Computer Networks 133 (2018) 166–182

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

MOSC: a method to assign the outsourcing of service function chain across multiple clouds☆

Huan Chen, Xiong Wang, Yangming Zhao, Tongyu Song, Yang Wang, Shizhong Xu*, Lemin Li

Key Laboratory of Optical Fiber Sensing and Communication, Education Ministry of China, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China

ARTICLE INFO

Article history:
Received 3 May 2017
Revised 15 October 2017
Accepted 17 January 2018
Available online 3 February 2018

Keywords:
Network Function Virtualization
Cloud computing
Service function chain
Hidden Markov Model

ABSTRACT

As Network Function Virtualization (NFV) becomes reality and cloud computing offers a scalable pay-as-you-go charging model, more network operators would like to outsource their Service Function Chains (SFC) to the public clouds in order to reduce the operational cost. Unfortunately, challenges of Quality of Service guarantee still exist while minimizing the operational cost with outsourcing SFC to public clouds. In this paper, we investigate this problem when there are a large number of candidate cloud providers with the consideration of diverse pricing schemes of network functions, additional latency caused by public network, and the relationship between the Virtual Network Function (VNF) performance and its cost. Compared with our previous conference version, we design D-MOSC, an improved deviation based heuristic algorithm to assign the Outsourcing of SFC across multiple clouds based on Hidden Markov Model (HMM). The extensive simulations show that MOSC saves up to 79.2% cost compared with that of deploying network functions in the local network. MOSC also achieves up to 50.7% cost savings compared with the result of the first-fit based optimization algorithm. Compared with the greedy version, D-MOSC achieves up to 26.7% cost savings with the guarantee of latency requirements.

1. Introduction

In today's operator networks, the communication traffic should go through a series of middleboxes, such as Network Address Translation (NAT), Firewall (FW), and Intrusion Detection/Prevention System (IDS/IPS). Usually, these middleboxes should be traversed in a specific order. For example, for a secure server whose accesses need to be highly restricted, its incoming traffic may first traverse an FW and then an IDS/IPS. Such ordered middleboxes is referred as Service Function Chains (SFC) [1].

Traditionally, the SFCs is implemented by steering flows to the hardware middleboxes located in local networks that incur high deployment and maintenance cost [2]. To solve this problem, Network Function Virtualization (NFV) implements middleboxes in software running on VMs [3–5], and some technologies are proposed to face the dynamic requirements of services [6,7]. In addition, several architectures (e.g., APLOMB [2], Jingling [8]) are proposed to migrate VNFs from local networks to clouds. As public clouds usually offer a pay-as-you-go charging model and elastic service delivery, the operational cost and complexity of maintenance can be reduced. More and more operators would like to outsource the SFC to the public cloud by renting computation resources.

When the network operators make the cost-efficient outsourcing plans, there are several issues that should be addressed. First of all, there are a large number of cloud providers (e.g., Amazon, Azure, Linode, Aliyun) which have diverse pricing schemes (as shown in Table 1 [9]) and different technical specifications (e.g., cloud latency). Moreover, redirecting flows to the clouds may introduce extra delays, thus how to guarantee the Quality of Service (QoS) of flows should be considered. At last, some network functions cannot be outsourced due to the security or privacy reasons (e.g., sensitive data storage).

More broadly, RFC7498 [1] proposed the topological dependencies of Service Function Chaining. The Network Functions need to be deployed on the network path or be traversed by corresponding

☆ The preliminary version of this paper titled "Towards Optimal Outsourcing of Service Function Chain Across Multiple Clouds" was published in the proceedings of IEEE ICC 2016. In this longer version, we present the following extended work. (1) We present more background on Service Function Chaining and how the SFC-enabled domains work. (2) We discuss the motivation example with extensive simulations to show the impacts with a larger dataset. (3) We propose an improved deviation based algorithm to find tradeoff between cost and latency, and make the algorithm in the conference version as the baseline. (4) We present the evaluation of our improved algorithm with that of the greedy based algorithm in the conference version. (5) We provide more concentrate discussion on modeling our problem as a Hidden Markov Model, and discuss some new related works.

* Corresponding author.
E-mail addresses: wangxiong@uestc.edu.cn (X. Wang), xsz@uestc.edu.cn (S. Xu).

https://doi.org/10.1016/j.comnet.2018.01.020
1389-1286/

Table 1
Pricing schemes of different cloud storage providers [9].

	S3 USA, EU ($)	Rackspace ($)	Nirvanix ($)
Data transfer in (GB)	0.10	0.08	0.18
Data transfer out (GB)	0.15	0.22	0.18
Storage (GB/month)	0.15	0.15	0.25

flows with traffic steering. Therefore, determining the outsourcing plan of Service Functions Chain is not an easy task, which sometimes forces the network administrators to create complicated routing schemes or provide an alternate virtual topology to satisfy the requirements [1], these mechanisms introduce extra network delays. Moreover, the performance of VNFs on different cloud environment is different. For example, DPI may only have 1 Gbps throughput with legacy NICs, but can be 10 times faster while using DPDK (Data Plane Development Kit) accelerated version of VNF [10,11]. A cloud server equipped with SATA-3 SSD (Solid State Drive) may have extreme high speeds close to 3.2 Gbps for data operations, but the price per gigabyte is 10 times greater than legacy drives [12]. Furthermore, other metrics of the cloud performance (e.g., bandwidth, CPU share) is also tightly related to the price [13]. Despite the performance, the pricing also matters the data reliability, using full-replication roughly doubles the hosting costs [9].

In view of this, we are to investigate how to migrate the SFCs to the public clouds in order to minimize the cost. Meanwhile, we should also take the QoS into consideration, i.e. the latency requirement of each SFC should be satisfied. In order to solve the problem, we first formulate it as an Integer Linear Programming (ILP) model. As this problem is *NP-hard*, the ILP model is intractable in large size networks. Accordingly, we propose a heuristic to schedule the outsourcing of SFC across multiple clouds with QoS guarantee (MOSC). The key idea of MOSC is to model the cloud selection as hidden stats transition in Hidden Markov Model (HMM), and then leverage Viterbi algorithm [14] to derive the SFC outsourcing plan by predicting the most-likely state sequences.

The technical contributions of this paper are summarized as follows:

- We formulate the service function chain outsourcing problem as an ILP model and propose deviation algorithm based MOSC, an efficient heuristic algorithm to solve this problem.
- We compare the results of MOSC with that of ILP model in small-scale networks. The results show that MOSC achieves near-optimal results in small-scale networks.
- We conduct extensive simulations to evaluate the performance of MOSC in large-scale networks. The results show that 79.7% cost is saved compared with that of deploying VNFs in the local network. Moreover, MOSC achieves up to 50.7% cost savings compared with that of the first-fit based optimization algorithm.
- Comparing with the conference version [15], we conduct extensive simulations to evaluate the performance improvements of Deviation algorithm based MOSC (D-MOSC) to that of the previous Greedy algorithm based MOSC (G-MOSC).

This paper is organized as follows. Firstly we discuss the outsourcing problem by a motivation example in Section 2. After that, the service function chain outsourcing problem is formulated as an ILP model in Section 3. In Section 4, we design an efficient heuristic algorithm for the problem, followed by simulation results in Section 5. The related works are presented in Section 6. At last, the conclusion is given in Section 7.

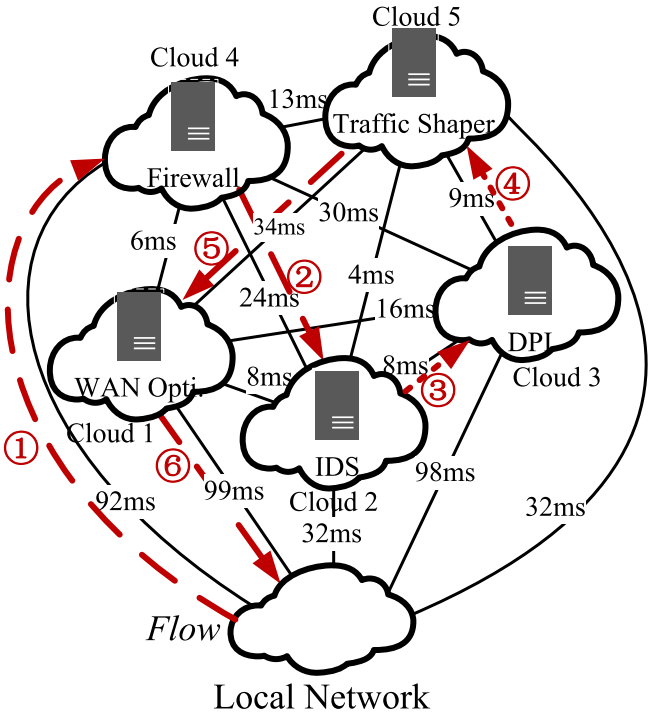


Fig. 1. Motivation example (cost efficient deployment).

Table 2
Pricing schemes of virtual network functions for flow of unit size.

	Firewall ($)	IDS ($)	DPI ($)	Traffic shaper ($)	WAN opti. ($)
Cloud 1	6	8	12	11	3
Cloud 2	13	5	28	19	24
Cloud 3	5	9	8	12	18
Cloud 4	2	9	16	10	13
Cloud 5	8	6	31	5	6
Local Network	13	18	32	22	28

2. Motivation

2.1. Motivation example

There are several works focusing on the VNF placement problem [16–20]. In this section, we explain why existing solutions are not suitable for the service function chain outsourcing problem through an example shown in Fig. 1.

We suppose that there is one flow of unit size in the local network, and the SFC of the flow is {Firewall → Intrusion Detection System (IDS) → DPI → Traffic Shaper → Wide Area Network (WAN) Optimizer}. Both the local network and public clouds can provide the five types of VNFs. The pricing schemes of the five VNFs in public clouds, as well as that in the local network, are given in Table 2.

RFC7665 [21] defined the components of core SFC architecture. The components include Service Function Forwarder (SFF), Service Functions, proxies, etc. All of these components compose a specific SFC-enabled domain. And different domains are interconnected with SFC encapsulation. The high-level architecture of these components is shown in Fig. 2.

In each SFC-enabled domain, the Service Function Forwarder (SFF) is responsible for injecting the SFC encapsulated traffic into public network and receiving the traffic. However, the encapsulation is not used for routing or path selection, it relies on the outer network transport [21]. The underlying devices (e.g., router, switch)

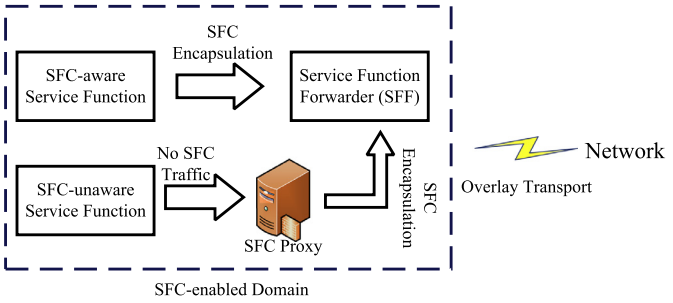


Fig. 2. The components of core SFC architecture.

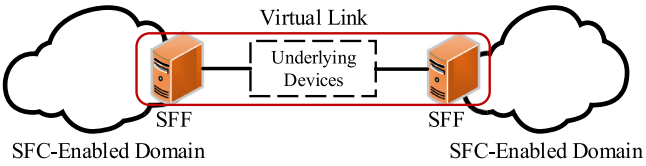


Fig. 3. The network overlay transport between SFC-enabled domains.

Table 3
Delay of links.

	Cloud 1	Cloud 2	Cloud 3	Cloud 4	Cloud 5	Local
Cloud 1	0	8 ms	16 ms	6 ms	34 ms	99 ms
Cloud 2	8 ms	0	8 ms	24 ms	4 ms	32 ms
Cloud 3	16 ms	8 ms	0	30 ms	9 ms	98 ms
Cloud 4	6 ms	24 ms	30 ms	0	13 ms	92 ms
Cloud 5	34 ms	4 ms	9 ms	13 ms	0	32 ms
Local	99 ms	32 ms	98 ms	92 ms	32 ms	0

carry the SFC encapsulated traffic and do not consult the SFC encapsulation and inner payload [21].

Obviously, as shown in Fig. 3, we can regard that the SFC-enabled domains are directly connected to each other by virtual links (consists of underlying devices). The cost to set up a virtual link (referred as *link* in this paper) is *not* negligible. For simplicity, we assume the cost to set up one virtual link for a flow of unit size is $2.

Traditionally, network operators implement the SFC by deploying VNFs in local networks in which no virtual link is needed, so the overall cost is $113. This scheme is referred as local placement scheme in this paper.

Another implementation of SFC is outsourcing VNFs to the public clouds. We first present the optimal placement given by NFV Location-LP [16] as shown in Fig. 1(a). The reason why we choose NFV Location-LP as the placement scheme is that it can achieve the minimization of the overall cost including the costs of VNFs and the connection costs of links when there are multiple candidate locations of VNFs. In our example, the optimal result of NFV Location-LP is shown in Fig. 1(a) in which six virtual links are used. The optimal cost is $35 and 69% cost is saved compared with $113 of local placement scheme.

Since NFV Location-LP, as well as other previous works focusing on the VNF placement problem, does not consider the extra delay incurred by outsourcing VNFs to public clouds, the path of the flow may include high-delay links (Table 3 shows the delay of links). As shown in Fig. 1(a), the total latency is 266 ms which cannot satisfy the QoS requirements of typical applications (e.g., VoIP, real-time video streaming, online games).

More broadly, to better identify the problem, we conduct an extensive simulation to evaluate the proportion of the flows whose latency requirements are not satisfied (referred as *failure rate*) when we adopt NFV Location-LP as the mechanism to solve the service function chain outsourcing problem.

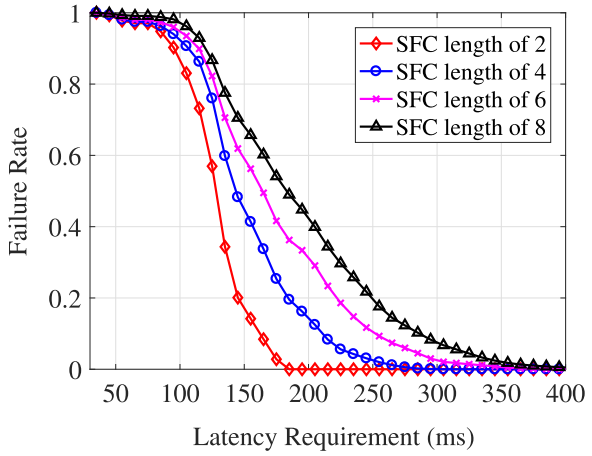


Fig. 4. Failure rate vs. latency requirement.

2.1.1. Simulation and models

To identify how the traditional placement scheme works, we need to determine the price schemes of VNFs in different nodes, the costs of links, and the network latency.

The costs of links are generated randomly in the range of 10–100, and the delays of links are set to random values with an average of 40 ms. In this simulation, we assume that there are 20 different cloud providers in our simulations, and 1024 flows are injected into the network with random traffic rates in the range of 10–100 Mbps.

For simplicity, considering each VNF i in the set of VNFs U, the processing cost of VNF i per 1 Mbps in the local network is regarded as the base price P_{ib}. P_{ib} of different VNF i is set to a random value in the range of 10–100.

The average price of VNF i per 1 Mbps in clouds, denoted as $\overline{P_{ic}}$, is a fraction of P_{ib}. That is $\overline{P_{ic}} = \alpha P_{ib}$, where *the coefficient* α *is a value in range of 0–1*. After that, the price of VNF i in cloud j per 1 Mbps, denoted as P_{ij}, is set to a random value in range of $0.8\overline{P_{ic}}$–$1.2\overline{P_{ic}}$. In this motivation simulation, α is set to 0.2, more results with different values of α will be discussed in Section 5.

The SFC is a sequence of VNFs with a length of m, which is generated by randomly picking m VNFs from U. Fig. 4 shows the results with different requirements of latencies.

2.1.2. Implications

From Fig. 4, it can be found out that when the length of SFC is 4, the latency requirements of all flows can be satisfied only if they are larger than 300 ms. As mentioned in [22], the latency requirement of flows of typical applications varies from 10 ms to 100 ms. In this case, even if the latency requirement is 100 ms, the failure rate is larger than 80%. That means if we adopt the NFV Location-LP to optimize the overall cost, the QoS requirements of most flows cannot be satisfied.

2.2. Impacts of VNF processing time

Despite the link latency, as we talked in Section 1, the processing time of the VNFs on clouds also acts as an important role. Moreover, the processing times of VNFs on different clouds are tightly related to the cost that the tenants are willing to pay, higher cost leads to higher technical specification of virtual machines which allows lower processing time.

To evaluate the impact of the processing time of VNFs on the results of our previous algorithms, we conduct another simulation. In this simulation, we use the same simulation setup. We add

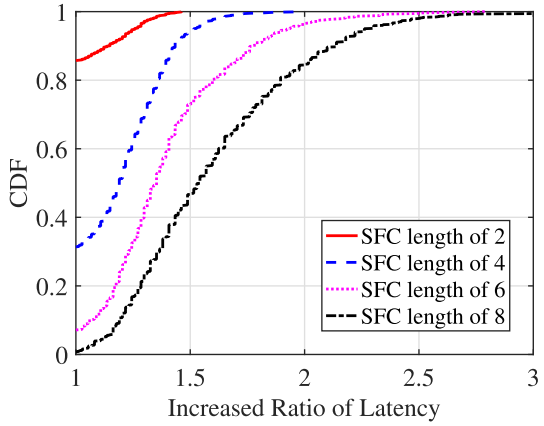


Fig. 5. Increasedlatency per flow vs. the length of SFC.

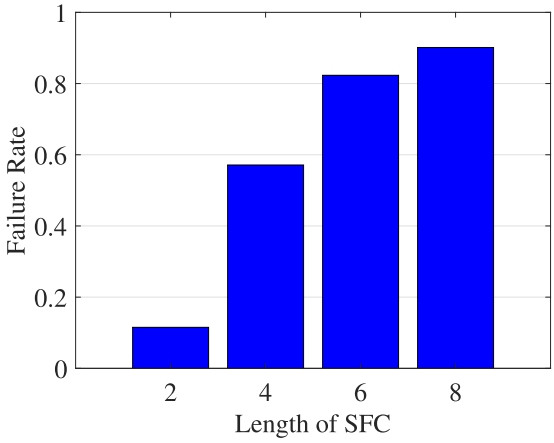


Fig. 6. Latency violation with VNF processing time.

the VNF processing time to the result of the previous placement scheme.

2.2.1. Model of processing time

We regard the processing latency of VNF i per 1 Mbps on node j which has the lowest price of deploying this function as the maximum processing latency T_{ij}. To ensure that the processing latency of VNF i associates with its price inversely, the average processing latency of VNF i per 1 Mbps on any other node k, denoted as $\overline{T_{ik}}$, is calculated as

$$\overline{T_{ik}} = \frac{P_{ij}}{\overline{P_{ik}}} \times T_{ij}$$

Recall, P_{ij} is the price of VNF i on node j. At last, the processing latency of VNF i per 1 Mbps on node k is set to a random value in range of $0.8\overline{T_{ik}}$–$1.2\overline{T_{ik}}$

2.2.2. Increased latency

Fig. 5 shows the cumulative distribution of increased latencies. From the result, we can see that the increased latency is proportional to the length of SFC. With the SFC length of 8, there are around 60% flows whose completion times increase by more than 1.5x! Obviously, the additional latency will grow much more significant while continuously increasing of SFC length.

2.2.3. Latency violation

Fig. 6 shows the failure rate of with different length of service function chain. From the result, we can figure out that the processing latency will cause the latency violation at a high level.

2.3. Latency optimization deployment

In above motivation example and extensive simulations, we argue that the service function chain outsourcing problem cannot be solved with the solution of traditional VNF placement problem (e.g., NFV Location-LP), since the placement schemes obtained by these solutions cannot meet the latency requirement in most of the time. However, there does exist solutions that aim at minimizing the network latency during the network function placement or take the latency factors into consideration in their optimization models [23–25]. In this subsection, we present the optimal placement of the same motivation example with the consideration of link latency in Fig. 7. For simplicity, we neglect the VNF processing latency here.

Firstly, while outsourcing VNFs to the clouds, we can achieve a latency optimal placement scheme by choosing Cloud 2 which has the minimal link delay to the local network, the placement is shown in Fig. 7(a). In this solution, the network latency is hugely decreased, compared with 266 ms of the cost-efficient placement shown in Fig. 1, the latency is decreased to 64 ms, which is 75.9% lower.

Although the latency is minimized with this solution, the overall cost to deploy all the VNFs in Clouds 2 is $91. Compared with the local placement scheme mentioned in Section 2.1, it only saves 19% cost. Meanwhile, compared with the cost-efficient placement, the tradeoff to minimize the network latency is over 60% extra cost!

In the motivation example, the second latency optimal placement scheme is shown in Fig. 7(b). By placing the VNFs in two low-latency clouds instead of only Cloud 2, the placement scheme achieves lower cost by $88, and incurs extra 4 ms delays. The cost is still 57% higher than that of the cost-efficient placement.

2.4. Discussion

From the above examples and simulations, we clarified that the deployment cost of VNFs and the network latency are the two key factors of the service function chain outsourcing problem. Also, from previous works, it is pointed out that whether the providers can guarantee the deadline of delays is the major concern of the users [26–28]. In view of this, we argue that the service function chain outsourcing problem should be addressed by finding a tradeoff of the overall outsourcing cost to satisfy all the deadline requirement rather than minimizing the latency for each user.

In order to achieve this, let's review the motivation example in Fig. 1. We suppose the latency requirement of the flow is 100 ms. To satisfy it, another feasible placement scheme is shown in Fig. 8. We deploy the Firewall in the local network rather than Cloud 4. Comparing with the result of NFV Location-LP, an extra cost of $9 is spent, while the delay decreases significantly, i.e., 81 ms. In addition, we deploy the WAN Optimizer in Cloud 5 rather than Cloud 1, which incurs extra $1 while saving 38 ms. In this scheme, the total latency of the path is reduced to 81 ms, and the cost is only $37 which saves 67.2% compared with that of local placement scheme.

Compared with the cost of the low latency deployment in Fig. 7, the cost is saved by 59%. Meanwhile, although the latency is 81 ms which is larger than that of the low latency deployment scheme, it still meets the latency requirement.

Takeaway: From the above examples, we argue that the solution of VNF placement problem (e.g., NFV Location-LP) is *not* suitable for the service function chain outsourcing problem. Also, we cannot simply minimize the network latency without consideration of the cost. This problem should be solved by jointly optimization with consideration of *pricing schemes and technical specifications of multiple cloud providers, requirement of SFC, performance of different VNFs, and QoS requirements of flows.*

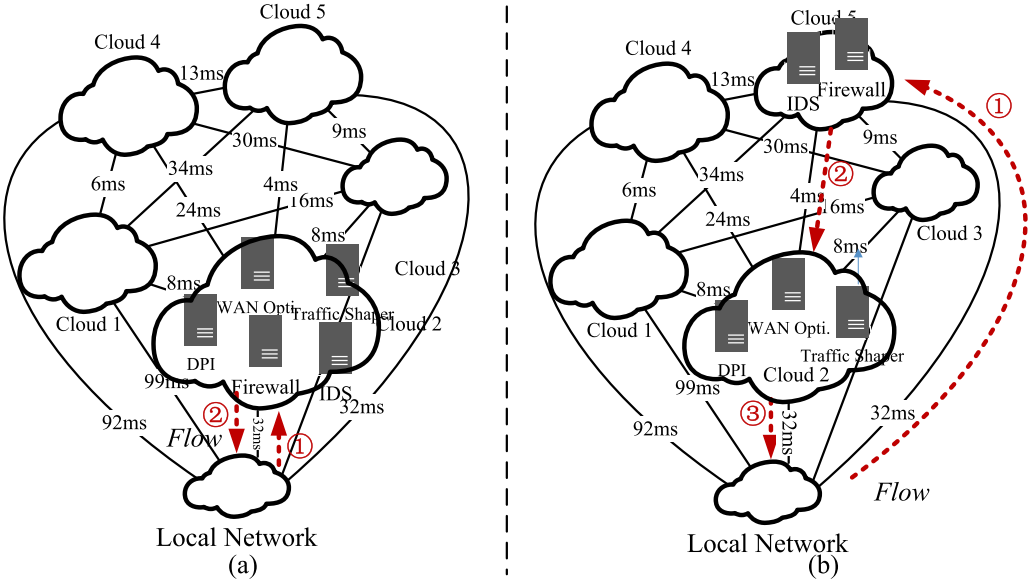


Fig. 7. Motivation example (low latency deployment).

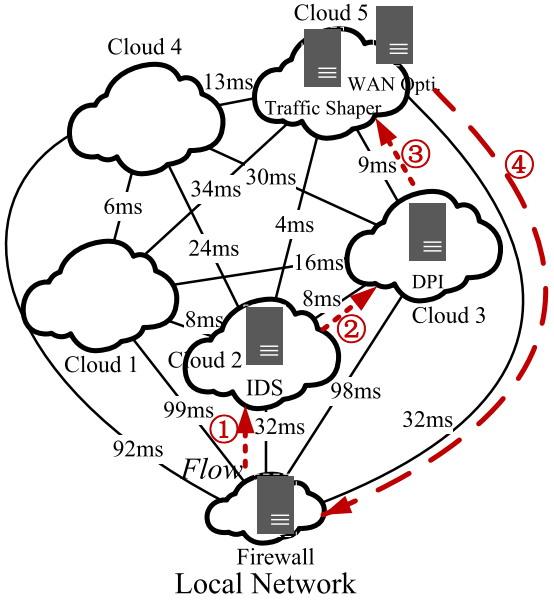


Fig. 8. Motivation example (joint optimization deployment).

3. Problem formulation

In this section, we formulate the service function chain outsourcing problem as an Integer Linear Programming (ILP) model. All the notations used in this section are summarized in Table 4.

3.1. Network model

The topology of the network is abstracted as an undirected graph $G = (N, E)$. Suppose there are K cloud providers, R_i, $1 \le i \le K$. Let $N = \{L\} \cup \{R_i\}_{i=1}^{K}$, L denotes the local network. E denotes the set of edges, edge (i, j) indicates that node i and j can directly communicate with each other. As discussed in Section 2, since the underlying devices (e.g., router, switch) carry the SFC encapsulated traffic and do not consult the SFC encapsulation and inner payload, we only discuss the virtual links between the SFC-enabled domains in this paper, and the underlying routing will only impact the delay of virtual links. In view of this, the network topology can be regarded as full-meshed with virtual links. Each VNF can be deployed locally or be outsourced to public clouds.

3.2. ILP formulation

The objective of the model is minimizing the overall cost including costs of VNFs on nodes and connection costs of links.

At first, the flow conservation constraints should be satisfied. Since we only discuss intra-traffic in the local network, both the source and destination of one flow is the local network when the SFC is outsourced to public clouds. To ensure this, we add two dummy VNF D_0 and D_1 to the SFC of each flow, and the SFC should start with D_0 and end with D_1. Moreover, D_0 and D_1 must be located in the local network. Accordingly, the constraints are described as

$$\sum_{j\in N, l\in V} n_t^{imjl} - \sum_{j\in V, l\in V} n_t^{jlim} = \begin{cases} 1 & \text{if} m = D_0 \\ -1 & \text{if} m = D_1 \\ 0 & \text{otherwise} \end{cases} \quad \forall f_t \in F, \forall i \in N, \forall m \in V \tag{1}$$

In this paper, we limit the implied order of SFC is a linear progression, described by v_t^{ml}. As mentioned in Section 1, the flow must traverse the VNFs in a required order. Thus, n_t^{imjl} should be restricted as

$$\sum_{i\in N, j\in N} n_t^{imjl} = v_t^{ml} \qquad \forall f_t \in F, \forall m, l \in V \tag{2}$$

Obviously, function m should be deployed on node i if any flow goes through function m on node i, i.e.,

$$z_t^{im} \ge \frac{n_t^{imjl} + n_t^{jlim}}{2} \qquad \forall f_t \in F, \forall i, j \in N, \forall m, l \in V \tag{3}$$

In this paper, we assume that in the service function chain of a flow, a specific function m appears only once, i.e.,

$$\sum_{i\in N} z_t^{im} = 1 \qquad \forall f_t \in F, \forall m \in V \tag{4}$$

Note that different VNFs might be deployed on the same node, to ensure the order of SFC, there has the possibility that loop exists in the path of SFC.

Table 4
Notations used in this section.

Notation	Description
f_t	The traffic rate in Mbps of the tth flow. For convenience, it also denotes the flow itself.
v_t^{ml}	Binary parameter indicating whether function l is the successor of function m in the SFC of flow f_t
M_k	The resource requirements of function k per 1 Mbps
C_i	The resource capacity of node i
B_{ij}	The bandwidth capacity of link (i, j)
d_{ij}^m	The delay from VNF m on node i to any VNF on node j, including the delay of link (i, j) and the processing time of VNF m on node i.
l_t	The latency requirement of flow f_t
λ_{im}	The processing cost of network function m on node i per 1 Mbps
θ_{ij}	The bandwidth cost of link (i, j) per 1 Mbps
n_t^{imjl}	A binary variable indicating whether f_t traverses from function m on node i to function l on node j
z_t^{im}	A binary variable indicating whether f_t goes through function m on node i
N	The set of nodes in network
E	The set of links in network
F	The set of flows
V	The set of VNFs

In addition, the capacity constraints on each node and each link are given in (5) and (6) respectively. It should be pointed out that the capacities of nodes which denote the public clouds are usually very large. Meanwhile, the virtual links are usually established with the on-demand provision, thus the capacities of virtual links are also large in practice.

$$\sum_{m\in V}\sum_{f_t\in F} f_t z_t^{im} M_m \le C_i \quad \forall i \in N \tag{5}$$

$$\sum_{f_t\in F}\sum_{m,l\in V} f_t n_t^{imjl} \le B_{ij} \quad \forall (i,j) \in E \tag{6}$$

After that, the latency constraint is formulated as follow,

$$\sum_{(i,j)\in E}\sum_{m,l\in V} n_t^{imjl} d_{ij}^m \le l_t \quad \forall f_t \in F \tag{7}$$

It should be pointed out that the processing time of the dummy VNF D_0 and VNF D_1 is 0.

Accordingly, we get the ILP model to describe the *service function chain outsourcing problem*:

$$\text{minimize} \sum_{i\in N, m\in V}\sum_{f_t\in F} f_t z_t^{im}\lambda_{im} + \sum_{(i,j)\in E}\sum_{f_t\in F}\sum_{m,l\in V} f_t n_t^{imjl}\theta_{ij}$$

subject to: (1)–(7)

3.3. Discussion

In the formulation, we assume that the structure of SFC is linear chain, this is

This problem can be reduced to *NFV location problem* [16] by relaxing the constraint of the order of VNFs and the constraint of flow latency. The NFV location problem has been proved as an NP-hard problem [16]. Thus, *the service function chain outsourcing problem is an NP-hard problem*. In view of this, a heuristic algorithm is proposed in Section 4.

4. Heuristic algorithm design

In this section, we propose the method of outsourcing SFC to multiple clouds with QoS guarantee (MOSC) to solve the problem formulated in the last section. The key spirit of MOSC is to sequentially determine the SFC of each flow. For a specific flow, we leverage the Hidden Markov Model (HMM) to formulate the cloud selection procedure and concrete the most-likely sequence in the HMM to be the SFC instance of this flow. Since this SFC instance may not satisfy the QoS requirement, a deviation based algorithm is used to realize tradeoff between cost and QoS.

4.1. Hidden Markov Model

The first step of MOSC is to find a cost-efficient path for each flow. This procedure is inspired by finding the most-likely sequence in HMM [29]. In traditional Markov models, we can directly observe the state and input the state transition probabilities as the only parameter. In HMM, the states are invisible directly, but the sequence of hidden states can be predicted by some directly visible states (called *observed state*). In HMM, three probabilities are used to predict the most-likely sequence, the initial probabilities, the transition probabilities, and the emission probabilities. Initial probabilities and transition probabilities are similar to traditional Markov model. The emission probabilities are the probabilities of observed state under the specific hidden state. Next, we present the detail of modeling the SFC outsourcing problem as an HMM model.

In our HMM model, state $S_{v_i}^{n_j}$ denotes node n_j is chosen to serve VNF v_i, and the procedure of traversing through all VNFs in the SFC can be regarded as the transition of states as shown in Fig. 9. Since the transition of states cannot be observed directly, these states are *hidden states* in Hidden Markov Model (HMM) [29]. However, there are several factors can be directly observed (e.g., costs of VNFs, resource usage). These factors can be used to predict the most-likely transition sequence of hidden states which corresponds to a cost-efficient path.

4.2. Finding the cost-efficient path for each flow

The SFC of a flow is a sequence of VNF which can be denoted as $V = \{v_1, v_2 \dots v_n\}$. And the nodes in the network are described by $N = \{n_1, n_2 \dots n_k\}$. As mentioned above, the hidden state $S_{v_i}^{n_j}$ means that the flow goes through function v_i on node n_j.

There are several factors which can be directly observed and dependent on the hidden states, e.g., the costs of VNFs and links, the resource usage of nodes. These factors can be used to construct the three-tuple (Π, A, B) of HMM. Π is the set of initial probabilities, A is the transition probability matrix and B is the emission probability matrix. The transition of hidden states and the parameters of HMM is shown in Fig. 9.

$\Pi = \{\pi_1, \pi_2 \dots \pi_k\}$ is the set of initial probabilities. π_i is the probability of choosing node n_i to serve VNF v_1 (the first VNF in the SFC).

To determine π_i, we use the cost of VNF v_1 on node n_i (referred as $\lambda_{n_i v_1}$) and the cost of the link between local network (n_0) and node n_i (referred as $\theta_{n_0 n_i}$). $\theta_{n_0 n_i} = 0$ when node n_i is also the local

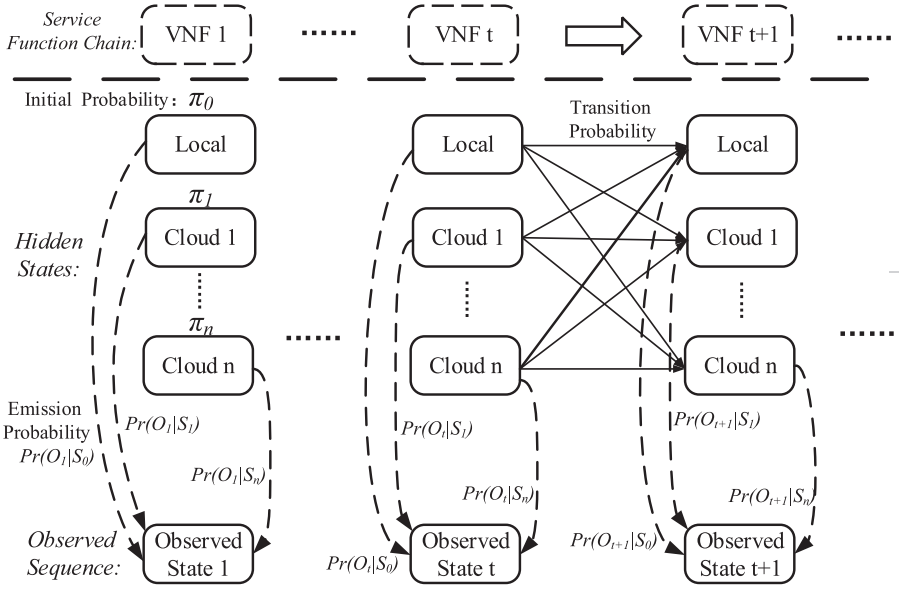


Fig. 9. State transition of Hidden Markov Model.

network. Thus, the initial probability π_i is given as

$$\pi_i = \frac{\sum_{n_i \in N}(\lambda_{n_i v_1} + \theta_{n_0 n_i}) - (\lambda_{n_i v_1} + \theta_{n_0 n_i})}{(|N|-1)\cdot(\sum_{n_i \in N}(\lambda_{n_i v_1} + \theta_{n_0 n_i}))} \tag{8}$$

A is the transition probability matrix. $Pr(S_{v_i}^{n_y}|S_{v_{i-1}}^{n_x})$ denotes the probability of transition from state $S_{v_{i-1}}^{n_x}$ to state $S_{v_i}^{n_y}$. In our case, the transition means that flow traverses through VNF v_{i-1} on node n_x to VNF v_i on node n_y.

$$A = \begin{pmatrix} Pr(s_{v_i}^{n_1}|s_{v_{i-1}}^{n_1}) & \dots & Pr(s_{v_i}^{n_1}|s_{v_{i-1}}^{n_k}) \\ \vdots & \ddots & \vdots \\ Pr(s_{v_i}^{n_k}|s_{v_{i-1}}^{n_1}) & \dots & Pr(s_{v_i}^{n_k}|s_{v_{i-1}}^{n_k}) \end{pmatrix}_{k\times k}$$

The cost to set up function VNF v_i on node n_y is the sum of the cost of VNF v_i on node n_y (referred as $\lambda_{n_y v_i}$) and cost of link (n_x, n_y) (referred as $\theta_{n_x n_y}$, n_x is the node used to serve v_{i-1}). Accordingly, the transition probability $Pr(s_{v_i}^{n_y}|s_{v_{i-1}}^{n_x})$ can be denoted as

$$Pr(S_{v_i}^{n_y}|S_{v_{i-1}}^{n_x}) = \frac{\sum_{n_y \in N}(\lambda_{n_y v_i} + \theta_{n_x n_y}) - (\lambda_{n_y v_i} + \theta_{n_x n_y})}{(|N|-1)\cdot(\sum_{n_y \in N}(\lambda_{n_y v_i} + \theta_{n_x n_y}))} \tag{9}$$

B is the emission probability matrix. Recall, emission probability $Pr(O_{v_i}|S_{v_i}^{n_x})$ is the probability of observed state O_{v_i} under the condition $S_{v_i}^{n_x}$. In our model, there are two observed states, one is node n_x is *suitable* for function v_i (referred as $O_{v_i}^{yes}$), the other is not suitable (referred as $O_{v_i}^{no}$).

$$B = \begin{pmatrix} Pr(O_{v_i}^{yes}|s_{v_i}^{n_1}) & \dots & Pr(O_{v_i}^{yes}|s_{v_i}^{n_k}) \\ Pr(O_{v_i}^{no}|s_{v_i}^{n_1}) & \dots & Pr(O_{v_i}^{no}|s_{v_i}^{n_k}) \end{pmatrix}_{2\times k}$$

One variable in our network is the resource usage of node n_x (referred as a non-zero positive value U_x). Thus, we define the emission probability in Eq. (10). With the definition, one node with more VNFs deployed is more likely to be chosen to place one new VNF. In this situation, it is expected that fewer links will be used, so the delay and cost can be reduced.

$$\begin{aligned} Pr(O_{v_i}^{yes}|s_{v_i}^{n_x}) &= U_x/(1+U_x) \\ Pr(O_{v_i}^{no}|s_{v_i}^{n_x}) &= 1 - Pr(O_{v_i}^{yes}|s_{v_i}^{n_x}) \end{aligned} \tag{10}$$

Moreover, if VNF v_i cannot be outsourced to public clouds due to security or privacy issues, the observed state of VNF v_i on the cloud nodes is *no* with the probability 1. In other words, $Pr(O_{v_i}^{yes}|s_{v_i}^{n_x}) = 0$ in this case.

After determining the three-tuple (Π, A, B) in HMM, the most-likely sequence of hidden states can be predicted by Algorithm 1 which is based on Viterbi algorithm [14].

Algorithm 1: Finding the Most-Likely State Sequence

Require: Topology $(\boldsymbol{N}, \boldsymbol{E})$, flow set $\boldsymbol{F}$, service chain $\boldsymbol{V}$ of each flow, observed sequence $\boldsymbol{O}$, hidden state set $\boldsymbol{S}$, three-tuple (Π, A, B) of HMM

Ensure: Most-likely hidden state sequence $\boldsymbol{R}$

1: Sort $\boldsymbol{F}$ in a descending order of traffic rate
2: **for** $f \in F$ **do**
3: $v_1 = V[0], \forall S_{v_1}^{n_k} \in S\ Pr(n_k|v_1) = P(O_{v_1}^{yes}|s_{v_1}^{n_k}) \cdot \pi_k$
4: **for** other $v_i \in V$ **do**
5: $\forall S_{v_i}^{n_k} \in S\ Pr(n_k|v_i) =$ $\max_{S_{v_i}^{n_k} \in S} Pr(O_{v_i}^{yes}|S_{v_i}^{n_k}) \cdot Pr(S_{v_i}^{n_k}|S_{v_{i-1}}^{n_x}) \cdot Pr(n_x|v_{i-1})$
6: $Ptr(n_k|v_i) \leftarrow S_{v_{i-1}}^{n_x}$ (x is the value used to get $Pr(n_k|v_i)$)
7: **end for**
8: For the last v_i, $R_f^i = argmax_{S_{v_i}^{n_k} \in S}(Pr(n_k|v_i))$
9: **for** other $v_i \in V$ **do**
10: $R_f^i = Ptr(n_k|v_{i+1})$
11: **end for**
12: **end for**
13: **return** R

From Line 3 to Line 7 of Algorithm 1, Viterbi algorithm gets the most-likely state sequence which produces the observed states. $Pr(n_i|v_k)$ is the probability of hidden state $S_{v_k}^{n_i}$ which is calculated is Line 5, and the most probable hidden state is the state with maximum $Pr(v_k|n_i)$. After that, Viterbi algorithm retrieves the *Viterbi path* by backtracking the hidden states with a reverse pointer Ptr (shown in Line 6). The *Viterbi path* corresponds to the path and the locations of network functions of one flow.

4.3. Deviation of the Viterbi path

In Section 4.2, the most-likely cost-efficient path of the SFC is obtained. However, although less number of links is expected in

HMM, it is still hard to determine the latency of the full path of one flow at each hidden state described in Section 4.2. Accordingly, if the path violates the latency requirement of the flow, we need to find a method to eliminate the latency violation by choosing some node with higher cost but lower latency to serve part of these VNFs. Inspired by the Yen's algorithm [30,31], we propose *Deviation based MOSC (D-MOSC)* which finds the deviation of the *Viterbi path* to trade-off the cost and latency to meet the requirement of flows.

4.3.1. Notations and definitions

Recall, the SFC of a flow is a sequence which can be denoted as $V = \{v_1, v_2 \dots v_n\}$. And the nodes in the network are described by $N = \{n_1, n_2 \dots n_k\}$. As mentioned above, the hidden state $s_{v_i}^{n_j}$ means that the flow goes through function v_i on node n_j. The potential path can be denoted as

$$p = \left\{s_{v_1}^{n_i}, s_{v_2}^{n_j} \dots s_{v_n}^{n_k}\right\} \qquad \{n_i, n_j, \dots, n_k\} \subseteq N \tag{11}$$

Hereafter, P_{ij} is the set of paths from $s_{v_i}^{n_a}$ to $s_{v_j}^{n_b}$ ($n_a, n_b \in N$), and c_p is the total cost of path p. As we only discuss intra-traffic in the local network, both the source and destination of one flow is the local network (n_0). In addition, we add two dummy functions d_0 and d_1 as the initial and terminate function respectively. Correspondingly, $S_{d_0}^{n_0}$ and $S_{d_1}^{n_0}$ denote the initial and terminate state, and P denotes the set $P_{d_0 d_1}$.

Given two states $s_{v_x}^{n_a}$ and $s_{v_y}^{n_b}$ of path $p \in P$, $q \in P_{xy}$ denotes a subpath in p which coincides with p from $s_{v_x}^{n_a}$ to $s_{v_y}^{n_b}$; such subpath is denoted as $sub_p(x, y)$. And given two paths, $p \in P_{d_0 i}$ and $q \in P_{i d_1}$, the concatenation of p and q is denoted as $p \diamond q$.

4.3.2. Tradeoff between cost and QoS

In Section 4.2, we model the service function chain outsourcing problem as an HMM and use the Viterbi algorithm to derive a most-likely cost-efficient path p_1 in P. However, the latency of p_1 may not satisfy the QoS requirement. When p_1 exceeds the latency threshold, we cannot use the most-likely cost-efficient path. Instead, we try other paths with larger cost in order to reduce the path latency.

Fig. 10 shows a simplified example with SFC length of 2 and latency requirement of 38 ms and only one public cloud. The Viterbi path (p_1) obtained by Algorithm 1 is described as red lines, with minimal cost of 10 but the latency of 60 ms. After that, we can find another 2 candidate paths which are shown in dash lines and dotted lines. These two paths have a higher cost of 12 and 15 respectively, but with lower latency. Next, we are going to elaborate our algorithm to realize the tradeoff between cost and QoS.

4.3.3. Deviation algorithm

The high-level idea of realizing tradeoff between cost and QoS is to construct a new path which deviates from the nodes of the original cost-efficient path. During the deviation procedure, the

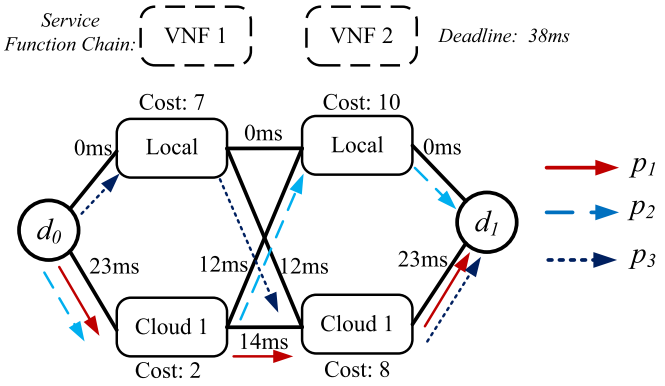


Fig. 10. Ranking paths.

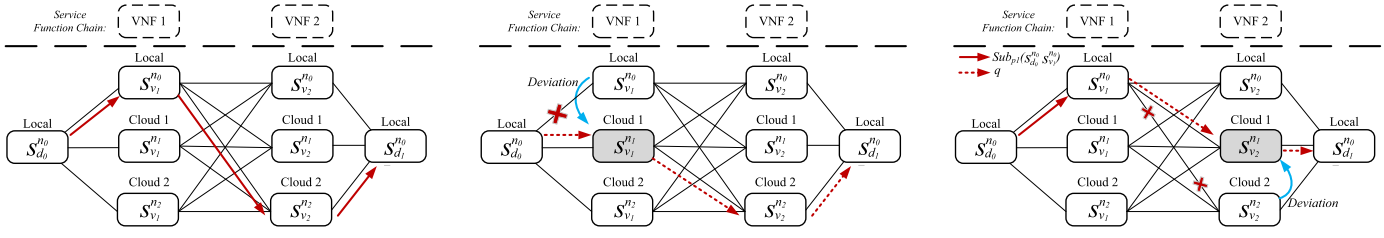
cost to realize the SFC increases, but the latency decreases. The deviation ends until we find a path which satisfies the latency requirement. The deviation path on the kth deviation is denoted by $p_k = \left\{s_{d_0}^{n_0}, s_{v_1}^{n_a} \dots s_{v_i}^{n_b}, s_{v_{i+1}}^{n_c}, \dots, s_{v_n}^{n_x}, s_{d_1}^{n_0}\right\}$ ($\{n_a, n_b \dots\} \subseteq N$).

p_1 is calculated by Algorithm 1 as the initial path, and the some paths will candidate for p_2. These candidate paths deviate p_1 from one of its nodes (i.e., the hidden state, we use *state* and *node* interchangeably) $s_{v_i}^{n_x}$, and for each node, one subpath $sub_{p1}(s_{v_i}^{n_x}, s_{d_1}^{n_0})$ needs to be calculated so that it coincides with p_1 by $sub_{p1}(s_{d_0}^{n_0}, s_{v_i}^{n_x})$ and $sub_{p1}(s_{v_i}^{n_x}, s_{d_1}^{n_0})$. $sub_{p1}(s_{t_i}^{n_x}, s_{d_1}^{n_0})$ cannot contain any node from $sub_{p1}(s_{d_0}^{n_0}, s_{v_i}^{n_x})$, except $s_{v_i}^{n_x}$ itself. Moreover, the first link in $sub_{p1}(s_{v_i}^{n_x}, s_{d_1}^{n_0})$ cannot be $(s_{v_i}^{n_x}, s_{v_{i+1}}^{n_y})$ in p_1, which means we need to set the transition probability $Pr(s_{v_{i+1}}^{n_y} | s_{v_i}^{n_x})$ to 0.

Next, we explain the algorithm with the network shown in Fig. 11. $p_1 = \left\{s_{d_0}^{n_0}, s_{v_1}^{n_0}, s_{v_2}^{n_2}, s_{d_1}^{n_0}\right\}$ is shown in red lines in Fig. 11(a).

After determining p_1, states $s_{d_0}^{n_0}$, $s_{v_1}^{n_0}$, and $s_{v_2}^{n_2}$ will be deviated. Starting with $s_{d_0}^{n_0}$ (as shown in Fig. 11(b)) which means we need to find a most-likely cost-efficient path from $s_{d_0}^{n_0}$ to $s_{d_1}^{n_0}$, and its first link is not $(s_{d_0}^{n_0}, s_{v_1}^{n_0})$. After removing the link $(s_{d_0}^{n_0}, s_{v_1}^{n_0})$ (by setting the transition probability to 0), the most-likely cost-efficient path obtained by Algorithm 1 is $\left\{s_{d_0}^{n_0}, s_{v_1}^{n_1}, s_{v_2}^{n_2}, s_{d_1}^{n_0}\right\}$ (The detail computation procedure is discussed in Section 4.2). This path is one of the candidate paths of p_2, we denote it as p_2^1.

Then we discuss state $s_{v_1}^{n_0}$ (Fig. 11(c)), we need to find the path with the form $sub_{p1}(s_{d_0}^{n_0}, s_{v_1}^{n_0}) \diamond q = (s_{d_0}^{n_0}, s_{v_1}^{n_0}) \diamond q$ where $q \in P_{s_{v_1}^{n_0}, s_{d_1}^{n_0}}$ and its first link is not $(s_{v_1}^{n_0}, s_{v_2}^{n_2})$. By removing this link (as shown in Fig. 11(c)), the obtained q is $\left\{s_{v_1}^{n_0}, s_{v_2}^{n_1}, s_{d_1}^{n_0}\right\}$. Therefore, the path $(s_{d_0}^{n_0}, s_{v_1}^{n_0}) \diamond q$ is also a candidate path of p_2, we denote it as p_2^2.



(a) p_1 (b) Deviation on $s_{d_0}^{n_0}$ (c) Deviation on $s_{v_1}^{n_0}$

Fig. 11. Determining p_2.

Note that there is no candidate paths generated when considering $s_{v_2}^{n_2}$ since there is only one link from $s_{v_2}^{n_2}$ to $s_{d_1}^{n_0}$. Therefore, all the candidates of p_2 is $\{p_2^1, p_2^2\}$. p_2 is the 2-nd most-likely path which is the one with minimum cost among the candidates.

Without loss of generality, assuming that the deviation paths, $p_1, \ldots, p_{k-1}$ are determined. The candidates of p_k should deviate from p_{k-1} starting from one of its nodes. Accordingly, a similar procedure to the one to determine p_2 will be used. Before each deviation, the initial links of the network should be restored, and when we find each deviation of the previous path, we check if it satisfies the requirement of the latency. If so, the deviation procedure will stop and the outsourcing plan is obtained. Since we assume the inner latency of local network is negligible, the algorithm will be convergence with the worst case that all VNFs are deployed in local networks.

As described above, the outline of the deviation algorithm is provided in Algorithm 2 .

Algorithm 2: Deviation of Path

Require: Topology $(\boldsymbol{N}, \boldsymbol{E})$, $p \leftarrow$ cost-efficient path from $s_{d_0}^{n_0}$ to $s_{d_1}^{n_0}$, $X = \{p\}$, $k \leftarrow 0$, $d(p) = s_{d_0}^{n_0}$, latency requirement t

Ensure: The deviation path that satisfy the latency requirement

1: **while** $X \neq \phi$ and $k < K$ **do**
2: $p_k \leftarrow \left\{s_{d_0}^{n_0}, s_{v_1}^{n_x} \ldots, s_{v_n}^{n_y}, s_{d_1}^{n_0}\right\}$, $p_k \in X$
3: $X \leftarrow X - \{p_k\}$
4: Remove link $(d(p_k), i)$, $i \in E$, of $p_1, \ldots, p_{k-1}$
5: **for** $s_{t_i}^{n_x} \in \left\{d(p_k), \ldots, s_{v_n}^{n_x}\right\}$ **do**
6: Remove link $(s_{v_i}^{n_x}, s_{v_{i+1}}^{n_y})$
7: $q \leftarrow$ Most-likely cost-efficient path from $s_{v_i}^{n_x}$ to $s_{d_1}^{n_0}$
8: $p \leftarrow sub_{p_k}(s_{d_0}^{n_0}, s_{v_i}^{n_x}) \diamond q$
9: $d(p) \leftarrow s_{v_i}^{n_x}$, $X \leftarrow X \cup \{p\}$
10: **end for**
11: Restore the deleted links
12: Sort X with ascending order of total cost
13: **for** $p \in X$ **do**
14: **if** get_latency$(p) < t$ **then**
15: **return** p
16: **end if**
17: **end for**
18: **end while**

4.4. Greedy algorithm to adjust the latency violated paths

In our earlier work in the conference version, we propose a greedy algorithm to adjust the Viterbi path obtained by Algorithm 1 when it exceeds the latency requirement. As a baseline to evaluate our algorithm, we also present the greedy based algorithm here to make *Greedy based MOSC (G-MOSC)*. Correspondingly, we call the algorithm proposed in Section 4.3 as *D-MOSC*.

Algorithm 3 adjusts it by replacing one or more nodes in the path until the latency requirement is satisfied. The key idea of Algorithm 3 is reducing the total latency of the path by migrating VNFs in the SFC to other nodes until the latency requirement is satisfied. In each step of the algorithm, the increase of cost is minimized.

In Line 6–11 of Algorithm 3, for each VNF in SFC, we migrate it to another node such that a new path p_{new} is obtained. Then we add p_{new} to the set of candidate paths $\boldsymbol{C}$. In Line 13–18, G-MOSC finds the path p_{new} with less latency, and p_{new} should be the path which increases the least cost. If the latency of all candidate paths is greater than that of the old path, we choose the one with minimum latency to continue the procedure.

Algorithm 3: Greedy Algorithm to Adjust Paths

Require: Topology $(\boldsymbol{N}, \boldsymbol{E})$, flow set $\boldsymbol{F}$, set of searched paths $\boldsymbol{P}$, set of candidate paths $\boldsymbol{C}$

Ensure: The set of path of all flows $\boldsymbol{R}$

1: **for** $f \in F$ **do**
2: $L_{max} \leftarrow$ the max latency requirement of f
3: $L \leftarrow$ the latency of path p of f
4: **while** $L > L_{max}$ **do**
5: $\boldsymbol{P} \leftarrow \boldsymbol{P} \cup p$
6: **for** each node $h \in p$ **do**
7: $h \leftarrow n, \forall n \in N$ & $n \neq h$. Get p_{new}
8: **if** $p_{new} \notin \boldsymbol{P}$ **then**
9: $\boldsymbol{C} \leftarrow \boldsymbol{C} \cup p_{new}$
10: **end if**
11: **end for**
12: Sort $\boldsymbol{C}$ in ascending order of cost increased
13: **for** $p_{new} \in C$ **do**
14: **if** the latency of p_{new} is less than L **then**
15: $p \leftarrow p_{new}$
16: break
17: **end if**
18: **end for**
19: **if** all the latency of $\boldsymbol{C}$ is greater than L **then**
20: $p \leftarrow$ the minimum latency path in $\boldsymbol{C}$
21: **end if**
22: $L \leftarrow$ latency of p
23: **end while**
24: $\boldsymbol{R} \leftarrow \boldsymbol{R} \cup p$
25: **end for**
26: **return** R

In Section 5, we will present the evaluation of the performance of MOSC in different scenarios. Furthermore, the comparison between D-MOSC and G-MOSC is also presented.

4.5. System deployment

Benefit from the flexible routing offered by Software-defined Networking (SDN), deploying MOSC in the real production environment is convenient. Fig. 12 shows the workflow of MOSC in typical SDN data centers.

The D-MOSC agent needs to be installed in the SDN controller. Firstly, the agent gathers information including latency, cost from multiple public cloud providers, and stores them in the local database. Then the agent will run D-MOSC algorithm to calculate the outsourcing scheme of VNFs and set up the VNF instance in corresponding nodes (local switches and public clouds).

While the new request comes into the data center, its SFC and latency requirements will be reported to the D-MOSC agent by the control channel between the switches and SDN controller. Then D-MOSC agent will determine the route of the incoming request, if some hops in the path locate in the public cloud, the agent will steer the flows through the public network gateway with existing approaches performing the outsourcing job [2,8].

5. Simulation

In this section, the performance of MOSC is studied. We first introduce the network topology and the parameters in Section 5.1. After that the simulation results are presented in Section 5.2.

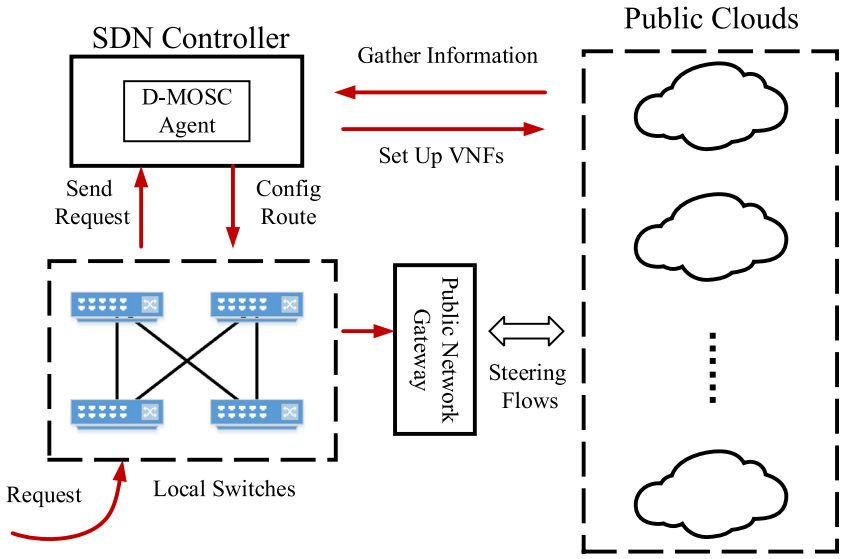


Fig. 12. System diagram of MOSC.

5.1. Simulation setup

We developed a flow level network simulator with Python, and the simulator runs on a server with Intel Xeon E5-2620 CPU, and 32 GB memory. The simulator takes the SFC requests, network topology, VNF pricing scheme, link latencies as inputs, and outputs the optimized SFC assignments across multiple cloud providers based on the results of D-MOSC.

5.1.1. Network topology and traffic dataset

As mentioned in Section 2, the network topology is full-meshed. The costs of links are generated randomly in the range of 10–100, and the delays of links are set to random values with an average of 40 ms. Unless otherwise specified, there are 20 different cloud providers in our simulations. And 1024 flows are injected into the network with random traffic rates in the range of 10–100 Mbps. In addition, the capacities of nodes are set to unlimited due to the scalability of public clouds. Since the virtual links can be established with the on-demand provision, unless otherwise specified, the capacities of links are also set to unlimited.

5.1.2. Input parameters

Now let's recall the input parameters settings mentioned in Section 2. The parameters that affect the result of MOSC include the price schemes of VNFs in different nodes, the latency requirements of flows, and the requirements of SFCs.

For simplicity, considering each VNF i in the set of VNFs U, the processing cost of VNF i per 1 Mbps in the local network is regarded as the base price P_{ib}. P_{ib} of different VNF i is set to a random value in the range of 10–100.

The average price of VNF i per 1 Mbps in clouds, denoted as $\overline{P_{ic}}$, is a fraction of P_{ib}. That is $\overline{P_{ic}} = \alpha P_{ib}$, where *the coefficient α is a value in range of 0–1*. After that, the price of VNF i in cloud j per 1 Mbps, denoted as P_{ij}, is set to a random value in range of $0.8\overline{P_{ic}}$ to $1.2\overline{P_{ic}}$.

The SFC is a sequence of VNFs with a length of m, which is generated by randomly picking m VNFs from U. m is the length of SFC in the simulations.

In the simulations, the coefficient α, the latency requirement of each flow, and the length of SFC is used to evaluate the overall cost of MOSC in different scenarios.

Table 5
Comparison between ILP Model and MOSC.

Number of flows	Result	
	ILP model	D-MOSC
64	118,482	120937
128	249,998	252314
256	510,983	522319

5.2. Simulation results

In this section, we present the results of ILP model and D-MOSC. As a benchmark, the first-fit based optimization algorithm is also evaluated. The first-fit based optimization algorithm (referred as *FF*) obtains the cost-efficient path by the first-first algorithm [32] firstly, then adjusts the path to satisfy the QoS requirements with Algorithm 3. Moreover, we evaluate the algorithm which outsources all the VNFs to the clouds without consideration of latency requirements (referred as *delay-ignored algorithm*). At last, we compare D-MOSC and G-MOSC to see how the deviation algorithm improves the performance.

5.2.1. Comparison between results of D-MOSC and ILP model

We compare the result of MOSC with the optimal solution of ILP model obtained by applying CPLEX. Since the problem is NP-hard, we present the optimal solution in small-scale networks which include five public clouds and several flows. In this scenario, the length of SFC is set to 5, and coefficient α is set to 0.2.

From Table 5, we can see that the result of D-MOSC is near optimal. The gap between the result of D-MOSC and the optimal solution is less than 4%.

5.2.2. Results with different α

We evaluate the overall cost of MOSC with that of local placement scheme and first-fit based optimization algorithm. In this scenario, the latency requirements of flows are randomly generated with an average of 120 ms, and the length of SFC is set to 5. The results under different coefficient α from 0.2 to 0.8 are presented.

From Fig. 13, we can see that when α is 0.2, MOSC saves about 79.2% cost compared with that of local placement scheme. And 21.5% cost is saved when α is 0.8. In addition, D-MOSC achieves up to 50.7% cost savings compared with that of the first-fit based optimization algorithm.

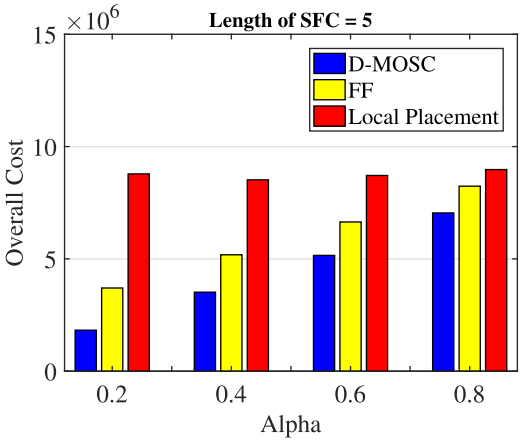


Fig. 13. Overall cost vs. the coefficient α.

Table 6
The proportion of outsourced VNFs with different α.

Alpha	Proportion of outsourced VNFs(%)
0.2	73.8
0.4	59.7
0.6	48.4
0.8	19.9

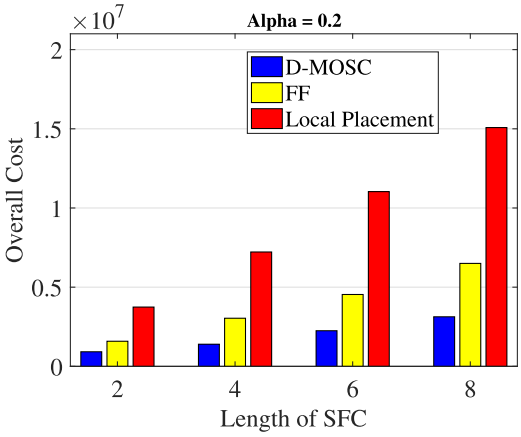


Fig. 14. Overall cost vs. length of service chain.

Since the costs of links are not negligible, fewer VNFs will be outsourced to public clouds when the costs of VNFs in clouds are almost the same as that of the local network. As a result, the improvement of D-MOSC is relatively low when α is 0.8. To study the impact of different pricing schemes on the amount of outsourced VNFs, the proportion of the outsourced VNFs to public clouds with different α is given in Table 6.

5.2.3. Results with different length of SFC

We evaluate the overall cost of MOSC with that of local placement scheme and first-fit based optimization algorithm. In this scenario, the latency requirements of flows are randomly generated with an average of 120 ms, and the coefficient α is set to 0.2. The results under different length of SFC from 2 to 8 are presented in Fig. 14.

Though the overall cost increases with the length of SFC, MOSC always saves about 70% cost compared with that of the local placement scheme and about 45% cost compared with that of FF. The reason is that D-MOSC always tries to find the most-likely next state in HMM, thus the increase of SFC length (corresponding to the length of Markov Chain in HMM) has no effect on the selection of the next state.

Table 7
The proportion of outsourced VNFs with link capacity constraint.

Link capacity (Gbps)	Proportion of outsourced VNFs(%)
20	16.4
40	30.1
60	45.2
80	57.9
100	71.8

5.2.4. Results of D-MOSC with different QoS requirement

Fig. 15 shows the results of D-MOSC as well as that of local placement scheme and delay-ignored algorithm with latency requirement varying from 30ms to 200ms. The coefficient α is set to 0.2.

Since all the VNFs are deployed in the local network, the overall cost obtained by local placement scheme remains unchanged. On the contrary, since the coefficient α is small, all the VNFs will be outsourced to the public clouds with the lowest price by the delay-ignored algorithm. Thus the result of the delay-ignored algorithm also remains unchanged.

The cost of MOSC lies between that of the local placement scheme and that of the delay-ignored algorithm. If the flows are latency sensitive, the possibility of outsourcing VNFs to the public cloud is low, so the resulted cost of D-MOSC is close to that of the local placement scheme as shown on the left side of the chart. On the contrary, if the flows are insensitive to latency, the possibility of outsourcing VNFs is high, so the resulted cost of D-MOSC is close to that of the delay-ignored algorithm as shown in the right side of the chart.

5.2.5. Results of D-MOSC with link capacity constraint

In the above simulation, we made an assumption that the link capacity is unlimited since the virtual links can be established with the on-demand provision. However, due to the physical links constraint, this assumption may not be made at all time, especially in high traffic scenarios. In view of this, we add the simulations that show the performance of D-MOSC with the link capacity constraints.

Recall, in the simulation setup, there are 1024 flows with random traffic rates in the range of 10–100 Mbps. To ensure the coverage of all the situations, we vary the link capacity from 0 to 100 Gbps, and we study the overall cost achieved by the D-MOSC. By limiting the link capacity to 0 Gbps, it means we can only deploy the VNFs in the local network, and respectively the maximum capacity allows all the VNFs be outsourced to the public clouds. In the scenario, the latency requirements of flows are randomly generated with an average of 120 ms, and the coefficient α is set to 0.2. The results under different SFC length from 2 to 8 are presented in Fig. 16.

In Fig. 16, it can be figured out that along with the increasing of the link capacity, the overall cost is decreasing in a nearly linear manner, with exceptions in some parts (e.g., the beginning and end of the curve). By inspecting the actual deployment of the SFC, we find out that the exception is caused by choosing the alternative path of SFC. While the link is full, D-MOSC is trying to find another deviation path to the candidate clouds which has enough capacity to serve the flows until all the candidate paths are full, then it will turn into next outsourcing scheme which may incur higher cost.

To better understand how the link capacity constraint affects G-MOSC, we present the proportion of outsourced VNFs with different link capacities in Table 7. From Table 7, we can find out that

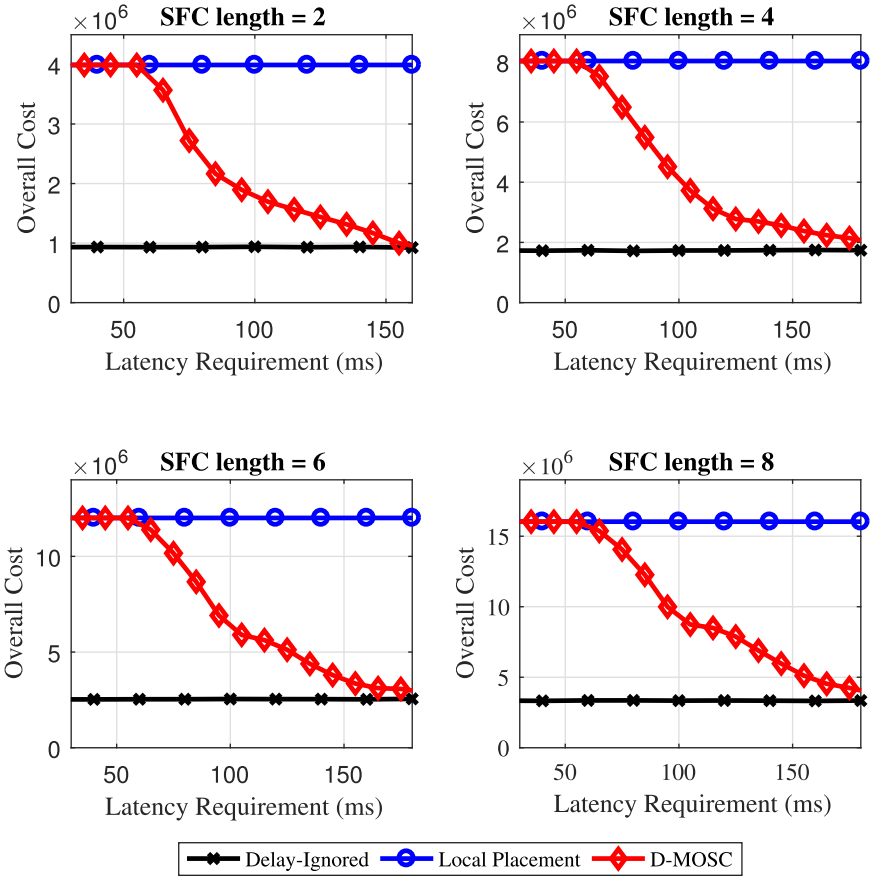


Fig. 15. Overall cost vs. latency requirement.

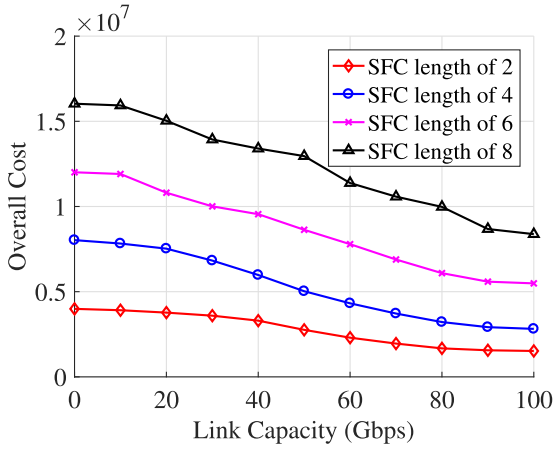


Fig. 16. Overall cost vs. link capacity.

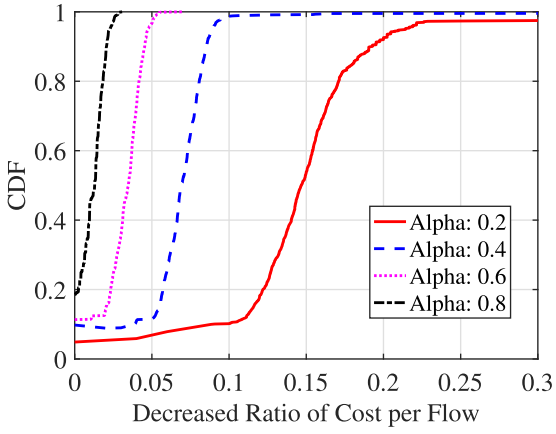


Fig. 17. Decreased ratio of cost of D-MOSC per flow vs. the coefficient α.

with lower link capacity, fewer VNFs are outsourced. This is the reason that causes the increasing of cost along with the decreasing of link capacity.

5.2.6. Comparison between D-MOSC and G-MOSC

In this section, we conduct the simulation to see the performance of D-MOSC while taking G-MOSC as the baseline. To evaluate the performance of D-MOSC, we compare these two algorithms under different coefficient α and different length of Service Function Chain, similarly, at each scene, the First-Fit algorithm is also evaluated as the benchmark.

Result with different α: The evaluation on D-MOSC and G-MOSC with different α contains two parts, first we show the CDF of decreased ratio of cost per flow in Fig. 17, the decreased ratio is defined as

$$ratio = \frac{c_{greedy} - c_{deviation}}{c_{greedy}}$$

Secondly, we present the overall cost of three compared algorithms in Fig. 18.

From Fig. 17, we can see that when $\alpha = 0.2$, the median of decreased ratio per flow is around 15%, and tail is around 30%. With higher α, the average cost per flow is lower. The overall costs with different α which is shown in Fig. 18 show that compared with that of G-MOSC, D-MOSC has lower cost, the decreased ratio is

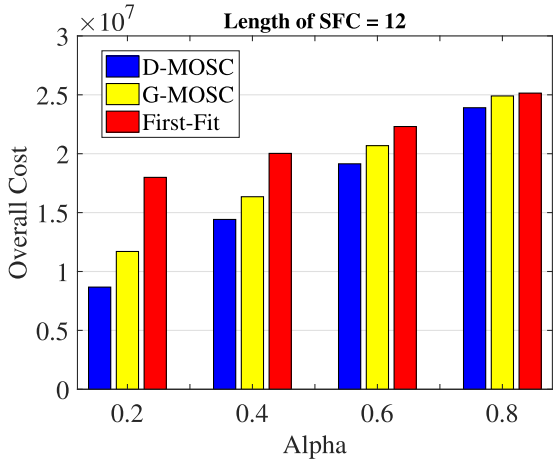


Fig. 18. Overall cost vs. the coefficient α.

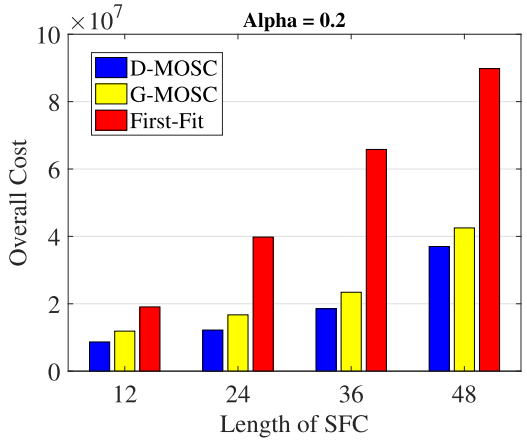


Fig. 20. Overall cost vs. length of SFC.

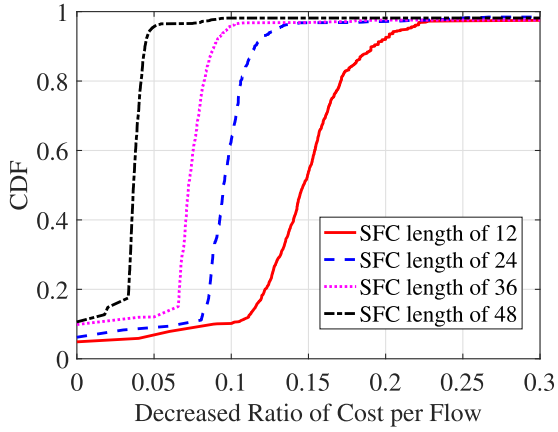


Fig. 19. Decreased ratio of cost of D-MOSC per flow vs. length of SFC.

ranging from 4% – 25.8%. Meanwhile, G-MOSC still outperforms the First-Fit algorithm up to 34.9%.

The above simulation results show D-MOSC does a better job when finding the tradeoff between the cost and latency compared with G-MOSC. The reason is that D-MOSC benefits from the most-likely path seeking procedure of HMM, but G-MOSC does a separate job when adjusting the latency violation paths.

Result with different length of SFC: In Fig. 17, we found some very interesting results. The decreased ratio of the cost per flow is higher while the SFC length is lower. This is because when we have a small number of SFC length, the latency of the entire SFC will be affected by the rescheduling of a small number of VNFs. In order to clearly show how the SFC length affects the latency increasing, we increased the SFC length into a wider range from 12 to 48.

From Fig. 19, we can see that with higher SFC length, the ratio of decreased latency is lower, since each VNF will account for a smaller portion in the entire SFC when the total length is longer. In Fig. 20, it can be figured out that the D-MOSC outperforms G-MOSC by 12.7%–26.7%, and decreases up to 52.7% compared with that of the first-fit based optimization algorithm.

6. Related work

There are several previous works focusing on optimization of virtual network function placement problems [16–20,33,34]. The setup costs and connection costs of VNFs are analyzed and optimized in [16,18]. And mathematic models and efficient algorithms for traffic engineering or resource utilization are proposed in [17–20]. However, some of these works[16,17,20] do not consider the service function chains, while the other works [18,19] do not consider the possibility of outsourcing VNFs to multiple public clouds.

The placement of virtual machines across multiple clouds is discussed in [35–37]. In these work, the pricing and specification of multiple cloud providers are discussed. Nevertheless, the work only focuses on the optimal placement of virtual machines without consideration of SFC and QoS of flows.

Cloudward Bound optimizes the cost by outsourcing the functions to public clouds [38]. In Cloudward, the benefit of dynamic planning the migration of enterprise applications is discussed. However, the technical background of Cloudward Bound is workflows of enterprise network such that Cloudward Bound cannot deal with the characteristic of network flow and SFC (e.g., QoS, the order of VNFs). Meanwhile, Cloudward Bound does not consider the specifications of multiple cloud providers either.

There are various works on the Virtual Network Embedding (VNE) problem which has the similar theoretical background of our work. More broadly, CoordVNF [39] proposed a quick heuristic to solve the allocation problem in large-scale networks. Savi et al. develop an optimization method which is the extension of VNE problem [40]. Dräxler et al. [41] formulated the embedding process as a joint optimization problem, in which the optimization objective is determined as a function of data rates. Although these works have the similar theoretical background, the accurate QoS requirements of specific flows make it hard to simply add or modify some constraints in their solution to solve the SFC outsourcing problem, since all of their models are working in the aspect of the whole network rather than flows. Adding the concept of flows into the traditional VNE models requires fundamental changes to the problem formulation. In view of this, we argue that the problem needs a totally different model.

VNF-P [42] discusses the hybrid scenario that the network services may be deployed in physical hardware and virtual services. The researchers propose a formal model that can be used to allocate resources in hybrid NFV networks. This macro solution can allocate the resources in network-wide, however, similar to the VNE problem, it is hard to use this solution to consider the QoS requirement of a specific flow.

Bhamare et al. proposed another similar solution that optimizes the VNF placement in multi-cloud SFC architecture [25]. The researchers develop an optimization model to reduce the latency to the end users by reducing the inter-cloud traffic. However, as we

observe, data center operators also have concerns of how to meet the strict demand of widely varying deadlines [27] with efficient operational cost. Inter-cloud deployment of VNFs can benefit from the different pricing scheme to achieve higher cost efficiency. In view of this, this problem needs to be revisited in consideration of the trade-off between operational cost and network latencies. In our work, we mainly focus on the cost-efficient outsourcing plan of network functions with the satisfaction of deadlines.

To the best of our knowledge, we are the first to propose an approach to optimize the overall cost with consideration of service function chain, the QoS requirements of flows, and multiple cloud providers.

7. Conclusion

In this work, we formulate the service function chain outsourcing problem as an ILP model and propose an efficient heuristic method to outsource SFC with QoS guarantee (MOSC). Compared with our previous conference paper, we propose an improved deviation based algorithm (D-MOSC) to find the tradeoff between cost and latency. Rather than two separate optimization part of the greedy algorithm in the conference paper, D-MOSC benefits from the combination of the most-likely path seeking procedure of Hidden Markov Model and the path deviation. We have conducted extensive simulations to show the performance of G-MOSC which derives a near optimal cost in small-scale networks and achieves up to 79.2% cost savings compared with that of local placement scheme. Furthermore, compared with that of the greedy based algorithm (G-MOSC) we proposed in the conference version, the deviation based MOSC achieves up to 26.7% cost savings with the guarantee of latency requirements.

Acknowledgment

This work is partially supported by the National Basic Research Program (973 Program) (2013CB329103), NSFC Fund (61271165, 61301153, 61401070, 61571098, 61671130), Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT), the 111 Project (B14039), and Science and Technology Program of Sichuan Province (2016GZ0138).

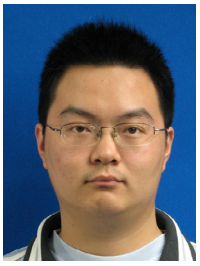
References

[1] Problem statement for service function chaining, (http://www.rfc-editor.org/rfc/rfc7498.txt). (accessed 2-May-2017).
[2] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, V. Sekar, Making middleboxes someone else's problem: network processing as a cloud service, in: Proceedings of the ACM SIGCOMM, ACM, 2012, pp. 13–24, doi:10.1145/2377677.2377680.
[3] N. ETSI, Network functions virtualisation, (2012).
[4] S. Rajagopalan, D. Williams, H. Jamjoom, A. Warfield, Split/merge: System support for elastic execution in virtual middleboxes., in: Proceedings of the USENIX NSDI, 2013, pp. 227–240.
[5] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, A. Akella, Opennf: Enabling innovation in network function control, in: Proceedings of the ACM SIGCOMM, 2014, pp. 163–174, doi:10.1145/2619239.2626313.
[6] T. Wen, H. Yu, G. Sun, L. Liu, Network function consolidation in service function chaining orchestration, in: Proceedings of the IEEE International Conference on Communications (ICC), 2016, pp. 1–6, doi:10.1109/ICC.2016.7510679.
[7] V. Eramo, E. Miucci, M. Ammar, F.G. Lavacca, An approach for service function chain routing and virtual function network instance migration in network function virtualization architectures, IEEE/ACM Trans. Netw. PP (99) (2017) 1–18, doi:10.1109/TNET.2017.2668470.
[8] G. Gibb, H. Zeng, N. McKeown, Outsourcing network functionality, in: Proceedings of the ACM Workshop on Hot Topics in Software Defined Networks, 2012, pp. 73–78, doi:10.1145/2342441.2342457.
[9] H. Abu-Libdeh, L. Princehouse, H. Weatherspoon, Racs: a case for cloud storage diversity, in: Proceedings of the ACM Symposium on Cloud Computing, 2010, pp. 229–240, doi:10.1145/1807128.1807165.
[10] Data plane development kit, (http://www.intel.com/content/www/us/en/communications/data-plane-development-kit.html). (accessed 2-May-2017).
[11] M.-A. Kourtis, G. Xilouris, V. Riccobene, M.J. McGrath, G. Petralia, H. Koumaras, G. Gardikis, F. Liberal, Enhancing VNF performance by exploiting SR-IOV and DPDK packet processing acceleration, in: Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), 2015, pp. 74–78, doi:10.1109/NFV-SDN.2015.7387409.
[12] V. Moreno, J. Ramos, J.L. García-Dorado, I. Gonzalez, F.J. Gomez-Arribas, J. Aracil, Testing the capacity of off-the-shelf systems to store 10gbe traffic, IEEE Commun. Mag. 53 (9) (2015) 118–125, doi:10.1109/MCOM.2015.7263355.
[13] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, A view of cloud computing, Commun. ACM 53 (4) (2010) 50–58, doi:10.1145/1721654.1721672.
[14] G.D. Forney Jr, The Viterbi algorithm, Proc. IEEE 61 (3) (1973) 268–278, doi:10.1109/PROC.1973.9030.
[15] H. Chen, S. Xu, X. Wang, Y. Zhao, K. Li, Y. Wang, W. Wang, et al., Towards optimal outsourcing of service function chain across multiple clouds, in: Proceedings of the IEEE International Conference on Communications (ICC), 2016, pp. 1–7, doi:10.1109/ICC.2016.7510996.
[16] R. Cohen, L. Lewin-Eytan, J.S. Naor, D. Raz, Near optimal placement of virtual network functions, in: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), 2015, pp. 1346–1354, doi:10.1109/INFOCOM.2015.7218511.
[17] B. Addis, D. Belabed, M. Bouet, S. Secci, Virtual network functions placement and routing optimization, in: Proceedings of the IEEE International Conference on Cloud Networking (CloudNet), 2015, pp. 171–177, doi:10.1109/CloudNet.2015.7335301.
[18] M. Bari, S.R. Chowdhury, R. Ahmed, R. Boutaba, On orchestrating virtual network functions in nfv, in: Proceedings of the IEEE International Conference on Network and Service Management (CNSM), 2015, pp. 50–56, doi:10.1109/CNSM.2015.7367338.
[19] M. Xia, M. Shirazipour, Y. Zhang, H. Green, A. Takacs, Network function placement for nfv chaining in packet/optical datacenters, J. Lightwave Technol. 33 (8) (2014) 1565–1570.
[20] J.W. Jiang, T. Lan, S. Ha, M. Chen, M. Chiang, Joint vm placement and routing for data center traffic engineering, in: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), 2012, pp. 2876–2880, doi:10.1109/INFCOM.2012.6195719.
[21] Service function chaining (sfc) architecture, (http://www.rfc-editor.org/rfc/rfc7665.txt). (accessed 2-May-2017).
[22] M. Alizadeh, A. Greenberg, D.A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, M. Sridharan, Data center TCP (DCTCP), in: Proceedings of the ACM SIGCOMM, 2010, pp. 63–74, doi:10.1145/1851182.1851192.
[23] M. Alicherry, T. Lakshman, Network aware resource allocation in distributed clouds, in: Proceedings of the IEEE INFOCOM, IEEE, 2012, pp. 963–971.
[24] S. Mehraghdam, M. Keller, H. Karl, Specifying and placing chains of virtual network functions, in: Proceedings of the IEEE 3rd International Conference on Cloud Networking (CloudNet), IEEE, 2014, pp. 7–13.
[25] D. Bhamare, M. Samaka, A. Erbad, R. Jain, L. Gupta, H.A. Chan, Optimal virtual network function placement in multi-cloud service function chaining architecture, Comput. Commun. 102 (2017) 1–16.
[26] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, S. Davy, Design and evaluation of algorithms for mapping and scheduling of virtual network functions, in: Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft), IEEE, 2015, pp. 1–9.
[27] S. Luo, H. Yu, L. Li, Decentralized deadline-aware coflow scheduling for datacenter networks, in: Proceedings of the 2016 IEEE International Conference on Communications (ICC), IEEE, 2016, pp. 1–6.
[28] S. Luo, H. Yu, Y. Zhao, S. Wang, S. Yu, L. Li, Towards practical and near-optimal coflow scheduling for data center networks, IEEE Trans. Parallel Distrib. Syst. 27 (11) (2016) 3366–3380.
[29] S.R. Eddy, Hidden Markov models, Curr. Opin. Struct. Biol. 6 (3) (1996) 361–365.
[30] J.Y. Yen, Finding the k shortest loopless paths in a network, Manag. Sci. 17 (11) (1971) 712–716.
[31] E.Q.V. Martins, M.M.B. Pascoal. An algorithm for ranking optimal paths. Technical Report 01/004, CISUC, 2000 (http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.3707).
[32] R.E. Korf, A new algorithm for optimal bin packing, in: Proceedings of AAAI/IAAI, 2002, pp. 731–736.
[33] X. Li, C. Qian, The virtual network function placement problem, in: Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2015, pp. 69–70, doi:10.1109/INFCOMW.2015.7179347.
[34] R. Riggio, T. Rasheed, R. Narayanan, Virtual network functions orchestration in enterprise wlans, in: Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015, pp. 1220–1225, doi:10.1109/INM.2015.7140470.
[35] S. Chaisiri, B.-S. Lee, D. Niyato, Optimal virtual machine placement across multiple cloud providers, in: Proceedings of the IEEE Asia-Pacific Services Computing Conference (APSCC), 2009, pp. 103–110, doi:10.1109/APSCC.2009.5394134.
[36] J.T. Piao, J. Yan, A network-aware virtual machine placement and migration approach in cloud computing, in: Proceedings of the IEEE International Conference on Grid and Cloud Computing, 2010, pp. 87–92, doi:10.1109/GCC.2010.29.
[37] Y. Gao, H. Guan, Z. Qi, Y. Hou, L. Liu, A multi-objective ant colony system algorithm for virtual machine placement in cloud computing, J. Comput. Syst. Sci. 79 (8) (2013) 1230–1242, doi:10.1016/j.jcss.2013.02.004.
[38] M. Hajjat, X. Sun, Y.-W.E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, M. Tawarmalani, Cloudward 520 bound: Planning for beneficial migration of

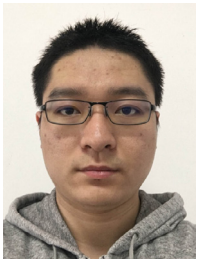
enterprise applications to the cloud, in: Proceedings of the ACM SIGCOMM, 2010, pp. 243–254, doi:10.1145/1851182.1851212.
[39] M.T. Beck, J.F. Botero, Scalable and coordinated allocation of service function chains, Comput. Commun. 102 (2017) 78–88.
[40] M. Savi, M. Tornatore, G. Verticale, Impact of processing costs on service chain placement in network functions virtualization, in: Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), IEEE, 2015, pp. 191–197.
[41] S. Dräxler, H. Karl, Z.Á. Mann, Joint optimization of scaling and placement of virtual network services, in: Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE Press, 2017, pp. 365–370.
[42] H. Moens, F. De Turck, VNF-P: a model for efficient placement of virtualized network functions, in: 2014 10th International Conference on Network and Service Management (CNSM), IEEE, 2014, pp. 418–423.

Huan Chen is currently a Ph.D. student at University of Electronic Science and Technology of China. His research interests include Software-defined Networking, Network Function Virtualization, and topics such as resource management, implementation on network system architectures.

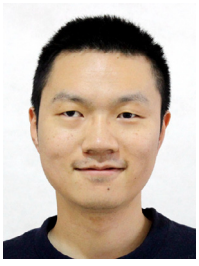
Xiong Wang received the B.S. degree in Electronic and Information Engineering from Chongqing University of Posts and Telecommunications and received the Ph.D. degree in communication and Information System from the University of Electronic Science and Technology of China in 2003 and 2008, respectively. Since then, he has been with the School of Communication and Information Engineering at the University of Electronic Science and Technology of China, where he is currently an Associate Professor.



Yangming Zhao is a Ph.D. candidate in University of Electronic Science and Technology of China (UESTC). He received his B.S. degree in Communication Engineering from UESTC in July 2008. His research interests include network optimization and data center networks.



Tongyu Song is currently a Ph.D. student at University of Electronic Science and Technology of China. His research interests include Software-defined Networking, applying Big Data in network, and topics such as cooperation between application and network, implementation on network system architectures.



Yang Wang is a Ph.D. student at University of Electronic Science and Technology of China. He received his B.S. degree in Communication Engineering from UESTC in July 2014. His research interest includes the architecture and implementation of Network Function Virtualization.

Shizhong Xu received the B.S., M.S., and Ph.D. degrees in electrical engineering from University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 1994, 1997 and 2000, respectively. He is now a Professor in UESTC. His research interests include broadband networks, all-optical network, and next-generation networks.

Lemin Li graduated from Jiaotong University, Shanghai, China in 1952, majoring in electrical engineering. From 1952 to 1956 he was with the Department of Electrical Communications at Jiaotong University. Since 1956 he has been with Chengdu Institute of Radio Engineering (now the University of Electronic Science and Technology of China). From August 1980 to Aug. 1982, he was a visiting scholar in the Dept. of Electrical Engineering and Computer Science at the University of California at San Diego, USA, doing research on digital and spread spectrum communications. His present research work is the area of communication networks including broadband networks and wireless networks.