Service Function Chain Placement in Edge Computing: A Topological Dependency-Informed Approach

Weibin Ma, Lena Mashayekhy
Department of Computer and Information Sciences, University of Delaware, Newark, Delaware 19716, USA
{weibinma, mlena}@udel.edu

***Abstract*—The integration of Network Function Virtualization (NFV) and Mobile Edge Computing (MEC) allows for efficient, advanced network services via Service Function Chains (SFCs). SFCs are sequences of ordered network functions designed to provide specific network services. However, the placement of SFCs is critical, especially for latency-sensitive applications such as telemedicine, due to spatial proximity between service functions, their processing order, and limited edge resources. This paper addresses the multi-SFC placement problem in MEC-NFV networks, aiming to reduce both deploying cost and routing cost. We propose an innovative algorithm based on topological sort, called TD-NFP, to solve this problem. The experimental results show that the proposed TD-NFP approach outperforms other benchmarks.**

***Index Terms*—Mobile Edge Computing, Network Function Virtualization, Service Function Chain Placement, Directed Acyclic Graph, Topological Sort.**

I. INTRODUCTION

Mobile Edge Computing (MEC) represents a pivotal shift in distributed cloud architecture, bringing significant computation and storage resources to the edge of networks through localized mini-datacenters or *cloudlets* [1]–[3]. Simultaneously, Network Function Virtualization (NFV) is transforming the management and operation of networks [4], [5]. NFV replaces traditional network devices with *Virtualized Network Functions (VNFs)*, providing efficient, cost-effective network operations.

The integration of MEC and NFV (i.e., MEC-NFV) allows for efficient, advanced network services via *Service Function Chains (SFCs)* [6]. An SFC comprises an ordered set of VNFs that must execute in a specific sequence to provide a network service. Each VNF in an SFC performs a specific function and passes its output to the next VNF in the sequence. These VNFs are executed on commodity servers within cloudlets to satisfy the service requests of mobile users. Consider a smart city application scenario, where a variety of network services are required for different functions. Smart traffic control systems require one sequence of network functions to prioritize, secure, and establish connections for real-time traffic video feeds, while emergency response systems require another sequence to direct, secure, and balance large data loads for high-priority alerts and video feeds from emergency scenes. These sequences, or SFCs, might share common network functions and need to operate simultaneously.

A significant research gap exists in efficiently placing multiple SFCs among cloudlets, which is more complex compared to traditional placement of independent VNFs due to the dependencies among VNFs in a chain and the constraints on available resources in cloudlets. The placement of SFCs can have a profound impact on network performance, influencing factors such as latency and load balancing, and the two key costs—deploying and routing. The deploying cost arises from deploying VNFs in cloudlets, while the routing cost originates from the sequential transfer of outputs for each SFC. Lowering total cost involves enabling multiple SFCs to share some common VNFs [7], [8], making the SFC placement problem more challenging. Inefficient SFC placement could lead to higher latency and unbalanced resource utilization, which could significantly degrade user experience and overall network performance. This motivates our study on developing efficient and effective multi-SFC placement strategies in MEC-NFV networks.

In this paper, we propose a novel approach to the SFC placement problem in MEC-NFV networks to reduce the deploying and routing costs while considering the complexities introduced by SFCs that share common VNFs. Instead of placing each SFC as a whole or placing SFCs sequentially, we introduce an approach constructing multiple SFCs into a Directed Acyclic Graph (DAG), called S-DAG, and then we propose an efficient topology-based approach to divide the complex multi-SFC placement problem into multiple independent VNF placement sub-problems. Each sub-problem focuses on solving the VNF placement problem while considering the dependencies between VNFs from the same SFC. Specifically, we introduce the topological sort algorithm to detach dependencies and group a set of independent VNFs. At each level, topological sort ensures that VNFs do not have any dependencies between each other. Then we focus on placing VNFs while considering the dependencies during the calculation of total cost for placing each VNF in each cloudlet. Our study is, to the best of our knowledge, the first to address the multi-SFC placement problem by constructing a DAG for SFCs and using a topological sort. This innovative approach is driven by a crucial need for more efficient and effective SFC placement strategies to improve network performance in MEC-NFV networks.

The rest of the paper is organized as follows. Section II reviews related work. The system model and problem formulation are described in Section III. In Section IV, we describe the proposed topological dependency-informed approach, TD-NFP. The performance evaluation is carried out in Section V.

Section VI concludes the paper and suggests future directions.

II. Related Work

In this section, we review existing studies that are related to our work from different standpoints.

VNF Placement. Many studies have primarily focused on efficient deployment of VNFs in MEC-NFV networks [9]–[12]. These studies have tackled various aspects, including capacitated allocation problem, minimization of latency, and reliability requirements. Fairstein *et al.* [9] formulated the VNF placement among MEC servers with capacitated resources and proposed multiple approximated algorithms to solve it. Yala *et al.* [10] investigated the VNF placement problem in MEC-NFV networks and formulated it as an optimization problem with the objective of minimizing access latency and maximizing service availability. They developed a genetic algorithm to achieve near-optimal solutions. Huang *et al.* [11] studied the reliability-aware VNF placement problem in MEC by considering the specific reliability requirements of mobile users. They proved the NP-hardness of the problem and proposed an approximation algorithm with a logarithmic approximation ratio. However, these studies mainly focused on individual VNF placement, and the dependencies among VNFs within the same network service were not considered.

SFC Placement. More recently, the SFC placement problem in MEC-NFV networks has received increasing attention [13]–[15]. Chen *et al.* [13] showed the SFC problem is NP-Complete and formulated it as a graph-based problem. They devised a heuristic algorithm that achieves a tradeoff between optimality and running time. Wang *et al.* [14] studied the SFC problem in MEC-NFV networks and reformulated it into two sub-problems. To solve them, they proposed a Hungarian-based algorithm. Yin *et al.* [15] also investigated the SFC problem in MEC-NFV networks while considering the SFC availability. They proposed a backup model to improve the SFC availability while reducing resource consumption. Moreover, a dynamic programming-based algorithm was developed. However, these works either concentrated solely on operating / deploying costs associated with VNF placement, routing costs resulting from dependencies, or sequentially placing SFCs.

Handling VNF Dependencies. Some studies have considered the dependencies among VNFs in the SFC placement problem [7], [16], [17]. A closely related study [7] to our work grouped SFCs in NFV-based Networks by proposing a heuristic solution built on the classic k-means algorithm. However, it is worth noting that our work substantially differs from this work. While [7] mainly aimed at identifying the optimal groups for the SFCs based on particular metrics, our approach extends this concept by adding a strong focus on the efficient placement of VNFs from SFCs in resource-limited edge networks. More importantly, we offer a novel perspective on the complexity of dependencies in the SFC placement problem. By transforming the SFC placement into a more manageable S-DAG placement problem, we are able to detach dependencies when solving it. This not only enhances the efficiency of our solution but also presents a new approach to handle dependencies in the SFC placement problem in MEC-NFV networks.

In summary, prior studies either focused on the individual VNF placement without considering the dependencies among VNFs, placing the SFC as a whole, or placed SFCs sequentially. In this paper, we design a novel approach to address the SFC placement problem in MEC-NFV networks. Our approach uniquely considers both the dependencies among VNFs within the same SFC and the limited resources of edge networks.

III. System Model and Problem Formulation

We consider a MEC-NFV network consisting of a network operator responsible for managing service deployment and a set of cloudlets distributed within the MEC-NFV network, providing various computation and storage resources for processing requested edge services. The cloudlets are interconnected by a high-speed network such as a wide-area network (WAN) or a local area network (LAN) [18]–[21]. We denote a set of cloudlets by $\mathcal{M} = \{1, ..., M\}$. Each cloudlet $j \in \mathcal{M}$ has varying resource capacity $B_j \in \mathcal{B}$ (e.g., computational or storage resources), where $\mathcal{B}$ represents the set of resource capacities for all cloudlets. We denote a set of VNFs by $\mathcal{V} = \{V_1, ..., V_K\}$ where K is the total number of VNF types in the MEC-NFV network. We denote a set of SFCs by $\mathcal{N} = \{\mathbf{v}_1, ..., \mathbf{v}_N\}$ where each SFC is composed of a sequence of ordered VNFs. Specifically, an SFC $\mathbf{v}_i \in \mathcal{N}$ is represented by a set of ordered VNFs $\mathbf{v}_i = \langle V_{i_1}, ..., V_{i_l} \rangle$, such that for $i_k \in \{i_1, ..., i_l\}$, we have $1 \le i_k \le K$ and $V_{i_k} \in \mathcal{V}$. Note that all VNFs in an SFC obey a specific order for that SFC. The required deployment resources for each VNF $V_{i_k} \in \mathcal{V}$ in each cloudlet $j \in \mathcal{M}$ is denoted as $W_{i_k j} \in \mathcal{W}$, with $\mathcal{W}$ representing the set of resource requirements for all VNFs. Thus, the deployment of each VNF V_{i_k} of SFC $\mathbf{v}_i$ on a cloudlet j must not exceed its available resources B_j.

We define $x_{i_k j} \in \{0, 1\}$ as the VNF selection decision variable such that $x_{i_k j} = 1$ if VNF V_{i_k} of SFC $\mathbf{v}_i$ is selected from cloudlet j, and 0 otherwise. This variable determines whether to select VNF V_{i_k} in cloudlet j for processing SFC $\mathbf{v}_i$ or not.

The deploying cost model represents the cost associated with deploying a VNF in a cloudlet. The deploying cost for placing VNF V_{i_k} of SFC $\mathbf{v}_i$ in cloudlet j and the total deploying cost for placing SFC $\mathbf{v}_i$ are represented as $c^d_{i_k j}$ and C^d_i, respectively. As a result, C^d_i can be calculated by:

$$C^d_i = \sum_{V_{i_k} \in \mathbf{v}_i} \sum_{j \in \mathcal{M}} x_{i_k j} c^d_{i_k j} \tag{1}$$

The routing cost model accounts for the dependency between consecutive VNFs in the same SFC. Each VNF in an SFC performs a specific function and transfers its output to the next VNF in the sequence. The routing cost model indicates the cost of transferring the output of one VNF to the next in the chain.

We denote the routing cost of transferring the output of VNF $V_{i_{k-1}}$ of SFC $\mathbf{v}_i$ from cloudlet j' (where VNF $V_{i_{k-1}}$ is placed) to cloudlet j (where VNF V_{i_k} is placed) as $c^r_{j'j}$. The total routing cost for SFC $\mathbf{v}_i$ is denoted as C^r_i and defined by:

$$C^r_i = \sum_{V_{i_k}\in\mathbf{v}_i}\sum_{j\in\mathcal{M}} x_{i_kj}\Big(\sum_{V_{i_{k-1}}\in\mathbf{v}_i}\sum_{j'\in\mathcal{M}} x_{i_{k-1}j'}c^r_{j'j}\Big) \quad (2)$$

Obviously, there is no routing cost incurred when two consecutive VNFs of the same SFC are placed in the same cloudlet.

We now formulate the SFC placement problem. Our goal is to minimize the total cost, which is the summation of the total deploying cost and the total routing cost if triggered. The dependencies between VNFs within each SFC is captured through the decision variable x_{i_kj}. The SFC placement problem is defined as follows:

$$\text{Minimize } \mathcal{C} = \sum_{i\in\mathcal{N}}(C^d_i + C^r_i) \quad (3)$$

Subject to:

$$\sum_{j\in\mathcal{M}} x_{i_kj} = 1, \qquad \forall V_{i_k}\in\mathbf{v}_i, i\in\mathcal{N}, \quad (4)$$

$$\sum_{i\in\mathcal{N}}\sum_{V_{i_k}\in\mathbf{v}_i} x_{i_kj}W_{i_kj} \le \mathcal{B}_j, \qquad \forall j\in\mathcal{M}, \quad (5)$$

$$x_{i_kj}\in\{0,1\}, \qquad \forall V_{i_k}\in\mathbf{v}_i, i\in\mathcal{N}, j\in\mathcal{M}. \quad (6)$$

The objective function (3) is to minimize the total costs of placing all SFCs in MEC-NFV networks. Constraint (4) ensures that each VNF is deployed and processed in only one cloudlet at once. Constraint (5) guarantees that the total required deployment resources for all VNFs in each cloudlet do not exceed its resource capacity. This prevents overloading any single cloudlet and ensures that each deployed VNF has sufficient resources for its operation. Constraint (6) is to ensure that the decision variable is binary.

The above SFC placement problem is NP-hard, which can be proved through a polynomial-time reduction from a well-known NP-hard problem called Generalized Assignment Problem (GAP) [22]. Moreover, it becomes more challenging to consider the dependency between consecutive VNFs in the same SFC when calculating the routing cost. To solve this challenging problem, we propose a topological dependency-informed approach (TD-NFP) that will be introduced in the next section.

IV. Topological Dependency-informed Network Function Placement

We introduce our proposed Topological Dependency-informed Network Function Placement (TD-NFP) approach that leverages graph theory to efficiently place SFCs on resource-limited cloudlets in MEC-NFV networks. Our approach involves two main steps: first constructing SFCs into a DAG formation, called S-DAG, to capture dependencies between VNFs, and then employing an efficient topological-based approach to properly place the VNFs based on the constructed S-DAG to cloudlets. For simplicity, we assume that provided SFCs can be constructed as a single DAG. This is a reasonable assumption as any cycles in the constructed DAG can be broken down into multiple DAGs. The implementation of our TD-NFP is summarized in Algorithm 1.

Algorithm 1 Topological Dependency-informed Network Function Placement (TD-NFP)

```
1:  Input: $\mathcal{M},\mathcal{N},\mathcal{W},\mathcal{B}$
2:  $\mathcal{C} = 0$ /*Total cost*/
3:  Initialize $\mathcal{A}, I, P$ as empty dictionaries $\emptyset$
4:  Initialize $Q$ as an empty queue
5:  S-DAG $G(V,E) \leftarrow$ DAGConstructor($\mathcal{N}$)
6:  for each $v \in V$ do
7:      for each successor $w$ of $v$ in S-DAG do
8:          $I[w]$ += 1
9:          $P[w]$.insert($v$)
10: for each $v \in V$ do
11:     if $I[v]$ = 0 then
12:         $Q$.enqueue($v$)
13: while $Q$ is not empty do
14:     /*Grouping VNFs*/
15:     $L \leftarrow \emptyset$ /* collect VNFs for current level*/
16:     $S \leftarrow Q$.size()
17:     for $i$ = 1 to $S$ do
18:         $v \leftarrow Q$.dequeue()
19:         $L$.insert($v$)
20:         for each successor $w$ of $v$ in S-DAG do
21:             $I[w]$ -= 1
22:             if $I[w]$ = 0 then
23:                 $Q$.enqueue($w$)
24:     /*Placement of VNF Groups*/
25:     $(C, A) \leftarrow$ Place($\mathcal{M},\mathcal{W},\mathcal{B},L,P,\mathcal{A}$)
26:     $\mathcal{C}$ += $C$
27:     $\mathcal{A} \leftarrow \mathcal{A} \cup A$
28: Output: $\mathcal{C},\mathcal{A}$
```

The algorithm is provided with four inputs: $\mathcal{M}$, a set of cloudlets; $\mathcal{N}$, a set of SFCs; $\mathcal{W}$, a set of resources requirements for all VNFs; and $\mathcal{B}$, a set of resource capacities for all cloudlets. TD-NFP initializes several variables: $\mathcal{A}$ to store the assignment of VNFs of the S-DAG to cloudlets (Line 3), Q as an empty queue for processing the VNFs (Line 4), and I and P two dictionaries (Line 3), where I maps each VNF to its indegree, indicating the number of unprocessed predecessors it has, while P maps each VNF to its predecessor(s), which will be used to calculate routing costs. Initially, TD-NFP constructs an S-DAG, $G(V,E)$, from the given set of SFCs $\mathcal{N}$ using the DAGConstructor algorithm, shown in Algorithm 2.

After obtaining the constructed S-DAG, TD-NFP calculates the indegree and a set of predecessor(s) for each VNF in S-DAG (Lines 6-9). VNFs with zero indegree (first level nodes) are immediately ready for deployment and are added to the queue to be processed in the first iteration (Lines 10-12). The algorithm proceeds to the deployment phase for all the VNFs

Algorithm 2 DAGConstructor

```
1: Input: $\mathcal{N}$
2: $G \leftarrow$ an empty dictionary $\emptyset$
3: for each SFC $\mathbf{v}_i \in \mathcal{N}$ do
4:     for each VNF $V_{i_k}$ in SFC $\mathbf{v}_i$ do
5:         if $V_{i_k} \notin G$ then
6:             $G[V_{i_k}] \leftarrow$ an empty {create a node}
7:         $G[V_{i_{k-1}}]$.append($V_{i_k}$) {add an edge}
8: Output: $G$
```

Algorithm 3 Place

```
1: Input: $\mathcal{M}, \mathcal{W}, \mathcal{B}, L, P, \mathcal{A}$
2: $H \leftarrow$ an empty cost dictionary $\emptyset$
3: for each VNF $v \in L$ do
4:     $P_v \leftarrow P[v]$ the predecessors of VNF $v$ in S-DAG
5:     for each cloudlet $j \in \mathcal{M}$ do
6:         $H[v,j] = c^d_{vj}$ deploying cost of placing VNF $v$ in cloudlet $j$
7:         for each predecessor $u$ of $v$ in $P_v$ do
8:             $j' \leftarrow \mathcal{A}[u]$ cloudlet that $u$ is placed in
9:             $H[v,j]$+= $c^r_{j'j}$ routing cost of sending data from $u$ in cloudlet $j'$ to $v$ in cloudlet $j$
10: $(C, A) \leftarrow Assign(L, \mathcal{W}, \mathcal{B}, H)$
11: Output: $C, A$
```

added to the queue.

For each level of the constructed S-DAG, TD-NFP uses a topological sorting approach to group VNFs that have no unprocessed predecessors (i.e., zero indegree) and hence can be placed independently (Lines 14-23). Specifically, it forms a group of VNFs with no predecessor(s) into L (Line 19), meanwhile, the indegree value of each successor of these VNFs is decreased by 1 (Line 21) and a set of new VNFs with no predecessor(s) are appended into Q for processing for the next level (Lines 22-23). After completing the grouping phase for a level, TD-NFP calls the VNF Placement function, Place, presented in Algorithm 3, to solve the VNF placement sub-problem (Lines 25) for the collected VNFs of each group. The Place function returns the cost and assignment for each VNF in the current group.

This two-phase approach ensures that TD-NFP places VNFs onto cloudlets in an efficient and cost-effective manner. At the end, TD-NFP returns the obtained cost value and the corresponding placement assignment for all VNFs.

DAGConstructor, Algorithm 2, initializes an empty dictionary G for constructing the S-DAG. This dictionary stores the mapping of each VNF to a set of its successors by using a VNF as a key and a set of its successors as its associated values. In doing so, it traverses each SFC $\mathbf{v}_i \in \mathcal{N}$. For every VNF V_{i_k} in the SFC, if the VNF does not already exist as a key in G, it is added to the dictionary and associated with an empty list (Lines 5-6). Subsequently, a directed edge, representing the dependency between the VNF V_{i_k} and its predecessor $V_{i_{k-1}}$ is added into G (Line 7). Upon completing

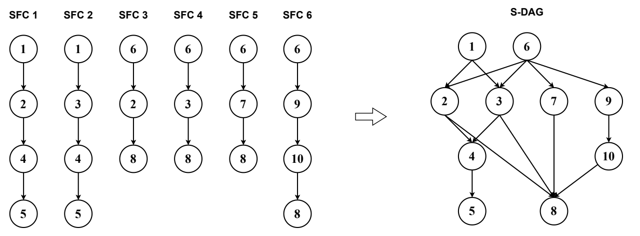


Fig. 1: Example of the S-DAG representation for six SFCs.

this process for all VNFs across all SFCs, the constructed S-DAG G is returned. The example shown in Fig. 1 represents a system with ten VNFs and six SFCs composed of a subset of these VNFs (left side) and the constructed S-DAG (right side). As shown in the figure, the SFCs share some VNFs to complete their processes, and we obtain four groups of VNFs as follows: $\{1,6\} \rightarrow \{2,3,7,9\} \rightarrow \{4,10\} \rightarrow \{5,8\}$ based on the topological sort.

Place function, Algorithm 3, takes $\mathcal{M}, \mathcal{W}, \mathcal{B}$, and the current group of VNFs, L, as inputs. It begins by initializing a cost dictionary H to empty for computing and storing the total cost associated with deploying all VNFs in the group L to all cloudlets in $\mathcal{M}$ (Line 3). Each element $H[(v,j)]$ of the cost dictionary represents the total cost of placing VNF v in cloudlet j, including both deploying cost and routing cost from the VNF's predecessors. The predecessors of each VNF v are derived from the S-DAG and are represented as P_v (Line 4). These predecessors need to process and pass data to VNF v. The value $H[(v,j)]$ encompasses the direct deploying cost, c^d_{vj}, of placing VNF v in cloudlet j and the routing costs, $c^r_{j'j}$, incurred when transferring data from v's predecessors to VNF v (Lines 5-9). The routing cost is an aggregate value, accumulated based on the placement of the predecessors. Upon determining the total cost values of placing VNFs on cloudlets stored in H, Place function solves the VNF placement problem by calling $Assign()$ function. The $Assign()$ function returns the minimum total cost and the corresponding placement assignments for the VNFs (Line 10). This function can be formulated as a general assignment problem (GAP) and solved using approximation algorithms or heuristics.

V. Experimental Results

To evaluate the performance of our approach, TD-NFP, for solving the multi-SFC placement problem with dependencies, it is compared with the following benchmarks:

- **Optimum (OPT) Policy**: This benchmark represents the theoretical best-case scenario. For each VNF of SFCs, the OPT finds an optimal placement minimizing the total cost as defined in Eq (3). The optimal results are obtained using the commercial solver CPLEX.
- **Least Load (LL) Policy**: This policy places each VNF of the SFCs in a cloudlet which currently serves the fewest VNFs. This strategy is a common load balancing approach aimed at ensuring a fair distribution of VNFs among available cloudlets.

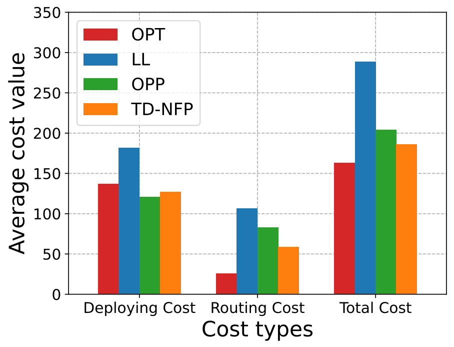


Fig. 2: Obtained cost

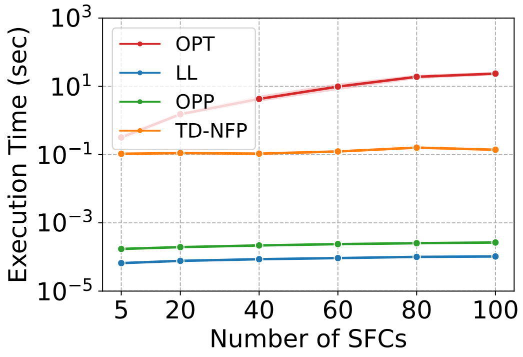


Fig. 3: Execution time

- **Optimal Path Placement (OPP) Policy**: S-DAG is composed of a set of paths, representing the SFCs, covering all VNFs. In this policy, each path (i.e., an individual SFC) is placed optimally among cloudlets to minimize the total sum.

All experiments were conducted using Python 3.9.4 on an Apple M1 Pro Chip with 16 GB RAM.

A. Experimental Setup

We consider multiple heterogeneous cloudlets deployed within the MEC networks, with multiple SFCs with limited number of different VNFs. For instance, an SFC for a surveillance application consists of six distinct VNFs [23]. For our experiments, the MEC initially has 10 cloudlets. There are 15 VNF types, and each SFC has a chain length varying from 1 to 10 VNFs chosen from the available VNF types. We consider 200 unique S-DAGs. To further analyze the algorithms, we examine different scenarios with the number of cloudlets ranging from 4 to 24, the number of distinct VNFs (VNF types) varying from 10 to 35, and the number of SFCs ranging from 5 to 100.

The total resource capacity for each cloudlet follows a uniform distribution between 5 to 25 units. The resource requirements for each VNF are uniformly selected to be between 1 to 3 units. Operational and routing costs are modeled as follows. The deploying cost for each VNF, considering potential multiple predecessors within an S-DAG, is uniformly selected from 5 to 15 units. Similarly, the routing cost between any two cloudlets is uniformly selected from 1 to 4 units. We repeat each scenario 30 times to obtain the average results.

B. Experimental Analysis

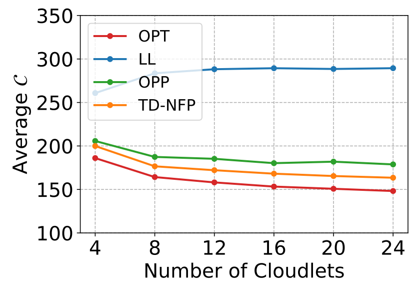
We evaluate the performance of our approach compared to the benchmarks considering various metrics.

Fig. 2 exhibits the average deploying cost (Eq (1)), routing cost (Eq (2)), and total cost (objective value in Eq (3)) obtained by the approaches across 200 distinct S-DAGs. LL performs the worst due to its focus on evenly distributing VNFs among cloudlets, neglecting deploying and routing costs. OPT presents the optimal total costs. OPP excels in deploying cost by optimally placing each individual SFC. However, since it does not account for dependencies from other SFCs, it incurs a higher routing cost. Both OPP and our proposed TD-NFP perform well with their obtained cost being near to the optimal (obtained by OPT). This is due to the fact that they both place VNFs among cloudlets considering the deploying cost and routing cost. However, TD-NFP outperforms OPP by solving VNF placement sub-problems with consideration of dependencies between VNFs from the same SFC, leading to a lower average total cost.

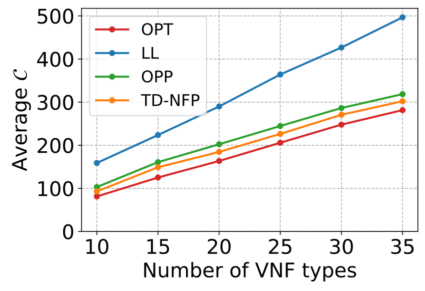
We analyze the execution time for solving the multi-SFC placement problem, with respect to number of SFCs. Fig. 3 shows the obtained results. LL, while not optimizing deploying and routing costs, achieves the fastest solution at about 0.05 milliseconds. OPP calculates a solution in roughly 0.2 milliseconds. Our proposed TD-NFP achieves a stable solution in about 0.1 second, making it more practical for real-world applications compared to OPT, which takes over 10 seconds for larger problem sizes. TD-NFP achieves this by detaching dependencies and breaking the original problem into smaller sub-problems, which significantly reduces execution time for solving the placement problem.

We further investigate the impacts of several important parameters on the total cost in Fig. 4. The average cost obtained by the approaches are evaluated using the same S-DAGs. Each scenario is repeated 30 times, and the average results are reported.

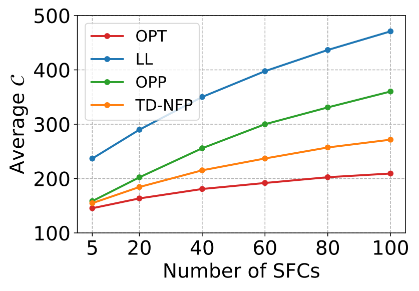
Fig. 4a shows the impact of varying the number of cloudlets. The average cost typically decreases as the number of cloudlets increases, except for the results of LL. An increased number of cloudlets offer more placement options, enhancing the likelihood of obtaining lower-cost assignments. OPP does not consistently yield reduced cost as the number of cloudlets increases due to its limited optimization concerning the routing cost. Fig. 4b shows that the average cost increases with an increase in the number of VNF types. This is likely because more VNF types can result in SFCs with longer chain lengths, leading to higher deploying and routing costs. Fig. 4c illustrates the impact of increasing the number of SFCs on total cost. An increase in the count leads to a higher number of VNFs being shared among the SFCs, and subsequently, it results in increased dependencies within the S-DAGs. The results show with an increase in the number of SFCs, the total



(a) different cloudlets



(b) different VNF types



(c) different SFCs

Fig. 4: Sensitivity analysis of objective value ($\mathcal{C}$)

costs of placement increase. LL performs worse compared to other benchmarks. Our TD-NFP consistently outperforms OPP and is close to OPT, demonstrating its effectiveness in handling complex scenarios with high-intensity S-DAGs, where more dependencies between VNFs exist.

In summary, TD-NFP consistently demonstrated its efficacy, offering competitive placement solutions close to optimal.

VI. Conclusion

In this paper, we addressed the SFC placement problem in MEC-NFV networks. To handle the complexity of placing multiple SFCs among cloudlets, we constructed SFCs as S-DAGs and transformed the problem to a more tractable S-DAG placement problem. Our novel topological dependency-informed approach, TD-NFP, efficiently detaches dependencies and divides the SFC placement problem into smaller subproblems. Experimental analysis demonstrates that TD-NFP outperforms other benchmarks, achieving lower costs while remaining scalable under different settings.

VII. ACKNOWLEDGMENT

This research was supported in part by NSF grant CNS-2145268.

References

[1] N. Sharghivand, L. Mashayekhy, W. Ma, and S. Dustdar, “Time-Constrained Service Handoff for Mobile Edge Computing in 5G,” *IEEE Transactions on Services Computing*, vol. 16, no. 3, pp. 2241–2253, 2023.

[2] D. Bhatta and L. Mashayekhy, “A Bifactor Approximation Algorithm for Cloudlet Placement in Edge Computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1787–1798, 2022.

[3] W. Ma and L. Mashayekhy, “Video Offloading in Mobile Edge Computing: Dealing with Uncertainty,” *IEEE Transactions on Mobile Computing*, pp. 1–14, 2024.

[4] J. Li, S. Guo, W. Liang, Q. Chen, Z. Xu, W. Xu, and A. Y. Zomaya, “Digital Twin-Assisted, SFC-Enabled Service Provisioning in Mobile Edge Computing,” *IEEE Transactions on Mobile Computing*, vol. 23, no. 1, pp. 393–408, 2024.

[5] T. V. Doan, G. T. Nguyen, M. Reisslein, and F. H. P. Fitzek, “SAP: Subchain-Aware NFV Service Placement in Mobile Edge Cloud,” *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 319–341, 2023.

[6] H. Hantouti, N. Benamar, and T. Taleb, “Service Function Chaining in 5G & Beyond Networks: Challenges and Open Research Issues,” *IEEE Network*, vol. 34, no. 4, pp. 320–327, 2020.

[7] Y. Chen, J. Wu, and R. Biswas, “Grouping Service Chains of Multiple Flows in NFV-Based Networks,” *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 377–388, 2020.

[8] D. Harutyunyan, N. Shahriar, R. Boutaba, and R. Riggio, “Latency-Aware Service Function Chain Placement in 5G Mobile Networks,” in *Proc. of the IEEE Conference on Network Softwarization*, 2019, pp. 133–141.

[9] Y. Fairstein, D. Harris, J. Naor, and D. Raz, “NFV Placement in Resource-Scarce Edge Nodes,” in *Proc. of the IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing*, 2020, pp. 51–60.

[10] L. Yala, P. A. Frangoudis, and A. Ksentini, “Latency and Availability Driven VNF Placement in a MEC-NFV Environment,” in *Proc. of the IEEE Global Communications Conference*, 2018, pp. 1–7.

[11] M. Huang, W. Liang, X. Shen, Y. Ma, and H. Kan, “Reliability-Aware Virtualized Network Function Services Provisioning in Mobile Edge Computing,” *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2699–2713, 2019.

[12] N. Kiran, X. Liu, S. Wang, and C. Yin, “VNF Placement and Resource Allocation in SDN/NFV-Enabled MEC Networks,” in *Proc. of the IEEE Wireless Communications and Networking Conference Workshops*, 2020, pp. 1–6.

[13] Z. Chen, S. Zhang, C. Wang, Z. Qian, M. Xiao, J. Wu, and I. Jawhar, “A Novel Algorithm for NFV Chain Placement in Edge Computing Environments,” in *Proc. of the IEEE Global Communications Conference*, 2018, pp. 1–6.

[14] M. Wang, B. Cheng, W. Feng, and J. Chen, “An Efficient Service Function Chain Placement Algorithm in a MEC-NFV Environment,” in *Proc. of the IEEE Global Communications Conference*, 2019, pp. 1–6.

[15] X. Yin, B. Cheng, M. Wang, and J. Chen, “Availability-aware Service Function Chain Placement in Mobile Edge Computing,” in *Proc. of the IEEE World Congress on Services*, 2020, pp. 69–74.

[16] H. Hawilo, M. Jammal, and A. Shami, “Network Function Virtualization-Aware Orchestrator for Service Function Chaining Placement in the Cloud,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 643–655, 2019.

[17] F. Tian, J. Liang, and J. Liu, “Joint VNF Parallelization and Deployment in Mobile Edge Networks,” *IEEE Transactions on Wireless Communications*, vol. 22, no. 11, pp. 8185–8199, 2023.

[18] T. Ouyang, Z. Zhou, and X. Chen, “Follow Me at the Edge: Mobility-Aware Dynamic Service Placement for Mobile Edge Computing,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, 2018.

[19] W. Ma and L. Mashayekhy, “Truthful Computation Offloading Mechanisms for Edge Computing,” in *Proc. of the IEEE Intl. Conference on Edge Computing and Scalable Cloud*, 2020, pp. 199–206.

[20] D. Bhatta and L. Mashayekhy, “Physics-Inspired Mobile Cloudlet Placement in Next-Generation Edge Networks,” in *Proc. of the IEEE International Conference on Edge Computing and Communications*, 2022, pp. 159–168.

[21] E. Farhangi Maleki, W. Ma, L. Mashayekhy, and H. La Roche, “QoS-aware Content Delivery in 5G-enabled Edge Computing: Learning-based Approaches,” *IEEE Transactions on Mobile Computing*, pp. 1–13, 2024.

[22] G. T. Ross and R. M. Soland, “A branch and bound algorithm for the generalized assignment problem,” *Mathematical programming*, vol. 8, no. 1, pp. 91–103, 1975.

[23] D. T. Nguyen, C. Pham, K. K. Nguyen, and M. Cheriet, “Placement and Chaining for Run-Time IoT Service Deployment in Edge-Cloud,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 459–472, 2019.