Network Function Placement Under Randomly Arrived Networking Traffic

Jie Sun, Feng Liu, *Member, IEEE*, Huandong Wang, *Member, IEEE*, Manzoor Ahmed, Yong Li, *Senior Member, IEEE*, Lianlian Zhang, and Hao Zeng

***Abstract*—The virtual network functions (VNFs) placement problem has drawn significant attention from both academia and industry in recent years. Most of the researchers have ignored the fact that the probability of traffic flows through VNFs cannot always be 100%. In this paper, we study the placement scheme for virtual network function considering randomized data traffic (VNFPRAT). Our objective is to determine optimal deployment locations for VNFs and minimize total end-to-end delay. We formulate the VNFPRAT problem as a 0-1 nonlinear programming problem and prove its NP-hardness. This formulation is linearized to obtain the optimal solution for small scale networks. Besides, two efficient metaheuristics, i.e., greedy and simulated annealing, are proposed quickly find a near-optimal placement solution. Extensive simulations demonstrate that our proposed approach achieves 38.8% less end-to-end delay than the generic algorithm.**

***Index Terms*—Network function virtualization, VNF placement, randomly arrived traffic.**

I. Introduction

NETWORK function virtualization (NFV) is an overarching concept enabling dynamic and flexible deployment of network functions (NFs) that is termed as virtual network functions (VNFs), such as deep packet inspection (DPI), firewall, proxy, and intrusion detection and prevention (IDP), etc. Using virtualization technology, VNFs are deployed in virtual machines (VMs) or containers of industry standard servers. The decoupling of NFs from dedicated equipments effectively reduces operational expenses (OPEX) and capital expenditures (CAPEX) on NFs' placement, management, and maintenance [1]. The integration of Software Defined Networking (SDN) [2] and NFV can automatically manage the life-cycle of VNFs and dynamically construct network services. Service function chain (SFC) within VNF paradigm refers to an ordered set of VNFs applied to traffic flows [3]. In distributed deployment, network performance is affected by the placement of VNFs, which necessitate to optimize the VNF placement [4].

There have been significant efforts from both academia and industry on how to place VNFs. Since that VNFs distributed in networks may bring a longer SFC path, some works have focused on optimizing the VNF placement to reduce the end-to-end delay. The references [5]–[7] paid attention to shortening total link delay to reduce the end-to-end delay. Among them, Xu *et al.* [5] proposed a Viterbi algorithm to reduce the end-to-end delay of the cross-domain SFC path. In addition, the references [8], [9] had taken the VNF processing delay into consideration to optimize the end-to-end delay. Marotta *et al.* [10] searched the VNF placement that satisfy the maximum tolerated end-to-end delay including queuing delay in switches. These works have covered different parts of end-to-end delay. However, they had not given enough attention to the link delay from a switch to its connected VNF. In our work, we aim to reduce total end-to-end delay that includes the link delay between switches and the VNFs connected to them.

Most of existing works about the VNF placement have ignored the fact that traffic flows traverse VNFs with probability, which cannot always be 100%. In real networks, traffic arrives to VNFs randomly. This is due to four reasons: 1) At the network level, communications in the opportunistic network rely on the probability that a mobile node encounters with another one [11]. 2) At the protocol level, multi-path communications brought by the stream control transmission protocol (SCTP) make traffic visit VNFs in separate paths with certain probability [12]. 3) At the queue level, the weighted random early detection (WRED) strategy randomly drops packets with probabilities when the forwarding queue becomes congested [13]. 4) Also considering the devices failure, e.g., any node (NFV servers or forwarding devices are down) or link fails (data cables cut) [14], [15], traffic may have to go through backup VNF instances. The stochastic manner of flow arrival at VNFs have influence on the statistical average delay spent on the link between a switch and its connected VNF. In addition, if VNFs have low visited probabilities, their occupations at the critical locations may prolong total end-to-end delay [16].

Manuscript received November 27, 2019; revised November 3, 2020 and April 6, 2021; accepted June 16, 2021. Date of publication June 23, 2021; date of current version December 9, 2021. This work was supported in part by the National Key Research and Development Program of China (Grant No. 2020YFA0711403) and National Natural Science Foundation of China (Grant Nos. 61231013, 91738301, 91438206 and 91638301), and Program for New Century Excellent Talents in University (Grant No. NCET-09-0025), and Fundamental Research Funds for the Central Universities. The associate editor coordinating the review of this article and approving it for publication was A. B. MacKenzie. *(Corresponding author: Yong Li.)*

Jie Sun, Feng Liu, Lianlian Zhang, and Hao Zeng are with the School of Electronics and Information Engineering, Beihang University, Beijing 100191, China (e-mail: sunjiehyit@buaa.edu.cn; liuf@buaa.edu.cn; lianlianzhang@buaa.edu.cn; zenghao950131@buaa.edu.cn).

Huandong Wang and Yong Li are with the Department of Electronic Engineering, Tsinghua University, Beijing 10084, China (e-mail: wanghuandong@tsinghua.edu.cn; liyong07@tsinghua.edu.cn).

Manzoor Ahmed is with the College of Computer Science and Technology, Qingdao University, Qingdao 266071, China (e-mail manzoor.achakzai71@yahoo.com).

Digital Object Identifier 10.1109/TCCN.2021.3091711

The location of a VNF depends not only on the number of processed flows but also depends on the probabilities of arrived flows. To fully exploit such a probability and attain higher network performance, there are four main challenges: 1) Random probabilities make servers' loads uncertain. 2) The routing path of each flow is difficult to determine. 3) The processing order of each VNF becomes more problematic. 4) The network traffic can be heavy, which need a scalable solution. Consequently, such factors make the problem non-trivial. Therefore, it necessitates to design an effective approach for jointly optimizing routing paths and VNFs' placements.

In this paper, we study the problem of placing VNFs for randomly arriving network traffic. We optimize VNF placement in accordance with the policy specifications [17]. Since arrival of traffic flow occurs in a probabilistic manner, the traffic load of each link/node is difficult to determine. To tackle such a challenge, Laghrissi *et al.* [18] analyzed the accessing probability in mobile edge computing (MEC) and presented a forecast-assisted method to place VNFs. Herker *et al.* [19] studied the failure probability of data-center infrastructure and proposed a novel deployment algorithm to improve the availability of SFCs. Sun *et al.* [20] investigated the probability of successfully chaining big-data-based VNFs and proposed a heuristic algorithm to strike a balance between service reliability and VNF deployment cost. The references [10], [21] proposed robust optimization approaches to tackle the uncertainties of traffic demands and VNFs' resource demands, respectively. The references [22], [23] designed the stochastic programming methods to handle the bandwidth demand uncertainty and the dynamism in the cloud environment, respectively. These works proposed effective algorithms to tackle different uncertainties in the network virtualization context. However, they do not consider different probability distribution of flow arrival at VNFs in the intra-SFC. In this paper, we work towards the uncertainty of data traffic arrival based on the expected value of the random variables.

The main contributions of our work are as follows.

- We formulate the VNF placement problem for randomly arrived traffic (VNFPRAT) as 0-1 nonlinear programming problem. Next we proved that the VNFPRAT problem is NP-hard via reducing the quadratic assignment problem [24] to it. By linearizing the quadratic items, we can obtain the optimal solutions using open source solvers.
- After proving the VNFPRAT problem is NP-hard, we present a greedy algorithm to overcome the scalability challenge. The greedy algorithm can reduce total end-to-end delay effectively. Next, to avoid trapping in local optimum, a simulated annealing algorithm is proposed to optimize the placement schemes globally.
- We conducted extensive simulations to demonstrate that our proposed algorithm is able to minimize end-to-end delay up to 38.8% in comparison with the algorithm proposed in [7].

We organize the rest of this paper as following. Section II gives a brief overview of the related works. Section III formulates the VNFPRAT problem as a 0-1 programming problem and proves the NP-hardness of this problem. Section IV presents an optimal algorithm and two heuristic algorithms. Section V portrays the simulation results, while Section VI concludes this paper.

II. Related Work

The construction of NFs is often based on the dedicated hardwares to guarantee the performance. The coupling between NFs and hardwares brings high OPEX and CAPEX, e.g., deploying, updating, and managing NFs. NFV can effectively solve this tightly coupling and automatically manage the life-cycle of VNFs [25]. According to [1], NFV makes networks become more flexible, and can save 20% to 35% CAPEX of constructing network services. The Internet Engineering Task Force (IETF) SFC Working Group (WG) proposes the definition of SFC to simplify the service construction by chaining VNFs [3]. To realize the potential benefits of NFV, we should address the problem of VNF placement that determines a series of performance, e.g., delay, energy, and costs [4].

Many efforts of optimizing VNF placement have been devoted to reduce the end-to-end delay [5]–[10], [26]. VNF can be flexibly placed in a NFV server rather than a access point that often hosts traditional NFs. The flexible placement also brings a longer SFC path and end-to-end delay. Xu *et al.* [5] proposed a Viterbi-based heuristic algorithm to search a cross-domain path with low end-to-end delay for embedding security service chains. To reduce the end-to-end delay in mobile network, Cziva *et al.* [6] applied the optimal stopping theory to dynamically replace VNFs. Liu *et al.* [7] proposed a greedy algorithm to reduce the total end-to-end delay of flows. The references [8], [9] proposed the novel heuristic algorithms to reduce the end-to-end delay that includes link delay and VNF processing delay. For the dynamic VNF placement, Jia *et al.* [26] considered VNFs' placement probabilities and proposed an online dependent rounding scheme to reduce the operational costs, e.g., the end-to-end delays. Marotta *et al.* [10] formulated packets in switches as a M/M/K/1 queuing system and searched the placements that satisfy the end-to-end delay constraints. The authors have focused on different parts of the end-to-end delay, e.g., link delay, processing delay in VNFs, and queuing delay in switches. However, they had not given enough attention to the link delay between a switch and the connected VNF. In this paper, we pay attention to the influence of stochastic flows on the link delay from a switch to a VNF connecting to it, and try to reduce the total end-to-end delay of flows.

The traffic flows through VNFs exhibit a stochastic behavior. Casazza *et al.* [14] investigated the probability of physical failure, in such situation traffic needs to go through a backup VNF. Randomly arriving traffic also exists in some special communications, e.g., the opportunistic network [11] and the SCTP protocol [12]. The traffic procession of VNFs in opportunistic networks depends on the randomly moving paths of user equipments [11]. The communication across geographically distributed terminals lead to different routing and service path using the SCTP protocol [12]. The WRED strategy randomly drops packets with probabilities

when congestion happens in forwarding queues [13]. In addition, some VNFs have the stochastic behaviors, e.g., the source network address translation (SNAT) randomly selects a external network address from a pool of address and port for a internal network application [27]. These works explained the source of the VNFPRAT problem, which refers to the VNF placement problem considering the randomized data traffic.

There are some works focusing on the probabilistic manner of traffic data in the VNF placement problem. Laghrissi *et al.* [18] focused on the accessing probability of each user equipment in edge computing and presented a forecast-based method to deploy VNFs. Herker *et al.* [19] addressed the problem of VNF placement and probabilistic device failure, and deployed redundant service chains to improve the service availability. Considering the probability of VNF failures, Chantre and da Fonseca [28] improved the reliability of network services via redundancy. Sun *et al.* [20] studied the reliability of physical nodes/links and designed a heuristic algorithm to maintain a balance between service availability and resources consumption. These research works considered probabilistic traffic in NFV network, and designed approaches to deploy the VNFs. Fischer [29] generated random scenarios with known solutions to evaluate the generality of VNF placement approaches. However, these works ignored different probability distribution of traffic arriving at VNFs of the intra-SFC. Our work focus on the stochastic manner of flows visiting VNFs and propose two heuristic algorithms to find near-optimal placements.

Some works use the stochastic or robust optimization approaches to address the uncertainty in different optimization problems. Bauschert *et al.* [21] proposed a robust optimization approach to reduce solution conservatism by taking into account spatial correlation between uncertain traffic demands, associated with user requests in network slices. Considering the uncertain bit rate and delay on small cells, Blanco *et al.* [30] employed a robust optimization algorithm to place mobile edge computing (MEC) services. These robust optimization approaches utilize the uncertainty set of the random variables rather than an accurate distribution function. However, robust optimization algorithms are often used for optimizing the worst-case scenario in the uncertainty set. For average cases, the conservative solutions obtained by robust optimization algorithms may result in lower network performance. Farkiani *et al.* [22] utilized a stochastic programming method and a sample average approximation technique to tackle the uncertainties of bandwidth demand of virtual link when embedding virtual networks. Xie *et al.* [23] proposed a rolling horizon approach for tackling a stochastic programming model used to represent the dynamism of network virtualization in a cloud environment. Marotta *et al.* [10] proposed a multi-step heuristic algorithm for solving complex robust optimization models that represent the uncertainties of VNFs' resource demands. These works utilized the stochastic or robust optimization methods to tackle different uncertainties, e.g., traffic and user requests in network slices [21], traffic in virtual networks [22], VNFs' resource demands [10], and the dynamism of network virtualization in a cloud environment [23]. However, they have not consider the uncertainties introduced from flows that visit VNFs with different probability distributions. In this paper, we utilize the expected value of the random variables to work towards the stochastic optimization problem of placing VNFs, i.e., the uncertainty of data traffic arrival at VNFs.

There are a number of studies that adopt stochastic optimization approaches to place SDN controllers into software-defined wireless networks [31] and software-defined cellular networks [32]. Regarding the stochastic delays of wireless links in software-defined wireless networks, Abdel-Rahman *et al.* [31] proposed a chance-constrained stochastic programming scheme to jointly optimize the placement and assignment of SDN controllers. In consideration of the uncertainty of user locations in software-defined cellular networks, Abdel-Rahman *et al.* [32] also utilized stochastic programming to minimize the number of SDN controllers and the response time to evolved Node-Bs (eNBs). All aforementioned stochastic programming approaches use the expected values to convert the models of stochastic optimization into deterministic programs for solving the stochastic controller placement problem. However, the differences between the stochastic optimization problems of VNF placement and SDN controller placement make the algorithms proposed in the above studies cannot be directly applied in the VNFPRAT problem. In particular, there are no sequence dependencies between controllers in SDN controller placement problems. In our work, the sequence dependencies of VNFs are totally specified by service function chains.

To tackle the uncertainty problem, the robust optimization and stochastic programming methods in above works are proposed from the perspectives of the worst performance and probability distribution, respectively. However, the robust optimization approaches pay more attention to the worst performance rather than the average performance. The conservative solutions obtained by robust methods may increase total delay, i.e., servers need to make room for potential resource demand of VNFs that have lower average placing probabilities, and it may prolong the paths that flows visit VNFs in most cases. The stochastic optimization approaches handle the flow uncertainty by presenting a more complex model based on a known probability distribution. However, for many realistic environments, the probability distributions of flows' arrival at VNFs are not given or inaccurate. In this paper, we consider the randomness of flows' arrival by adequately modelling the uncertainties, e.g., a binary matrix for expressing a flow through its SFC, the probability of placing a VNF, and VNFs' expected resource demands. In addition, the expected value-based algorithms are proposed to suit different distributions that have the same mean arrival probability, which guarantees the performance of end-to-end delay in terms of average cases.

Existing VNF placement researches determined the VNF placement ignoring the link delay from a switch to a VNF connecting to it and the stochastic manner of flows visiting VNFs. To solve this problem, we presented a 0-1 programming model and the expected-value-based heuristic algorithms to optimize the VNF placements. The robust optimization methods proposed in above references also inspire the improvement of our work. In our future work, we will further improve

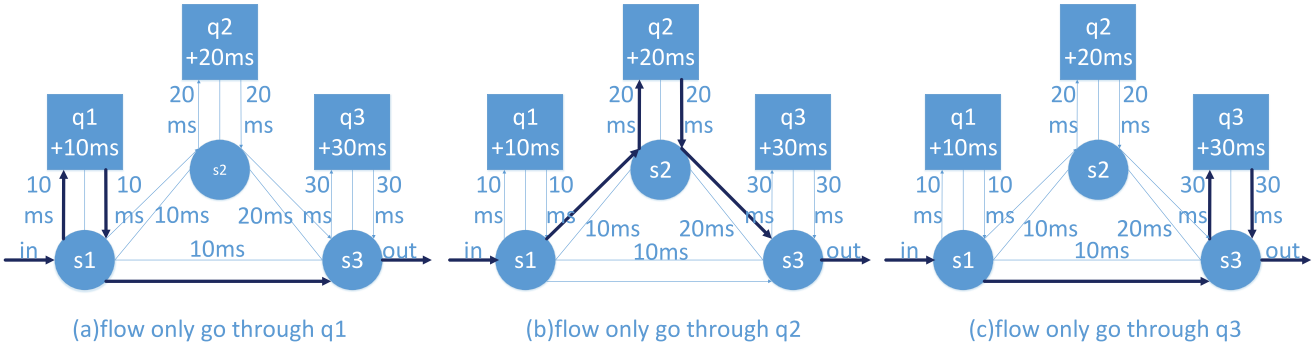


Fig. 1. Different arrivals in the VNFPRAT problem when the placement is given.

the tolerance of the traffic uncertainty based on a multi-stage robust optimization algorithm. In the first stage, we will use the expected-value-based algorithms to obtain the initial VNF placement. In the second stage, we will migrate a few VNFs away from servers that have a higher probability of resource contention. In the third stage, a few flows will be migrated away from the links that may have a potential congestion. This phased optimization method is straightforward and may achieve more robust solutions on stochastic VNF placement problems. Compared with other joint optimization approaches, a multi-stage heuristic algorithm is straightforward and may have lower computational complexity.

III. System Model and Problem Formulation

A. System Model

We consider a network comprising of SDN controller, openflow switches, softwarized network functions, and standard servers. A centralized controller program the openflow switches to route traffic. The virtual network functions run on VMs or containers to share the server resources. Based on the traffic flows and the requested service chains, the network operator carefully designs the placement schemes and deploying VNFs in servers. Then, the flow entries are proactively installed into openflow switches to guide traffic through VNFs. With this, the VNFs are chained as SFCs.

The information of the substrate network is given as the inputs of the VNFPRAT problem. We denote the underlying network by $\mathcal{G} = (\mathcal{S}, \mathcal{E})$, where $\mathcal{E}$ represents the set of links, and $\mathcal{S}$ stands for the set of switches. c_s denotes resources capacity of each switch $s \in \mathcal{S}$ for deploying VNFs. The resources refer to available CPU capacity or memory capacity of servers. If $(s_{l_1}, s_{l_2}) \in \mathcal{E}$, $d^l_{s_{l_1}, s_{l_2}}$ denotes link delay between s_{l_1} and s_{l_2}. Further, the shortest path delay $d^r_{s_{l_3}, s_{l_4}}$ between sl3 and sl4 is obtained by summing all the link delays. For instance, if $ShortestPath(s_{l_3}, s_{l_4}) = \{(s_{l_3}, s_l), (s_l, s_{l_4})\}, (s_{l_3}, s_l), (s_l, s_{l_4}) \in \mathcal{E}$, then $d^r_{s_{l_3}, s_{l_4}} = d^l_{s_{l_3}, s_l} + d^l_{s_l, s_{l_4}}$.

For the flow information, we use $\mathcal{F}$ to denote the set of flow in the NFV network. Then, a flow f_k is expressed as $f_k = \{e_k, c_k, i_k, d_k\}$, where e_k and i_k denote the switch ingress and egress, respectively. c_k denotes its corresponding service chain, and d_k denotes the end-to-end delay. The service chain is represented by $c_k = \{m^1_k, m^2_k, \ldots, m^{n_k}_k\}$, where n_k designates the amount of VNFs in c_k and m^i_k is the i-th VNF type that f_k can traverse. Each VNF $m^i_k \in c_k$ is then represented as $m^i_k = \{s_{m^i_k}, h_{m^i_k}, p^t_{m^i_k}, p^d_{m^i_k}\}$, where $s_{m^i_k}$ represents the switch that m^i_k is connected to, $h_{m^i_k}$ indicates a server in which m^i_k is deployed, $p^t_{m^i_k}$ denotes the probability of f_k goes through m^i_k, and $p^d_{m^i_k}$ denotes the probability of deploying m^i_k. Further, $r(q_i)$ and $d^t_{q_i}$ represent resources required to deploy q_i and the message processing delay of q_i, respectively. The complete set $\mathcal{Q}$ represents the group of VNFs. The notations used in the problem formulation and system model are summarized in Table I.

With the inputs of network and flow information, we attempt to design placement schemes for minimizing the end-to-end delay. To show the effect of the probability factor on the end-to-end delay, Fig. 1 varies a flow's arrival to VNFs that have been given a placement. Consider a network of three nodes s_1, s_2, and, s_3. Each node is connected to a server. The servers host $\{q_1, q_2, q_3\}$, respectively. The flow of f start at s_1 and finish at s_3 that arrives in a random fashion at VNFs. If f traverses only q_1, as shown in Fig. 1a, the total end-to-end delay is 40 ms, in which 10ms is the link delay from s_1 to q_1, while 10ms is the processing delay of q_1, followed by 10ms link delay from q_1 to s_1, and 10ms link delay between s_1 and s_3. The probability of this event is $p^t_{q_1}(1-p^t_{q_2})(1-p^t_{q_3})$. However, if f traverses only q_2 or q_3, as shown in Fig. 1b and 1c, respectively, the total end-to-end delay will be up to 100ms, and the probability changes to $(1-p^t_{q_1})p^t_{q_2}(1-p^t_{q_3})$ and $(1-p^t_{q_1})(1-p^t_{q_2})p^t_{q_3}$. We use p_i to denote the probability of the i-th event and d_i to represent the end-to-end delay brought by the i-th event. For all possible scenarios N, the expected value of end-to-end delay of flows can be obtained by

$$\sum_{\forall i \in N} d_i p_i. \tag{1}$$

The probability of processed traffic flow at a particular VNF often depends on the network status and service policies, e.g., WRED randomly drops packets with probabilities when the network congestion happens [13]. Based on the dataset provided by [33], we estimate the probability of flow arrival at VNFs by dividing the number of flows visiting core servers by

TABLE I
SYMBOLS USED IN PROBLEM FORMULATION AND SYSTEM MODEL

Symbol	Description
	Sets and set objects
c_k	Service chain that f_k need to go through
e_k,i_k	Egress/ingress switch
$f \in \mathcal{F}$	The flow set in the NFV network
h_{q_i}	Server which host the VNF q_i
m_k^i	ith VNF of c_k
$q \in Q$	The VNF set that needs to be placed
$s \in S$	The switch set in the NFV network
s_{q_i}	Switch which has an attached server h_{q_i}
s_n^i, s_n^c, s_n^e	nth ingress, core, and egress switches, respectively, mentioned in proposition 2
	Variables
$I_{a,b}$	The symbol indicating whether a equals to b or not
u_{s_l,q_i}	The expected value of the number of flows that i_k is s_l, while firstly traverse VNF q_i
u_{q_i,s_l}	The expected value of the number of flows that e_k is s_l, while lastly traverse VNF q_i
u_{q_i,q_j}	The expected value of the number of flows that immediately traverses q_j after traversing q_i
u_{q_i}	The expected value of the number of flows traversing through q_i
x_{q_i,s_l}	Binary variable indicating whether q_i is placed on s_l
y_{f_k,m_k^i}	Binary decision variables to indicate whether the flow f_k arrives m_k^i or not
y^k	0-1 vectors to indicate whether f_k arrives at each VNF m_k^i in c_k or not
y_h^k	The equivalent 0-1 vector of the decimal number h
	Parameters
c_{s_l}	Switch resource capacity
d_k	The expected value of the end-to-end delay of f_k in case of all possible values of y^k
d_h^k	The end-to-end delay of f_k in case of y_h^k
$d_{s_{l_1},s_{l_2}}^l$	Link delay between s_{l_1} and s_{l_2}
$d_{s_{l_1},s_{l_2}}^r$	Shortest path delay of from s_{l_1} to s_{l_2}
$d_{q_i}^t$	Message processing delay of q_i
n_k	Number of VNFs in c_k
$p_{m_k^i}^t$	The probability of f_k arrives to m_k^i
$p_{q_i}^d$	The probability of deploying q_i
r_{s_l}	The remaining resources in switch s_l
r_{q_i}	The required resource for deploying VNF q_i
	Fuctions
D^t	Expected value of end-to-end delay for all flows
$f_{q_i,s_l}^o, f_{q_i,s_l}^e$	The costs of placing q_i in s_l, which can be obtained from the functions in the greedy algorithm
$g(q_i), r(q_i)$	the output placements obtained from the greedy and simulated annealing algorithm, respectively

the total number of flows generated by edge devices. Besides, we only count the cases that traffic is processed by virtualized network services, i.e., the cases that none of flows visits core servers are ignored. Then, for this real datacenter network, the obtained average probabilities of flows' arrivals range from 10% to 100%. Therefore, in the rest of this work, we make the parameter settings of flows' arrival probabilities fall into this range. Predicting the probabilities based on traffic matrices and traffic demands in network will be in our future work.

To illustrate the importance of VNF placement to randomly arrived traffic, we consider a scenario as shown in Fig. 2, where there are three VNFs {q1, q2, q3} need to be deployed. To focus on the impact of uncertainty on network performance, we pre-generate the probability of traffic flow arrival at each VNF in this paper. We make the probabilities of *Flow1* through q1 and q2 take value 50%. Probabilities of *Flow2* through q1 and q3 are 40% and 70%, respectively. We assume that events of *Flow1* and *Flow2* arrive at VNFs are independent, i.e., whether *Flow1* through VNFs or not, it has no consequence on *Flow2* traversing them, and vice versa. In the remaining paper, we assume that flows go through VNFs independently. Moreover, the Fig. 2 depicts that the network contains 6 switches. Here, we present two feasible placement schemes: Case A places {q1, q2, and q3} at {S3, S4, and S5}, respectively. Case B separately deploys {q1, q2, and q3} at {S4, S2, and S3}. According to Eqn. (1), the expected end-to-end delay of cases A and B are 96.1ms and 114.92ms, respectively. Results illustrate that the scheme B is worse than the scheme A, which means that service chaining performance is affected by VNF placement scheme. Thus, it is essential to study VNF placement problem with randomly arrived traffic.

B. Problem Formulation

1) Decision Variables: To represent VNF placement scheme, we define binary decision variables x_{q_i,s_l}, where $x_{q_i,s_l} = 1$ if the server hosting q_i is linked to switch s_l, otherwise, $x_{q_i,s_l} = 0$. Hence, x_{q_i,s_l} set a constraint given as follows:

$$\sum_{\forall s_l \in S} x_{q_i,s_l} = 1, \forall q_i \in Q, \tag{2}$$

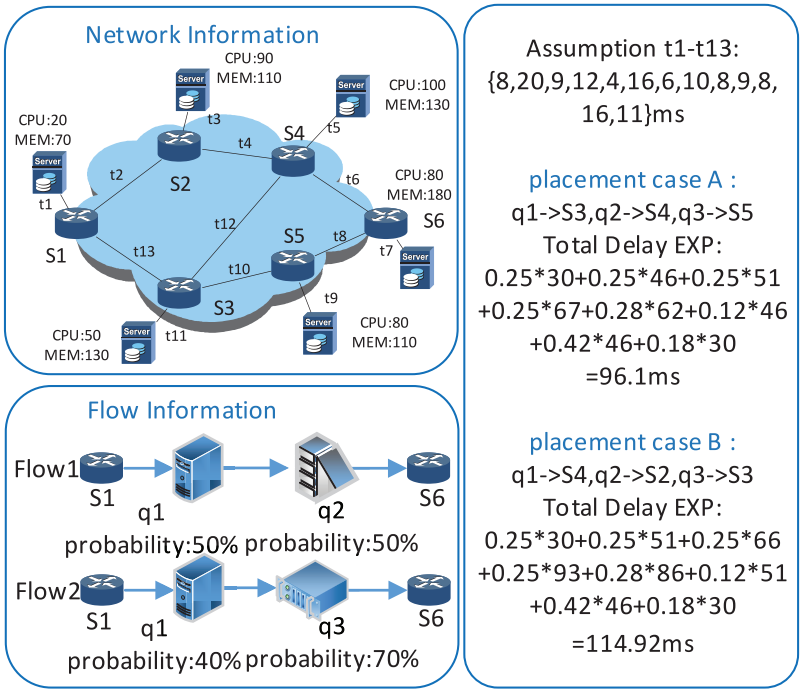


Fig. 2. VNF placement with randomly arriving traffic.

which makes sure that each VNF is deployed in only one place.

2) System Constraints: Any given VNF $q \in Q$ is essential only if a flow $f \in F$ arrives to q. We assume that each flow $f \in F$ traversing through a VNF q is independent, and in each SFC, VNFs are visited independently. As for the flows $\{f_k \in F : q_i \notin c_k\}$, the probability of unnecessary deployment of q_i is clearly 100%. The flow values $\{f_k \in F : q_i \in c_k\}$ is obtained by multiplying the unvisited probabilities of all items, i.e., $\prod_{m_k^\gamma \in c_k, I_{m_k^\gamma, q_i}=1} (1 - p^t_{m_k^\gamma})$. Since the probability of unnecessary deployment of q_i is $\prod_{\forall f_k \in F} \prod_{m_k^\gamma \in c_k, I_{m_k^\gamma, q_i}=1} (1 - p^t_{m_k^\gamma})$, thus the probability of necessary deployment of q_i can be calculated as:

$$p^d_{q_i} = 1 - \prod_{\forall f_k \in F} \prod_{m_k^\gamma \in c_k, I_{m_k^\gamma, q_i}=1} \left(1 - p^t_{m_k^\gamma}\right). \tag{3}$$

The connectivity of q_i to s_l does not depend on connecting another VNF to s_l. Thus, we calculate the expected value of total required resources deploy VNFs at $\sum_{\forall q_h \in Q} p^d_{q_h} r_{q_h} x_{q_h, s_l}$. Eq. 4 guarantees that the resource capacity is bigger than the total resource requirement, i.e.,

$$\sum_{\forall q_h \in Q} p^d_{q_h} r_{q_h} x_{q_h, s_l} \leq C(s_l), \forall s_l \in S. \tag{4}$$

VNF can be deployed at a location that meets the above constraint in SDN network. Whereas, owing to physical reasons, e.g., geographical locations of terminals, available power supplies, and resource capacity of servers. Many VNFs can only be deployed on several specified locations in realistic scenarios. To take these scenarios into consideration, we use s_{q_i} to represent the location set, which can carry the VNF q_i. Undeployment of q_i is defined as:

$$x_{i,l} = 0, \forall q_i \in Q, \forall s_l \in S \backslash S_{q_i}. \tag{5}$$

Since the performance of network applications is reflected by the end-to-end delay, it is essential to minimize this delay. Therefore, we explore the delay index in our work.

3) Objective Functions: The traffic flows traverse VNFs with certain probabilities in realistic scenarios. To deal with a random arrival of traffic flows, we evaluate the expected value of end-to-end delay by enumerating all possible cases of traversing VNFs. We define binary variables y_{f_k, m_k^i} to indicate whether the flow f_k arrives at m_k^i or not, where $y_{f_k, m_k^i} = 1$ denotes that f_k arrives m_k^i, otherwise, $y_{f_k, m_k^i} = 0$. Then, a case of whether f_k arrives each VNF m_k^i in c_k or not is expressed as $y^k = \{y_{f_k, m_k^1}, y_{f_k, m_k^2}, \ldots, y_{f_k, m_k^i}\}$. For example, if $y^k = \{1, 0, \ldots, 0\}$, then it indicates that f_k only traverses m_k^1. To guarantee that each flow gets service, the case of flow f_k through none of the VNFs in c_k is ignored in this paper. All possible values of y^k are mathematically expressed as follows:

$$\begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix}. \tag{6}$$

By converting each binary row vector to its decimal number equivalent, we notice that these decimal numbers range from 1 to $2^n - 1$. Thus, we use y^k_h to denote the equivalent 0-1 vector of the decimal number h. The length of the vector y^k_h is n_k. Further, let $p_{y^k_h}$ represents the probability of the occurrence of the y_h case. To facilitate the calculation of $p_{y^k_h}$, we preprocess y^k_h. We use a vector b^k_h to record the index of all the values

of y_h^k that are equal to 1 in an ascending order. The length of b_h^k is equal to l_h^k. For example, if y_3={0,..., 0,1,1}, then $b_3^k = [n-1, n]$, $b_3^k[1] = [n-1]$, $b_3^k[2] = [n]$, and $l_3^k = 2$. The $p_{y_h^k}$ value can be obtained as follows:

$$p_{y_h^k} = \prod_{\forall i \in N, \forall i \notin b_h^k, 1 \le i \le n_k} \left(1 - p_{m_k^i}^t\right) \prod_{\forall j \in b_h^k} p_{m_k^j}^t, \tag{7}$$

where the range of the first multiplication is the complementary set of b_h^k. To simplify expressions in the following sections, we use i_h^k to denote the $b_h^k[i]$-th VNF in c_k. f_h^k and l_h^k are used to represent the $b_h^k[1]$-th and $b_h^k[l_h^k]$-th VNF in c_k, respectively. Besides, we use $\{s_{k,h}^f, s_{k,h}^i, s_{k,h}^l\}$ to designate the switches connected with $\{f_h^k, i_h^k, l_h^k\}$, respectively. While $\{h_{k,h}^f, h_{k,h}^i, h_{k,h}^l\}$ separately denote the servers that execute VNFs.

After obtaining the value of $p_{y_h^k}$, for y_h^k the total delay $d_{k_h}^s$ of f_k is evaluated. In general, the end-to-end delay of a flow consists of four parts: (1) the delay from i_k to $s_{m_k^1}$, $s_{m_k^{n_k}}$ to e_k, (2) between switches that attaches adjacent VNF pairs in c_k, (3) the sum of the processing delay $d_{q_i}^t$, (4) the round-trip time between the switch that q_i is connected, and the NFV server that hosts q_i. For example, the first part of the delay equals to $d_{(i_k, s_{k,h}^f)}^r$ in case of y_h^k. Then, we determine the expected value of the first part by enumerating all possible traversing cases y^k, i.e., $\sum_{h=1}^{2^{n_k}-1} p_{y_h^k} d_{(i_k, s_{k,h}^f)}^r$. After summing the expected value in every part, we obtain the expected value of the end-to-end delay d_k as follows:

$$\begin{aligned} d_k = & \sum_{h=1}^{2^{n_k}-1} p_{y_h^k} d_{i_k, s_{k,h}^f}^r \\ & + \sum_{h=1}^{2^{n_k}-1} p_{y_h^k} d_{s_{k,h}^l, e_k}^r \\ & + \sum_{h=1}^{2^{n_k}-1} \sum_{\gamma=1}^{l_h^k - 1} p_{y_h^k} d_{s_{k,h}^\gamma, s_{k,h}^{\gamma+1}}^r \\ & + \sum_{h=1}^{2^{n_k}-1} \sum_{\eta=1}^{l_h^k} p_{y_h^k} \left(d_{s_{k,h}^\eta}^t + 2 d_{s_{k,h}^\eta, h_{k,h}^\eta}^l \right). \end{aligned} \tag{8}$$

In Eq. (8), $2d_{s_{k,h}^\eta, h_{k,h}^\eta}^l$ is the round-trip time between $s_{k,h}^\eta$ and $h_{k,h}^\eta$. We obtain the expected value of the total end-to-end delay by summing the expected value of the end-to-end delay over all flows, represented as D^t, given below:

$$\begin{aligned} D^t = & \sum_{\forall f_k \in F} \sum_{h=1}^{2^{n_k}-1} p_{y_h^k} d_{i_k, s_{k,h}^f}^r \\ & + \sum_{\forall f_k \in F} \sum_{h=1}^{2^{n_k}-1} p_{y_h^k} d_{s_{k,h}^l, e_k}^r \\ & + \sum_{\forall f_k \in F} \sum_{h=1}^{2^{n_k}-1} \sum_{\gamma=1}^{l_h^k - 1} p_{y_h^k} d_{s_{k,h}^\gamma, s_{k,h}^{\gamma+1}}^r \\ & + \sum_{\forall f_k \in F} \sum_{h=1}^{2^{n_k}-1} \sum_{\eta=1}^{l_h^k} p_{y_h^k} \left(d_{s_{k,h}^\eta}^t + 2 d_{s_{k,h}^\eta, h_{k,h}^\eta}^l \right). \end{aligned} \tag{9}$$

Eq. (9) does not show the coefficients of decision variables intuitively. For the purpose of analysing $x_{m_k^i, s_l}$ and their products, we evaluate the expected value of the total end-to-end delay in terms of device pairing.

Before dwelling into detail, we introduce several definitions, such as u_{s_l, q_i} represents the expected number of flows that ingress switch s_l, where q_i is the first VNF of the service chain. Contrary to ingress, u_{q_i, s_l} represents the expected number of flows that egress switch s_l and the last traversed VNF is q_i. However, u_{q_i, q_j} is the expected value of the number of flows immediately traversing q_j after traversing q_i, u_{q_i} is the expected value of flows traversing q_i. These definitions are mathematically expressed as follows:

$$\begin{aligned} & \forall s_l \in S, \forall q_i \in Q, u_{s_l, q_i} = \sum_{\forall f_k \in F} \sum_{h=1}^{2^{n_k}-1} I_{i_k, s_l} I_{f_h^k, q_i} p_{y_h^k}, \\ & \forall s_l \in S, \forall q_i \in Q, u_{q_i, s_l} = \sum_{\forall f_k \in F} \sum_{h=1}^{2^{n_k}-1} I_{e_k, s_l} I_{l_h^k, q_i} p_{y_h^k}, \\ & \forall q_i, \forall q_j \in Q, u_{q_i, q_j} = \sum_{\forall f_k \in F} \sum_{h=1}^{2^{n_k}-1} \sum_{\gamma=1}^{l_1^h - 1} \\ & \quad \times I_{\gamma_h^k, q_i} I_{[\gamma+1]_h^k, q_j} p_{y_h^k}, \\ & \forall q_i \in Q, u_{q_i} = \sum_{\forall f_k \in F} \sum_{h=1}^{2^{n_k}-1} \sum_{\eta=1}^{l_1^h} I_{\eta_h^k, q_i} p_{y_h^k}, \end{aligned} \tag{10}$$

where $I_{a,b}$ is an indicative symbol, which represents when a is equal to b, then $I_{a,b} = 1$, otherwise, $I_{a,b} = 0$. For instance, the delay between the switch s_l and VNF q_i is equal to $d_{(s_l, s_{l_1})} r$, when q_i is placed at s_{l_1}. Under the constraint of (2), there is only one location of l_1 that makes $x_{q_i, s_{l_1}} = 1$. Thus, summing up $d_{(s_l, s_{l_1})}^r u_{s_l, q_i} x_{q_i, s_{l_1}}$ over all possible locations of l_1, we obtain the expression of the delay of this pair as $\sum_{\forall s_{l_1} \in S} d_{(s_l, s_{l_1})}^r u_{s_l, q_i} x_{q_i, s_{l_1}}$. After performing the summation of each pair delay, the expected value of total delay is evaluated. The following Eq. (11) consists of coefficients of decision variables and their products.

$$\begin{aligned} D^t = & \sum_{\forall q_i \in Q} \sum_{\forall s_l \in S} \sum_{\forall s_{l_1} \in S} d_{(s_l, s_{l_1})}^r u_{s_l, q_i} x_{q_i, s_{l_1}} \\ & + \sum_{\forall q_i \in Q} \sum_{\forall s_l \in S} \sum_{\forall s_{l_1} \in S} d_{(s_{l_1}, s_l)}^r u_{q_i, s_l} x_{q_i, s_{l_1}} \\ & + \sum_{\forall q_i \in Q} \sum_{\forall q_j \in Q} \sum_{\forall s_{l_1} \in S} \sum_{\forall s_{l_2} \in S} d_{(s_{l_1}, s_{l_2})}^r u_{q_i, q_j} x_{q_i, s_{l_1}} x_{q_j, s_{l_2}} \\ & + \sum_{\forall q_i \in Q} \sum_{\forall s_l \in S} \left(d_{q_i}^t + 2 d_{s_l, h_{q_i}}^l \right) u_{q_i} x_{q_i, s_l}. \end{aligned} \tag{11}$$

An intuitive interpretation of (9) and (11) is that (11) simplifies (9) by merging the same variables, as presented in the following Proposition 1.

Proposition 1: The expected value of total delay obtained from Eqn. (11) is equal to the expected value of total delay obtained from Eqn. (9).

Proof 1: The proof is by induction on the number of cases that flows' arrivals at each VNF, i.e., the number of vectors in the front of Eqn. (6) that uses a binary matrix to describe the cases of whether a flow arrives each VNF in its SFC. For simplicity, we just give the proving procedure of the equality of the first item in (9) and (11). This procedure can be easily extended to prove that Eqn. (9) equals to Eqn. (11).

With the case described by the first one vector in the matrix (6), the first item of Eqn. (9) can be expressed as $\sum_{\forall f_k \in F} p_{y_1^k} d^r_{i_k, s^f_{k,1}}$. The first item of Eqn. (11) can be expressed as $\sum_{\forall q_i \in Q} \sum_{\forall s_l \in S} \sum_{\forall s_{l_1} \in S} d^r_{s_l, s_{l_1}} (\sum_{\forall f_k \in F} I_{i_k, s_l} I_{f_1^k, q_i} p_{y_1^k}) x_{q_i, s_{l_1}}$, and the equality of two expressions can be proved by moving the fourth summation to the front.

We assume that the first item in (9) equals to the first item in (11) with the cases described by the first m vectors in the matrix (6). When flows' arrival cases match the description of the first $m+1$ vectors in the matrix (6), the expression of the first item in (11) can be obtained from that in (9) through the following transformation:

$$
\begin{aligned}
&\sum_{\forall f_k \in F} \sum_{h=1}^{m+1} p_{y_h^k} d^r_{i_k, s^f_{k,h}} \\
&= \sum_{\forall f_k \in F} \left(\sum_{h=1}^{m} p_{y_h^k} d^r_{i_k, s^f_{k,h}} + \sum_{h=m+1}^{m+1} p_{y_h^k} d^r_{i_k, s^f_{k,h}} \right) \\
&= \sum_{\forall f_k \in F} \sum_{h=1}^{m} p_{y_h^k} d^r_{i_k, s^f_{k,h}} + \sum_{\forall f_k \in F} \sum_{h=m+1}^{m+1} p_{y_h^k} d^r_{i_k, s^f_{k,h}} \\
&= \sum_{\forall f_k \in F} \sum_{h=1}^{m} p_{y_h^k} d^r_{i_k, s^f_{k,h}} \\
&\quad + \sum_{\forall f_k \in F} p_{y^k_{m+1}} \left(\sum_{\forall q_i \in Q} \sum_{\forall s_l \in S} d^r_{s_l, s_{q_i}} I_{i_k, s_l} I_{f^k_{m+1}, q_i} \right) \\
&= \sum_{\forall f_k \in F} \sum_{h=1}^{m} p_{y_h^k} d^r_{i_k, s^f_{k,h}} \\
&\quad + \sum_{\forall f_k \in F} p_{y^k_{m+1}} \left(\sum_{\forall q_i \in Q} \sum_{\forall s_l \in S} d^r_{s_l, s_{q_i}} I_{i_k, s_l} I_{f^k_{m+1}, q_i} \right) \\
&= \sum_{\forall f_k \in F} \sum_{h=1}^{m} p_{y_h^k} d^r_{i_k, s^f_{k,h}} \\
&\quad + \sum_{\forall f_k \in F} p_{y^k_{m+1}} \left(\sum_{\forall q_i \in Q} \sum_{\forall s_l \in S} \sum_{\forall s_{l_1} \in S} d^r_{s_l, s_{l_1}} \right. \\
&\qquad \left. \times I_{i_k, s_l} I_{f^k_{m+1}, q_i} x_{q_i, s_{l_1}} \right) \\
&= \sum_{\forall q_i \in Q} \sum_{\forall s_l \in S} \sum_{\forall s_{l_1} \in S} d^r_{s_l, s_{l_1}} \\
&\quad \times \left(\sum_{\forall f_k \in F} \sum_{h=1}^{m} I_{i_k, s_l} I_{f_h^k, q_i} p_{y_h^k} \right) x_{q_i, s_{l_1}} \\
&\quad + \sum_{\forall q_i \in Q} \sum_{\forall s_l \in S} \sum_{\forall s_{l_1} \in S} d^r_{s_l, s_{l_1}} \\
&\quad \times \left(\sum_{\forall f_k \in F} I_{i_k, s_l} I_{f^k_{m+1}, q_i} p_{y^k_{m+1}} \right) x_{q_i, s_{l_1}} \\
&= \sum_{\forall q_i \in Q} \sum_{\forall s_l \in S} \sum_{\forall s_{l_1} \in S} d^r_{s_l, s_{l_1}} \\
&\quad \times \left(\sum_{\forall f_k \in F} \sum_{h=1}^{m+1} I_{i_k, s_l} I_{f_h^k, q_i} p_{y_h^k} \right) x_{q_i, s_{l_1}}. \qquad (12)
\end{aligned}
$$

In the above transformations, we use an alternate expression to replace $d^r_{i_k, s^f_{k,h}}$. This technique can also be utilized to obtain the equality of the other items of (9) and (11). Since the expected value of the total delay is equal to the sum of the four items, we declare that the expected value of the total end-to-end delay achieved from (11) is equal to which achieved from (9) with the cases described by the first $m+1$ vectors in the matrix (6). By the induction hypothesis, this conclusion is right for any given number of cases of flows' arrivals that includes $2^{n_k} - 1$ cases, i.e., $\sum_{\forall f_k \in F} \sum_{h=1}^{2^{n_k}-1} p_{y_h^k} d^r_{i_k, s^f_{k,h}}$ equals to $\sum_{\forall q_i \in Q} \sum_{\forall s_l \in S} \sum_{\forall s_{l_1} \in S} d^r_{s_l, s_{l_1}} u_{s_l, q_i} x_{q_i, s_{l_1}}$. Therefore, this proposition that Eqn. (9) equals to Eqn. (11) is proved.

Our objective is to minimize the expected total delay and formally define the VNFPRAT problem accordingly as:

$$
\begin{aligned}
&min \;\; D^t \quad according \quad to \quad (11) \\
&s.t. \;\; (2),\ (4),\ (5). \qquad (13)
\end{aligned}
$$

By reducing a well-known NP-hard problem to the VNFPRAT problem, we have proved that our formulated VNFPRAT problem is NP-hard.

Proposition 2: The virtual network function placement problem of (13) is an NP-hard problem.

Proof 2: The basis of our proof is that if an NP-hard problem can be reduced to a problem $\mathcal{P}$, then $\mathcal{P}$ is also NP-hard [34]. Therefore, we try to reduce a classical NP-hard problem, i.e., the quadratic assignment problem (QAP) [24], to our formulated problem. QAP minimizes the total cost of deploying n facilities at n locations. This is expressed as:

$$
\begin{aligned}
&min \quad \sum_{l_1, l_2 = 1}^{n} \sum_{\alpha, \beta = 1}^{n} c_{l_1, l_2} f_{\alpha, \beta} x_{\alpha, l_1} x_{\beta, l_2} + \sum_{l_1, \alpha = 1}^{n} b_{\alpha, l_1} x_{\alpha, l_1} \\
&s.t. \quad \sum_{l=1}^{n} x_{\alpha, l} = 1, \qquad \alpha = \{1, 2, \ldots, n\}; \\
&\qquad \sum_{\alpha=1}^{n} x_{\alpha, l} = 1, \qquad l = \{1, 2, \ldots, n\}; \\
&\qquad x_{\alpha, l} \in \{0, 1\}, \qquad \alpha, l = \{1, 2, \ldots, n\}, \qquad (14)
\end{aligned}
$$

where c_{l_1, l_2} represents the distance from location l_1 to location l_2, $f_{\alpha, \beta}$ represents a flow between facility α and facility β, and

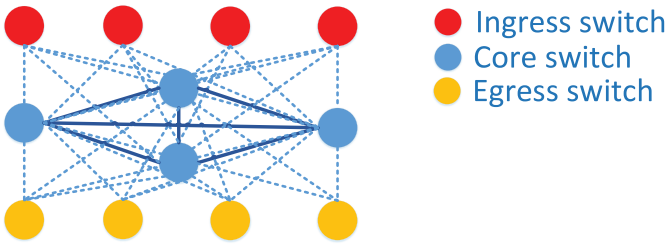


Fig. 3. This topology is used for simplifying the QAP problem to the VNFPRAT problem.

b_{α,l_1} indicates the cost of deploying facility α at the location l_1. x_{α,l_1} is a decision variable, if $x_{\alpha,l_1}=1$, then the facility is deployed at location l_1, otherwise $x_{\alpha,l_1}=0$.

To reduce QAP (14) to the VNFPRAT problem (13), we build a network, in which the VNFPRAT problem has the same objective function and constraints as QAP's. We consider to place n VNFs to a constructed network. Switch set includes n ingress, n core, and n egress-switches, and they are represented as $s_1^i, s_2^i, \ldots, s_n^i$; $s_1^c, s_2^c, \ldots, s_n^c$, and $s_1^e, s_2^e, \ldots, s_n^e$, respectively. These devices are linked as: each core switch connects to each ingress switch, each egress connects to each core switch, and core switches connect with each other, i.e., $E=\{(s_i^i,s_j^c)\}\cup\{(s_i^c,s_j^c)\}\cup\{(s_i^c,s_j^e)\}$. For better illustration, Fig. 3 shows the established topology when $n=4$.

Based on the established topology information, we make the resource requests for VNFs $r(q_i)=1, \forall 1\le i\le n$ and resource capacities of switches $\mathrm{C}\ (s_l^i)=0$, $\mathrm{C}\ (s_l^c)=1$, $\mathrm{C}\ (s_l^e)=0$, $\forall 1\le l\le n$. The resource settings entail that each core switch can host at most one VNF. Besides, for each VNF q_i, at least one of the arrival probabilities of the processed flows is made to take value 100%, i.e., $\exists p^t_{m_k^j} I_{m_k^j,q_i}=1, \forall m_k^j\in c_k, \forall f_k\in F, \forall q_i\in Q$. The specified arrival probabilities make the placing probabilities of VNFs equal to one. The probability and resource settings make each VNF be placed at exactly one core switch. We select i_k from ingress switches, e_k from egress switches, and carefully design each service chain and determine the probabilities of flow arrival at each VNF making $u_{q_\alpha,q_\beta}=f_{\alpha,\beta}$. Lastly, we make $d^r_{(s^c_{l_1},s^c_{l_2})}=c_{l_1,l_2}$, $d^r_{(s^i_l,s^c_{l_1})}=\frac{b_{\alpha,l_1}}{4\sum_{l=1}^n u_{s_l,q_i}}$, $d^r_{(s^c_{l_1},s^i_l)}=\frac{b_{\alpha,l_1}}{4\sum_{l=1}^n u_{q_i,s_l}}$, $d^t_{q_\alpha}=\frac{b_{\alpha,l_1}}{4u_{q_\alpha}}$, $d^l_{s^c_l,h_{q_i}}=\frac{b_{\alpha,l_1}}{8u_{q_\alpha}}$.

Sufficiency: Considering the above constructed inputs, the VNFPRAT problem (13) have the same formulation with the QAP problem (14). Therefore, if the assignment scheme of the QAP problem exists, the VNFPRAT problem has the same feasible solutions. The expected end-to-end delay of feasible solutions in (13) equals to the corresponding value of objective function of QAP in (14). Therefore, the optimal solutions of the QAP problem are also the optimal solutions of the VNFPRAT problem.

Necessity: On the contrary, x_{q_i,s_l} equals to 1 when q_i is assigned to s_l, otherwise, it equals to 0. $\sum_{k=1}^n x_{i,k}=1$ guarantees that q_i is connected to exactly one switch. Therefore, when the solutions satisfy the constraints in the VNFPRAT problem (13), they are also the feasible solutions of the QAP problem (14).

The above constructed topology reduces the QAP problem to the VNFPRAT problem. Besides, the QAP problem is known as NP-hard problem. According to the complexity theory [34], the VNFPRAT problem is also an NP hard problem. Therefore, in the subsequent section, we propose two heuristic algorithms to solve VNFPRAT problem.

IV. Proposed Algorithms

For small scale problems, we design an optimal algorithm. Since large scale problems are NP-hard, we propose two heuristic algorithms for such problems. These proposed algorithms are based on the greedy strategy and simulated annealing, to achieve near optimal solutions for large scale problems.

A. Exact Optimization Approach

In the VNFPRAT problem, although the constraints are linear, there is a second-order item in the objective function according to (11). In addition, the variables $x_{m_k^i,s_l}$ are binary, so the VNFPRAT problem in Eq. (13) is a 0-1 quadratic programming problem. In order to solve the problem optimally with a linear programming solver, it needs to linearize the second order items in the objective function. The linearization technique of the product of binary decision variables is to introduce a variable to replace nonlinear items. The constraints of the substitution variable are determined by the products of the upper/lower bounds of the decision variables. Thus, we use $y_{i,j,l_1,l_2}=x_{i,l_1}x_{j,l_2}$ to replace $x_{i,l_1}x_{j,l_2}$. Because both x_{i,l_1} and x_{j,l_2} are binary values, the constraints of y_{i,j,l_1,l_2} are obtained as follows:

$$\begin{cases} y_{i,j,l_1,l_2}\le x_{i,l_1}; \\ y_{i,j,l_1,l_2}\le x_{j,l_2}; \\ y_{i,j,l_1,l_2}\ge x_{i,l_1}+x_{j,l_2}-1; \\ y_{i,j,l_1,l_2}\in\{0,1\}. \end{cases} \tag{15}$$

We replace the variable $x_{i,l_1}x_{j,l_2}$ with y_{i,j,l_1,l_2}, and add the constraints of variables y_{i,j,l_1,l_2} to Eq. (13). We utilized the open source linear programming solvers, i.e., Gurobi [35], to solve the problem.

B. Heuristic Solution Approaches

Next, we propose the heuristic algorithms to efficiently find a solution for the VNFPRAT problem.

1) Greedy Algorithm: The basic idea of the greedy algorithm is to determine the location of each VNF iteratively following a greedy selection approach. For the VNF deployment, we need to consider the dependency relationships between VNF pairs. Our strategy is to perform VNF deployment in a manner to minimize delay to other switches and VNFs. Furthermore, few VNFs have high flow traversing probabilities, therefore, they are set at high priority for location selection.

With above considerations, we have designed a greedy algorithm as shown in Algorithm 1. The algorithm consists of two steps. The first step is to sort the VNFs in a descending order based on deployment probability. According to (3), a higher

Algorithm 1: The Algorithm Based on Greedy Strategy

Input: VNF Set $\mathcal{Q}$; Flow set $\mathcal{F}$; Switches set $\mathcal{S}$; Path delay $d^r_{s_{l_1},s_{l_2}}$;
Output: Placement scheme $g(q_j)$, $\forall q_j \in Q$

```
//Initialization;
g(q_j) = -1, ∀q_i ∈ Q; O = φ;
//step1:VNF sort;
for each q_i ∈ Q, s_l ∈ S do
    calculate p^d_{q_i}
sort q_i ∈ Q in descending order according to p^d_{q_i}
//step2:Iterative placement;
for each q_i ∈ Q, s_l ∈ S do
    score_min = inf, id_min = -1;
    for each s_l ∈ S do
        calculate the total resource demand in s_l;
        if s_l satisfies resource constraint (4) then
            calculating score(s_l, q_i) according to (16);
            if incorporating unplaced VNFs then
                calculating score(s_l, q_i) according to (17);
            if score(s_l, q_i) < score_min then
                score_min = score(s_l, q_i); id_min = l;
    if id_min ≠ -1 then
        g(q_i) = id_min; Q = Q - {q_i}; O = O + {q_i}
return g(q_j)
```

possibility of deployment means that the VNF has a higher flow traversing probability.

Then, the VNFs are deployed iteratively in a sorted order. We define the cost score f_{q_i,s_l} as the increment of the expected total delay when q_i is placed at s_l. Then, cost scores are calculated for those switches satisfying the resource constraints. The cost of placing q_i in s_l is obtained as:

$$\begin{aligned} f^o_{q_i,s_l} = & \sum_{\forall s_{l_1} \in S} d^r_{s_{l_1},s_l} u_{s_{l_1},q_i} + d^r_{s_l,s_{l_1}} u_{q_i,s_{l_1}} \\ & + \sum_{\forall q_j \in O} \left(d^r_{s_l,g(q_j)} u_{q_i,q_j} + d^r_{g(q_j),s_l} u_{q_j,q_i} \right) \\ & + \left(d^t_{q_i} + 2 d^l_{s_l,h_{q_i}} \right) u_{q_i}. \end{aligned} \tag{16}$$

Some VNFs may not have been placed when deploying q_i. To incorporate the impact of the undeployed VNFs, a weighted delay is used as an approximation of the expected value of the total end-to-end delay between unplaced and placed VNFs. Then, we add approximations to switches' cost score. Considering the unplaced VNFs, the cost of placing q_i in s_l is changed to:

$$\begin{aligned} f^e_{q_i,s_l} = & f^o_{q_i,s_l} + \frac{\sum_{\forall s_{l_1} \in S} d^r_{s_l,s_{l_1}} r_{s_{l_1}}}{\sum_{\forall s_{l_1} \in S} r_{s_{l_1}}} \sum\nolimits_{\forall q_j \in Q-O} u_{q_i,q_j} \\ & + \frac{\sum_{\forall q_j \in Q} \left(d^t_{q_j} + 2 d^l_{s_l,h_{q_j}} \right) r(q_j)}{\sum_{\forall q_j \in Q} r(q_j)} \sum_{\forall q_j \in Q-O} u_{q_j}, \end{aligned} \tag{17}$$

where $r_{s_{l_1}}$ denotes the residual resources of the switch s_{l_1} and $\mathcal{O}$ designates the set of VNFs and their locations are determined before q_i.

There are $|\mathcal{Q}|$ iterations in the greedy algorithm, and $|\mathcal{S}|$ switches' scores are calculated in each iteration. We determine the total complexity of the proposed algorithm by summing up times of counting scores. The times of counting switches' score is $|\mathcal{Q}||\mathcal{S}|$, and the complexity of counting each score is $|\mathcal{Q}|+|\mathcal{S}|$. The total complexity becomes $O(|\mathcal{Q}||\mathcal{S}|(|\mathcal{Q}|+|\mathcal{S}|))$.

2) Simulated Annealing Algorithm: Simulated annealing algorithm is a general-purpose algorithm to find the near optimal solution in a large search space. We use the simulated annealing algorithm to solve the VNFPRAT problem. We carefully customize settings, such as neighbor generation and temperature, which can greatly influence the performance of our algorithm. Next, we introduce the work flow and specific setting details.

As shown in Algorithm 2, the algorithm requires an initial placement *r*. The output of the greedy algorithm is used as the initial placement scheme. Then, we generate a new neighbor placement r', count the total delay change that is denoted by $\Delta d^t = d^t(r') - d^t(r)$, and perform search placement in an iterative manner. The probability that the algorithm updates the placement *r* to the neighbor placement r' is defined as follow:

$$p(\Delta d^t) = \begin{cases} 1, & if\ \Delta d^t < 0; \\ e^{-\frac{\Delta d^t}{t}}, & else, \end{cases} \tag{18}$$

Algorithm 2: The Algorithm Based on Simulated Annealing

Input: VNF set $\mathcal{Q}$; Flow set $\mathcal{F}$; Switches set $\mathcal{S}$; Path delay $d^r_{s_{l_1},s_{l_2}}$; Output of greedy algorithm $g(q_j)$

Output: VNF placement scheme $r(q_j)$, $\forall q_j \in Q$

```
//Initialization;
r = g; d_c^t = d^t(r); d_min^t = d_c^t; r_min = r; t = t_0; L = 2|Q||S|; IteNum = L|Q|; i=0;
while i < ItrNum do
    r' = new generated neighbours of r;
    count Δd^t; p(Δd^t);
    if random(0,1) < p(Δd^t) then
        r = r'; d_c^t = d_c^t + Δd^t;
        if d_c^t < d_min^t then
            d_min^t = d_c^t;
    i++;
    if mod(i, L) == 0 then
        t = the updated temperature
return placement scheme r;
```

TABLE II
SIMULATION SETTINGS

Parameter	Value Settings
link delay $d^l_{(s_i,s_j)}$ (ms)	[10, 25]
link delay $d^l_{(s_{q_i},h_{q_i})}$ (ms)	[4, 6, 8, 10]
VNF resource demand	1
available switch resource	[1, 2]
number of VNFs in each flow	[2, 3, 4, 5]
number of VNFs	[8, 11, 20]
Service chain number	[3, 10]
probability of flow go through VNF	random

where t represents the current temperature. Eq. (18) shows that if $\Delta d^t > 0$, it allows a search process to move to worse solutions, which guarantees that the algorithm would not fall into a local minima. Finally, it updates temperatures and moves to the next search process. The search repeats with a predefined iteration number. The algorithm finally returns placement schemes with the minimum expected value of the total delay.

a) Neighbor functions: We design two neighbor generating functions: (1) randomly select two VNFs and swap their connected switches; (2) randomly select a VNF and connect it to a new randomly selected switch. For both functions, the generated neighbor placement should satisfy resource constraints. In each iteration, the probability of selecting the two functions is equal.

b) Calculate the expected change of delay: Noting that many VNFs have the same placements after neighbor generation, the delay change denoted as Δd^t only exists in VNFs chosen to be replaced. Assuming that the first neighbours generation chooses q_j and q_i to exchange their switches, we calculate Δd^t as:

$$\begin{aligned}\Delta d^t = &\sum_{\forall s_l \in S}\left(d^r_{l,r_{q_j}} - d^r_{l.r_{q_i}}\right)\left(u_{s_l,q_i} - u_{s_l,q_j}\right)\\ &+\sum_{\forall s_l \in S}\left(d^r_{r_{q_j},l} - d^r_{r_{q_i},l}\right)\left(u_{q_i,s_l} - u_{q_j,s_l}\right)\\ &+\sum_{\forall q_g \in Q}\left(d^r_{r_{q_g},r_{q_j}} - d^r_{r_{q_g},r_{q_i}}\right)\left(u_{q_g,q_i} - u_{q_g,q_j}\right)\\ &+\sum_{\forall q_g \in Q}\left(d^r_{r_{q_j},r_{q_g}} - d^r_{r_{q_i},r_{q_g}}\right)\left(u_{q_i,q_g} - u_{q_j,q_g}\right). \quad (19)\end{aligned}$$

Eq. (19) provides delay expected change by using total neighboring link delay in new locations of q_i and q_j to minus that in their original location. For the second neighbor generation algorithm, we assume that the selected VNF is q_i, and the new location for deploying q_i is $r'(q_i)$. For the second neighbor generation algorithm, the change of the total delay can be calculated in a similar way:

$$\begin{aligned}\Delta d^t = &\sum_{\forall s_l \in S}\left(d^r_{l,r'(q_i)} - d^r_{l,r(q_i)}\right)u_{s_l,q_i}\\ &+\sum_{\forall s_l \in S}\left(d^r_{r'(q_i),l} - d^r_{r(q_i),l}\right)u_{q_i,s_l}\\ &+\sum_{\forall q_g \in Q}\left(d^r_{r(q_g),r'(q_i)} - d^r_{r(q_g),r(q_i)}\right)u_{q_g,q_i}\\ &+\sum_{\forall q_g \in Q}\left(d^r_{r'(q_i),r(q_g)} - d^r_{r(q_i),r(q_g)}\right)u_{q_i,q_g}. \quad (20)\end{aligned}$$

c) Temperature: As shown in (18), the temperature t is used to control the probability of moving to a worse placement. A high temperature is more likely to accept bad solutions [36]. Temperature t_0 is set and updated according to the techniques described in [36]. Specifically, we can obtain t_0 by calculating the mean Δd^t between initial placement scheme and its randomly generated neighbours: $t_0 = (1-\lambda)\Delta d^t_{avg} + \lambda \Delta d^t_{\min}$, where we set λ equal to 0.5. Temperature is updated by $t = \frac{t}{1+\beta t}$ with step of $2|Q||S|$.

In the above algorithm, the iteration number is $IteNum = 2|Q|^2|S|$ and each iteration's computational complexity is $O(|S| + |Q|)$. Therefore, the simulated annealing algorithmic total complexity is $O(|Q|^2|S|(|S| + |Q|))$.

TABLE III
CHARACTERISTICS OF NETWORK TOPOLOGIES

	Abilene	ARPANET	Fattree-4	Fattree-16
Node number	11	29	20	320
Link number	14	32	32	2432
Average hops	3.19	5.52	3.46	4.02
Network scenario	Backbone	Backbone	Data center	Data center

TABLE IV
PERFORMANCE EVALUATION DETAIL OF ALL COMPARED ALGORITHMS

Scenario	Evaluation Platform	OPT	GA-ORG	GA-ED	SA	BASE	ISCP
		Average delay (ms) in Abilene					
Case1	Simulation	35	80	80	45	85	120
	Testbed	41. 47	90. 48	90. 22	52. 03	95. 99	129. 55
Case2	Simulation	75	95	90	85	100	150
	Testbed	80.40	101.48	96.30	90.87	104.26	161.93
		Average delay (ms) in Fattree-4					
Case1	Simulation	55	65	60	60	70	100
	Testbed	70. 81	81. 48	76. 11	76. 42	86. 78	108. 05
Case2	Simulation	90	105	105	95	125	140
	Testbed	97. 44	113. 99	114. 52	104. 67	136. 68	156. 74

V. EVALUATION

In this section, we evaluate the performance of our proposed scheme against two baseline schemes (the proposal in [7] and traditional placement) under different system parameters. First, we use the testbed based experiments to compare the performance gap between the testbed and the simulation, as well as evaluate their performances in the real service chaining. Then, we use extensive simulations to quantitatively analyse the performance of the proposed scheme in larger scale networks.

We observe that VNFs are traditionally deployed on access switches [37], which is used to design our baseline. The first step of the baseline algorithm is to sort the VNFs by their deployed possibilities in descending order. Its second step is to iteratively connect VNFs to access switches. In each iteration, a switch connected to the minimum expected number of VNFs is selected, i.e., $s* = \arg_{s_l \in S}\{u_{s_l,q_i} + u_{s_l,q_i}\}$.

A. Simulation Setup

For evaluation purpose, we devise ARPANET, Abilene, and Fattree to generate network and policy information [17], [38]. The characteristic settings of utilized models are summarized in the Table III. Among them, ARPANET [39] refers to the first experiment network of packet switching and has gradually evolved to today's Internet. Abilene [40] is an American backbone network created by the Internet2 community and Fattree [41] represents a layered structural network that is often used to model data center networks. In this simulation, FatTree-4 and FatTree-16 adopt three level networks and have {4, 16} level-2 sub-trees, respectively. Each switch on the edge level has exactly one port dedicated to connecting terminals. We place VNFs in ARPANET and Abilene to evaluate the algorithmic performance in wide area networks (WANs). The FatTree-4 and FatTree-16 are used to evaluate the performance of proposed algorithms in datacenter networks. Besides, we also explore the performance of proposed algorithms on a real datacenter network (named EDU1) which has a 2-Tiered topology architecture [33].

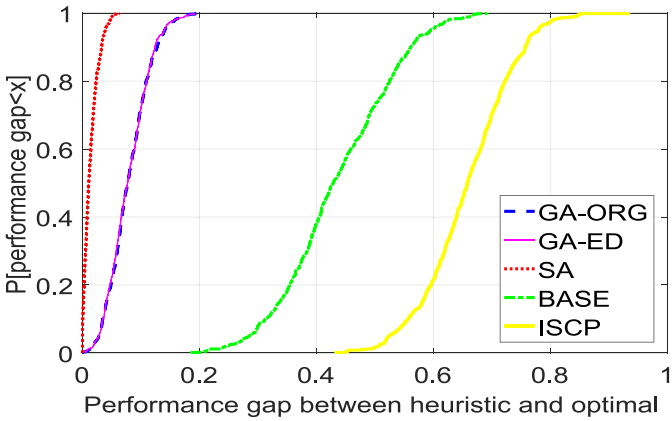


Fig. 4. CDF of our algorithms' performance over optimal solutions.

We generate policy specifications according to [17], [38]. Moreover, we generate flows between switch pairs and randomly assign a service chain to each flow that probabilistically arrive at each NFV. Each service chain is a random sequence of VNFs that is randomly selected from $\mathcal{Q}$. After assigning link and VNFs' processing delays, the network controller route traffic from one VNF to another through the shortest path.

In the following evaluations, an OPT represents an optimal algorithm. GA-ORG denotes the greedy-based algorithm. GA-ED refers to the evolved greedy that incorporates the impact of undeployed VNFs. SA represents the algorithm based on the simulated annealing strategy. Similarly, an ISCP represents a greedy algorithm proposed in [7]. BASE represents the baseline algorithm used for comparing algorithmic performance.

ESCAPE is used for evaluating the algorithmic performance gaps between testbed and simulation results. ESCAPE is an NFV emulation system, which uses Click modular router to implement VNFs. Also, Mininet emulator is utilized to emulate the network topology, and POX controller to guide

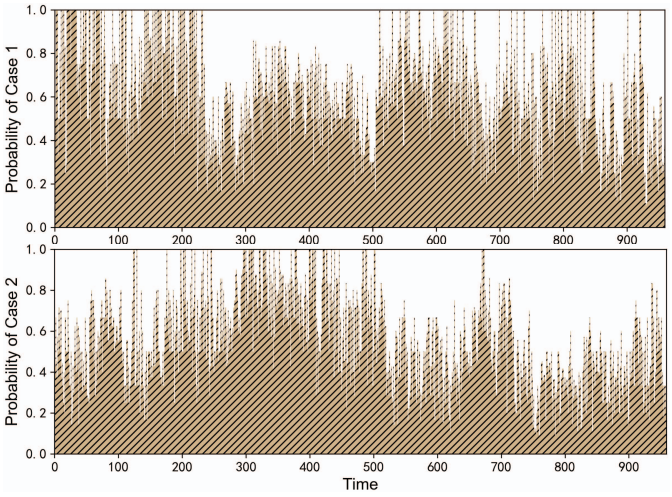


Fig. 5. Time-varying arrival probabilities in data center networks.

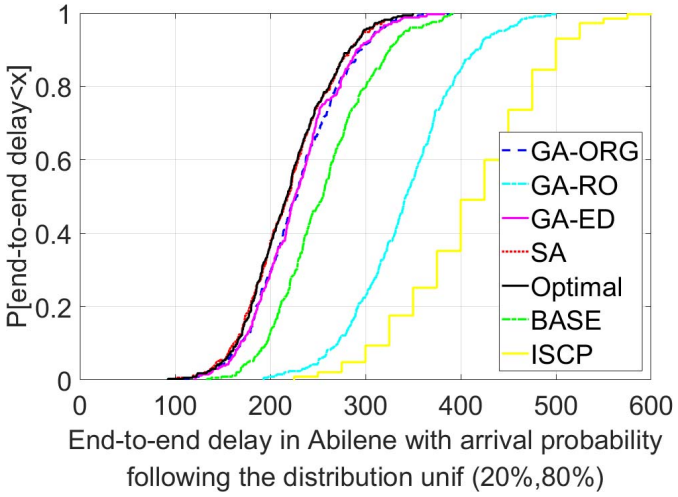


Fig. 6. CDF of end-to-end delays of flows guided by RO-based and our algorithms.

packets [42]. We use it to simulate Abilene and FatTree-4 topologies. For both topologies, each switch connects a host for generating traffic, and a linux container for running VNFs. We define the total number of VNFs [8, 11], however, each service chain contains [2, 3] VNFs. In total, 10 service chains are generated. Between every switch pairs, we generate 10 flows, and we specify random service chain for each flow. To simplify the implementation of testbed, each flow is guided through VNFs with 50% or 100% probability. After obtaining the solutions, ESCAPE install the VNFs and flow tables proactively. Then, we use Iperf [43] to generate traffic and Ping [44] to measure the end-to-end delay.

We randomly generate two placement scenarios of Abilene and FatTree-4, respectively, and simulation results are shown in Table IV. The obtained results depict that the testbed-based results are increased from 4.2% to 28.7% compared to the simulation-based results. Taking GA-ED of *case2* in Abilene for example, the testbed delay is 96.3 and increases 7% as compared with the simulation delay, i.e., 90. The increase (less than 28.7%) on the delay is caused by store-and-forward mode of network, i.e., the ping packets are queueing at switches. However, overall results obtained from the testbed are consistent with the obtained results through the simulation, which means that the simulation results are objective and reasonable.

The performance metrics studied in our work are the end-to-end delay of each flow and total end-to-end delay of flows. Based on the metrics, we evaluate our algorithmic performance to investigate the performance gains: (1) the performance gap between the proposed algorithms and baselines (2) the performance gap between the proposed heuristics and the optimal algorithm (3) the performance changes when varying system parameters.

B. Performance of Greedy Algorithm

For evaluating the algorithmic performance, total 600 simulation scenarios are generated. Network and flows are generated as specified in the previous subsection, and randomly select each scenario parameters from the set that is shown in Table II.

To evaluate the proposed heuristic algorithms with respect to the exact solver, we analyze an the expected delay distribution in the different number of VNF instances and different networks. Fig. 7 depicts the end-to-end delay of flows with the different number of VNF instances placed in different network. As shown in Fig. 7a, when placing 8 VNFs in ARPANET, about {62.1%, 61.9%, 55.7%, 55.6%, 13.5%, 3.8%} of flows spectively guided by algorithms {Optimal, SA, GA-ORG, GA-ED, BASE, ISCP} achieve less than 300 ms end-to-end delay. The obtained results show that the placement strategy has a significant impact on the end-to-end delays of each flow. When placing 11 VNFs in Abilene, this ratio changes to {100%, 100%, 98.7%, 99.6%, 87.5%, 82.7%} as shown in Fig. 7b. Compared with Fig. 7a, Fig. 7b shows that the smaller network diameter of Abilene (the average route hops of Abilene and ARPANET are 3.19 and 5.52, respectively) leads to shorter end-to-end delay of flows. When 20 VNFs are placed in Abilene, Fig. 7c shows that the ratio becomes {91.8%, 90.6%, 87.6%, 86.9%, 75.1%, 9.3%}. In a resource-limited network, a larger number of VNF instances may result that VNFs are more evenly distributed and flows have longer average routing paths and delays as shown in Fig. 7c.

To explore the performance gaps, Fig. 4 depicts the cumulative distribution function (CDF) of the end-to-end delay of different algorithms over the optimal results. There are small algorithmic performance gaps between the heuristics and the optimal, i.e., the mean end-to-end delay is inflated by 1.4%, 7.9%, 8%, 43.7%, and 65.9% with placement scheme generated by SA, GA-ED, GA-ORG, BASE, and ISCP, respectively. Specifically, the performance gap between GA-ED and OPT is less than 12.4% for 90% of scenarios. Furthermore, the proposed algorithm has a large performance gap with ISCP, which demonstrates that our proposed algorithms can reduce an end-to-end delay when dealing with the VNFPRAT problem. Compared with ISCP, {SA, GA-ED, GA-ORG} could reduce {38.8%, 34.9%, 34.8%} end-to-end delay, respectively. Fig. 4 also illustrates that SA outperforms GA-ORG and GA-ED. This is because GA-ED, GA-ORG are easy

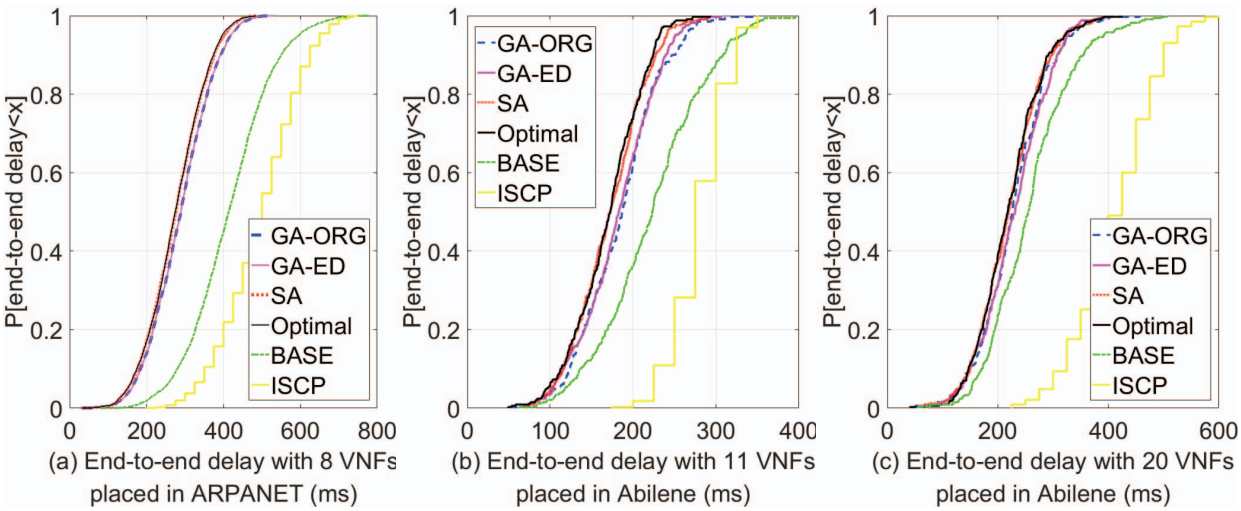


Fig. 7. CDF of the end-to-end delay of each flow.

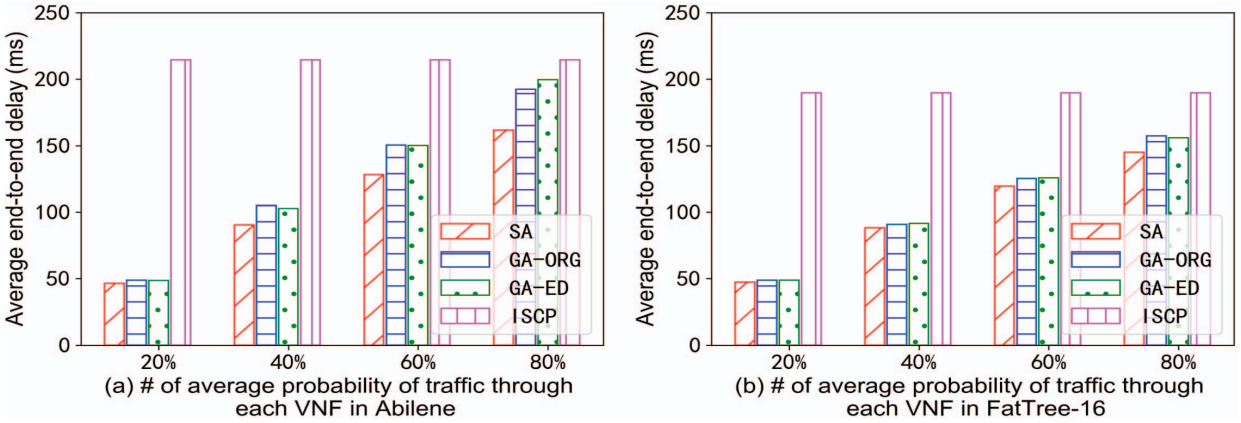


Fig. 8. Average end-to-end delay when varying the probabilities of flows through VNFs.

to fall into local optimum. We use SA to optimize the solutions globally, and take the solutions obtained from GA-ORG and GA-ED as its initial value to improve SA's search efficiency. On average, SA has a 6.11% and 6.05% improvement than GA-ORG and GA-ED, respectively. This is because SA is not trapped to the local minima, while both GA-ORG and GA-ED determine the local minima.

We provide a performance comparison with other stochastic optimization techniques carried out to address the uncertainties of the arrival rate of traffic flow. Based on the characteristics of parameter values, the optimization problems can usually be categorized into deterministic optimization, stochastic programming (SP), robust optimization (RO), and distributionally robust optimization. Because the probability of flow arrival at each VNF is a single stochastic parameter, the VNFPRAT problems can be well regarded as stochastic programming and robust optimization problems. In terms of optimization under uncertainty, the SP-based algorithms need to obtain the distribution of the random variables in advance. Fig. 5 depicts the time-varying arrival probabilities of two cases estimated based on the dataset provided by [33], which is captured by a Cisco switched port analyzer (SPAN), spans multiple days, and ranges about 16 minutes. The different distributions of two cases show that real cases have various arrival probabilities even in the same network. The distribution assumption of arrival probability made by stochastic programming may be correct in some scenarios, but not correct in other scenarios. Because a distribution can not be formed to satisfy all scenarios, the SP-based algorithms are not applied to solve the VNFPRAT problem.

As for the cases where the distribution parameters of the random variables are not known, the RO-based algorithms utilize the uncertainty set of the random variables rather than an accurate distribution function to obtain conservative solutions. To evaluate the performance gap between our algorithms and RO-based algorithms, we estimate the uncertainty set of arrival probability and propose an RO-based algorithm. We set arrival probabilities to follow the uniform distribution on the interval (20%, 80%) that falls into the range estimated from dataset [33]. We use GA-RO to denote the proposed RO-based algorithm. The GA-RO algorithm has a similar process to the GA-ORG algorithm, except that the RO-based algorithm calculates the order of placing VNFs, resource demands, and placing costs using the maximum of arrival probability, i.e., $p_{q_i}^t = 80\%$. As shown in Fig. 6, the GA-RO algorithm prolongs 49.7% end-to-end delay than the

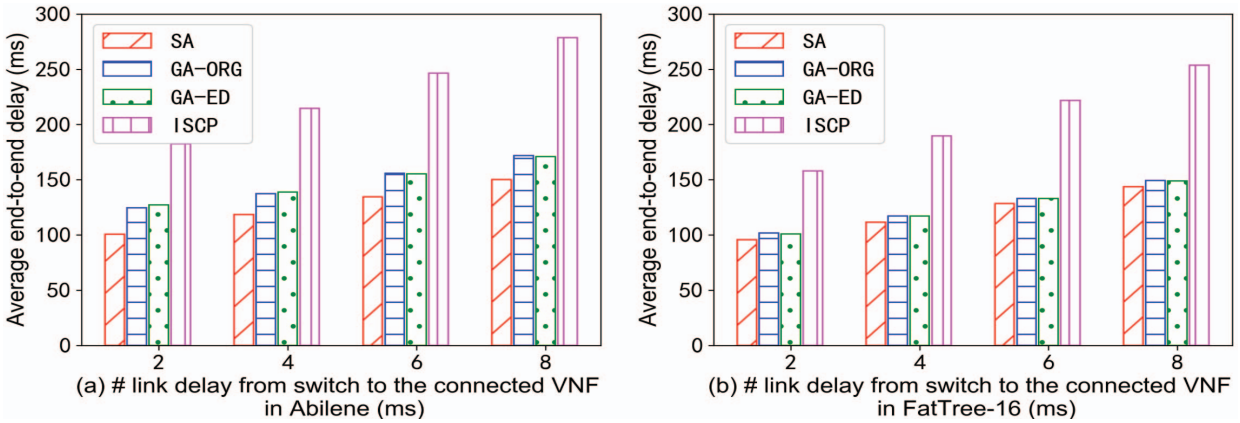


Fig. 9. Average end-to-end delay when varying the link delay from switch to the connected VNF.

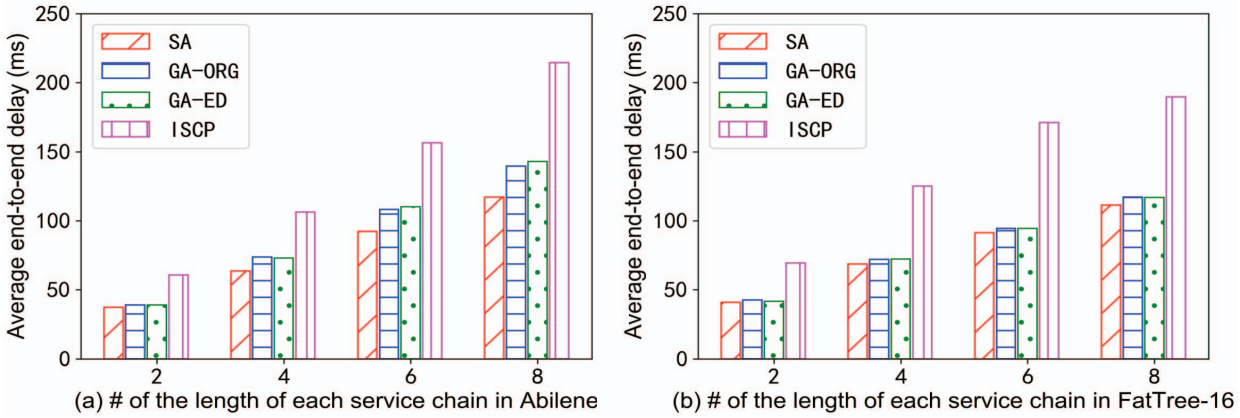


Fig. 10. Average end-to-end delay when varying the length of each service chain.

GA-ORG algorithm. Besides, {95.4%, 94.8%, 91.5%, 91.8%, 79.6%, 22.7%, 9.3%} of flows spectively guided by algorithms {Optimal, SA, GA-ORG, GA-ED, BASE, GA-RO, ISCP} achieve less than 300 ms end-to-end delay. That is because robust optimization algorithms are often used for optimizing the worst-case scenario in the uncertainty set. The conservative placement obtained from the RO-based algorithm need to make room for potential resource demand of VNFs that have lower placing probabilities, which results in lower network performance, e.g., prolong end-to-end delay caused by a longer SFC path.

C. Performance of Simulated Annealing Algorithm

Fig. 8 depicts the impact of randomly arriving traffic while we change the flow arriving probability at each VNF. Based on the statistical analysis on real packet traces provided in [33], we carefully set the average probabilities of flow arrivals as 20%, 40%, 60%, and 80%, which are fallen into the estimated range. These different expected values of arrival probabilities are set to simulate the variations of traffic loads arriving at VNFs. From these results, we can see that end-to-end delay of our proposed algorithms increases linearly with an increasing mean of flows probability arriving at each VNF, in accordance with our intuition. When the mean probability of flow arrival takes value 40% in Abilene, the average end-to-end delay of {SA, GA-ED, GA-ORG, ISCP} are {90.2, 102.6, 104.9, 214.5}ms, respectively. But the results of ISCP remain constant, which is the end-to-end delay upper limit of our proposed algorithms. For example, as shown in Fig. 8b, the average end to end delay obtained by our proposed algorithms with the different probabilities {20%, 40%, 60%, 80%} are 48.37, 90.18, 123.56, and 152.83 respectively, and will slowly approach its upper limit 189.75, which is the result of ISCP. This is because the probability of flow arriving at each VNF is always 100% in ISCP.

To study the impact of network specifications, we change link delay between VNFs and their connected switches, and the results are shown in Fig. 9. Link delays are set to 2ms, 4ms, 6ms, and 8ms for both the Abilene and FatTree-16 network. Fig. 9 illustrates that the end-to-end delay increases linearly with increasing $d^l_{s_{q_i},h_{q_i}}$. When the mean link delay from switch to the connected VNF in FatTree-16 is set to 8ms, the average end-to-end delay obtained by {SA, GA-ED, GA-ORG, ISCP} are {143.6, 148.9, 149.1, 253.7}ms, respectively. In addition, an ISCP algorithm has a fast increase in end-to-end delay than our proposed algorithms, because our algorithms consider flows' arrival at VNFs with certain probability. For instance in Fig. 9a, SA's end to end delay with different link delays are 100.58ms, 118.42ms, 134.39ms, and 150.07ms. ISCP's end to end delay become 182.54ms, 214.55ms, 246.54ms, and 278.55ms when varying link delays.

Since ISCP considers that flows must go through all VNFs in service chain, the increase of link delay $d^l_{s_{q_i},h_{q_i}}$ results in a fast rise in end-to-end delay.

Fig. 10 depicts the performance of placement strategy in case of varying number of VNFs on each service chain. In the simulation, each service chain includes 2, 4, 6, and 8 VNFs. When the mean length of SFCs in Abilene is set to 8, the average end-to-end delay of {SA, GA-ORG, GA-ED, ISCP} achieve {117.1, 139.5, 142.8, 214.5}ms, respectively. In Abilene, GA-ED's end to end delays with different lengths of service chain are 38.91ms, 72.99ms, 110.02ms, and 142.82ms. As the results show, the end-to-end delay is prolonged when the number of VNFs on each service chain increases, which is also in accordance with our intuition.

In the evaluation, the proposed heuristic algorithms are demonstrated to work well (a reduction of up to 38.8% delay) and could achieve sub-optimal solutions with tiny optimality gaps (1.4%-8%). The algorithms are applicable to several scenarios with different parameter values, including flow arriving probability, link delay, and SFC length. Simulation results show that the proposed algorithms perform better than the baselines in response to parameter variations.

VI. Conclusion

In this paper, we investigated the VNFPRAT problem brought by stochastic behavior in networks to find an optimal VNF placement to minimize an expected value of the end-to-end delay. We formulated the VNFPRAT problem into a 0-1 quadratic programming problem and proved that it is an NP-hard problem. To minimize total end-to-end delay, we proposed greedy algorithm and simulated annealing algorithm to obtain near optimal solutions. Extensive simulations are performed to demonstrate that our algorithms are able to minimize end-to-end delay up to 38.8% compared to ISCP. Moreover, the algorithmic performance is near to the optimal values.

In future work, we will estimate the arrival probabilities using the demand and traffic matrices, infer the distribution of arrival probability employing the hypothesis testing, fitting, and learning techniques, and take consideration of full distribution to achieve better performance. Besides, we will try other stochastic programming or robust optimization techniques to tackle the uncertainty of traffic flow arrival at VNFs.

References

[1] (2019). *Building Business Benefits for NFV*. [Online]. Available: https://www.sdxcentral.com/cisco/service-provider/info/analysis/building-nfv-business-benefits/

[2] N. Mckeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[3] (2019). *The Internet Engineering Task Force (IETF) Service Function Chaining (SFC) Working Group (WG). Documents*. [Online]. Available: https://datatracker.ietf.org/wg/sfc/documents/

[4] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, 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.

[5] Q. Xu, D. Gao, T. Li, and H. Zhang, "Low latency security function chain embedding across multiple domains," *IEEE Access*, vol. 6, pp. 14474–14484, 2018.

[6] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vNF placement at the network edge," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Honolulu, HI, USA, 2018, pp. 693–701.

[7] J. Liu, Y. Li, Y. Zhang, L. Su, and D. Jin, "Improve service chaining performance with optimized middlebox placement," *IEEE Trans. Services Comput.*, vol. 10, no. 4, pp. 560–573, Jul./Aug. 2017.

[8] H. Xing *et al.*, "An integer encoding grey wolf optimizer for virtual network function placement," *Appl. Soft Comput.*, vol. 76, pp. 575–594, Mar. 2019.

[9] Y. Cai, Y. Wang, X. Zhong, W. Li, X. Qiu, and S. Guo, "An approach to deploy service function chains in satellite networks," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Taipei, Taiwan, 2018, pp. 1–7.

[10] A. Marotta, E. Zola, F. d'Andreagiovanni, and A. Kassler, "A fast robust optimization-based heuristic for the deployment of green virtual network functions," *J. Netw. Comput. Appl.*, vol. 95, pp. 42–53, Oct. 2017.

[11] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic networking: Data forwarding in disconnected mobile ad hoc networks," *IEEE Commun. Mag.*, vol. 44, no. 11, pp. 134–141, Nov. 2006.

[12] P. Natarajan, J. R. Iyengar, P. D. Amer, and R. R. Stewart, "SCTP: An innovative transport layer protocol for the Web," in *Proc. 15th Int. Conf. World Wide Web*, 2006, pp. 615–624.

[13] L. B. Lim, L. Guan, A. Grigg, I. W. Phillips, X. Wang, and I. U. Awan, "RED and WRED performance analysis based on superposition of N MMBP arrival proccess," in *Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, Perth, WA, Australia, 2010, pp. 66–73.

[14] M. Casazza, P. Fouilhoux, M. Bouet, and S. Secci, "Securing virtual network function placement with high availability guarantees," in *Proc. IFIP Netw. Conf. (IFIP Networking) Workshops*, Stockholm, Sweden, 2017, pp. 1–9.

[15] P. Vizarreta, M. Condoluci, C. M. Machuca, T. Mahmoodi, and W. Kellerer, "QoS-driven function placement reducing expenditures in NFV deployments," in *Proc. IEEE Int. Conf. Commun.*, Paris, France, 2017, pp. 1–7.

[16] C. Shao, S. Leng, Y. Zhang, A. Vinel, and M. Jonsson, "Performance analysis of connectivity probability and connectivity-aware MAC protocol design for platoon-based VANETs," *IEEE Trans. Veh. Technol.*, vol. 64, no. 12, pp. 5596–5609, Dec. 2015.

[17] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, 2013, pp. 27–38.

[18] A. Laghrissi, T. Taleb, M. Bagaa, and H. Flinck, "Towards edge slicing: VNF placement algorithms for a dynamic & realistic edge cloud environment," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Singapore, 2017, pp. 1–6.

[19] S. Herker, X. An, W. Kiess, S. Beker, and A. Kirstaedter, "Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, San Diego, CA, USA, 2015, pp. 1–7.

[20] J. Sun, G. Sun, D. Liao, Y. Li, M. Ramachandran, and V. Chang, "Reliable and efficient deployment for virtual network functions," in *Proc. Int. Conf. Smart Comput. Commun.*, 2017, pp. 375–384.

[21] A. Baumgartner, T. Bauschert, F. D'Andreagiovanni, and V. S. Reddy, "Towards robust network slice design under correlated demand uncertainties," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, 2018, pp. 1–7.

[22] B. Farkiani, B. Bakhshi, and S. A. MirHassani, "Stochastic virtual network embedding via accelerated benders decomposition," *Future Gener. Comput. Syst.*, vol. 94, pp. 199–213, May 2019.

[23] Y. Xie, B. Wang, S. Wang, and L. Luo, "Virtualized network function provisioning in stochastic cloud environment," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, 2020, pp. 1–6.

[24] E. M. Loiola, N. M. M. D. Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," *Eur. J. Oper. Res.*, vol. 176, no. 2, pp. 657–690, 2007.

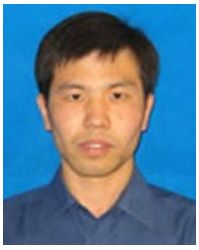
[25] *Network Functions Virtualisation (NFV); Architectural Framework*, Standard ETSI GS NFV 002 V1.2.1, 2019. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf

[26] Y. Jia, C. Wu, Z. Li, F. Le, and A. Liu, "Online scaling of NFV service chains across geo-distributed datacenters," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 699–710, Apr. 2018.

[27] T. Koponen, R. Zhang, P. Thakkar, and M. Casado, "Connection identifier assignment and source network address translation," U.S. Patent 8 913 611, Dec. 16 2014.

[28] H. D. Chantre and N. L. S. da Fonseca, "Redundant placement of virtualized network functions for LTE evolved multimedia broadcast multicast services," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, 2017, pp. 1–7.
[29] A. Fischer, "Generating random service function chain embedding problems," in *Proc. IEEE Conf. Netw. Funct. Virtualization Softw. Defined Netw. (NFV-SDN)*, Berlin, Germany, 2017, pp. 1–6.
[30] B. Blanco, I. Taboada, J. O. Fajardo, and F. Liberal, "A robust optimization based energy-aware virtual network function placement proposal for small cell 5G networks with mobile edge computing capabilities," *Mobile Inf. Syst.*, vol. 2017, Oct. 2017, Art. no. 2603410.
[31] M. J. Abdel-Rahman, E. A. Mazied, A. MacKenzie, S. Midkiff, M. R. Rizk, and M. El-Nainay, "On stochastic controller placement in software-defined wireless networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, San Francisco, CA, USA, 2017, pp. 1–6.
[32] M. J. Abdel-Rahman, E. A. Mazied, K. Teague, A. B. MacKenzie, and S. F. Midkiff, "Robust controller placement and assignment in software-defined cellular networks," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Vancouver, BC, Canada, 2017, pp. 1–9.
[33] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 267–280.
[34] D. E. Knuth, "Postscript about NP-hard problems," *ACM SIGACT News*, vol. 6, no. 2, pp. 15–16, 1974.
[35] (2019). *Gurobi Optimization*. [Online]. Available: http://www.gurobi.com/
[36] M. Alfonsas, "A modified simulated annealing algorithm for the quadratic assignment problem," *Informatica*, vol. 14, no. 4, pp. 497–514, 2003.
[37] B. Soewito and Hirzi, "Building secure wireless access point based on certificate authentication and firewall captive portal," in *Proc. EPJ Web Conf.*, vol. 68, 2014, Art. no. 00029.
[38] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture," in *Proc. 9th USENIX Conf. Netw. Syst. Design Implement.*, 2012, p. 24.
[39] M. Hauben, *History of ARPANET*, vol. 17, L'Instituto Superior de Engenharia do Porto, Porto, Portugal, 2007.
[40] (2019). *Abilene Core Topology*. [Online]. Available: https://itservices.stanford.edu/service/network/internet2/abilene
[41] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.
[42] A. Csoma *et al.*, "ESCAPE: Extensible service ChAin prototyping environment using mininet, click, NETCONF and POX," in *Proc. ACM Conf. SIGCOMM*, 2014, pp. 125–126.
[43] A. Tirumala, L. Cottrell, and T. Dunigan, "Measuring end-to-end bandwidth with Iperf using web100," in *Proc. Passive Active Meas. Workshop*, 2003, pp. 1–8.
[44] C. Pelsser, L. Cittadini, S. Vissicchio, and R. Bush, "From Paris to Tokyo: On the suitability of ping to measure latency," in *Proc. Conf. Internet Meas. Conf.*, 2013, pp. 427–432.

Jie Sun received the B.S. degree in computer and information engineering college from the Huaiyin Institute of Technology, Huaian, China, in 2012, and the M.S. degree from the School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, China, in 2016. He is currently pursuing the Ph.D. degree with Beihang University, Beijing, China. His research interests include software defined network, network function virtualization, and the space-ground integrated network.



Feng Liu (Member, IEEE) received the M.S. and Ph.D. degree from Xi'an Jiaotong University, Xi'an, China, in 1997 and 2000, respectively. He did two years research in Tsinghua University as a Postdoctoral Researcher. He is a Professor with the School of Electronics and Information Engineering, Beihang University. His research interests include space-ground communication network, delay tolerant network, and self-organizing mobile network.

Huandong Wang (Member, IEEE) received the first B.S. degree in electronic engineering, the second B.S. degree in mathematical sciences, and the Ph.D. degree in electronic engineering from Tsinghua University, Beijing, China, in 2014, 2015, and 2019, respectively, where he is currently a Postdoctoral Research Fellow with the Department of Electronic Engineering. His research interests include urban computing, mobile big data mining, and machine learning.

Manzoor Ahmed received the B.E. and M.S. degrees in electrical engineering and computer science from Balochistan Engineering University, Khuzdar, Pakistan, in 1996 and 2010, respectively, and the Ph.D. degree in communication and information systems from the Beijing University of Posts and Telecommunications, China, in 2015. From 1997 to 2000, he served as a Lecturer with Balochistan Engineering University and then served as a Telecomm Engineer in government owned telecommunication service provider NTC, Pakistan, from 2000 to 2011. Recently, he worked as a Postdoctoral Researcher with the Electrical Engineering Department, Tsinghua University, China, from 2015 to 2018. He is currently a Faculty Member with the Department of Computer Science and Technology, Qingdao University. He has several research publications in IEEE top journals and conferences. His research interests include resource allocation and offloading in vehicular communications and networking, fog and edge computing, socially aware D2D communication, and physical layer security. He received several awards, including Distinction Award from the President of Pakistan, the Best Employee Award in NTC, and the Best Paper Award in 2014 GameNets Conference.

Yong Li (Senior Member, IEEE) received the Ph.D. degree in electronic engineering from Tsinghua University in 2012, where he is currently a Tenured Associate Professor with the Department of Electronic Engineering, Tsinghua University. His research interests include machine learning and big data mining, particularly, automatic machine learning and spatial-temporal data mining for urban computing, recommender systems, and knowledge graphs. He has published over 100 papers on first-tier international conferences and journals, including KDD, WWW, UbiComp, SIGIR, AAAI, TKDE, and TMC, and his papers have total citations more than 8300. Among them, ten are ESI Highly Cited Papers in Computer Science, and five receive conference Best Paper (run-up) Awards. He received IEEE 2016 ComSoc Asia–Pacific Outstanding Young Researchers, Young Talent Program of China Association for Science and Technology, and the National Youth Talent Support Program. He has served as a general chair, a TPC chair, and a SPC/TPC member for several international workshops and conferences, and he is on the editorial board of two IEEE journals.

Lianlian Zhang received the B.S. and M.S. degrees in electronics and information engineering from Beihang University, Beijing, China, in 2008 and 2011, respectively, where she is currently pursuing the Ph.D. degree. Her research interests include SDN, space-ground communication network, and network function virtualization.

Hao Zeng received the B.S. degree from Beihang University, Beijing, China, in 2017, where she is currently pursuing the postgraduate degree. Her major is electronic engineering. Her research field includes high-speed data processing, software-defined network, and data exchange.