A Mathematical Model and Dynamic Programming Based Scheme for Service Function Chain Placement in NFV

Yansen XU†(a), Student Member and Ved P. KAFLE††, Member

SUMMARY  Service function chain (SFC) is a series of ordered virtual network functions (VNFs) for processing traffic flows in the virtualized networking environment of future networks. In this paper, we present a mathematical model and dynamic programming based scheme for solving the problem of SFC placement on substrate networks equipped with network function virtualization (NFV) capability. In this paper, we first formulate the overall cost of SFC placement as the combination of setup cost and operation cost. We then formulate the SFC placement problem as an integer linear programming (ILP) model with the objective of minimizing the overall cost of setup and operation, and propose a delay aware dynamic programming based SFC placement scheme for large networks. We conduct numeric simulations to evaluate the proposed scheme. We analyze the cost and performance of network under different optimization objectives, with and without keeping the order of VNFs in SFC. We measure the success rate, resources utilization, and end to end delay of SFC on different topologies. The results show that the proposed scheme outperforms other related schemes in various scenarios.

key words: SFC, NFV, resource management, network management

1. Introduction

In the era of the fifth generation (5G) mobile communication, new services and applications are emerging, and the network is required to provide various types of services. For example, 4K video services require enhanced mobile broadband (eMBB) communication, self-driving cars require high bandwidth, ultra-reliable and low latency communication (URLLC), and smart cities with densely deployed IoT devices require massive machine type communication (mMTC)[1].

With the advent of new technologies such as network function virtualization (NFV) [2] and software-defined networking (SDN), the different types of services be offered by creating virtual network slices over the same physical network infrastructure. The NFV and SDN also provide a platform to adjust the resources (e.g., CPU, memory, storage and bandwidth) allocated for a network function of the virtual network slice as the service demand increases. However, we require new mechanisms to determine the required amount of resources by network service whose resource demand changes dynamically so that the network resources allocated to the service processing the traffic can be timely adapted in order to maintain the quality of service requirements. Improperly network resource allocation for the service will result in either high resource waste, or invalid service level agreement, and decreases the user experiences. Therefore, the network resources must be utilized efficiently to reduce the capital expenditure (CAPEX) and operating expenditure (OPEX).

In addition to the data forwarding, network functions (NF) are essential to improve security, enhance performance and reduce cost [3]. Traditional network functions are vendor-specific, lack of scalability and flexibility because they are permanently fixed with hardware [4]. Due to this hardware-based fashion, it is difficult and error-prone for traditional networks to add or expand services since the capacity, location and type of functions have been fixed once the networks are deployed. Thus, the network resources must be over-provisioned to cope with the potential increase in the future demand of resources.

Network function virtualization (NFV) decouples the network functions from hardware. NFV enables the network function to run on a piece of generic hardware instead of dedicated one. By utilizing virtualization technology, network functions can be virtualized by installing the virtualized network function (VNF) into virtual machine or container. The VNF can be installed, configured, migrated and adjusted automatically for services to adapt to the dynamically changing demands of network and computation resources.

Despite these advantages of NFV, there many issues yet to be solved. One fundamental issue is the VNF placement of a service function chain (SFC). An SFC defined by the Internet Engineering Task Force (IETF) is an ordered set of network functions for processing traffic flows [5]. In the context of NFV, an SFC is composed of several VNFs, which could be deployed across the physical network. An SFC placement problem is to find out a set of appropriate substrate nodes and path through the substrate network for SFC realizing functions, so that the CPU and bandwidth requirements of the SFC are fully satisfied. The selected set of substrate nodes and path must guarantee the end to end latency requirement and the required order of VNFs in the SFC. The placement of VNFs in SFC is significantly important since a proper placement of VNFs can achieve maximum profit and minimum cost by efficiently utilizing network resources such as CPU, memory and bandwidth, as well as improving the performance, availability and reliability. Figure 1 shows an example of SFC deployment on substrate network.
In this paper, we separate the overall cost of deploying SFC into operation cost and setup cost, and formulate the SFC placement problem as an Integer Linear Programming (ILP) model with the objective of minimizing the overall costs of setup and operation, while satisfying the bandwidth and CPU constraints. We propose a dynamic programming based SFC placement scheme to adapt to large networks as well as guarantee the end-to-end delay requirements of SFC. We also consider guaranteeing the intended order of VNFs, which is a critical requirement for SFCs.

The contributions of this work are as given below.

1. We define the setup and operation costs and formulate the SFC placement problem as an ILP model with the objective of minimizing these costs. The model takes CPU and bandwidth resources as constraints, and satisfies the latency requirements and the order of VNFs in SFC.

2. We propose a dynamic programming based SFC placement scheme to guarantee the end-to-end delay of SFC in the substrate network as well as satisfying the CPU, bandwidth and order constraints.

3. We carry out experimental study to evaluate the cost performance of the network under variation optimization objectives, and we found the setup cost is affected most under different objectives.

4. We perform experimental study on the cost performance of network with and without keeping the order of VNFs in SFC requests. We found that the overall cost of network is higher with keeping the order of VNFs in SFC.

5. We perform experimental study of delay aware dynamic programming based SFC placement scheme. The results show that the scheme can adapt to different network topologies.

The rest of this paper is organized as follows. Section 2 presents the related works. Section 3 defines the setup and operation costs and describes the ILP model. Sections 4 and 5 present the experimental study of the cost and performance. Section 6 concludes this paper.

2. Related Work

Standardization activities and research are ongoing in the field of NFV. There are many published white papers [2], [6], use cases [7], and surveys [8]–[10] of NFV.

Authors in [11]–[14] investigated the virtual network embedding (VNE) problems to find and allocate network resources for configuring virtual networks. Although the SFC placement problem is related to VNE problem, it is more specific to guaranteeing the order of VNFs in the SFC.

A. Dwaraki et al. [15] proposed an adaptive SFC placement based on layered graphs [18] to achieve minimum end to end latency of SFC. N. Huin et al. [16] also utilized the layered graphs and formulated the SFC placement problem into two ILP. The authors also investigated the best compromise between the number of nodes to host one type of VNFs and the bandwidth requirements. Y. Xu et al. [17] proposed a reliable SFC placement scheme by avoiding a single-point failure in the network of layered graphs.

A. Tomassilli et al. [19] formulated the problem as a set cover problem and minimized the total deployment cost while keeping the order of VNFs in SFC. The authors proposed a naïve and faster greedy algorithm, and an optimal algorithm for tree topologies. R. Cohen et al. [20] formulated the SFC placement problem as a facility location problem and a generalized assignment problem to minimize overall network cost. J. Liu et al. [21] proposed a dynamic service function chain deployment and readjustment model that jointly optimize the deployment of new users’ SFCs and the readjustment of in-service users’ SFCs and consider the trade-off between resource consumption and operational overhead.

However, these works only consider network resources such as CPU and bandwidth but do not take the latency or order of VNFs in SFC into consideration. They assume that the SFC paths in the substrate network have been predefined and consider the placement of VNFs along the predefined paths only. They do not consider the operation or setup cost.

3. Network Cost and Problem Model

In this section, we define the network setup and operation costs. Then, we formulate the SFC placement problem as an ILP model with the objective of minimizing the overall cost of setup and operation.

3.1 Network Setup and Operation Costs

The network setup cost is defined as the cost charged only once during the process of placing a VNF on a substrate node, such as the license fee of the VNF. This cost is supposed to be machine type independent and only related to the type of VNF the machine hosts and the number of machines to host it, without considering how much resource this type of VNF consumed or how many SFCs share this.
type of VNF on the same machine. The setup cost can be minimized by the collocation of same type VNFs in the same machine.

The operation cost is the cost of processing a unit of VNF on a substrate node. Differing from the setup cost, the operation cost is machine type dependent, which means different machines process the same type of VNF at different costs. The operation cost is charged based on the amount of CPU requirement of the VNF as well as the type of machine. The operation cost can be minimized by finding the optimal location for processing VNFs for each SFC request at the least cost.

3.2 Modeling of Substrate Network

We model a substrate network as a directed graph \( G = (V, E) \), where \( V \) and \( E \) represent the set of substrate network nodes and edges, respectively. \( C_e \) denotes the CPU resource capacity of \( v \in V \). \( B_{(u,v)} \) and \( L_{(u,v)} \) denote the bandwidth resource capacity and the latency of edge \((u, v)\), respectively, where \((u, v) \in E\).

A. Modeling of type of VNF

We use the following notations to represent the type and cost of VNF.

\( \varphi_\gamma \): a set of VNF types and \( \gamma (\gamma \in \Gamma) \) is a type of VNF.

\( \omega_{\gamma,v} \): per CPU operation cost for \( \gamma \) type VNF on substrate network node \( v \).

The operation cost for an SFC, which requests \( a \) amount of CPU to operate VNF \( f \) on the substrate network node \( v \), is calculated as

\[
\text{operation cost} = a \cdot \omega_{\gamma,v}
\]

B. Modeling of SFC request

The SFC request is modeled with 5-tuple \((s_i, d_i, l_i, b_i, \gamma_i)\), where \( i \) is the \( i \)-th SFC request in the SFC requests set.

\( s_i \): source node of SFC \( i \) in the substrate network, where \( s_i \in V \).

\( d_i \): destination node of SFC \( i \) in the substrate network, where \( d_i \in V \) and \( d_i \neq s_i \).

\( l_i \): maximum end-to-end latency requirement of SFC \( i \). The end-to-end latency of a path for an SFC request cannot exceed this value.

\( b_i \): bandwidth requirement of SFC \( i \). Here, we assume that the bandwidth requirements between any two neighbor VNFs in SFC are identical.

\( \gamma_i \): a vector of ordered VNFs of SFC \( i \), and \( g_{i,j} \) is the \( j \)-th VNF in \( \gamma_i \).

\( \gamma_i \): a vector of ordered VNF types for SFC \( i \). It has the same size as \( \gamma_i \). Specifically, \( t_{i,j} \in \Gamma \) represents the type of \( g_{i,j} \), i.e., the type of the \( j \)-th VNF in \( \gamma_i \).

\( c_i,j \): a vector of CPU resource requirement for SFC \( i \). It has the same size as \( \gamma_i \). Specifically, \( c_{i,j} \) represents the CPU resource requirement of \( g_{i,j} \), i.e., the CPU resource requirement of the \( j \)-th VNF in \( \gamma_i \).

C. Decision variables

\( x_{\gamma,v} \): a Boolean variable that equals 1 if the type of VNF \( \gamma \) is deployed on substrate network node \( v \), and 0 otherwise.

\( y_{i,j,v} \): a Boolean variable that equals 1 if the \( j \)-th VNF in the \( i \)-th SFC is deployed on the substrate network node \( v \), and 0 otherwise.

\( z_{i,(u,v)} \): a Boolean variable that equals 1 if the edge \((u, v)\) is on the path of the \( i \)-th SFC where \((u, v) \in E\).

\( k_{i,v} \): an integer variable that indicates the rank of substrate node \( v \) along the path of the \( i \)-th SFC in the substrate network. This variable ranks the node along the path in the substrate network for one SFC request to avoid loop and keep VNFs’ order in SFC.

D. Constraints

1) Node CPU capacity constraints:

\[
\sum_j y_{i,j,v} \cdot c_{i,j} \leq C_v \quad \forall v \in V
\]

Equation (1) ensures that for all SFC requests, the total CPU requirements of VNFs that deployed on the substrate network node \( v \in V \) cannot exceed the CPU capacity of node \( v \).

2) Placement constraints:

\[
\sum_v y_{i,j,v} = 1 \quad \forall j \text{ in } i, \quad \forall i
\]

Equation (2) ensures that for any VNF in an SFC request, and for any SFC request, the VNF should be deployed on one and only one substrate network node.

\[
\sum_v y_{i,j,v} \leq 1 \quad \forall i, \forall v \in V
\]

Equation (3) ensures that no more than two VNFs in one SFC request are deployed on the same substrate network node, i.e., Eq. (3) prevents the collocation of two or more VNFs in one SFC request.

\[
x_{i,v} = 1 \text{ if } y_{i,j,v} = 1 \quad \forall i, \forall v \in V
\]

Equation (4) ensures that if the \( j \)-th VNF in the \( i \)-th SFC is a \( t_{i,j} \) type of VNF, and this VNF is placed on the substrate network node \( v \), then node \( v \) must be able to host the \( t_{i,j} \) type VNF.

\[
y_{i,j,v} = 0, \text{ if } v = s_i, \quad \forall i, \quad j \tag{5}
\]

\[
y_{i,j,v} = 0, \text{ if } v = d_i, \quad \forall i, \quad j \tag{6}
\]

Equations (5) and (6) ensure that VNFs in SFC requests cannot be deployed on the source node and destination node of SFC in the substrate network.

3) Bandwidth constraints:

\[
\sum_v z_{i,(u,v)} \cdot b_i \leq B_{(u,v)} \quad \forall (u, v) \in E
\]
Equation (7) ensures that the total bandwidth requirement of all SFC requests cannot exceed the total capacity of substrate network bandwidth for any substrate edge \((u, v) \in E\).

4) Latency constraints:
\[
\sum_{(u,v) \in E} z_{i,(u,v)} \cdot L_{(u,v)} \leq l_i \quad \forall i
\]  
(8)

Equation (8) ensures that for any SFC request, the path latency in the substrate network for this SFC request cannot exceed the maximum latency requirement of the SFC.

5) Flow constraints:
\[
\sum_{(u,v) \in E} z_{i,(u,v)} - \sum_{(v,u) \in E} z_{i,(v,u)} = 0
\]
\[\forall u, v \in E, u \neq s_i, v \neq d_i, \forall i\]  
(9)

\[
\sum_{u \in R} z_{i,(u,v)} - \sum_{v \in R} z_{i,(v,u)} = 0
\]
\[\forall v \in R, R \text{ is a set of } v \text{'s neighbor nodes}, v \neq s_i, v \neq d_i, \forall i\]  
(10)

Equation (11) ensures that for any substrate network node \(v\), \(R\) is a set of \(v\)'s neighbor nodes. If \(v\) is neither source node nor destination node, the amount of incoming flow entering node \(v\) is always equal to the amount of outgoing flow leaving node \(v\) (i.e., the number of edges ending at node \(v\) is always equal to the number of edges starting from node \(v\)).

Equation (12) ensures that if substrate network node \(v\) is source node, there are always a certain amount of outgoing flow leaving node \(v\) (i.e., one edge starting from node \(v\)); if substrate network node \(v\) is destination node, there are always a certain amount of incoming flow starting from node \(v\) (i.e., one edge ending at node \(v\)).

\[
z_{i,(u,v)} + z_{i,(v,u)} \leq 1 \quad \forall (u,v) \in E \quad \forall i
\]  
(13)

Equation (13) ensures that one edge in the substrate network can be traveled at most once.

\[
\sum_{(u,v) \in E} z_{i,(u,v)} = 1 \text{ if } y_{i,j,v} = 1 \quad \forall (u,v) \in E \quad \forall i
\]  
(14)

\[
\sum_{(u,v) \in E} z_{i,(v,u)} = 1 \text{ if } y_{i,j,v} = 1 \quad \forall (u,v) \in E \quad \forall i
\]  
(15)

Equations (14) and (15) ensure that the path must travel through the substrate network node \(v\) if the \(j\)-th VNF in the \(i\)-th SFC is deployed on node \(v\).

\[
k_{i,u} - k_{i,v} \geq 1 - |V| \cdot (1 - z_{i,(u,v)}) \quad \forall (u,v) \in E \quad \forall i
\]  
(16)

\[
k_{i,u} = 0 \text{ if } u = s_i \quad \forall i
\]  
(17)

Equations (16) and (17) ensure that no isolated loop is generated in the network. An isolated loop is a path that starts from a substrate node and ends at the same substrate node. All nodes in the loop satisfy constraints (11), but this loop is apart from the path of SFC, which starts from source node, ends destination node and connects the nodes that host VNFs. To avoid this isolated loop, we assign the substrate node along the path of SFC a rank number as shown in (16) (17). If \(z_{i,(u,v)}\) is equal to 1, that \((u,v) \in E\) is on the path of SFC, and node \(v\) comes after \(u\) as it is a directed graph, the rank of \(v\), \(k_{i,v}\), is always higher than the rank of \(u\), \(k_{i,u}\). If \(z_{i,(u,v)}\) is equal to 0, it has no impact on the rank of nodes \(u\), \(v\), since \(|V|\) is the number of nodes in substrate network. It is obvious that the destination node will always have a higher rank than the source node. Thus, the path will never travel back to the source node, and the possibility of isolated loops is eliminated.

7) VNFs order constraints:
\[
k_{i,u} - k_{i,v} \geq 0 \text{ if } y_{i,j,+1,v} = 1 \text{ and } y_{i,j,v} = 1
\]
\[\forall i, j, \forall u, v \in V\]  
(18)

Equation (18) ensures that if substrate network node \(u\), \(v\) can host the \((j+1)\)-th and \(j\)-th VNF in one SFC request, respectively, then substrate node \(u\) should come after substrate node \(v\) in the flow path to keep the order of VNFs.

E. Optimization objective

The objective is to minimize the setup cost and operation cost. Total setup cost \(S\) is defined as
\[
S = \sum_{\gamma \in \Gamma} \sum_{v \in V} x_{v,\gamma} \cdot \varphi_{v}\gamma
\]

Here, \(x_{v,\gamma}\) is a Boolean variable that equals 1 if the type of VNF \(\gamma\) is deployed on substrate network node \(v\) and \(\varphi_{v}\gamma\) is the setup cost of \(\gamma\) type VNF on one substrate node.

\[
\sum_{v \in V} x_{v,\gamma} \cdot \varphi_{v}\gamma
\]
indicates the number of nodes in the substrate which host \(\gamma\) type of VNF, and \(\sum_{v \in V} x_{v,\gamma} \cdot \varphi_{v}\gamma\) is the summation of setup cost for \(\gamma\) type VNF.

By minimizing the setup cost, one type of VNF can be deployed on as minimum number of substrate network nodes as possible. Here we have assumed that VNF license fee is issued accorded to the number of machines hosting it.

Operation cost \(O\) is defined as
\[
O = \sum_{i} \sum_{j} \sum_{v} y_{i,j,v} \cdot c_{i,j} \cdot \omega_{i,j,v}
\]

Here, \(\omega_{i,j,v}\) is per CPU operation cost for substrate node \(v\) operating VNF \(g_{i,j}\), and \(c_{i,j}\) is the amount of CPU requirement for VNF \(g_{i,j}\). Thus, \(y_{i,j,v} \cdot \omega_{i,j,v}\) is the total operation cost for substrate node \(v\) operating VNF \(g_{i,j}\). This objective sums up all the cost of VNFs of all SFC requests. By minimizing this operation cost, the VNFs can be deployed with a minimum cost of substrate network nodes.

The final optimization objective is to minimize \(S\),
which is the summation of setup and operation costs, as given below:

\[
\text{Minimize } \mathcal{C} = S + O
\]

F. Network resource utilization ratio

The network resource utilization ratio is defined as:

\[
r = \frac{\xi_u}{\xi_t}
\]

Where \(\xi_u\) is the summation of network resource requirements of all deployed SFC requests, and \(\xi_t\) is the summation of total network resource capacity. Network resource can be bandwidth or CPU, and if it is bandwidth resource, \(\xi_u\) and \(\xi_t\) are defined as:

\[
\xi_u = \sum_{i} \sum_{(u,v) \in E} z_{i,(u,v)} \cdot b_i \quad \xi_t = \sum_{(u,v) \in E} B_{(u,v)}
\]

and if it is CPU resource, \(\xi_u\) and \(\xi_t\) are defined as:

\[
\xi_u = \sum_{i} \sum_{j} \sum_{v \in V} y_{i,j,v} \cdot c_{i,j} \quad \xi_t = \sum_{v \in V} C_v
\]

4. Dynamic Programming Based Optimal Algorithm

In this section, we propose a dynamic programming (DP) based SFC placement scheme which achieves the minimum end to end latency in the SFC with respect to the CPU and bandwidth constraints, as well as guaranteeing the order of VNFs specified in SFC. In this scheme, we assume that the source (src) and destination (dst) of SFC have been decided prior to running this algorithm. Both src and dst do not request CPU resource, but only request bandwidth resources.

4.1 Algorithm Description

In the SFC placement, the basic problem is to find a suitable location in the substrate network for deploying a VNF while satisfying all constraints. The SFC placement problem is solved by finding appropriate nodes for all VNFs of the SFC. Intuitively, when deploying one VNF, we must consider and recalculate the location for its precedents and decedents, which is incomputable in large networks. By leveraging DP technique, our scheme calculates the location for one VNF only once, without recalculating its precedents and decedents. The scheme calculates feasible locations and stores the information of the feasible locations for VNFs as the order of VNF in SFC. By backtracking the path from the destination to source node, the scheme can guarantee the order of VNFs, which is a critical requirement for SFC placement.

To examine the feasibility of a substrate node for deploying a VNF, the scheme uses the following three metrics: CPU resource sufficiency, connectivity, and bandwidth resource sufficiency.

The scheme also introduces four matrices \(A\), \(L\), \(P\), and \(T\) to store the information of feasible locations. Specifically, \(A(v, f)\) is a Boolean value that is set to true, if substrate node \(v\) is feasible to host VNF \(f\). This matrix is used for examining the connectivity between two neighboring VNFs. \(L(v, f)\) records the minimum latency from the substrate node that hosts src to the substrate node \(v\) on which VNF \(f\) is deployed. This matrix is used to find the minimum end to end latency of SFC. \(P(v, f)\) stores the path composed of the substrate nodes along this path. This matrix is used for backtracking the complete path from src to dst. \(T(v, f)\) is a temporary substrate network state of residual CPU resources and residual bandwidth resources, if the VNF \(f\) was placed on substrate node \(v\). This matrix is necessary since once a VNF is deployed on a substrate node, the deployment of succeeding VNFs needs to refer to a new network state, such as available amount of CPU and bandwidth resources.

Algorithm 1 and Algorithm 2 show the detail of the scheme. In Algorithm 1, for any VNF \(f\) in SFC, the mapping scheme checks the feasibility of all substrate nodes whether they could host \(f\). If all substrate node cannot host
vnf, the algorithm terminates and returns fail. If at least one node is feasible to host f, the algorithm continues to map the next VNF.

Algorithm 2 shows the detail of checking the feasibility of nodes one by one for hosting a VNF f. The procedure first examines whether substrate node v has sufficient available CPU resources for hosting f. If n has no sufficient CPU resources, the algorithm immediately returns false (note A(v, f) is initialized as false). If v has sufficient available CPU resources for f, the algorithm then checks the connectivity and bandwidth sufficiency between n and all other substrate nodes that could host the precedent of v.

From line 6 in Algorithm 2, for any substrate node v' (v' ≠ v), if A(v', f) = true, which indicates that v' is feasible to host f_p, where f_p is the precedent of f, the algorithm finds shortest path p_{ov'} between v and v' in terms of latency. If the shortest path p_{ov'} exists, and p_{ov'} has sufficient available bandwidth to satisfy the bandwidth request between f_p and f, the algorithm calculates the summation of L(v', f_p) (i.e. the latency from src to v') and the latency of p_{ov'}. The algorithm returns false if one of the following three conditions is satisfied: A(v', f_p) is not true, no shortest path p_{ov'} exists, and the available bandwidth of p_{ov'} is not sufficient.

From line 11 in Algorithm 2, it tries to find the minimum latency path between all pairs of v and v'. After all iterations of v' ∈ substrate nodes, v' ≠ v, L(v, f) is set to the lowest value of the latencies of all possible paths from src to v, via v'. Simultaneously, the procedure updates A(v, f) by true, which indicates that v is feasible for hosting f, updates P(v, f) by concatenating P(v', f_p) and p, which is the path from src to v, updates T(v, j) by T(v', f_p), which is the network state if f_p were deployed on v'. It finally allocates CPU and bandwidth values on T(v, f) as requested for f.

Since Algorithm 1 starts from the first VNF (the one next to src since the location of src has been pre-determined), to iterate VNFs in SFC one by one, the specific order of VNFs in SFC can be guaranteed. For any VNF f, if there is a substrate node v, so that Algorithm 2 could return true, there must exist a path in substrate network, which connect all VNFs orderly from src to f. And it can be easily understood that this path has sufficient CPU and bandwidth resources, as well as a minimum latency for allocation.

4.2 Extraction of Route Information

In scheme, if the connectivity, CPU and bandwidth requirements are satisfied, the algorithm will return success and the recorded information P(v_{src}, dst) where v_{src} is the substrate network node which hosts dst in the SFC. P(v_{src}, dst) is composed of a set of paths for connecting two neighboring VNFs in SFC. For example, P(v_{src}, dst) = [v_{src}, [v_{f_1},..., v_{f_p}], [v_{f_1},..., v_{f_p}],..., [v_{f_1},..., v_{f_p}], [v_{f_1},..., v_{f_p}]] and each element of P, e.g. [v_{f_1},..., v_{f_p}], is a set of substrate node in which the first node (say v_{f_1}) hosts VNF f_{p_1}, and the last node (say f) hosts VNF f. f_{p_1} and f are neighbors and f_{p_1} is preceding f. The nodes between v_{f_1} and v_{f_2} in [v_{f_1},..., v_{f_p}] are in the path from v_{f_1} to v_{f}. Thus, once the algorithm succeeds to obtain P, it can easily extract the route information.

4.3 Matrices Initialization

The scheme refers to the information stored in the nodes which is feasible to host the precedent of a VNF, checks the connectivity, and selects the minimum latency path between two substrate nodes that host src and dst of SFC. Thus, the scheme needs the initialization of matrices for backtracking from dst to src. Since src is pre-determined, the matrices are initialized as follows:

Set A(v_{src}, src) = true and all other elements in A as false, where v_{src} is the substrate node which hosts src of SFC. Set L(v_{src}, src) = 0 and all other elements in L as infinity. Set P(v_{src}, src) = [[src]] and all other elements in P as null. Set T(v_{src}, src) as a copy of network state which contains the currently available CPU and bandwidth resources, and all other elements in T as null.

4.4 Computation Complexity

The scheme finds proper locations in the substrate network for hosting the VNFs one by one in the specific order of SFC. This requires |V| times iterations, which is equal to the number of VNFs in SFC. For each VNF, the scheme examines all substrate network nodes |V| times, and in each examination of substrate node, the scheme checks the connectivity |V| − 1 times. Thus, the computation complexity is $O(|V|^2)$.

4.5 Consideration on Ultra-Large-Scale Network

For ultra-large-scale substrate network, we consider that the network could be divided into clusters. Each cluster is composed of proper number of nodes, such as 100. Then, within this range, it can be solved very well by applying our scheme. In addition, the cluster method is also necessary if taking latency into account. For instance, due to the latency constraint, the solution can be obtained from only within a certain size area of the network, thus, no need to search the whole network to find the solution. However, how to divide the large network into clusters is outside the scope of this paper, and we add this consideration to our future work.

5. Experimental Evaluation of ILP Model

In this section, we study the effects of different optimization objectives on the overall network cost. We also study the cost and performance of network with and without maintaining the order of VNFs in SFC requests.

5.1 Parameter Setting

The substrate network topology is consisting of six nodes and nine edges as shown in Fig. 1. Similar to [21], the CPU capacities of nodes and bandwidth capacities of edges are
randomly chosen from [80, 100] units, and the latency of each edge is randomly chosen from [2, 3] units. The substrate network supports 4 VNF types. The number of VNFs in one SFC is chosen from [2, 4] randomly. The CPU requirement for one VNF is chosen from [3, 5] units, and the bandwidth requirement is chosen from [4, 6] units, randomly. The end to end latency requirement of one SFC is a value obtained by multiplying the number of VNFs in the SFC and a number randomly chosen from [4, 6]. The source and destination nodes of an SFC request are randomly chosen from the substrate network nodes, and they cannot be the same node. To evaluate the proposed SFC placement model, and for comparison of different costs, setup cost and operation cost are modeled as uniformly distributed within [5, 10]. This setting of costs can be suitable for early stage of operation of the network, when the setup cost and operation cost are comparable. The numbers of SFC requests for once experiment vary from 5 to 40. All the SFC requests arrive at the network at the same time. To increase the statistical accuracy, we generated 50 independent SFC requests and averaged the results.

5.2 Cost with Different Optimization Objectives

Figure 2 shows the results of different costs at three different objectives: (1) $S + O$, where both setup cost and operation cost are considered jointly as the objective, (2) only $O$, where only operation cost is considered as the objective, and (3) only $S$, where only setup cost is considered as the objective. Figure 2 (a) shows the results of setup cost at different objectives. The $x$-axis is the number of SFC requests. The setup cost has an upper-limit since the setup cost depends only on the type of VNFs and the numbers of substrate network nodes which host the VNFs. The value of setup cost is small when the number of SFC requests is small. This indicates that same type VNFs in different SFC requests are collocated on one or few substrate network nodes. As the number of SFC request increases, the setup cost increases, since only a small number of substrate network nodes can no longer satisfy the requirements of SFC requests for one type of VNF. Thus, the number of substrate network nodes for hosting one type of VNF must be increased to provide more resources, which increases the setup cost. Finally, the setup cost reaches the upper-limit, indicating that for one type of VNF has been deployed on almost all nodes in the substrate network. An interesting result shows that the setup cost increases much faster when the optimization objective is only $O$. This is reasonable since when the objective is only minimizing operation cost, the model finds solutions that have least operation cost without considering the collocation of the same type of VNF even when the network workload is light, which results in a high setup cost. However, when the optimization objective is only $S$, the collocation of same type of VNF is considered as the most important target to slow increasing of setup cost.

Figure 2 (b) shows the results of operation cost at different optimization objectives. Note that the operation cost incurs in the substrate node processing VNFs, and this cost varies according to the type of substrate nodes as well as the type of VNF. When only setup cost (only $S$) is taken as the objective, operation cost is higher than that at other objectives, since the model tries to find as few substrate nodes as possible to collocate the same type VNFs without considering whether these nodes process the VNF at a lower cost.

Figure 2 (c) shows the overall cost which is the summation of setup cost and operation cost. The overall cost is in a near-linear fashion which is similar to the operation cost shown in Fig. 2 (b). Comparing operation cost with setup cost in $y$-axis, the operation cost dominates the main part of the overall cost which results a similar trend as operation cost.

5.3 The Impact of Keeping Order of VNFs in an SFC

Keeping the specified order of VNFs in SFC is critical for some SFCs to realize the whole service, such as an encryption function must come before the decryption function. However, in some other scenarios, VNFs in SFC are independent and unrelated, and placing them in an order is not necessary. For instance, as illustrated in [5], in an SFC composed of three VNFs of Intrusion Detection System (IDS), Intrusion Prevention System (IPS) and Firewall, the VNFs can be in an arbitrary order. In addition, with the rapid softwareization of network functions, new types of VNFs may emerge and they may or may not require to be placed in an order. Therefore, as in [24], to analysis how the order affects the cost and resource utilization, we conduct the experiments by keeping and not keeping the order of VNFs in an SFC request, and provide a comparison in terms of setup.
cost, operation cost, and bandwidth utilization.

Figure 3 shows setup cost, operation cost and bandwidth utilization ratio with and without keeping the order of VNFs. The setup cost without keeping the order of VNFs is much smaller compared to that with keeping the order. This is because the model has more degrees of freedom to find optimal solutions when relaxing the constraint of the order of VNFs in the SFC requests. It shows that the setup cost increases gradually when not keeping order of VNFs. This is because the model would not look for a new node to arrange the resource provisioning if one node hosting a type of VNF has sufficient CPU resources.

Figure 3 (b) shows the operation cost with and without keeping the order of VNFs in SFC requests. The operation cost without keeping the order of VNFs is smaller than the one with keeping the order of VNFs since the model can find and use the substrate node as solution which has the least operation cost for one type of VNF when not considering the order constraint, and this leads a lower operation cost.

Figure 3 (c) shows the results of bandwidth utilization ratio with and without keeping the order of VNF in SFC requests. The bandwidth utilization without keeping the order of VNFs is larger than that with keeping the order of VNFs in SFC requests. Since the summation of setup cost and operation cost is the optimizing objective, which is dependent on both the number of nodes, and location of nodes, the model only solves the problem by finding the most optimal nodes which have least cost, without taking the constraint of order of VNFs in SFC.

6. Experimental Evaluation of Dynamic Programming-Based Scheme

We conducted a numerical simulation to evaluate our dynamic programming-based scheme by a simulator written in Python. We generated the network topology on basis of Erdős-Rényi graph [22], [23].

Similar to [14], [15], [19], the number of nodes in substrate network is taken as 50, and the connectivity probability of any two nodes is set to 0.1, 0.2, and 0.5. Specifically, the probability value of 0.1 indicates that the substrate network is sparsely connected with few links, such as Germany50[19], [25], which contains 50 nodes and 88 links; the probability value of 0.5 indicates that the substrate network is a densely connected network; such as a partial mesh network; the probability value of 0.2 indicates that the substrate network is in between the sparse and dense networks. The CPU capacity of a node and bandwidth capacity of a link in substrate network are randomly chosen from 50 to 100 units, respectively, and the latency of a link is randomly chosen from 1 to 5 units. The CPU capacity of a node and bandwidth capacity of a link in substrate network are randomly chosen from 50 to 100 units, respectively, and the latency of a link is randomly chosen from 1 to 5 units.

Similar to [14], the number of VNFs in an SFC is randomly chosen from 2 to 6. The CPU request of a VNF and the bandwidth request between two neighboring VNFs are randomly chosen from 1 to 50 units, respectively. Different from the experiments of ILP, which all SFC requests arrived at the same time, in this experiment, the SFC requests arrived at and left from the substrate network dynamically. SFC request arrived in a Poisson process with a mean of 5 seconds and each request had an exponential lifetime with a mean of 500 seconds. If the two or more SFC requests arrived in a very short interval and the previous one had not yet been deployed, following SFC requests were inserted into a queue to wait. The total number of SFC requests were 500.

6.1 Performance of Deployment Success Rate

Figure 4 (a) shows the success rate as the SFC requests arrive at the substrate network. We evaluated the success rate on three different substrate network topologies, whose edge connectivity probability $p$ were 0.1, 0.2, and 0.5. Three topologies have the same trend, that, at low workload, the success rate is 100% since the network resource is sufficient for deploying all incoming SFC requests. As more SFC requests arrived, the network load got heavier and the success rate decreased and finally converged to around 70% when $p = 0.2$ and $p = 0.5$, and 60% when $p = 0.1$.

6.2 Performance of CPU Utilization

Figure 4 (b) shows the CPU utilization ratio as the number
of SFC request arrived at the substrate network. We evaluated the CPU utilization on three different substrate network topologies, whose edge connectivity probability $p$ were 0.1, 0.2, and 0.5. At the low network workload, the trend of CPU utilization rate is gradually increased since the resource of network is sufficient for SFC requests. As more SFC requests arriving, the workload grows high and finally converged at a high value of about 80%.

Figure 4(c) shows the end to end latency of SFC on three substrate network topologies, whose edge connectivity probability $p$ are 0.1, 0.2, and 0.5. We vary the number of VNFs in SFC and statistic the average end to end latency of SFCs, which have been successfully deployed on the substrate network. The end to end latency increases as the number of VNFs in SFC increasing since a longer SFC will need a longer path in the substrate network and results in a longer end to end latency. From the view of substrate network, the end to end latency of SFC gets larger when the edge connectivity probability is small, such as $p = 0.1$. This is because a lower network edge connectivity probability results in a smaller number of edges in the substrate network, and traffic from one VNF needs travel a longer path to the next VNF which results in a longer end to end latency.

7. Conclusion and Future Work

In this paper, we separated the cost of deploying VNFs of SFC into setup cost and operation cost, and formulated the SFC placement as an ILP model with the objective of minimizing overall costs by taking CPU and bandwidth resources as constraints, as well as satisfying the latency requirements and order of VNFS in SFC. We also proposed a delay-aware dynamic programming based SFC placement scheme for large network. The experiment study demonstrated that different optimization objectives can affect the network cost, especially the setup cost. We also found that the setup cost and operation cost increased when keeping the order of VNFS in SFC. The numeric simulation on the proposed scheme showed that our scheme outperformed in different scenarios.

In future work, we extend our scheme with the dynamic resource adjustment for SFC requests according to different service level agreement (SLA) on the means of real-time resource utilization prediction. For ultra-large-scale substrate network, we consider to divide the network into small clusters to obtain the optimal solution. We conduct the experiments on different topologies and more practical environments in addition to numeric simulations.

References

[1] Recommendation ITU-R M.2083-0. IMT Vision—Framework and Overall Objectives of the Future Development of IMT for 2020 and Beyond, Int’l Telecommunication Union-Radiocommunication Sector, 2015.

[2] Network functions virtualization: an introduction, benefits, enablers, challenges & call for action, ETSI, Oct. 2012.

[3] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, “Making middleboxes someone else’s problem: network processing as a cloud service,” ACM SIGCOMM Computer Communication Review, vol.42, no.4, pp.13–24, 2012.

[4] M.C. Luizelli, L.R. Bays, L.S. Buriol, M.P. Barcellos, and L.P. Gaspar, “Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions,” Proc. IFIP/IEEE Int. Symp. Integ. Netw. Manage. (IM), pp.98–106, May 2015.

[5] P. Quinn and T. Nadeau, “Problem statement for service function chaining,” IETF RFC 7498, 2015.

[6] Network functions virtualisation: white paper on NFV priorities for 5G, ETSI, Feb. 2017.

[7] “Network functions virtualisation (NFV): Use cases,” ETSI, Sophia Antipolis, France, Tech. Rep., Oct. 2013. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/nfv/001_099/001/01.01.00/gs

[8] W. Xu, Y. Jiang, and C. Zhou, “Problem Statement of Network Functions Virtualization Model,” Internet-Draft, IETF Internet-Draft draft-xjz-nfv-model-problem-statement-00, Sept. 2013.

[9] B. Yi, X. Wang, K. Li, S.K. Das, and M. Huang, “A comprehensive survey of Network Function Virtualization,” Computer Networks, vol.133, pp.212–262, 14 March 2018.

[10] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Boulen, F. De Turck, and R. Boutaba, “Network function virtualization: State-of-the-art and research challenges,” IEEE Commun. Surveys Tuts., vol.18, no.1, pp.236–262, 1st Quart., 2016.

[11] M. Yu, Y. Yi, J. Rexford, and M. Chiang, “Rethinking virtual network embedding: substrate support for path splitting and migration,” ACM SIGCOMM Computer Communication Review, vol.38, no.2, pp.17–29, 2008.

[12] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, “Virtual network embedding through topology-aware node ranking,” ACM SIGCOMM Computer Communication Review, vol.41, no.2, pp.38–47, 2011.

[13] N.M.M.K. 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.

[14] L. Gong, Y. Wen, Z. Zhu, and T. Lee, “Toward profit-seeking
virtual network embedding algorithm via global resource capacity,” IEEE International Conference on Computer Communications (INFOCOM), pp.1–9, April 2014.

[15] A. Dwaraki and T. Wolf, “Adaptive service-chain Routing for virtual network functions in software-defined networks,” Workshop on Hot topics in Middleboxes and Network Function Virtualization (HotMiddlebox), pp.32–37, 2016.

[16] N. Huin, B. Jaumard, and F. Giroire, “Optimization of network service chain provisioning,” IEEE International Conference on Communications (ICC), pp.1–7, May 2017.

[17] Y. Xu and V. P. Kafle, “Reliable service function chain provisioning in software-defined networking,” International Conference on Network and Service Management (CNSM), pp.1–4, Nov. 2017.

[18] S. Choi, J. Turner, and T. Wolf, “Configuring sessions in programmable networks,” Computer Networks, vol.41, no.2, pp.269–284, 2003.

[19] A. Tomassilli, F. Giroire, N. Huin, and S. Perennes, “Provably Efficient Algorithms for Placement of Service Function Chains with Ordering Constraints,” IEEE Proc. INFOCOM, 2018.

[20] R. Cohen, L. Lewin-Eytan, J.S. Naor, and D. Raz, “Near optimal placement of virtual network functions,” IEEE International Conference on Computer Communications (INFOCOM), pp.1346–1354, April 2015.

[21] J. Liu, W. Lu, F. Zhou, P. Lu, and Z. Zhu, “On Dynamic service function chain deployment and readjustment,” IEEE Trans. Netw. Service Manag., vol.14, no.3, pp.543–553, Sept. 2017.

[22] P. Erdős and A. Rényi, “On random graphs,” Publ. Math., 6, 290, 1959.

[23] E.N. Gilbert, “Random graphs,” Ann. Math. Stat., vol.30, no.4, pp.1141–1144, 1959.

[24] S. Mehraghdam, M. Keller, and H. Karl, “Specifying and placing chains of virtual network functions,” IEEE 3rd International Conference on Cloud Networking (CloudNet), pp.7–13, Oct. 2014.

[25] S. Orłowski, R. Wessły, M. Pióro, and A. Tomaszewski, “Sndlib 1.0—survivable network design library,” Networks, vol.55, no.3, pp.276–286, 2010.