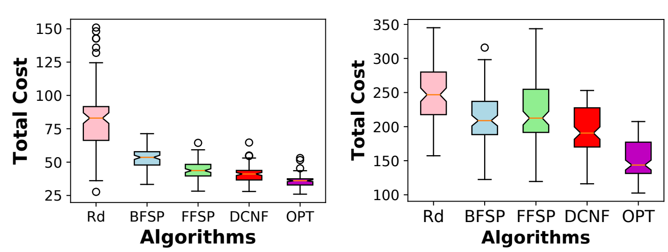
(a) Different Network topologies (b) Different Chains for Topo 1

Fig. 3: Approximation ratios of DCNF on topologies with 8 nodes

2) Approximation Ratio Verification: We do simulations on the number of chain m from 1 to 5 with 5 VNFs for each chain. Here we use the MIP solver to compute the optimal solutions of SFCP. The approximation ratio in Fig. 3 is computed by dividing the total cost of DCNF by that of OPT in each simulation.

In Fig. 3a, we can see the ratios of DCNF to OPT in different network topologies are among interval $[1, 2.375]$, far smaller than the upper bound proved in Theorem 4. Besides, comparing the outcomes of different topologies, we can find DCNF works better on the topologies with fewer links or higher symmetry.

Fig. 3b shows the performance of DCNF with different numbers of SFCs. The results with 1 and 2 chains demonstrate the cases limited at the edge when DCNF is reduced to CNF. The results with 3 and 4 chains are the critical cases when DCNF may need cloud resources while OPT need not. And the results with 5 chains are the cases when there are no adequate resources at the edge and cloud resources must be used. In Fig. 3b, we can see that the worst performance of DCNF is achieved on the critical point when the edge is just enough by OPT.



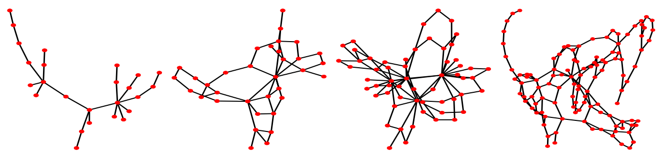
(a) 2 chains (b) 5 chains

Fig. 4: Performance comparisons of diff. algorithms on Topo1

3) Performance Comparisons with Benchmarks: In this section, we do simulations on Topo 1 with $m = 2$ when SFCP is limited at the edge and $m = 5$ when cloud resources must be used. In each simulation, we run Rd, FFSP, BFSP, DCNF, OPT. The different performances of them are displayed in Fig. 4. As for $m = 2, 5$, DCNF always has the best performance compared with three benchmarks.

C. Simulations on 4 Large-Scale Real Network Topologies

1) Network Topology and Simulation Setup: Moreover, we also perform simulations on 4 larger real network topologies, shown in Fig. 5, from the Internet topology zoo [32]: (1) AMRES (25 nodes and 24 links), (2) ARNES (34 nodes and 46 links), (3) DFN (58 nodes and 87 links), (4) ITCDeltacom (113 nodes and 160 links).



(a) AMRES (b) ARNES (c) DFN (d) ITCDeltacom

Fig. 5: Different real network topologies

Let the capacities of edge servers follows the normal distribution of mean of 12 and standard variance of 4, with the limitation of no less than 6. Additionally, we set $\alpha = 1, \beta = 1, \gamma = 2, \zeta = 1, f_{i,j} \sim \mathbb{N}(2, 0.5), l_i \sim \mathbb{N}(\overline{f_i}, 0.25), L_i \sim \mathbb{N}(\overline{f_i}, 0.25)$, where $\overline{f_i}$ is the mean size of all VNFs $F_{i,j}$ of a SFC i.

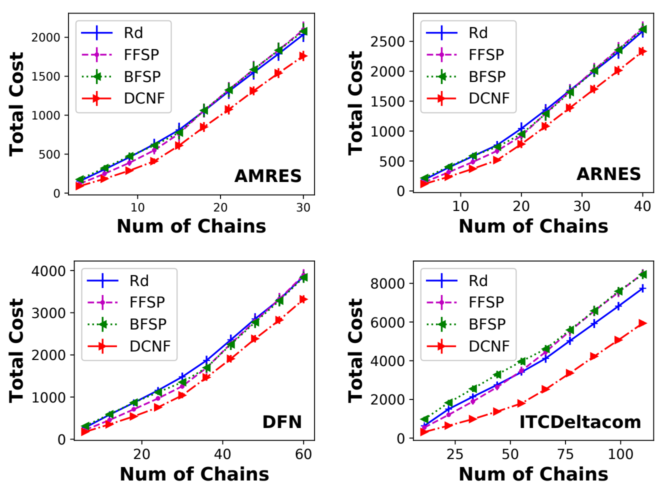


Fig. 6: Performance comparisons of different algorithms on 4 different real network topologies, respectively called AMRES, ARNES, DFN, ITCDeltacom.

2) Performance Comparisons with Benchmarks: In this part, we do simulations on 4 different real network topologies with different m. In each simulation, all three benchmarks and our DCNF are operated. In Fig. 6, we can see DCNF always have a better performance than three benchmarks.

3) Time Cost Comparisons with Benchmarks: In Fig. 7, we draw the time cost of simulations on the network topology called ITCDeltacom by different algorithms. In the figure, we

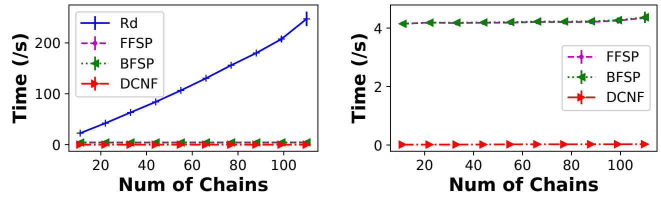
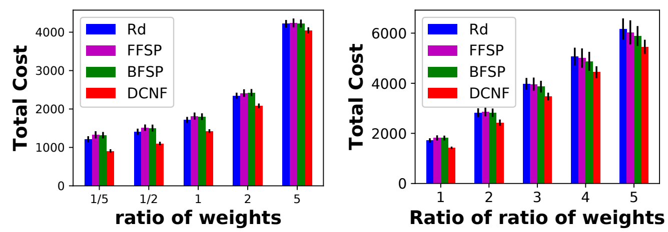


Fig. 7: Time cost of of different algorithms on Deltacom

can get the conclusion that DCNF is a very fast algorithm. It has a dramatic advantage compared with the three benchmarks.



(a) ratio $= \frac{\alpha}{\beta} = \frac{\gamma}{\zeta}, \beta = \zeta = 1$ (b) ratio $= \frac{\gamma}{\alpha} = \frac{\zeta}{\beta}, \alpha = \beta = 1$

Fig. 8: Performance comparisons of different algorithms with different settings of weighting factors

4) Performance Comparisons with Different Settings of Weighting Factors: Here we do simulations with 30 SFCs and different settings of weighting factors on the network topology called AMRES. In detail, we first adjust the relative importance between resource weights and latency weights by setting the ratio of $\frac{\alpha}{\beta}$, $\frac{\gamma}{\zeta}$ from $\frac{1}{5}$ to 5. Besides we also adjust the relative importance between cloud weights and edge weights by setting the ratio of $\frac{\gamma}{\alpha}$, $\frac{\zeta}{\beta}$ from 1 to 5. Fig. 8 reveals DCNF always has a better performance than the benchmarks in all cases.

VII. CONCLUSION

In this paper, we propose an approximation algorithm, DCNF, to deploy SFCs on the edge and the cloud. It balances the resource and latency and optimizes the sweet-spot of resource cost on the edge and the cloud and all corresponding communication latency. Specifically, the Next Fit (NF) strategy employed in it ensures efficient resource utilization and avoids the redundant data traffic. Besides, a designed sub-algorithm called double spanning tree (DST) for virtual network embedding plays a critical role in reducing latency. In all, we prove DCNF has a constant approximation ratio to the optimal solution. Simulation results demonstrate its superiority. Due to the space limitation, in this paper, we only solves SFCP in a batch manner. In future, we will further settle SFCP in online manner. Since NF can sequentially deal with SFCs in their arriving order, we can promote our CNF to the online manner by simply substituting the nearest neighbor algorithm for DST.

VIII. ACKNOWLEDGE

This research work was supported in part by the U.S. National Science Foundation under grant number CCF-1526162, CCF-1717731.

REFERENCES

[1] ETSI, and NFVISG, "Network Functions Virtualization-Introductory White Paper," *SDN and OpenFlow World Congress*, 2012.

[2] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," *2014 IEEE 3rd International Conference on the cloud Networking(CloudNet)*, pp.7–13, 2014.

[3] W. Shi, J. Cao, et al, "Edge computing: Vision and challenges," *IEEE internet of things* vol. 3, no. 5, pp. 637–646, 2016.

[4] D. S. Johnson, "Near-optimal bin packing algorithms," *PhD diss., Massachusetts Institute of Technology*, 1973.

[5] P. Jin, X. Fei, et al, "Latency-aware VNF Chain Deployment with Efficient Resource Reuse at Network Edge,"*IEEE Conference on Computer Communications (INFOCOM)*, pp.267–276, 2020.

[6] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal VNF placement at the network edge,"*IEEE Conference on Computer Communications (INFOCOM)*, pp. 693–701, 2018.

[7] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, PP. 2179–2192, 2018.

[8] H. Ren, Z. Xu, W. Liang, Q. Xia, P. Zhou, O. F. Rana, A. Galis, and G. Wu, "Efficient algorithms for delay-aware NFV-enabled multicasting in mobile edge clouds with resource sharing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 9, pp. 2050–2066, 2020.

[9] J. Son, and R. Buyya, "Latency-aware Virtualized Network Function provisioning for distributed edge clouds,"*Journal of Systems and Software* no. 152,pp. 24–31, 2019.

[10] J. Martín-Peréz, F. Malandrino, et al, "OKpi: All-KPI Network Slicing Through Efficient Resource Allocation,"*IEEE Conference on Computer Communications (INFOCOM)*, pp. 804–813, 2020.

[11] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang, and X. Fu, "Delay-aware virtual network function placement and routing in edge clouds,"*IEEE Transactions on Mobile Computing*, 2019.

[12] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary. "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 98–106, 2015.

[13] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," *11th International Conference on Network and Service Management (CNSM)*, pp. 50–56, 2015.

[14] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," *2015 IEEE 4th International Conference on the cloud Networking (CloudNet)*, pp. 171–177, 2015.

[15] R. Cohen, L. L. Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," *IEEE Conference on Computer Communications (INFOCOM)*, pp. 1346–1354, 2015.

[16] X. Shang, Z. Liu, and Y. Yang, "Network Congestion-aware Online Service Function Chain Placement and Load Balancing," *Proc. of the 48th International Conference on Parallel Processing*, pp.1–10, 2019.

[17] D. Li, P. Hong, and K. Xue, "Virtual network function placement considering resource optimization and SFC requests in cloud datacenter," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 7, pp.1664–1677, 2018.

[18] Y. Sang, B. Ji, et al, "Provably efficient algorithms for joint placement and allocation of virtual network functions," *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pp.1–9, 2017.

[19] S. Agarwal, F. Malandrino, C. F. Chiasserini, and S. De, "Joint VNF placement and CPU allocation in 5G,"*IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 1943-1951, 2018.

[20] J. liu, H. Xu, G. Zhao, C. Qian, X. Fan, and L. Huang, "Incremental Server Deployment for Scalable NFV-enabled Networks,"*IEEE Conference on Computer Communications (INFOCOM)*, pp. 2361–2370, 2020.

[21] X. Shang, Y. Huang, Z. Liu, and Y. Yang, "Reducing the Service Function Chain Backup Cost over the Edge and Cloud by a Self-adapting Scheme,"*IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 2096–2105, 2020.

[22] D. Zheng, C. Peng, X. Liao, L. Tian, G. Luo, and X. Cao, "Towards Latency Optimization in Hybrid Service Function Chain Composition and Embedding,"*IEEE Conference on Computer Communications (INFOCOM)*, pp. 1539–1548, 2020.

[23] V. Valls, G. Iosifidis, G. d. Mel, and L. Tassiulas, "Online Network Flow Optimization for Multi-Grade Service Chains,"*IEEE Conference on Computer Communications (INFOCOM)*, pp. 1329–1338, 2020.

[24] Z. Wan, "Cloud Computing infrastructure for latency sensitive applications," *IEEE 12th International Conference on Communication Technology*, pp. 1399–1402, 2010.

[25] C. Long, Y. Cao, T. Jiang and Q. Zhang, "Edge computing framework for cooperative video processing in multimedia IoT systems," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp.1126–1139, 2017.

[26] Y. Huang, Y. Lu, F. Wang, X. Fan, J. Liu and V.C. Leung, "An edge computing framework for real-time monitoring in smart grid," *IEEE International Conference on Industrial Internet (ICII)*, pp. 99–108, 2018.

[27] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp.1738–1762, 2019.

[28] Y. Sahni, J. Cao, L. Yang and Y. Ji, "Multi-Hop Multi-Task Partial Computation Offloading in Collaborative Edge Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp.1133–1145, 2020.

[29] Q. Xia, W. Ye, Z. Tao, J. Wu and Q. Li, "A Survey of Federated Learning for Edge Computing: Research Problems and Solutions," *High-Confidence Computing*, 2021.

[30] C. Zhan, H. Hu, Z. Liu, Z. Wang and Shiwen Mao, "Multi-UAV-Enabled Mobile Edge Computing for Time-Constrained IoT Applications," *IEEE Internet of Things Journal*, 2021.

[31] K. Magzhan, and H. M. Jani, "A review and evaluations of shortest path algorithms," *International journal of scientific and technology research*, vol. 2, no. 6, pp. 99–104, 2013.

[32] S.Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo,"*IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.