5G Network Slices Embedding with Sharable Virtual Network Functions

Chengli Mei, Jiayi Liu, Jinyan Li, Lei Zhang, and Menghan Shao

Abstract: **Network slicing (NS) is recognized as a key technology for the 5G mobile network in enabling the network to support multiple diversified vertical markets over a shared physical infrastructure with efficiency and flexibility. A 5G NS instance is composed of a set of virtual network function (VNF) instances to form the end-to-end (E2E) virtual network for the slice to operate independently. The deployment of a NS is a typical virtual network embedding (VNE) problem. We consider a scenario in which VNF instances can be shared across multiple slices to further enhance the utilization ratio of the underlying physical resources. For NSs with sharable VNF instances, the deployment of the slice instances is essentially the embedding of multiple virtual networks coupled by the VNFs shared among slices. Hence, we formulate this sharable-VNFs-based multiple coupled VNE problem (SVM-VNE) through an integer linear program (ILP) formulation, and design a back-tracking coordinated virtual network mapping algorithm. Simulation results demonstrate that VNF-sharing can enhance the slice acceptance ratio with the same physical network, which represents higher physical resource utilization. Moreover, our approach achieves higher acceptance ratio by comparing to a baseline algorithm.**

Index Terms: **5G, end-to-end slices, network slicing, virtual network embedding, VNF sharing.**

I. INTRODUCTION

NETWORK slicing is recognized as a key technology for the fifth generation (5G) mobile network in enabling the network to support multiple diversified vertical markets with efficiency and flexibility [1], [2]. Enabled by new maturing paradigms, such as network function virtualization (NFV) and software defined networking (SDN) [3], network slicing empowers the infrastructure provider (InP) to compose multiple separated end-to-end (E2E) logical networks over a shared common physical infrastructure and delivers different types of services based on various requirements and business needs of each vertical. It improves the intelligence of the 5G network, delivers the Network-as-a-Service business model for value creation and reduces both capital expenditure (CAPEX) and operating expense (OPEX) of the InP [4].

The 5G network slice (NS) spans across multiple parts of the 5G network, including the 5G radio access network (RAN), the 5G transport network (TN) and the 5G core network (CN). In the literature, research works related to 5G network slicing can be roughly classified into two categories [5]. Some works have been conducted on virtualizing and softwarizing the radio resource for RAN slicing [6]–[8]; whereas some works, known as E2E network slicing (or CN slicing) investigate the placement of virtual network functions (VNFs) towards the underlying physical infrastructure to form the corresponding virtual network for the slices to operate independently [9], [10]. There are also works design the interfaces and protocols between RAN slices and CN slices [11]. In the current work, we focus on the deployment of E2E 5G NSs, in which each instance of the slice must contain all the required VNFs to compose the NS's service across the whole 5G system, from the RAN to the CN. Hence, the realization of such an E2E 5G NS is essentially the deployment of the corresponding virtual network which involves the optimization of resource allocations by considering network and computing resources. The mapping from the slice instance virtual network towards the physical infrastructure is a typical virtual network embedding (VNE) problem which has been intensively studied in the literature. However, for 5G E2E NS deployment, the specific requirements of the 5G system should be considered to reformulate the VNE problem.

One of the specific requirements of network slicing is the isolation of slice instances, whereby each slice is composed of isolated VNFs and carries specific application traffic. Hence in the literature, related works normally tackle the optimization for the deployment of multiple independent virtual networks. However, for 5G slicing, different levels of isolation can be considered [12], and some common network services or functions can be shared among multiple slice instances, such as mobility management, the network address translation (NAT) function, etc. The working of VNFs is actually realized by the instantiation of VNF instances that are software modules executed on virtual machines, and towards which a certain amount of physical resources are allocated. By allowing these common VNF to be shared among slices, less VNF instances are required to be instantiated and physical resources could be saved. Hence, in this work we consider a loosed level of isolation in which the non-security-critical VNF instances can be shared across multiple slices to further improve the utilization of the underlying physical resources [8], [13].

For NSs with sharable VNF instances, the deployment of the slice instances is essentially the embedding of multiple virtual networks coupled by the VNFs shared among these slices. Hence, the typical VNE problem should be reformulated in the context of sharable VNFs for the 5G network. In this paper, we formulate this 5G E2E sharable-VNFs-based multiple coupled VNE problem (short for SVM-VNE problem) through an integer linear program (ILP) formulation. Especially, we consider

Manuscript received March 10, 2020; revised August 29, 2020; approved for publication by Hyoil Kim, Division III Editor, October 3, 2020.

C. Mei, J. Li, and L. Zhang are with the China Telecom Research Institute, email: {meichl, lijinyan, zhanglei}@chinatelecom.cn.

J. Liu and M. Shao are with the State Key Laboratory on ISN, Xidian University, email: jyliu@xidian.edu.cn, mhshao@stu.xidian.edu.cn.

J. Liu is the corresponding author.

Digital Object Identifier:10.1109/JCN.2020.000026

the specific topological characteristics of the 5G network. Then, we distinguish VNF types into sharable and non-sharable ones, and formulate the different resource requirements of these two. We leverage on the sharing property of VNFs to minimize the physical resources consumption. Due to the complexity of the problem, we design an heuristic algorithm to implement these NS virtual networks. Firstly, by measuring total and sharable resource requirements, we define a scheme which determines the deployment sequence of the multiple NSs. Then, for each NS instance, we adopt a back-tracking coordinated virtual node and link mapping mechanism to reduce bandwidth resource consumption. For sharable VNF mapping, we prioritize physical nodes with the same types of VNFs to reduce physical resource consumption. Finally, we conducted intensive simulations to evaluate our algorithm. The results show that VNF-sharing is beneficial for improving physical resource utilization ratio.

The remainder of this paper is organized as follows. Related works are summarized in Section II. Then, Section III describes the system model and presents the coupled multiple network slices embedding problem formulation. Our proposed sharable VNFs oriented network slices embedding algorithm is presented in Section IV. In Section V we evaluate the performance gain of our proposed algorithm. Finally, we conclude this work in Section VI.

II. RELATED WORKS

A. 5G Network Slicing

The basic idea of Network Slicing can be traced back to the infrastructure-as-a-service (IaaS) model for cloud computing, where a pool of physical resources is shared by multiple different tenants to create their logically isolated virtual systems to run their own businesses [5]. The NS is gaining extreme research interest in the development of the 5G network, because the 5G aims to explore new business models for value creation by providing tailored network services towards different types of vertical applications, such as Industry 4.0, wireless virtual/augmented reality (VR/AR) streaming, etc. The NS has been also the focus of multiple standardization organizations, for example, the logical end-to-end slicing has been included in the 3GPP Release 15 as one of the fundamental technologies for the 5G network. In [1], the authors outline the evolution from network sharing principle toward network slicing concept for the 5G network. In [2], the authors review the state-of-the-art in 5G network slicing and present a widely adopted three-layer generic framework for 5G NS.

5G NS can be roughly classified into two categories: (1) RAN slicing includes dynamic RAN composition and slice-oriented radio resource virtualization and scheduling mechanisms; (2) E2E slicing (or CN slicing) refers to the embedding of the verticals' virtual networks towards the physical infrastructure [5], [14]. There are also works design the interfaces and protocols between RAN slices and CN slices [11]. For RAN slicing, the main issue is to allocate radio resources among multiple slices according to the channel conditions and service-level agreement (SLA) requirements of tenants, to achieve slice isolation without compromising multiplexing gain over the shared radio resources. In [7], the authors present Orion, a RAN slicing system that achieves isolation among RAN slices based on the abstraction for the virtualization of the hardware and spectrum of RAN. In [15], the authors specify a subset of RAN slice configuration parameters to drive the radio resources allocation based on the quality of service (QoS) requirement of slices.

On the other hand, the E2E slicing also stands for an important part for 5G NS. NS is service oriented, hence implementing service specific slices is achieved by the means of E2E slicing, such as information centric network (ICN) slices [16], content distribution network (CDN) slices [17], evolved multimedia broadcast multicast service (eMBMS) slices [18]. In the current paper, we focus on E2E 5G NS which spans across RAN, TN, and CN.

In [3], the authors survey the two NS enabling technologies, software defined networking (SDN) and network function virtualization (NFV), which bring virtualization and softwarization into the mobile packet core network architecture to offer the flexibility for establishing on-demand E2E network slices. In [9], the authors first determine the NS network level parameter by measuring E2E traffic at aggregation points in RAN, then, based on the formed virtual network, a virtual network embedding algorithm is proposed to implement the corresponding slices. In [10], the authors propose model and algorithm for E2E 5G NSs embedding by considering the topological information of NS virtual networks. In [19], the authors investigated 5G network function virtualization to minimize energy consumption. The VNFs deployment is formulated as a mixed integer linear program and heuristic algorithm is designed. In [20], the authors present a solution for the 5G transport network slicing problem in terms of both mixed integer linear programming formulations and heuristic algorithms.

B. Virtual Network Embedding Problem

The essential problem for 5G E2E NS implementation is the embedding of the NS corresponding virtual network, which is composed of VNFs to form the vertical's serving system, towards the physical infrastructure. In the literature, some works consider that the slices are formed by service function chains (SFCs) with chaining structure [13], [21]. While chains can be modeled to support some types of applications, however we choose to use the more general virtual network (VN) to represent slice's customized E2E network structure. As a matter of fact, a large part of the current related works on 5G E2E slicing use VN to represent the NS structure and model the VN by a graph [22]. Implementing an instance of a NS involves two steps: Firstly the VN is embedded by localizing both virtual nodes and virtual links, then traffic steering rules are established by the SDN controller to configure switches to route traffic accordingly. In this process, the VN embedding is essentially a VNE problem, which has been intensively studied in the literature [23].

The VNE problem can be formulated as an ILP model, and the complexity of the problem is known as NP-hard [23], [24]. In [21], the authors propose an exact mathematical model based on layered graph and column generation for the embedding of SFCs. In [25], the authors presents an approach to accelerate VNE algorithms by reducing the searching space and extracting the VN into subgraphs. There are also a large amount of efforts devoted in designing heuristic algorithms to achieve a

practical solution with efficiency. Some works adopt the two-step embedding method in which the virtual nodes are firstly placed, then based on the position of virtual nodes, virtual links are mapped as multi-commodity flow problem [10], [24], [26]. This method neglects the relation between network nodes and links, which may result in less efficient usage of physical resources. Another class of methods try to place virtual nodes and links coordinately within one stage. In [27], the authors utilize the one-stage method to map nodes and links at the same time and show the benefits of this method on obtaining a better physical resource utilization. In [28], the authors apply the breadth-first search to determine the sequence of embedding virtual nodes and map links during the same stage along the node embedding process. However, for embedding 5G E2E NSs with sharable VNFs (the above defined SVM-VNE problem), it requires to extend the VNE problem and redesign algorithms by considering the specific requirements of the 5G system.

C. Network Slicing with Sharable VNFs

Despite the large amount of related works on the topic of 5G E2E slicing, the number of works on NS with sharable VNFs is very limited. In the 5G network, some common network services can be shared among multiple slice instances, such as mobility management, the network address translation (NAT) function, etc. The reutilization of these sharable VNFs can further improve the physical resource utilization ratio. In [12], the authors state that NS isolation level and strength depends on slicing requirements and usage scenarios, and a network slice instance could be partially isolated from another one. In [29], the authors investigate the VNE embedding problem to collocate multiple virtual networks within the same substrate resources by sharing underlying resource blocks dynamically to satisfy VN's working probability and tolerance threshold. In [30], VNF locations of different types are already deployed in the physical infrastructure, the authors optimize VNF placement and routing problem with sharable VNF nodes. In [31], the authors study a sharing based SFC scheduling problem, in which SFC flows are scheduled through the already equipped VNF instances in the network to satisfy service requirements. However, the above works are not about implementing NSs with sharable VNFs among multiple different slice instances in the 5G system. In [13], the authors study the chaining-structured SFCs embedding problem with VNF instances sharing among multiple chains in the context of 5G network. As we stated in the previous section, NS is service oriented, virtual network is more general for presenting slice's customized service system. Moreover, comparing to VNFs located in the CN, the quantity of RAN VNFs (such as virtualized BBU or MEC nodes) is larger. Hence, in our work, we consider the more general VN to represent tenant's slice instance. We conducted simulations to compare the performance of our proposed algorithm towards the works stated in [13], the results are detailed in Section V.

III. SYSTEM MODEL AND PROBLEM FORMULATION

In Fig. 1, we illustrate 5G E2E network slices with sharable VNFs. This paper investigates the embedding of multiple NSs coupled with the sharable VNFs. In this section, we first describe the underlying physical infrastructure model. The physical infrastructure network is owned by one InP. Then, we model NS requests. The NS requests are issued by tenants based on their service and SLA requirements. Especially, we consider the scenario in which some specific VNF types could be shared among NSs to improve the network resource utilization ratio. Further, we formulate the problem of allocating physical network nodes and links resources to implement these coupled NSs as an ILP. For ease of understanding, a table of notation is summarized in Table 1.

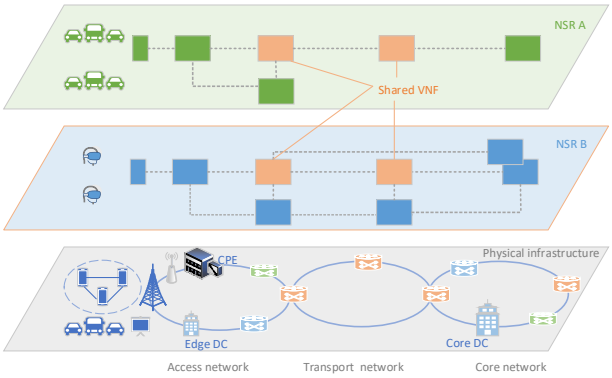

Fig. 1. 5G E2E network slices with sharable VNFs.

Table 1. Notations.

G_I, N_I, E_I	Physical network, physical nodes and physical links
n_u^I, e_{uv}^I	One physical node, one physical link
C_u^I, B_{uv}^I, L_{uv}^I	Physical node resource, link bandwidth and delay
NSR_k	The kth NSR, $k \in [1, K]$
G_S^k, N_S^k, E_S^k	NSR_k's virtual network, virtual nodes, virtual links
n_{ik}^S, e_{ijk}^S	NSR_k's one virtual node and one virtual link
B_{ijk}^S, L_{ijk}^S	NSR_k's virtual link bandwidth and delay
F, f, F_s	VNF types set, one VNF type, sharable VNF types
$T(n_{ik}^S)$	the VNF type of virtual node n_{ik}^S
N_{kf}^S	Virtual nodes in NSR_k with VNF type f
C_f	Instantiation resource requirement of VNF type f
C_{ik}^S	Running resource requirement of n_{ik}^S
loc_{ik}^u	Location specification parameter
x_{ik}^u, y_{ijk}^{uv}	Node and link mapping decision variables

A. Physical Infrastructure Model

The physical infrastructure network is modeled by a graph $G_I(N_I, E_I)$ where N_I denotes for the physical network nodes and E_I stands for physical communication links. One physical

network node is denoted by $n_u^I \in N_I$. We consider the C-RAN architecture in the RAN side, such that remote radio unit (RRU) and base band unit (BBU) are seperated, and the BBU can be virtualized. Hence, we distinguish four kinds of physical nodes:

- The physical RRU nodes set N_I^R is allowed to host virtual RRUs.
- The access nodes set N_I^A, such as BBU hotels and edge data centers (DCs) are capable for accommodating virtual BBU and virtual mobile edge computing (MEC) servers.
- The transport network nodes set N_I^T which denotes TN optical switches.
- The core network nodes set N_I^C, which includes core DCs for CN data plane and control plane VNFs virtualization.

and we have $N_I = N_I^R \cup N_I^A \cup N_I^T \cup N_I^C$. Without loss of generality, we assume that N_I^R nodes can only host virtual RRUs; N_I^A and N_I^C nodes are suitable for accommodating multiple other types of virtualized VNFs for the flexible deployment of NSs. Moreover, each physical node $n_u^I \in N_I^A \cup N_I^C$ is characterized by its capacity C_u^I which represents the physical resources of the node (such as computing resources).

Physical communication links are represented by E_I, with $e_{uv}^I \in E_I$ denote the physical link connecting two physical nodes n_u^I and n_v^I. Especially, RRUs are connected to the C-RAN central locations via common public radio interface (CPRI) fronthaul transport, and the other types of links (including backhaul and TN CN links) are based on IP transportation. For each link $e_{uv}^I \in E_I$, we denote by B_{uv}^I the bandwidth capacity of the link, and L_{uv}^I the transmission delay of the link.

B. Network Slice Request Model

Network slices are service oriented. Different services requires various different composition of network functions to form the specific service system. In order to describe all kinds of network slice requests (NSRs), we follow the general way to represent a 5G E2E network slice by a VN, and model it by an undirected graph. For example, as shown in Fig. 2, we depict a 5G E2E NS for providing ICN services. The NS contains the following VNFs:

- ICN-CR: The ICN cache enabled router.
- ICN-GW: The ICN gateway, for ICN-IP interworking.
- The MME, SDN-Controller and virtual network infrastructure orchestrator (VNIO), etc., which are sharable among multiple NSs.

Clearly, the topology of the NS is a mesh based VN, which is represented by an undirected graph.

We consider K NSRs. Each network slice request NSR_k is represented by a virtual network and modeled as a graph $G_S^k(N_S^k, E_S^k)$, $k \in [1, K]$, with N_S^k denotes for the virtual network nodes in NSR_k, and E_S^k stands for virtual links connecting the virtual nodes. One virtual network node is denoted by $n_{ik}^S \in N_S^k$. Each virtual network node represents a virtual VNF instance, such as virtual RRU, virtual BBU, virtual MEC server, virtual encoder (for mobile video service slices) and virtual gateway, etc. Due to the specific position requirement of virtual RRU, we denote by N_S^{Rk} the set of virtual RRU nodes. Then, we denote by C_{ik}^S the physical resource requirement of the virtual node n_{ik}^S. A virtual link is represented by $e_{ijk}^S \in E_S$, with B_{ijk}^S denoting its bandwidth requirement and L_{ijk}^S the link delay requirement.

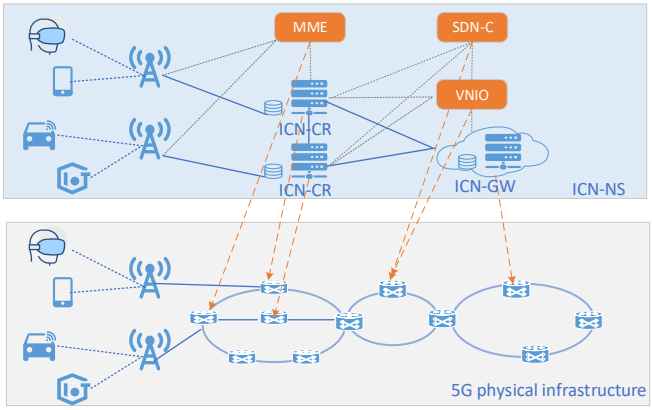

Fig. 2. An illustrative example: A 5G ICN NS.

Specifically, we consider the scenario in which some types of VNFs can be shared among these K NSs. The sharing policy depends on the InP's and tenants' requirements. In this case, an instance of the VNF can participate into multiple NSs, to avoid redundantly instantiate more VNFs and to improve the utilization of the underlying physical network resources. Assume that the VNFs can be classified into F types, and each type is denoted by $f \in F$. A subset of these types, noted as $F_s \subseteq F$ are defined as sharable VNF types. For a virtual node n_{ik}^S, its corresponding VNF type is denoted as $T(n_{ik}^S)$, consequently, if $T(n_{ik}^S) \in F_s$, this virtual node can be co-localized with other virtual nodes with the same sharable type. We define the set $N_{kf}^S = \{n_{ik}^S | T(n_{ik}^S) = f\}$, as the set of virtual nodes with VNF type f in NSR_k.

We define the basic operational resource requirement to instantiate a VNF of type f by C_f. The afore-defined resource requirement C_{ik}^S of virtual node n_{ik}^S is actually the additive resource requirement to run the VNF for the corresponding network services, which is related to the workload of the virtual node. By this definition, for a non-sharable VNF, the instantiation resource requirement is always $C_f + C_{ik}^S$; whereas for a sharable VNF, the first instance on one physical node requires $C_f + C_{ik}^S$, the subsequent virtual nodes with the same type allocated to this physical node just requires the additive resource part C_{ik}^S.

C. SVM-VNE Problem Formulation

Deploying these NSs correspond to mapping the K virtual networks towards the physical infrastructure. We define two sets of binary variables:

- The node map decision variable x_{ik}^u takes the value 1 if the virtual node n_{ik}^S is placed on the physical node n_u^I, and takes the value 0 otherwise.
- The link map decision variable y_{ijk}^{uv} takes the value 1 if the virtual link e_{ijk}^S is deployed through the physical link e_{uv}^I, and takes the value 0 otherwise.

We first formulate the 5G network topology related constraints. In the 5G network, VNFs are position specific. For example, the virtual RRU can only be deployed on physical RRUs

to provide wireless transmission in a certain area. In MEC service scenarios, the related VNFs should be placed on the edge of the network. To describe such location specification, we define a binary location parameter loc_{ik}^{u} which takes value 1 if the corresponding virtual node n_{ik}^{S} can be placed on the corresponding physical node n_{u}^{I}, and 0 otherwise. Hence, we introduce a location constraints in (1).

$$x_{ik}^{u} \leq loc_{ik}^{u}, \quad \forall\, n_{u}^{I} \in N_I, \;\; n_{ik}^{\in} N_S^k \tag{1}$$

Especially, the location parameter loc_{ik}^{u} should be properly defined. The position and traffic requirements of virtual RRUs can be determined by analysing the spatial temporal traffic pattern of each NSs. Each virtual RRU is anchored toward one specific physical RRU node, and the corresponding loc parameter is set to 1. Hence, $\sum_u loc_{ik}^{u} = 1$, $\forall k$ and $n_{ik}^{S} \in N_S^{Rk}$. Moreover, for access VNFs, $loc_{ik}^{u} = 0, \forall n_{u}^{I} \in N_I^R \cup N_I^T \cup N_I^C$; for core VNFs, $loc_{ik}^{u} = 0, \forall n_{u}^{I} \in N_I^R \cup N_I^A \cup N_I^T$.

Then, we formulate the resource constraints for the sharable VNFs from (2) to (4). For the K coupled NSRs, a specific constraint is the physical node resource constraint which bounds the number of placed virtual nodes. First, we count the physical resource consumed by sharable VNFs placed on a physical node:

$$C_{u}^{I_s} = \sum_{f \in F_s} \sum_{k} \sum_{n_{ik}^{S} \in N_{kf}^{S}} (x_{ik}^{u} C_{ik}^{S} + C_f \cdot 1\{x_{ik}^{u} > 0\}), \quad \forall\, n_{u}^{I}, \tag{2}$$

where $1\{x_{ik}^{u} > 0\}$ is the indicator function which takes value 1 if $x_{ik}^{u} > 0$, and 0 otherwise. Then, we compute the physical resource required by non-sharable VNFs placed on a physical node:

$$C_{u}^{I_n} = \sum_{f \in \{F - F_s\}} \sum_{k} \sum_{n_{ik}^{S} \in N_{kf}^{S}} x_{ik}^{u} (C_f + C_{ik}^{S}), \quad \forall\, n_{u}^{I}. \tag{3}$$

Clearly, the physical resource constraint is represented as:

$$C_{u}^{I_s} + C_{u}^{I_n} \leq C_{u}^{I}, \quad \forall\, n_{u}^{I}. \tag{4}$$

Finally, we formulate the VNE embedding constraints for each NSR from Constraint (5) to (9). Firstly, each virtual nodes must be mapped to one physical node.

$$\sum_{n_{u}^{I} \in N_I} x_{ik}^{u} = 1, \quad \forall\, n_{ik}^{S} \tag{5}$$

With the location Constraint (1), this constraint ensures that the virtual RRU is deployed on its specific position, and each virtual node of the NSR is properly deployed. Then, the sharable VNFs could save some physical node resources, however, the bandwidth requirements can not be reduced. Hence, the physical link resource constraint should be respected.

$$\sum_{k} \sum_{e_{ijk}^{S} \in E_S^k} B_{ijk}^{S} y_{ijk}^{uv} \leq B_{uv}^{I}, \quad \forall\, e_{uv}^{I} \tag{6}$$

The mapped physical links should be bounded by the virtual link delay requirement.

$$\sum_{e_{uv}^{I} \in E_I} L_{uv}^{I} y_{ijk}^{uv} \leq L_{ijk}^{S}, \quad \forall\, e_{ijk}^{S} \tag{7}$$

One virtual link e_{ijk}^{S} is mapped as a flow from the physical source to sink. Hence, we impose the following flow related constraints.

$$\sum_{n_v^I \in N(n_u^I)} y_{ijk}^{uv} \leq 1, \quad \forall\, n_{u}^{I},\, e_{ijk}^{S}, \tag{8}$$

$$\sum_{n_v^I \in N(n_u^I)} y_{ijk}^{uv} - y_{ijk}^{vu} = x_{ik}^{u} - x_{jk}^{u}, \quad \forall\, n_{u}^{I},\, e_{ijk}^{S}, \tag{9}$$

in which $N(n_u^I)$ stands for the neighbor set for the physical node n_u^I. Constraint (8) ensures that there is no loop along the physical path mapped for the virtual link. Constraint (9) satisfies the flow conservation constraints.

Finally, we set forth the problem objective formulation. One reasonable strategy to allocate the infrastructure for these K NSRs would be to minimize the utilization of the network resources. The physical resource consumed on each physical node is calculated as: $C_{u}^{I_s} + C_{u}^{I_n}$, $\forall\, n_{u}^{I}$. The bandwidth consumption on each physical link is: $\sum_k \sum_{e_{ijk}^{S}} B_{ijk}^{S} y_{ijk}^{uv}$, $\forall\, e_{uv}^{I}$.

Then, the SVM-VNE problem is formulated as Problem (P1), subjects to the NSRs' SLA requirements and physical network capacity:

$$\min_{\{x_{ik}^{u}\},\{y_{ijk}^{uv}\}} \quad \sum_{n_u^I} (C_{u}^{I_s} + C_{u}^{I_n}) + \sum_{e_{uv}^{I}} \sum_{k} \sum_{e_{ijk}^{S}} B_{ijk}^{S} y_{ijk}^{uv} \tag{P1}$$

$$\text{s.t. (3), (4), (5), (6), (7), (8), and (9).}$$

The complexity of the Problem (P1) is NP-hard, because when $K = 1$, this problem is reduced to the NP-hard VNE problem. Hence we focus on developing fast algorithms to practically tackle the problem for real and large system setting.

IV. SVM-VNE ALGORITHM

The minimization of the objective of the Problem (P1) comes from two parts: (1) By integrating the sharable VNFs into less instances, the physical node resource consumption could be reduced; (2) by mapping virtual links into shorter physical paths, the bandwidth resource could be saved. Inspired through these two observations, we design our SVM-VNE algorithm to especially tackle these two aspects of the problem.

A. Network Node Importance

We first introduce how to measure network *node importance* by considering both resource and topology information. This measurement is applicable for both physical network nodes and virtual nodes to distinguish the importance of nodes for determining the sequence of slices and nodes mapping. The following concepts of node importance are conventional in general networks, hence we manifest the measurement of the virtual nodes for one NS, that of the physical nodes can be derived similarly.

A.1 Resource Importance

We consider the measurement for virtual nodes in NSR_k, the topology is given by G_S^k. Firstly, we define the resource impor-

tance of n_{ik}^S as:

$$RI(n_{ik}^S) = C_{ik}^S \cdot \sum_{e_{ijk}^S} B_{ijk}^S.$$

Thus, we consider both the computing resource required by n_{ik}^S, and the bandwidth requirement of virtual links connected to n_{ik}^S in G_S^k.

A.2 Topology Importance

We utilize the following conventional concepts to measure the topological characteristics of n_{ik}^S.

- Degree centrality. The degree of n_{ik}^S measures the number of links that connect to it in G_S^k. The normalized node degree is defined as $d'_{n_{ik}^S} = d_{n_{ik}^S}/(|N_S^k| - 1)$, where $d_{n_{ik}^S}$ is node degree of n_{ik}^S and $|N_S^k|$ is the number of virtual nodes in NSR_k.
- Betweenness centrality. The betweenness centrality calculates the fraction of all-pairs shortest paths pass through the node in the topology graph, which is defined as $b_{n_{ik}^S} = \sum_{j\neq i\neq m} p_{jm}^k(n_{ik}^S)/\sum_{j\neq i\neq m} p_{jm}^k$. Here, p_{jm}^k is the number of shortest (n_{jk}^S, n_{mk}^S)-paths in G_S^k, and $p_{jm}^k(n_i^S)$ is the number of those pathes goes through node n_{ik}^S in G_S^k. We further normalize it into $b'_{n_{ik}^S} = 2b_{n_{ik}^S}/(|N_S^k| - 1)(|N_S^k| - 2)$.
- Closeness centrality. The closeness centrality measures the distance of the node towards all other nodes in the topology graph, which is defined as $a_{n_{ik}^S} = 1/\sum_{j\neq i} d(n_{ik}^S, n_{jk}^S)$. Here, $d(n_{ik}^S, n_{jk}^S)$ denotes the distance between node n_{ik}^S and n_{jk}^S in G_S^k. We also normalize it into $a'_{n_{ik}^S} = a_{n_{ik}^S}(|N_S^k| - 1)$.

Then, the topology importance of n_{ik}^S is defined as:

$$TI(n_{ik}^S) = d'_{n_{ik}^S} + b'_{n_{ik}^S} + a'_{n_{ik}^S}.$$

Finally, the node importance of n_{ik}^S is calculated as:

$$NI(n_{ik}^S) = RI(n_{ik}^S) \cdot TI(n_{ik}^S). \tag{10}$$

B. Deployment Scheme for K NSRs

To deploy the K NSRs' virtual networks, the first step would be to determine a sequence of deployment. We first measure the total resource requirement and sharable resource requirement of NSRs. Firstly, the total resource requirement of NSR_k is calculated as:

$$Z(NSR_k) = \alpha \sum_{n_{ik}^S} (C_{f=T(n_{ik}^S)} + C_{ik}^S) + \beta \sum_{e_{ijk}^S} B_{ijk}^S, \tag{11}$$

in which α and β are two tuneable positive parameters, which drive the emphasis on computing or bandwidth resources. The sharable resource requirement of NSR_k, which can be potentially integrated, is calculated as:

$$R(NSR_k) = \sum_{f\in\{F_s\}} \sum_{n_{ik}^S \in N_{kf}^S} TI(n_{ik}^S) C_f, \tag{12}$$

Algorithm 1 Deployment algorithm for the K NSRs

1: **Input**: The K NSRs.
2: **Output**: The mappings from the K NSRs to G_I.
3: **for** Each NSR_k **do**
4: Calculate $L(NSR_k)$ based on (13).
5: **end for**
6: Sort the K NSRs in $L(NSR_k)$ decreasing order and push into the list $L(NSR)$.
7: **while** $L(NSR) \neq \emptyset$ **do**
8: Remove the first NSR from $L(NSR)$.
9: Deploy the NSR based on Algorithm 4.
10: Update parameters according to the deployment.
11: **end while**

Algorithm 2 Virtual node sorting algorithm for NSR_k

1: **Input**: The G_S^k of NSR_k.
2: **Output**: A sorted list of virtual nodes $L(N_S^k)$.
3: **for** $n_{ik}^S \in N_S^k$ **do**
4: Calculate $NI(n_{ik}^S)$ based on (10).
5: **end for**
6: Find $\hat{n_{ik}^S}$ as the virtual node with the highest NI value.
7: Build the searching tree $T(\hat{n_{ik}^S})$ on top of G_S^k, by setting $\hat{n_{ik}^S}$ as the root and using the Breadth-First Search algorithm.
8: For each layer of $T(\hat{n_{ik}^S})$, sort the virtual nodes in NI decreasing order, and push these sorted virtual nodes into $L(N_S^k)$.

which is weighted by the topology importance of the sharable VNF virtual node. Hence, a more topological important sharable virtual node is more emphasized in determining the sharable resource requirement. Based on the above two definition, we finally sort all K NSRs based on:

$$L(NSR_k) = Z(NSR_k) + \gamma R(NSR_k), \tag{13}$$

where γ is also a tuneable positive parameter. The process of deploying the K NSRs are depicted in Algorithm 1. Note that NSRs with more resource requirement and sharable resources will be prioritized in the sequence of deploying to facilitate the integration of sharable VNFs.

C. Coordinated Mapping Algorithm for One NSR

C.1 Virtual Node Sorting

Then, we introduce our algorithm to deploy one NSR. Assume the to-be-deployed NSR is NSR_k. We adopt the coordinated virtual node and link mapping method to reduce physical bandwidth consumption [28]. Moreover, we restrict that two neighboring virtual nodes are mapped within h hops in the physical infrastructure. We adopt the back-tracking mechanism such that if the current virtual node cannot be mapped to the current candidate physical node, the algorithm will try the next best one. This mechanism can further explore more mapping opportunities and improve NSR acceptance ratio. We first determine a sequence based on which the virtual nodes of NSR_k are mapped.

The virtual node sorting algorithm first select the virtual node

Algorithm 3 Candidate physical nodes selection algorithm for n_{ik}^S

1: **Input**: The virtual node n_{ik}^S of NSR_k.
2: **Output**: A sorted list of candidate physical nodes $CPN(n_{ik}^S)$.
3: Define $Loc(n_{ik}^S) = \{n_u^I | loc_{ik}^u = 1, \forall n_u^I \in N_I\}$.
4: Define $VPN(T(n_{ik}^S))$ as the set of physical nodes with VNFs of type $T(n_{ik}^S)$ mapped, and $\overline{VPN}(T(n_{ik}^S))$ as its complementary set.
5: Let n_{par}^S be the parent virtual node of n_{ik}^S in the BFS tree; Let n_{par}^I be the physical node to which n_{par}^S is mapped.
6: Define $PN_h(n_{par}^I)$ as the set of physical nodes within h-hops from n_{par}^I in G_I.
7: **if** n_{ik}^S is the ROOT of the BFS tree **then**
8: **if** $T(n_{ik}^S) \in F_s$ **then**
9: $CPN_1(n_{ik}^S) = \{n_u^I | C_u^I > C_{ik}^S\} \cap VPN(T(n_{ik}^S)) \cap Loc(n_{ik}^S)$.
10: $CPN_2(n_{ik}^S) = \{n_u^I | C_u^I > (C_f + C_{ik}^S)\} \cap \overline{VPN}(T(n_{ik}^S)) \cap Loc(n_{ik}^S)$.
11: Sort physical nodes in $CPN_1(n_{ik}^S)$ and $CPN_2(n_{ik}^S)$ in NI decreasing order respectively.
12: $CPN(n_{ik}^S) = CPN_1(n_{ik}^S) + CPN_2(n_{ik}^S)$.
13: **else**
14: $CPN(n_{ik}^S) = \{n_u^I | C_u^I > (C_f + C_{ik}^S)\} \cap Loc(n_{ik}^S)$.
15: Sort physical nodes in $CPN(n_{ik}^S)$ in NI decreasing order.
16: **end if**
17: **else**
18: **if** $T(n_{ik}^S) \in F_s$ **then**
19: $CPN_1(n_{ik}^S) = \{n_u^I | C_u^I > C_{ik}^S\} \cap VPN(T(n_{ik}^S)) \cap PN_h(n_{par}^I) \cap Loc(n_{ik}^S)$.
20: $CPN_2(n_{ik}^S) = \{n_u^I | C_u^I > (C_f + C_{ik}^S)\} \cap \overline{VPN}(T(n_{ik}^S)) \cap PN_h(n_{par}^I) \cap Loc(n_{ik}^S)$.
21: Sort physical nodes in $CPN_1(n_{ik}^S)$ and $CPN_2(n_{ik}^S)$ in NI decreasing order respectively.
22: $CPN(n_{ik}^S) = CPN_1(n_{ik}^S) + CPN_2(n_{ik}^S)$.
23: **else**
24: $CPN(n_{ik}^S) = \{n_u^I | C_u^I > (C_f + C_{ik}^S)\} \cap PN_h(n_{par}^I) \cap Loc(n_{ik}^S)$.
25: Sort physical nodes in $CPN(n_{ik}^S)$ in NI decreasing order.
26: **end if**
27: **end if**

with the highest NI value as the root and build the BFS tree from G_S^k. Then, from the higher layer of the tree, the virtual nodes are pushed into the sorted list based on their NI value. Hence, the virtual nodes are sorted based both on their distances from the root and the NI values.

C.2 Candidate Physical Nodes Selection

For a given virtual node, the NSR mapping algorithm selects a set of candidate physical nodes, from which to map the corresponding virtual node. We define $CPN(n_{ik}^S)$ as the candidate physical nodes set for n_{ik}^S. The candidate physical nodes selection algorithm is depicted in Algorithm 3. We classify four cases based on the role of the virtual node n_{ik}^S:

- n_{ik}^S is the root of the BFS tree and its VNF type is sharable (line 8-12). In this case, $CPN(n_{ik}^S)$ contains two parts: (1) $CPN_1(n_{ik}^S)$ (line 9) contains physical nodes with VNFs of the type $T(n_{ik}^S)$ already mapped, and with sufficient large physical resources, and these physical nodes can accommodate the n_{ik}^S; (2) in $CPN_2(n_{ik}^S)$ (line 10), we also consider physical nodes without the $T(n_{ik}^S)$ type VNFs in case $CPN_1(n_{ik}^S) = \emptyset$. Then, these two sets are sorted based on NI value decreasing order respectively, and we append $CPN_2(n_{ik}^S)$ at the end of $CPN_1(n_{ik}^S)$ to form the $CPN(n_{ik}^S)$ set.
- n_{ik}^S is the root of the BFS tree and its VNF type is non-sharable (line 14-15). In this case, $CPN(n_{ik}^S)$ is composed of physical nodes with sufficient large resources and suitable locations. The $CPN(n_{ik}^S)$ set is also sorted based on NI value decreasing order.
- n_{ik}^S is not the root of the BFS tree and its VNF type is sharable (line 18-22). When n_{ik}^S is not the root of the tree, it means that it is not the first to-be-mapped virtual node and some other virtual nodes (including its parent node in the BFS tree) have been already mapped. In the coordinated virtual node and link mapping mechanism, when mapping this node, the virtual links connecting this node and other mapped nodes should also be mapped. In order to save bandwidth resources, we select physical nodes that locate near (for instance, within h-hops, where h is a parameter) to the mapped nodes (parent node of n_{ik}^S). Hence, comparing to the CPN set of the root, we add a constraint on CPN such that the candidate physical nodes should locate within h-hops from the parent node of n_{ik}^S. Finally, $CPN(n_{ik}^S)$ is formed and sorted following the same logic as for the root.
- n_{ik}^S is not the root of the BFS tree and its VNF type is non-sharable (line 24-25). In this case, $CPN(n_{ik}^S)$ is composed of physical nodes with sufficient large resources, locate near to the parent node and with suitable locations. The $CPN(n_{ik}^S)$ set is also sorted based on NI value decreasing order.

C.3 SVM-VNE Algorithm

Then we describe our coordinated SVM-VNE algorithm. The algorithm is detailed in Algorithm 4:

- The input of the algorithm is the physical network model G_I and the NSR_k model G_S^k. The output of the algorithm is the mapping from the NSR to the physical network. If the NSR cannot be mapped due to lack of resource, a rejection is returned.
- The algorithm tries to map the virtual node in the list $\mathbb{L}(N_S^k)$ one by one (step 4). For the current to be mapped virtual node n_{ik}^S, obtain the set of candidate physical nodes $CPN(n_{ik}^S)$ from Algorithm 3. Then, depending on the role of n_{ik}^S in the virtual network, we further classify two subsituations.
- First, if n_{ik}^S is the root of the BFS tree (step 6), we directly map n_{ik}^S to the first physical nodes in $CPN(n_{ik}^S)$.
- If n_{ik}^S is not the root of the BFS tree, it implies both the virtual node and corresponding virtual links should be mapped. The idea is, we try to map n_{ik}^S to the physical nodes in $CPN(n_{ik}^S)$ one-by-one: if all virtual links con-

Algorithm 4 SVM-VNE algorithm for NSR_k

Input: The physical network G_I, the NSR G_S^k.
Output: The mapping from NSR G_S^k to G_I.
Obtain $\mathbb{L}(\mathbb{N}_{\mathbb{S}}^{\rceil})$ from Algorithm 2.
for $n_{ik}^S \in \mathbb{L}(\mathbb{N}_{\mathbb{S}}^{\rceil})$ **do**
Obtain $CPN(n_{ik}^S)$ from Algorithm 3.
if n_{ik}^S is the ROOT of the BFS tree **then**
Map n_{ik}^S to the first physical node in $CPN(n_{ik}^S)$.
else
for $n_u^I \in CPN(n_{ik}^S)$ **do**
Map the virtual link between n_{ik}^S and n_{par}^S:
Remove links in G_I with bandwidth lower than $B_{n_{ik}^S, n_{par}^S, k}^S$.
Find the *shortest-path* between n_u^I and n_{par}^I in G_I.
if The link delay is feasible **then**
Map other virtual links between n_{ik}^S and the already mapped virtual nodes.
else
Go to step 11 to try the next $n_u^I \in Cand(n_i^S)$ for mapping n_{ik}^S and the corresponding virtual links.
end if
end for
if n_{ik}^S cannot be mapped to any physical node in $CPN(n_{ik}^S)$. **then**
Reject the NSR.
end if
end if
end for

necting n_{ik}^S and the already mapped nodes can be mapped successfully (the shortest path can fulfill Constraint (7)), then n_{ik}^S is mapped to the current candidate physical node, otherwise we try the next physical nodes in the $CPN(n_{ik}^S)$. At the end, if all physical nodes cannot accommodate n_{ik}^S, which means n_{ik}^S cannot be successfully mapped, a rejection is returned.

Our algorithm is designed to save the physical resource consumption through the following means: (1) For sharable VNF deployment, we prioritize physical nodes hosting the same type of VNFs to exploit the possibility of integrating the sharable VNFs; (2) we utilize the parameter h to restrict that neighboring virtual nodes are mapped towards near physical positions (within h-hops). Consequently, the virtual links are mapped into shorter physical pathes, which reduces bandwidth consumption.

D. Practical Relevance

In real system, the NSRs arrive dynamically and randomly. When a NSR arrives, the InP allocates physical network resources to implement the corresponding NS based on Algorithm 4; and after its life time expires, the NS is deleted and physical network resources are released. It is trivial to treat the NS expiration event. For NSR arriving event, our proposed algorithm can be utilized to implement the single NSR based on the already deployed NSRs setting and available network resources.

Then we analyze the runtime complexity of our proposed algorithm. The algorithm utilized the following operations: (1) Sorting. The runtime complexity of Quicksort algorithm is $O(|N_S|\log|N_S|)$; (2) BFS with runtime complexity $O(|N_S||E_S|)$; (3) shortest path. The runtime complexity of Bellman algorithm is $O(|N_I||E_I|)$. Taking $N = \max\{|N_S|, |N_I|\}$ and $E = \max\{|E_S|, |E_I|\}$. Algorithm 4's complexity is $O(NE + N\log N + N(N\log N + N^2E))$, which is finally reduced into $O(N^3E)$. Then, for deploying K NSRs, the total complexity is $O(KN^3E)$.

Table 2. Simulation setting.

Physical network setting	Value
Node CPU capacity	U[50,70]
Link bandwidth capacity	U[100,200]
Link delay	U[3,5]
AN:TN:CN	3:4:3
Virtual network setting	**Value**
Node CPU requirement	U[20,30]
Link bandwidth requirement	U[1,10]
Link delay requirement	U[60,100]

V. EVALUATIONS

A. Simulation Setting

We build a simulation platform to evaluate the performance of our proposed algorithm. Firstly, in the experiment, the underlying 5G physical network and NSR's virtual network topology is generated by a modified Barabasi-Albert (BA) scale-free network model construction algorithm [32]. The BA model is largely utilized to represent the 5G system and NSR topology [10], [33]. Typically, as introduced in Section III, the physical network contains three types of nodes: Access nodes (AN), transport nodes (TN) and core nodes (CN). The network is generated from an initially fully connected CN network. Then, new nodes are added into the network one-by-one. Each time, the incoming new node is connected towards existing nodes in the network based on a probability $P_i = d_i/\sum_j^N d_j$, where N denotes the total number of current existing nodes and d_i is the degree of the incoming node. Moreover, TN nodes can be connected to TN or CN nodes, whilst AN nodes can only be connected to TN nodes.

The overall simulation setting is shown in Table 2. For the physical network, physical node CPU capacity follows uniform distribution on the interval $[50, 70]$; physical link capacity follows uniform distribution on the interval $[100, 200]$; and physical link delay follows uniform distribution on the interval $[3, 5]$. Moreover, the physical network is composed of AN, TN and CN nodes, the proportion of these three nodes follows $3:4:3$. For virtual network, virtual node CPU requirement follows uniform distribution on the interval $[20, 30]$; virtual link bandwidth requirement follows uniform distribution on the interval $[1, 10]$; and virtual link delay requirement follows uniform distribution on the interval $[60, 100]$. Virtual network composition (node location specification) follows the same as the physical network. 40% of virtual nodes are sharable, and the additive CPU requirement of sharable virtual nodes accounts for 20% of its overall CPU resource requirement. For algorithm parameters, we set the link mapping hops parameter $h = 1$, and α, β, γ are also set

to be 1. Simulations with the same setting are launched for 100 times, and the results are taken as the average value.

We conducted two sets of simulations. In the first set (detailed in Section V.B), we compare our SVM-VNE Algorithm towards its counterpart without VNF-sharing, and we denote this algorithm by NSVM-VNE Algorithm. In the second set (detailed in Section V.C), we compare SVM-VNE and NSVM-VNE Algorithms towards a baseline algorithm that is a SFC embedding algorithm presented in [13]. The difference of these two groups of simulation comes from the fact that the baseline algorithm is specifically designed for SFCs with chaining structure, whilst our algorithm is for NSs with general mesh based VN structure. Hence, for the first set of simulation, the NSs are meshes, whereas for the second set of simulation, in order to conduct comparable experiments, the NSs are chains. Simulation results show that (1) VNF sharing achieves higher NS acceptance ratio, hence higher physical resource utilization; (2) our algorithm still outperform the baseline algorithm for NSs with chaining structure.

B. Basic Performance

We present the results of our first set of simulation in this section. We evaluate SVM-VNE and NSVM-VNE Algorithms. In Fig. 3, we show the acceptance ratio of the two algorithms by varying the scale of the physical network from 40 nodes to 100 nodes. The acceptance ratio is calculated as the number of successfully embedded NSRs over the overall NSRs. The overall NSR number is set to 30, and each NSR contains 10 virtual nodes. As the physical network enlarges, the underlying resources increase and more NSRs could be embedded. For all physical network scales, the sharing based NSR embedding algorithms achieves higher acceptance ratio: 33.7% for 40 physical nodes and 99.1% for 100 physical nodes for the SVM-VNE algorithm; and 26.5% for 40 physical nodes and 66.3% for 100 physical nodes for the NSVM-VNE algorithm.

In Fig. 4, we show the acceptance ratio of the two algorithms for different number of NSRs. The physical network remains with 100 physical nodes. Each NSR contains 10 virtual nodes. We increase the number of NSRs from 30 to 100. The acceptance ratio decreases as the resource requirement increases. The sharing based SVM-VNE algorithm also achieves higher resource utilization ratio as the acceptance ratio is 96.2% for 30 NSRs and 25.0% for 100 NSRs. For NSVM-VNE algorithm, the acceptance ratio is 66.8% for 30 NSRs and 19.5% for 100 NSRs. In Fig. 5, we show the acceptance ratio of the two algorithms for different virtual network scales. The physical network remains with 100 physical nodes. The number of NSRs are 30. We vary the number of virtual nodes contained in each NSR from 9 to 36. As the scale of the virtual network increases, more resource are demanded, and the acceptance ratio decreases. The acceptance ratio is 96.8% with 9 virtual nodes and 20.0% with 36 virtual nodes in each NSR for the SVM-VNE algorithm, and the result is 67.7% with 9 virtual nodes and 16.7% with 36 virtual nodes in each NSR for the NSVM-VNE algorithm.

Finally, we show in Fig. 6 the number of VNF instances used per slice. For the non-sharing based NSVM-VNE algorithm, each virtual node should be implemented with one VNF instance, hence the VNF instance number is the same as the virtual node number. For our VNF-sharing based SVM-VNE algorithm, we set that 40% percent of virtual nodes are with sharable VNF types, hence the sharable virtual nodes can be co-localized into the same VNF instances, and the VNF instance number is less than the virtual nodes number. We also observe that the ratio of VNF instance over virtual node number decreases as the virtual network scale increases. As the physical network is saturated, more VNF instances are used.

C. Comparison to Baseline

In the second set of simulation, we aim to compare the performance of our algorithm towards the SFC embedding algorithm presented in [13]. We denote the algorithm presented in [13] as the *baseline* algorithm. The baseline algorithm is typically designed for SFCs with chaining structure, hence we conducted a set of simulation with chaining structured NSRs. Our algorithm is designed for general NSRs by graph model, hence it can be applied for the chaining structured NSRs, as chain is a typical structure which can also be represented by our model.

The results are presented in Fig. 7 to Fig. 10. Both the VNF-sharing based SVM-VNE algorithm and the baseline algorithm perform better than the non-sharing based one. Moreover, our algorithm achieves better performance as it obtains higher acceptance ratio. For Fig. 7 we increase the physical network scale with physical node number from 50 to 150. The number of NSR is 30 with each NSR contains a chain with 10 virtual nodes. SVM-VNE algorithm obtains the highest acceptance ratio. For instance, the acceptance ratio is 86.5% for 100 physical nodes with SVM-VNE, it is 80.1% for the baseline algorithm, and it is 66.0% for the NSVM-VNE algorithm.

In Figs. 8 and 9, we show the acceptance ratio with different number of NSRs and different NSR scales. The setting remains the same as for Figs. 4 and 5. The results show that SVM-VNE outperforms the baseline algorithm in achieving higher acceptance ratio. Finally, In Fig. 10, we also compare the number of VNF instances used by SVM-VNE and the baseline algorithms. Our algorithm uses less VNF instances, because we prioritize physical nodes with sharable VNF instances in the NSR embedding process. With more integrated VNF instances, our algorithm utilizes less physical node resources, hence achieves higher resource utilization ratio. Then, we further investigate the performance of our algorithm in some other aspects in Fig. 11 to Fig. 13. In Fig. 11, we show the NSR acceptance ratio by varying the percentage of VNFs which are sharable for our proposed algorithm and the baseline algorithm. We do not show this simulation result for the NSVM-VNE algorithm because in this case this percentage remains zero. The percentage of sharable virtual nodes vary from 10% to 60%. For this simulation, the physical network contains 100 physical nodes, and the number of NSRs is 30 with each NS contains 10 virtual nodes. As the percentage of sharable VNFs increases, the acceptance ratio also increases since more VNFs could be integrated. Our algorithm outperforms the baseline algorithm. Especially, when the percentage of sharable VNFs increases, the gap between the two algorithm also increases. For 10% VNFs are sharable, the acceptance ratio of our algorithm is 69.7%, and that of the baseline algorithm is 67.8%. Whilst for 60% VNFs are sharable, the acceptance ratio of our algorithm is 98.6%, and that of the baseline algorithm is

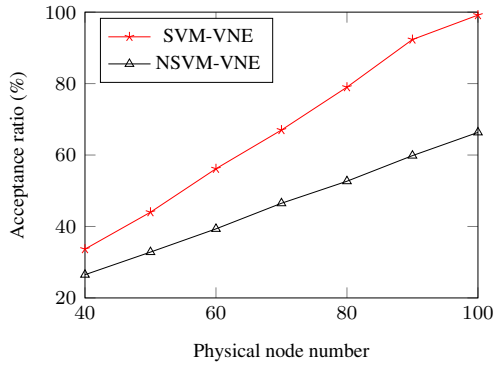

Fig. 3. Acceptance ratio with different physical networks.

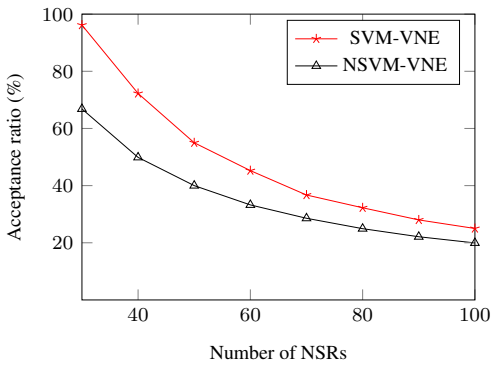

Fig. 4. Acceptance ratio with different NSRs.

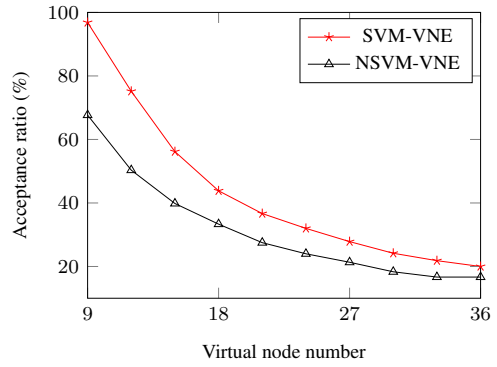

Fig. 5. Acceptance ratio with different virtual networks.

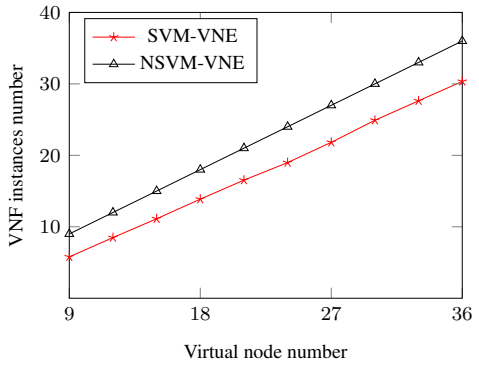

Fig. 6. VNF instances number with different virtual network.

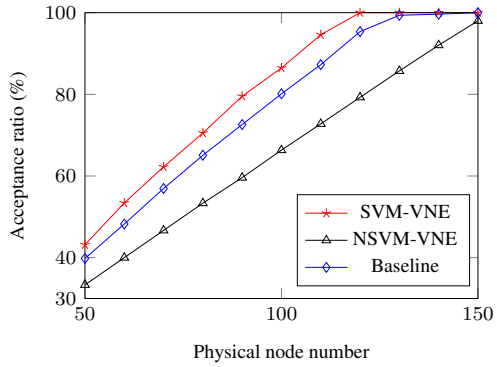

Fig. 7. Acceptance ratio with different physical networks.

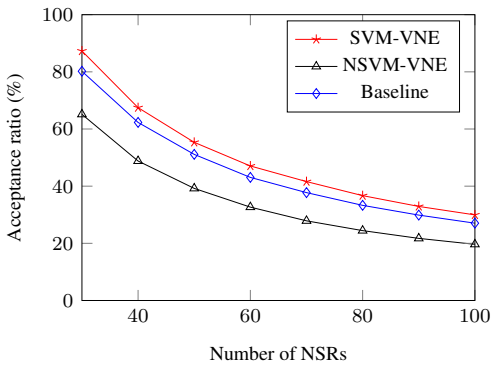

Fig. 8. Acceptance ratio with different NSRs.

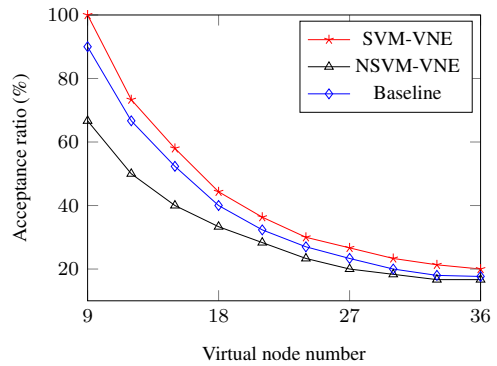

Fig. 9. Acceptance ratio with different virtual networks.

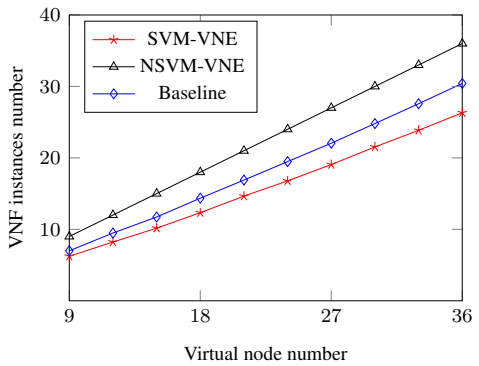

Fig. 10. VNF instances number with different virtual network.

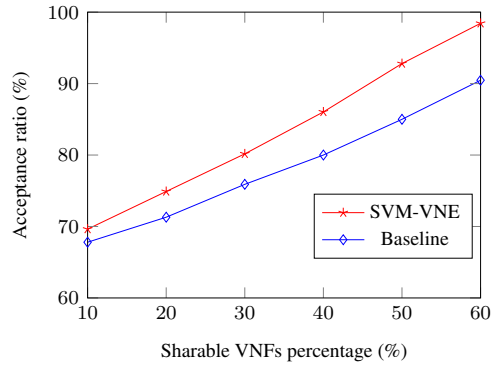

Fig. 11. Acceptance ratio with different VNF sharable ratio.

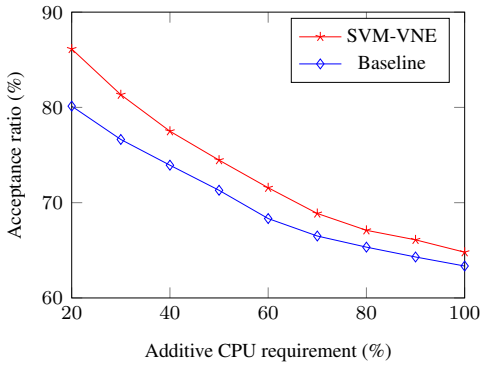

Fig. 12. Acceptance ratio vs. additive CPU requirement.

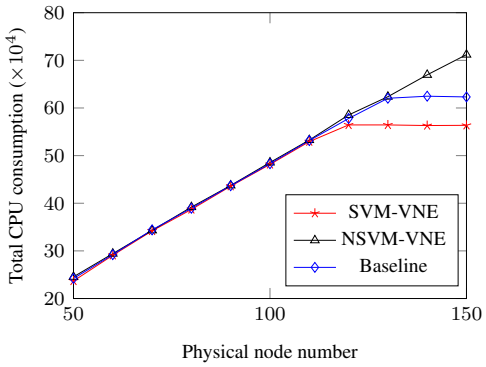

Fig. 13. Acceptance ratio with different VNF sharable ratio.

90.4%.

In Fig. 12, we show the NSR acceptance ratio by varying the additive CPU resource requirement of the sharable VNFs. For all the other figures, this parameter is set to 20%. The physical network setting is 100 physical nodes, and the number of NSRs is 30 with each NS contains 10 virtual nodes. The additive resource requirement is introduced in Section III.B, for the first VNF mapped on a physical node, the whole resource required by this VNF type is allocated, whereas for the following VNFs mapped on the same physical node with same shable type, the additive resource requirement is allocated to reduce resource consumption. The additive CPU requirement of sharable virtual nodes is represented by the percentage of its overall CPU resource requirement. Consequently, higher number shows higher resource requirement. We vary the additive CPU requirement from 20% to 100%, and the acceptance ratio of the two algorithms decreases because sharable VNFs require more resources. The result of NSVM-VNE is not shown because 100% additive CPU requirement actually refers to non-sharable VNFs of our algorithm. The performance of our algorithm outperforms the baseline algorithm. For 20% additive resource requirement, the acceptance ratio of our algorithm is 86.1%, and that of the baseline algorithm is 80.1%. Whilst for 80% additive resource requirement, the acceptance ratio of our algorithm is 76.6%, and that of the baseline algorithm is 75.4%.

Finally, we show the resource consumption of the three algorithms in Fig. 13 by varying physical network scale. The number of NSRs is 30, with each NS contains 10 virtual nodes. In the formulation of the SVM-VNE problem, our objective is to minimize the resource consumption (physical nodes resource that is represented by CPU, and bandwidth resources). These two kinds of resources are not measured by the same metric, and in our algorithm, we mostly focused on saving physical node resource by integrating sharable VNFs with the same type. Hence, we show here the total CPU consumption of the three algorithms with different physical network scales. In this figure, we calculated the CPU allocated towards the NSRs which are successfully mapped. Hence, the Fig. 13 should be analyzed together with Fig. 7 to show more comprehensive information. In Fig. 13, we could observe two parts: for physical network with less than 120 nodes; and for physical network with more than 120 nodes. For the first part, the three algorithms almost consume the same number of CPUs. However, as observed from Fig. 7, our SVM-VNE algorithm achieves the highest acceptance ratio, which means that with the same CPU consumption, our algorithm maps more NSRs. The similar CPU consumption comes from the facts that the physical network is underprovisioned, which is not sufficient to map all NSRs. Hence all the three algorithms are restricted by the physical resources. Then, for physical network with more than 120 nodes, firstly our algorithm maps all NSRs. As a result, the CPU consumption of our algorithm remains almost the same for physical nodes 120 to 150. Then, as the number of physical nodes increases, the baseline algorithm and the NSVM-VNE also maps all the 30 NSRs, and their CPU consumption also converges. However, for mapping the same number of NSRs, our algorithm consumes the least CPU resources.

VI. CONCLUSIONS

In this paper, we investigate the 5G E2E NS deployment problem with sharable VNF instances. We formulate the multiple coupled NS virtual network embedding problem through an ILP formulation. Our design goal is to minimize the resource consumption by integrating sharable VNF instances. We design VNF-sharing based NSR deployment algorithm. We conducted intensive simulation to evaluate the performance of our algorithm under different scenarios. We investigated the NSRs acceptance ratio, instantiated number of VNFs and CPU consumption under different physical network scales, NSRs resource requirements and sharable resources percentage. The simulation

results demonstrate that our algorithm achieves higher NSRs acceptance ratio, hence higher physical resource utilization, by comparing to a baseline algorithm. Further, our algorithm instantiates less VNFs, hence more VNFs are integrated. Finally, our algorithm consumes less CPU resources with higher NSRs acceptance ratio, which shows that our design goal is achieved.

REFERENCES

[1] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From network sharing to multi-tenancy: The 5G network slice broker," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 32–39, 2016.

[2] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, 2017.

[3] V. Nguyen, A. Brunstrom, K. Grinnemo, and J. Taheri, "SDN/NFV-based mobile packet core network architectures: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1567–1602, 2017.

[4] X. Zhou, R. Li, T. Chen, and H. Zhang, "Network slicing as a service: Enabling enterprises' own software-defined cellular networks," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 146–153, 2016.

[5] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commu. Surveys Tuts.*, vol. 20, no. 3, pp. 2429–2453, 2018.

[6] C. Liang and F. R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 358–380, 2015.

[7] X. Foukas, M. K. Marina, and K. Kontovasilis, "Orion: Ran slicing for a flexible and cost-effective multi-service mobile network architecture," in *Proc. ACM MobiCom*, 2017, pp. 127–140.

[8] C. Chang and N. Nikaein, "Ran runtime slicing system for flexible and dynamic service execution environment," *IEEE Access*, vol. 6, pp. 34 018–34 042, 2018.

[9] J. Li *et al.*, "Joint resource allocation and online virtual network embedding for 5G networks," in *Proc. IEEE GLOBECOM*, 2017, pp. 1–6.

[10] W. Guan, X. Wen, L. Wang, Z. Lu, and Y. Shen, "A service-oriented deployment policy of end-to-end network slicing based on complex network theory," *IEEE Access*, vol. 6, pp. 19 691–19 701, 2018.

[11] R. Ferrus, O. Sallent, J. Perez-Romero, and R. Agusti, "On 5G radio access network slicing: Radio interface protocol features and configuration," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 184–192, 2018.

[12] Z. Kotulski, T. W. Nowak, M. Sepczuk, M. Tunia, R. Artych, K. Bocianiak, T. Osko, and J. P. Wary, "Towards constructive approach to end-to-end slice isolation in 5G networks," *Eurasip J. Inform. Security*, vol. 2018, no. 1, p. 2, 2018.

[13] T. Truong-Huu, P. Murali Mohan, and M. Gurusamy, "Service chain embedding for diversified 5G slices with virtual network function sharing," *IEEE Commun. Lett.*, vol. 23, no. 5, pp. 826–829, 2019.

[14] S. Zhang, "An overview of network slicing for 5G," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 111–117, 2019.

[15] J. Pérez-Romero, O. Sallent, R. Ferrús, and R. Agustí, "On the configuration of radio resource management in a sliced ran," in *Proc. IEEE/IFIP NOMS*, 2018, pp. 1–6.

[16] R. Ravindran, A. Chakraborti, S. O. Amin, A. Azgin, and G. Wang, "5G-icn: Delivering icn services over 5g using network slicing," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 101–107, 2017.

[17] I. Benkacem, T. Taleb, M. Bagaa, and H. Flinck, "Optimal vnfs placement in cdn slicing over multi-cloud environment," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 616–627, 2018.

[18] J. Liu, Q. Yang, G. Simon, and W. Cui, "Migration-based dynamic and practical virtual streaming agent placement for mobile adaptive live streaming," *IEEE Trans. Network Service Manag.*, vol. 15, no. 2, pp. 503–515, 2018.

[19] A. N. Al-Quzweeni, A. Q. Lawey, T. E. H. Elgorashi, and J. M. H. Elmirghani, "Optimized energy aware 5G network function virtualization," *IEEE Access*, vol. 7, pp. 44 939–44 958, 2019.

[20] M. R. Raza *et al.*, "Dynamic slicing approach for multi-tenant 5G transport networks," *IEEE/OSA J. Optical Commun. Netw.*, vol. 10, no. 1, pp. A77–A90, 2018.

[21] N. Huin, B. Jaumard, and F. Giroire, "Optimal network service chain provisioning," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1320–1333, 2018.

[22] S. Agarwal, F. Malandrino, C. Chiasserini, and S. De, "Joint VNF placement and cpu allocation in 5G," in *Proc. IEEE INFOCOM*, 2018, pp. 1943–1951.

[23] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 2013.

[24] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 206–219, 2012.

[25] A. Blenk, P. Kalmbach, J. Zerwas, M. Jarschel, S. Schmid, and W. Kellerer, "Neurovine: A neural preprocessor for your virtual network embedding algorithm," in *Proc. IEEE INFOCOM*, 2018, pp. 405–413.

[26] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. IEEE INFOCOM*, 2014, pp. 1–9.

[27] A. Jarray and A. Karmouch, "Decomposition approaches for virtual network embedding with one-shot node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 1012–1025, 2015.

[28] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, Apr. 2011.

[29] Y. Mao, Y. Guo, H. Hu, Z. Wang, and T. Ma, "Sharing based virtual network embedding algorithm with dynamic resource block generation," *IEEE Commun. Lett.*, vol. 19, no. 12, pp. 2126–2129, 2015.

[30] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *Proc. IEEE CloudNet*, 2015, pp. 171–177.

[31] B. Yi, X. Wang, and M. Huang, "A generalized vnf sharing approach for service scheduling," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 73–76, 2018.

[32] A. L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[33] W. Chang, G. Shou, Y. Liu, Z. Guo, Y. Hu, and X. Jin, "Constructing scale-free topologies for low delay of 5G," in *Proc. IEEE PIMRC*, 2017, pp. 1–6.

Chengli Mei received his Ph.D. degree from Shanghai Jiaotong University. He is a Director and a Professorate Senior Engineer of China Telecom Technology Innovation Center. His research interests focus on the techniques of mobile communication networks.

Jiayi Liu obtained her Bachelors of Science degree in Electronic Engineering from Xidian University in Xi'an China, in 2007. She received her Master of Science and Ph.D., both in Computer Science, from Telecom-Bretagne and Rennes 1 University, France, in 2009 and 2013, respectively. Since 2014, she works as Lecturer with Xidian University, in School of Telecommunication Engineering and Guangzhou Institute of Technology. Her research interests include 5G network, content distribution, Network resource scheduling.

Jinyan Li received her Master's degree from the Beijing University of Post and Telecommunication. She is a Senior Engineer of China Telecom Technology Innovation Center. Her research focuses on the standardization and technique evolution of mobile networks.

Lei Zhang received the Master's degree from Beijing University of Post and Telecommunication. She is a Junior Researcher of China Telecom Technology Innovation Center. Her research interest focus on the techniques of mobile communication networks.

Menghan Shao received the B.S. degree in Communication Engineering from Xidian University, Xi'an, China, in 2018, where she is currently pursuing the M.S. degree in Electronics and Communication with the State Key Laboratory of Integrated Service Networks. Her research interests include 5G network, mobile edge computing.