Efficient Virtual Network Function Placement for Poisson Arrived Traffic

Jie Sun
Beihang University
sunjiehyit@buaa.edu.cn

Feng Liu
Beihang University
liuf@buaa.edu.cn

Manzoor Ahmed
Tsinghua University
manzoor.achakzai71@yahoo.com

Yong Li
Tsinghua University
liyong07@tsinghua.edu.cn

***Abstract*—Network Function Virtualization (NFV) and Software Defined Network have revolutionized data networks. For deployment of the VNFs, it is imperative to consider the queuing delay occurring in the NFV servers. The presented work suggests to incorporate the Poisson distribution of packet arrival rate and packet size along with the limited processing capacity of NFV server. The underlying phenomenon is framed as a 0-1 fractional programming problem. More precisely, the underlying problem is framed to 0-1 MILP to get the optimal solution. Since the VNF placement problem is NP-hard, therefore, we propose two heuristic algorithms to get the solution. Extensive simulations demonstrate that our algorithms effectively reduce 72.9% delay compared with the universal algorithm Improve Service Chaining Performance (ISCP).**

I. Introduction

Network Function Virtualization (NFV) is employed to deploy middleboxes (such as NAT, DPI, Firewall and so on) in the standard servers conveniently by incorporating virtual technologies. In recent years the concept of Software Defined Network (SDN) is put forward to guide network traffic towards any openflow-enabled location via flow tables specified by the controller. By combining NFV and SDN technology, a network operator can easily control the traditional network functions. NFV brings many advantages, but also requires to solve the middleboxes placement problem to improve network performance.

Many research efforts have been made for optimally placing middleboxes in the NFV. Considering the general middlebox placement scenario, Li et al. [1] formulate it to a 0-1 programming problem and design the heuristics to obtain sub-optimal solutions; Cohen et al. [2] propose a near-optimal solution to place middlexes since the placing problem is NP-hard. For various special network scenarios, Marotta et al. [3] focus on the VNFs placement problem incurred in the 5G architecture; Yang et al. [4] pay attention to energy conservation of the service chains in data center network. The authors argue that higher throughput [5] and availability [6] are the required parameters to improvise the service chains' performance.

Despite these detailed studies, none among the prevalent works has addressed the problem of queueing delay occuring in the NFV servers, which may even cause a very long delay due to the limited processing capacity of the NFV servers. Deval et al. [4] pay attention to response time reduction in a multicloud scenario by decreasing the traffic between inter-clouds. But, instead of the inter-clouds, the intra-cloud servers are the real processing units that follow M/M/1 queueing model in the NFV network [7]. Thus, our work concentrates on their performance for processing the classed arrived traffic, and take the end-to-end delay as our optimization target to provide better experience to the end users.

The queueing phenomenon often appears in real NFV networks because of various reasons. The first reason is the packet arrival rate of general network traffic that doesnt follow a uniform distribution or constant bit rate (CBR). For example, the packet rates usually follow a poisson distribution [8]. Secondly, the packet size varies in the network traffic, as this often follows an exponential distribution. Lastly, the processing capacity of the servers is limited and the CPU time occupation of various type of middleboxes is different [7].

Considering the complicated flow information and the aforementioned scenarios, the VNF placement problem becomes even more challenging. Firstly, in addition to the traditional network information, the processing capacity of the servers and the CPU time occupation of each middlebox must be considered. Secondly, as to the flow information, there are multi-class traffic flows with various distributions of the arrival rate and the packet size. Lastly, there are many parameters that impact the performance of the placement scheme, for instance, link delay, dependency relationship of the middleboxes in service chains, etc.

Thus, this is not a simple placement problem considering the non-uniform distribution based traffic in the placement scenario. The contributions of this paper are summed as follows:

- To the best of our knowledge, we are the first to focus on the virtual network function placement for Poisson arrived traffic, short for VNFPPAT, and formulate it as a 0-1 quadratic fractional programming problem. In order to find an exact solution for the VNFPPAT problem, we transform it to a mixed integer linear programming (MILP) problem.
- Since VNFPPAT is an NP-hard problem, we propose two heuristic algorithms: one employs a greedy strategy and the other is based on a simulated annealing algorithm.
- We evaluate our proposed algorithm performance based on extensive simulations. The proposed algorithms greatly reduce the total delay. On average, the simulated annealing algorithm reduces 72.9% of the total delay compared with the state-of-the-art algorithms.

978-1-5386-8088-9/19/$31.00 ©2019 IEEE

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on March 11,2023 at 15:54:00 UTC from IEEE Xplore. Restrictions apply.

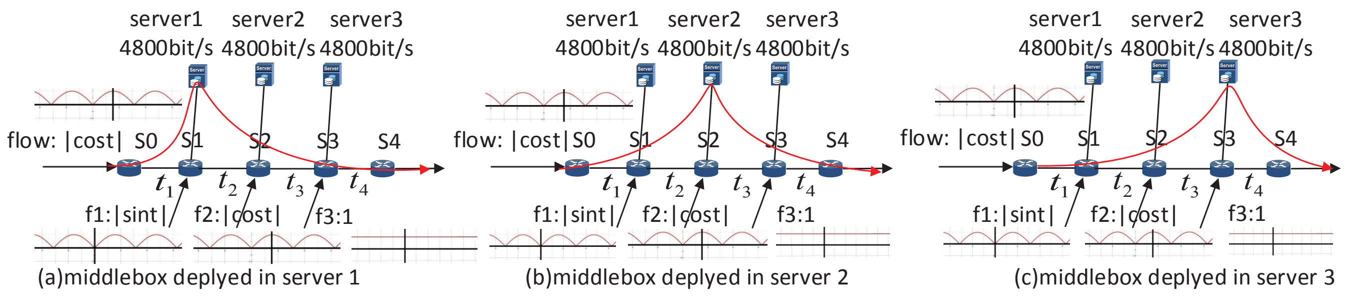

server1
server2
server3
4800bit/s
flow: |cost| S0
S1
S2
S3
S4
t_1
t_2
t_3
t_4
f1:|sint|
f2:|cost|
f3:1
(a)middlebox deplyed in server 1
(b)middlebox deplyed in server 2
(c)middlebox deplyed in server 3

Fig. 1. Illustration of the VNF placement for a single non-uniformly distributed arrived traffic.

The remaining contents of the paper are presented as follows. The system scenario of the general VNF placement is detailed in Section II. The VNFPPAT problem is formulated in Section III. The heuristic algorithms and their performance are introduced and discussed in Sections IV, V, respectively. Finally, we conclude our research in Section VI.

II. System Overview

The considered scenario incorporates openflow-enabled switches and standard NFV servers. The switches are controlled by a logical centralized controller. The path of flows is guided by openflow flow tables that are distributed by the controller. The network functions (middleboxes) are deployed in the standard NFV servers that are directly connected to the openflow switches. The controllers specify the service chains of the corresponding flows for guiding them through the middleboxes.

Fig. 1 illustrates the VNF placement problem while considering non-uniform traffic. We just ignore the queueing delay in the switches and the link delay between them and their corresponding server. We consider three background traffic flows traversing the corresponding servers with different rates. Their packet rates are assumed to be $|sint|$, $|cost|$, and 1. The arriving traffic or flow only needs to go through one middlebox with $|cost|$ packet/s. In each server, the processing capacity is 4800 bits per second, and the mean packet size is 200 bits. If the middlebox is deployed in Server1, the average of the end-to-end delay is formulated as $\frac{\int_0^{2\pi}\sum_{i=1}^{4} t_i+\frac{1}{24-|\cos t|-|\sin t|}dt}{2\pi}$. If the traffic only goes through Server2, the mean delay changes to $\frac{\int_0^{2\pi}\sum_{i=1}^{4} t_i+\frac{1}{24-2|cost|}dt}{2\pi}$. In the last situation, the flow arrives at the Server3 and its delay is formalized as $\frac{\int_0^{2\pi}\sum_{i=1}^{4} t_i+\frac{1}{23-|cost|}dt}{2\pi}$. Obviously, the end-to-end delay in the first situation is minimal. The deploying strategy in Fig. 1(c) is worse than that of Fig. 1(b).

To illustrate the VNFPPAT problem, we provide another example as shown in Fig. 3, which contains two arrival flows. The queueing model depicted in Fig. 2 is also called M/M/1, and is the most appropriate illustration of NFV server system [7]. Therefore, we use M/M/1 as a default model in our paper. The queueing delay of M/M/1 is $\frac{1}{u_{s_l}-\lambda_{s_l}}$ [8] where u_{s_l} refers to the average service rate of h_l and λ_{s_l} describes its total packet arrival rate. In fact, the queueing model of servers does not remain M/M/1 model over time, because the packet size could not change in most situations. In these situations, it is quite challenging to calculate the end-to-end delay. Thus, we introduce an independence assumption, i.e., each packet will be given a new random length in each server. This assumption is contradictory to the fact, but it can yield a result that is close to real incurred delay [9]. Based on the independence assumption and queueing model as M/M/1, the total queueing delay of f_k can be expressed as $\sum_{h_l\in c_k}\frac{1}{u_{s_l}-\lambda_{s_l}}$.

Thus, considering Fig. 3, we can easily compute the total delay with the aforementioned expression. There are two Poisson flows both traversing from S1 to S6 with 1 and 9 packets per second, respectively. Four middleboxes (FW, IPS, NAT, and Proxy) need to be deployed in this topology. Six NFV servers are used to deploy them, and six corresponding background traffics visit these servers with $\{6, 14, 8, 2, 1, 10\}$ packets/second, respectively. The processing capacity of each server is 4800 bits/s, and the mean packet length of flows arriving in them is 200 bits/packet. All link delays are set to 10ms. If four middleboxes are placed on Server 2, 3, 5, and 4, respectively, the total delay of the two flows is computed as

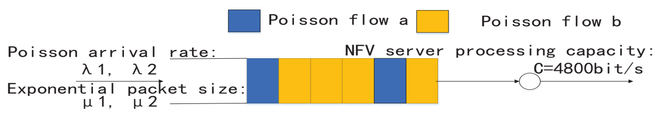
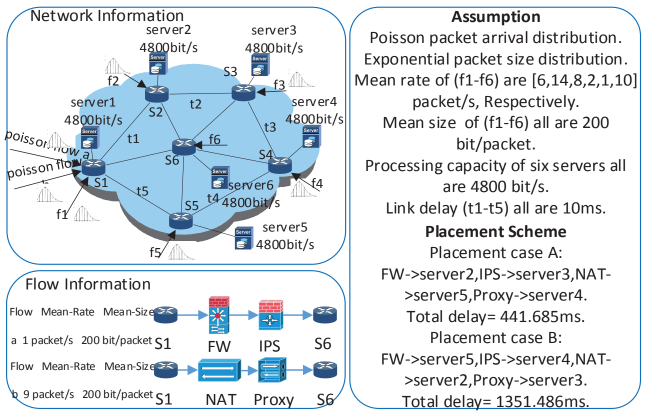


Fig. 2. A system of markovian multi-class, single-server queueing scenario.



Flow Mean-Rate Mean-Size
a 1 packet/s 200 bit/packet S1 FW IPS S6
Flow Mean-Rate Mean-Size
b 9 packet/s 200 bit/packet S1 NAT Proxy S6

Assumption
Poisson packet arrival distribution.
Exponential packet size distribution.
Mean rate of (f1-f6) are [6,14,8,2,1,10] packet/s, Respectively.
Mean size of (f1-f6) all are 200 bit/packet.
Processing capacity of six servers all are 4800 bit/s.
Link delay (t1-t5) all are 10ms.

Placement Scheme
Placement case A:
FW->server2,IPS->server3,NAT->server5,Proxy->server4.
Total delay= 441.685ms.
Placement case B:
FW->server5,IPS->server4,NAT->server2,Proxy->server3.
Total delay= 1351.486ms.

Fig. 3. An example of the middleboxes placement problem in case of multi-class non-uniform traffic.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on March 11,2023 at 15:54:00 UTC from IEEE Xplore. Restrictions apply.

441.7ms. The alternative placing scheme considers to deploy {FW, IPS, NAT, Proxy} to Server- {5, 4, 2, 3} separately. The total value changes to $10*6+(1/(24-14-9)+1/(24-8-9)+1/(24-1-1)+1/(24-2-1))*1000=1351.5ms$. The above results and their designs indicate the importance of placement schemes and their designs, since they greatly affect the queueing delay.

III. Problem Formulation

A Problem Formulation

To formulate the VNFPPAT problem, we use a series of notations to represent the variables that are used in our manuscript. The variables involve network information and flow information. For network information, we use ($\mathcal{S}$, $\mathcal{H}$, $\mathcal{E}$) to denote a network topology that refers to switch set, NFV server set, and link set, respectively. A row vector ($d^l_{s_l,s_{l_1}}$, h_l, c_{l_i}) is used to denote the attributes associated with the switch s_l, where $d^l_{s_l,s_{l_1}}$ represents the link delay to s_{l_1}, h_l represents the connected server, and c_{l_i} refers to capacity of h_l processing data yield from the middlebox q_i. The following (i_k, e_k, c_k, λ_k, l_k) vectors depict the properties of the flow f_k, i.e., these refer to ingress/egress switch, corresponding service chain, packet rate, and the mean packet length of f_k, respectively.

For simplifying the deployment procedure, we use the binary symbol x_{q_i,s_l} to denote whether q_i is deployed in h_l. Since a middlebox can be deployed to only one NFV server, the following constraint should be satisfied:

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

Owing to the limitation of processing capability, the arrival packet rate should be less than its server rate to guarantee the queueing system stability. Otherwise, storing more data of this system will need more queueing time. The stability of the queuing system should meet the following constraint:

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

where ρ_{q_i,s_l} refers to service intensity of q_i in h_l, that denotes the processing capability utilization. The left side of (2) expresses the total service intensity of h_l according to [10]. Except for the resource limitation, some middleboxes must run on the special platforms. For example, DPI may need to be deployed in a NFV server with powerful data processing capabilities, such as GPU. With this consideration, the location of q_i should follow the constraint $x_{q_i,s_l} = 0, \forall s_l \in S \backslash S_{q_i}$.

The delay of traffic flow is the key parameter to get the user experience termed as Quality of Experience (QoE). Thus, we take the computation of the end-to-end delay as our objective function. Other parameters, such as bandwith, can be optimized in a similar way.

An end-to-end delay contains four parts in the NFV scenario, i.e., link delay between the ingress switch i_k and the first middlebox m^1_k, link delay from the last middlebox $m^{n_k}_k$ to the egress switch e_k, link delay of route path of service chain c_k, and queueing delay in each server. In this article, we directly ignore the forwarding delay in each switch since it is small enough compared to other delays.

Before expressing the end-to-end delay, we introduce the definitions for the four parts of the delay (v_{q_i,s_l}, v_{s_l,q_i}, v_{q_i,q_j}, v_{q_i}), which can explicitly show the coefficient of decision variables x_{q_i,s_l} in the objective function expression. v_{q_i,s_l} and v_{s_l,q_i} are used to designate the number of flows whose ingress/egress switch is s_l and the first/last visited middlebox is q_i. v_{q_i,q_j} refers to the number of flows that directly traverses q_j after arriving at q_i, and v_{q_i} represents the total number of times that a flow passes through q_i. These variables are defined as follows:

$$\begin{cases} \forall s_l \in S, \forall q_i \in Q, v_{s_l,q_i} = \sum\limits_{\forall f_k \in F} I_{i_k,s_l} I_{m^1_k,q_i}, \\ \forall s_l \in S, \forall q_i \in Q, v_{q_i,s_l} = \sum\limits_{\forall f_k \in F} I_{e_k,s_l} I_{m^{n_k}_k,q_i}, \\ \forall q_i \in Q, \forall q_j \in Q, v_{q_i,q_j} = \sum\limits_{\forall f_k \in F} \sum\limits_{\eta=1}^{n_k-1} I_{m^\eta_k,q_i} I_{m^{\eta+1}_k,q_j}, \\ \forall q_i \in Q, v_{q_i} = \sum\limits_{f_k \in F} z_{q_i,f_k}. \end{cases} \tag{3}$$

In the above expression, I_{i_k,s_l} is an indicative parameter that informs whether i_k equals to s_l. Considering that a flow f_k may go through one middlebox more than once, we define z_{q_i,f_k} as the number of times that flow f_k passes through middlebox q_i, which is calculated as $z_{q_i,f_k} = \sum\limits_{h=1}^{n_k} I_{m^h_k,q_i}$. The total value of each part of the end-to-end delay can be obtained by defining sums of the appropriate variables in (3) over all possible deployment locations, because q_i can only be deployed at one place due to constraint (2). For instance, the total delay between the ingress switches and the first middle box can be obtained as $\sum\limits_{\forall q_i \in Q} \sum\limits_{\forall s_l \in S} \sum\limits_{\forall s_{l_1} \in S} d^r_{s_l,s_{l_1}} v_{s_l,q_i} x_{q_i,s_{l_1}}$ that can acquire the total delay of all flows. Therefore, the total delay can be formulated as:

$$\begin{aligned} D^t = & \sum_{\forall q_i \in Q} \sum_{\forall s_l \in S} \sum_{\forall s_{l_1} \in S} d^r_{s_l,s_{l_1}} v_{s_l,q_i} x_{q_i,s_{l_1}} \\ & + \sum_{\forall q_i \in Q} \sum_{\forall s_l \in S} \sum_{\forall s_{l_1} \in S} d^r_{s_{l_1},s_l} v_{q_i,s_l} x_{q_i,s_{l_1}} \\ & + \sum_{\forall q_i \in Q} \sum_{\forall q_j \in Q} \sum_{\forall s_{l_1} \in S} \sum_{\forall s_{l_2} \in S} d^r_{s_{l_1},s_{l_2}} v_{q_i,q_j} x_{q_i,s_{l_1}} x_{q_j,s_{l_2}} \\ & + \sum_{\forall s_l \in S} \frac{\sum\limits_{\forall q_i,q_j \in Q} \rho_{q_i} v_{q_j} x_{q_i,s_l} x_{q_j,s_l} y'_{s_l}}{\sum\limits_{\forall q_g \in Q} \lambda_{q_g} x_{q_g,s_l} - \sum\limits_{\forall q_h,q_k \in Q} \rho_{q_h} \lambda_{q_k} x_{q_h,,s_l} x_{q_k,s_l}}. \end{aligned} \tag{4}$$

In (4), the variable y'_{s_l} indicates whether there exists at least one flow traversing through s_l, which can be expressed by $x_{q_1,s_l} \vee x_{q_2,s_l} \vee ... x_{q_n,s_l}$. The fourth part of the delay can be obtained by $\frac{\rho_{s_l}}{(1-\rho_{s_l})\lambda_{s_l}} v_{s_l}$, when at least one flow has arrived. The value changes to zero when packet arrival rate λ_{s_l} is zero. The binary linear programming formulations of y'_{s_l} are introduced in [11] and are defined as follows:

$$\left\{ x_{q_i,s_l} \le y'_{s_l}, \forall q_i \in Q; \sum_{\forall q_i \in Q} x_{q_i,s_l} \ge y'_{s_l}; y'_{s_l} \in \{0,1\}. \right. \tag{5}$$

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on March 11,2023 at 15:54:00 UTC from IEEE Xplore. Restrictions apply.

Thus, the VNFPPAT problem is formulated as:

$$\begin{aligned} \textbf{P1:} \quad & \min \quad D^t \quad \text{according to} \quad (4) \\ & \text{s.t.} \quad (1), (2), (5). \end{aligned} \tag{6}$$

B Problem transformation

The VNFPPAT problem formulated in **P1** is the 0-1 quadratic fractional programming problem. In order to solve the VNFPPAT problem using ILP software such as Gurobi, we need to change it to a linear programming (LP) problem. We divide this procedure into several steps: transform the objective function to an non-fractional form, and then map it to a 0-1 mixed linear programming problem. Similar to the strategy proposed by Charnes-Copper et al., the fraction $\frac{y'_{s_l}}{\sum\limits_{\forall q_g \in Q} \lambda_{q_g} x_{q_g,s_l} - \sum\limits_{\forall q_h \in Q} \sum\limits_{\forall q_k \in Q} \rho_{q_h} \lambda_{q_k} x_{q_h,s_l} x_{q_k,s_l}}$ is replaced by the variable t_{s_l} [12]. We use d_t^1, d_t^2, and d_t^3 to separately represent 1-3 part of total delay. Thus, the expression of total delay is converted into the following form:

$$\begin{aligned} D^t =& d_t^1 + d_t^2 + d_t^3 \\ &+ \sum_{\forall s_l \in S} \sum_{\forall q_i \in Q} \sum_{\forall q_j \in Q} \rho_{q_i} v_{q_i} x_{q_i,s_l} x_{q_j,s_l} t_{s_l}. \end{aligned} \tag{7}$$

Accordingly, the following equation relationships should be added to the constraint set:

$$\begin{cases} \sum\limits_{\forall q_g \in Q} \lambda_{q_g} x_{q_g,s_l} t_{s_l} - \sum\limits_{\forall q_h \in Q} \sum\limits_{\forall q_k \in Q} \rho_{q_h} \lambda_{q_k} x_{q_h,s_l} x_{q_k,s_l} t_{s_l} = y'_{s_l}; \\ \qquad t_{s_l} \geq 0. \end{cases} \tag{8}$$

Thus, the non-fractional form of VNFPPAT is formulated as:

$$\begin{aligned} \textbf{P2:} \quad & \min \quad D^t \quad \text{according to} \quad (7) \\ & \text{s.t.} \quad (1), (2), (5), (8). \end{aligned} \tag{9}$$

Secondly, we should linearize the higher order terms because the formulation has still quadratic and cubic terms. Therefore, new notations are used to replace them, i.e., $o_{i,l} = x_{q_i,s_l} t_{s_l}$, $y_{i,j,l_1,l_2} = x_{q_i,s_{l_1}} x_{q_j,s_{l_2}}$, and $w_{i,j,l} = x_{q_i,s_l} x_{q_j,s_l} t_{s_l}$. After replacing the variables, the expression for the total delay is changed to:

$$\begin{aligned} D^t =& d_t^1 + d_t^2 + \sum_{\forall s_l \in S} \sum_{\forall q_i \in Q} \sum_{\forall q_j \in Q} \rho_{q_i} v_{q_i} w_{i,j,l} \\ &+ \sum_{\forall q_i \in Q} \sum_{\forall q_j \in Q} \sum_{\forall s_{l_1} \in S} \sum_{\forall s_{l_2} \in S} d^r_{s_{l_1},s_{l_2}} v_{q_i,q_j} y_{i,j,l_1,l_2}. \end{aligned} \tag{10}$$

Now, the constraint of (8) changes its form to:

$$\left\{ \sum_{\forall q_g \in Q} \lambda_{q_g} o_{g,l} - \sum_{\forall q_h \in Q} \sum_{\forall q_k \in Q} \rho_{q_h} \lambda_{q_k} w_{h,k,l} = y'_{s_l}; t_{s_l} \geq 0. \right. \tag{11}$$

The variables which are used to replace the nonlinear terms should relax their feasible region with the Reformulation Linearization Technique (RLT) [13]. The RLT bound-factor constraints of y_{i,j,l_1,l_2}, $o_{i,l}$ and $w_{i,j,l}$ are listed as follows:

$$\begin{cases} y_{i,j,l_1,l_2} \leq x_{q_i,s_{l_1}}; y_{i,j,l_1,l_2} \leq x_{q_j,s_{l_2}}; \\ y_{i,j,l_1,l_2} \geq x_{q_i,s_{l_1}} + x_{q_j,s_{l_2}} - 1; \\ y_{i,j,l_1,l_2} \in \{0,1\}; \\ o_{i,l} \geq 0; t_{s_l} - o_{i,l} \geq 0; o_{i,l} - w_{i,i,l} = 0; \\ w_{i,j,l} \leq o_{i,l}; w_{i,j,l} \leq o_{j,l}; w_{i,j,l} \geq o_{i,l} + o_{j,l} - t_{s_l}. \end{cases} \tag{12}$$

$$\left\{ o_{i,l} \in \{0, t_{s_l}\}; w_{i,j,l} \in \{0, t_{s_l}\}. \right. \tag{13}$$

Now, the VNFPPAT problem is formulated as a Mixed Integer Nonlinear Programming (MINLP) problem presented by the following expressions:

$$\begin{aligned} \textbf{P3:} \quad & \min \quad D^t \quad \text{according to} \quad (10) \\ & \text{s.t.} \quad (1), (2), (5), (11) - (13). \end{aligned} \tag{14}$$

Third, there are two bi-linear constraints (13) that need to be linearized. For example, the constraint of (13) is the product of a binary variable x_{q_i,s_l} and a continuous variable t_{s_l}. According to [14], we can use a sufficiently large **M** to guarantee $o_{i,l}$ to be zero if x_{q_i,s_l} is zero, and be t_{s_l} when x_{q_i,s_l} is one. The introduced constraints are expressed as:

$$\begin{cases} o_{i,l} \leq M.x_{i,l}; o_{i,l} \leq t_{s_l}; o_{i,l} \geq t_{s_l} - M(1 - x_{i,l}); \\ w_{i,j,l} \leq M.y_{i,j,l,l}; w_{i,j,l} \leq t_{s_l}; w_{i,j,l} \geq t_{s_l} - M(1 - y_{i,j,l,l}). \end{cases} \tag{15}$$

Finally, the VNFPPAT problem is converted into a 0-1 MILP and expressed as follows:

$$\begin{aligned} \textbf{P4:} \quad & \min \quad D^t \quad \text{according to} \quad (10) \\ & \text{s.t.} \quad (1), (2), (5), (11) - (12), (15). \end{aligned} \tag{16}$$

IV. Proposed Algorithms

In this section, we present two heuristic algorithms to solve the VNFPPAT problem since the VNF placement problem has been proven to be an NP-hard problem [1]. One heuristic algorithm is based on a greedy strategy, and the other uses a simulated annealing technique.

A Greedy algorithm

The basic principle of designing greedy algorithm is to minimize the increment of the total delay during the placement iterations. Deploying a middlebox does not only cause its end-to-end delay, but also introduces more queueing delay for other middleboxes. Moreover, the order of deploying middleboxes need to be taken into account because of their different functionalities. Obviously, a middlebox with a faster data arrival rate has a greater impact on the total delay and it should have a higher priority in the placement procedure. Thus, we sort the middleboxes in descending order considering their data arrival rate which can be obtained by $\sum\limits_{f_k \in F} \lambda_k l_k z_{q_i,f_k}$.

When q_i is selected for deployment, the location resulting in a minimum total delay increment is selected. We define the increment as the cost of deploying q_i in s_l. The value contains the path delay from q_i to other switches, other middleboxes, and the queueing delay change of all flows that arrive at s_l. We use $f^o_{q_i,s_l}$ to represent the increment while we ignore the undeployed middleboxes, and it's expressed as follows:

$$\begin{aligned} f^o_{q_i,s_l} =& \sum_{\forall q_j \in O} (d^r_{s_l,g(q_j)} v_{q_i,q_j} + d^r_{g(q_j),s_l} v_{q_j,q_i}) \\ &+ \sum_{\forall s_{l_1} \in S} (d^r_{s_{l_1},s_l} v_{s_{l_1},q_i} + d^r_{s_l,s_{l_1}} v_{q_i,s_{l_1}}) + ({d^q_{s_l}}' {v_{s_l}}' - d^q_{s_l} v_{s_l}). \end{aligned} \tag{17}$$

In 17, $g(q_j)$ denotes the location of q_j. ${d^q_{s_l}}'$ and ${v_{s_l}}'$ designate the new queueing delay and new flow number traversing

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on March 11,2023 at 15:54:00 UTC from IEEE Xplore. Restrictions apply.

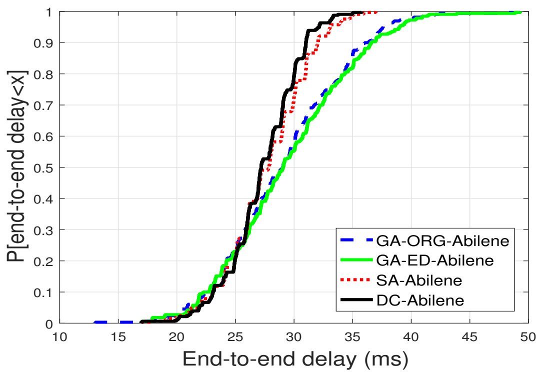


Fig. 4. CDF of the end-to-end delay.

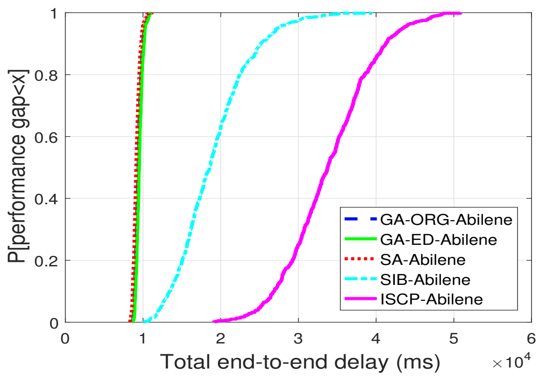


Fig. 6. CDF of total delay of heuristic algorithms.

s_l after deploying a middlebox, respectively. To estimate the link delay for the remaining unemployed middleboxes, we use the average link delay. The notation $f^e_{q_i,s_l}$ is used to denote the estimated total delay for all middleboxes, and it's calculated as follows:

$$f^e_{q_i,s_l} = f^o_{q_i,s_l} + \frac{\sum_{\forall s_{l_1} \in S} d^r_{s_l,s_{l_1}} r_{s_{l_1}}}{\sum_{\forall s_{l_1} \in S} r_{s_{l_1}}} \sum_{\forall q_j \in Q-O} v_{q_j} \tag{18}$$

In (18), $r_{s_{l_1}}$ denotes the remaining CPU resources of server s_{l_1}. The algorithmic complexity of the greedy algorithm can be obtained by multiplying the number of iterations of the nested loops. A total of $|\mathcal{Q}|$ middleboxes must be deployed, $|\mathcal{S}|$ costs are calculated during each iteration, and $|\mathcal{Q}|+|\mathcal{S}|$ operations are required for measuring a switch's cost. Therefore, its computational complexity is $\mathcal{O}(|\mathcal{Q}||\mathcal{S}|(|\mathcal{Q}|+|\mathcal{S}|))$.

B Simulated Annealing

To further improve the placement generated from the greedy algorithm, we propose the other heuristic algorithm based on simulated annealing. As a general optimization method, we still need to carefully set the initial value of temperature, its updating procedure, and neighbor generating methods. The setting of temperature parameter is related to the mean delta delay according to [15]. We set the initial temperature t_0 as $t_0 = (1-\lambda_1-\lambda_2)\Delta d^t_a + \lambda_1 \Delta d^t_m + \lambda_1 \Delta d^t_x$, in which Δd^t_a indicates the average delta delay, Δd^t_m denotes the minimal delta delay, and Δd^t_x designates the maximal delta delay. In our proposed work, λ_1 and λ_2 are set to 0.5 and 0, respectively. The temperature is updated by the formula $t = \frac{t}{1+t/t_0}$. We propose two neighbor generating functions in our simulated annealing algorithm. The first one randomly selects two middleboxes and exchanges their deployed NFV servers. The other randomly chooses a middlebox and places it to a randomly selected location.

V. Evaluation

In this section, three aspects must be considered: performance gap between our heuristics and the directly calculated solution, gap between our proposed algorithms and other proposals addressing the VNFPPAT problem, and performance variation in accordance with changes in the parameters. The comparison is made with a baseline heuristic algorithm that is inspired by the usual strategy in network deployment. In general non-NFV scenarios, the middleboxes can only be placed in the access points. Therefore, our baseline algorithm chooses the switch s_l with minimal service intensity from/to q_i, i.e., $\min(\sum_{\forall f_k \in F} \rho_{s_l,f_k} I_{i_k,s_l} + \sum_{\forall f_k \in F} \rho_{f_k,s_l} I_{s_l,e_k})$.

A Parameter Settings

In our proposed work, we use two common network topology: Abilene [16] and Fattree-8 [17]. Afterwards, we set the capacity value of processing each middlebox for each NFV server. According to [18], we generate the flow information that contains their ingress/egress switches, their packet size, the mean length of packets, and their corresponding service chain. We designate several flows between each pair of the access switches and randomly specify its visiting sequence

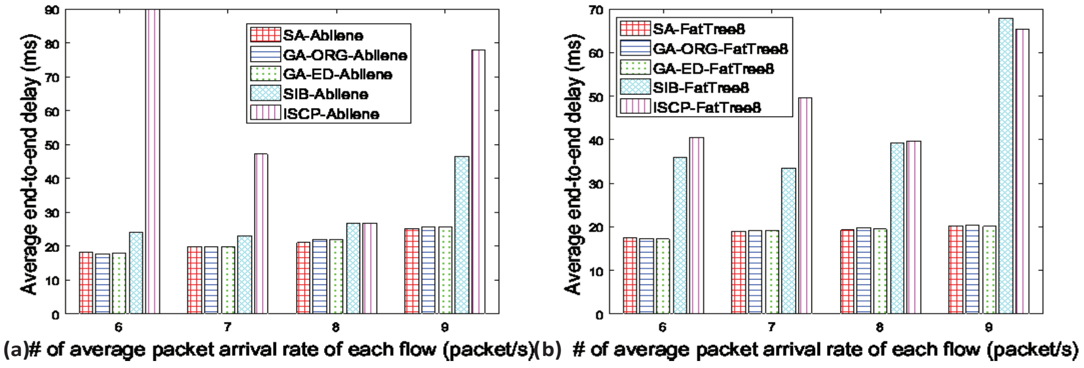


Fig. 5. The average delay with the variation in mean packet arrival rate of each flow.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on March 11,2023 at 15:54:00 UTC from IEEE Xplore. Restrictions apply.

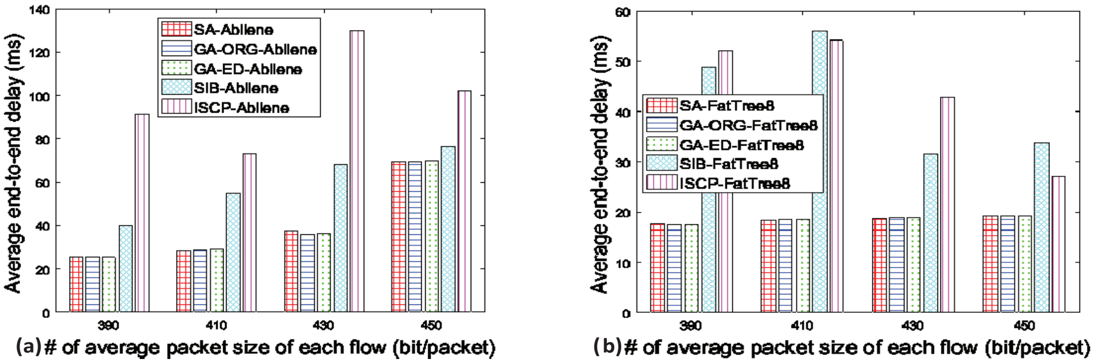


Fig. 7. The average delay when varying the mean packet size of each flow.

to a certain middleboxes. The detailed setting information is listed in Table I.

We use the following notations $\{GA-ORG$, $GA-ED$, $SA\}$ to designate the proposed algorithms: greedy strategy with the cost $f^{o}_{q_i,s_l}$ and $f^{e}_{q_i,s_l}$, simulated annealing algorithm. SIB is used to denote the baseline algorithm, DC refers to the directly calculated solution by solving the ILP (16), and $ISCP$ is used to represent the algorithm proposed in [1] (Improve Service Chaining Performance with optimized middlebox placement).

B Algorithmic Performance

The primary aspects of evaluation considered in our proposed work are: the algorithmic performance gap between the proposed heuristic algorithms and the directly calculated result, the difference with the baseline algorithm, and the variety of performance while varying the parameters.

Fig. 4 explicitly depicts the performance gap between our heuristic algorithms $\{GA-ORG$, $GA-ED$, $SA\}$ and DC in the form of a Cumulative Distribution Function (CDF). We generate 3 flows between each device pair in the Abilene network, and a service chain of each flow having 6 randomly selected middleboxes. The average packet arrival rate is set to be 8 packets per second and the mean packet size is 400 bits per packet. Other settings are randomly selected from Table I. The mean gaps between the heuristics and directly calculated is small, i.e., the values of $\{GA-ORG$, $GA-ED$, $SA\}$ are 6.5%, 7.7%, and 1.2%, respectively. 90% of the flows' delay inflated ratio of $GA-ORG$ are less than 32.2%, and the values generated by $GA-ED$ and SA are 36% and 24.8%, respectively. Considering the end-to-end delay, 79% of the flows have a delay of less than 30ms with DC, and this values reduces to {57.6%, 55.5%, 72.4%} with $\{GA-ORG$, $GA-ED$, $SA\}$, respectively. Based on the above results, we can conclude that our heuristics are reasonable and objective.

The performance measurements of our algorithms compared with other schemes in terms of total delay are presented in Fig. 6. In this simulation, we generate 600 NFV scenario cases in Abilene topology. A total of 14 middleboxes must be deployed and the mean server capacity for dealing with their data is 9.6×10^5 bits per second. The SA algorithm greatly reduces the end-to-end delay compared with SIB and $ISCP$, and has a similar performance as that of $GA-ORG$ and $GA-ED$. SA averagely reduces 72.9% total delay in contrast with ISCP which forgets considering the queueing phenomenon appearing in the NFV servers. Compared to $ISCP$, SIB scheme brings a small improvement and reduces

TABLE I
Evaluation Settings for Obtaining Algorithmic Performance

Parameter	Value Settings
The number of flows	[330, 992]
The number of middleboxes	[14, 40, 60]
The number of simulation cases	[1, 600]
The number of service chains	[1, 3]
The length of each service chain	[6]
The average processing capacity (bit/s)	[0.96, 1.1, 1.8, 1.9]$\times 10^6$
Link delay (ms)	[1]
The average packet rate (packet/s)	[8]
The average packet size (bit/packet)	[400, 450]

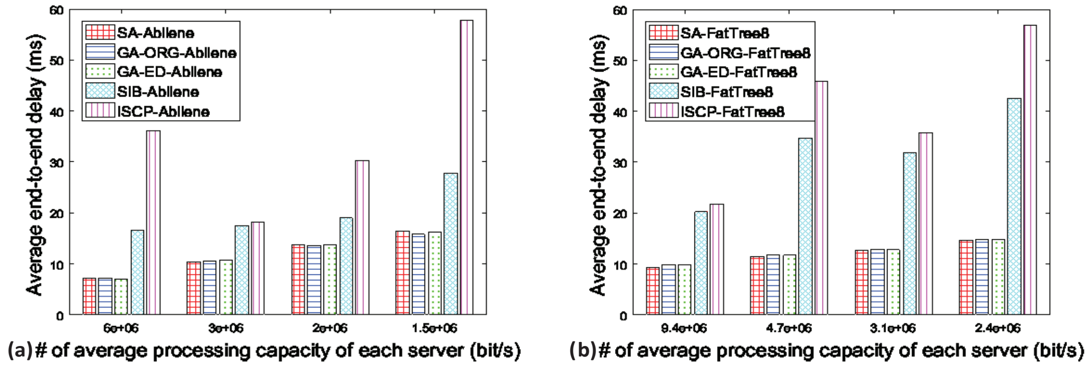


Fig. 8. The average delay when varying the mean processing capacity of each server.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on March 11,2023 at 15:54:00 UTC from IEEE Xplore. Restrictions apply.

44% of the delay because considering the data arrival rate can affect the time spent in the servers. However, owing to its simple strategy, SIB increases the total delay by 107.3% compared with SA. SA reduces the total delay by 2.8% and 3.7% compared with $GA-ORG$ and $GA-ED$, respectively.

Next, we examine the influence of three principal parameters employed in the VNFPPAT problem including packet arrival rate, packet size, and processing capacity of each server. We randomly generate a NFV scenario with the settings shown in Table I both for the Abilene and Fattree-8 topology. In the first place, the packet arrival rate is separately set to {6, 7, 8, 9} while simulating the scenario four times. As shown in Fig. 5(a), the mean end-to-end delay of our proposed algorithms $\{GA-ORG, GA-ED, SA\}$ are {17.9, 19.8, 21.7, 25.4}ms for four different packet rates in the Abilene topology. The average delays of SA in the Fattree-8 topology are {17.5, 18.9, 19.5, 20.3}ms and are depicted in Fig. 5(b). In the next phase, we set the mean processing capacity of each middlebox to {390, 410, 430, 450} bits per packet. As for the four setting values, the average delay of our heuristic algorithms is {25.5, 28.6, 36.4, 69.4}ms as is shown in Fig. 7(a). $GA-ORG$ obtains the result {17.4, 18.54, 18.9, 19.2}ms as presented in Fig. 5(b). Finally, we set the mean capacity of processing each middlebox to $\{6, 3, 2, 1.5\}\times 10^6$ and $\{9.4, 4.7, 3.1, 2.4\}\times 10^6$ bits per second for Abilene and Fattree-8, respectively. The heauristics achieve an average end-to-end delay of {7.2, 10.5, 13.7, 16.2}ms in Abilene network, as depicted in Fig. 8(a). As shown in Fig. 8(b), mean delays of {9.8, 11.9, 12.9, 14.7}ms are measured in the Fattree-8 topology with four different processing capacities.

The above results are in accordance with our intuition since a higher packet rate, larger packet size or less processing capacity all result in an increased queueing delay incurred in the servers. However, the results of $ISCP$ and SIB are inflated and irregular since they ignore to take the queueing delay into consideration.

VI. Conclusion

In this paper, we are the first to focus on the queueing delay in the NFV servers. To obtain the VNF placement scheme for Poisson based traffic, we formulated the VNFPPAT problem as a 0-1 quadratic fractional programming problem, and then transformed it to a 0-1 MILP problem. We proposed two heuristic algorithms: one of the heuristic algorithm is based on a greedy strategy, the other is incorporating simulated annealing. Extensive simulations indicate that SA offers huge performance improvement that can reduce delay by 72.9% compared with the $ISCP$ scheme. In the future work, we will study other distributions of packet arrival rate and packet size in different network scenarios.

References

[1] J. Liu, Y. Li, Y. Zhang, L. Su, and D. Jin, "Improve service chaining performance with optimized middlebox placement," *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 560–573, 2017.

[2] R. Cohen, L. Lewin-Eytan, J. S. Naor, and D. Raz, "Near optimal placement of virtual network functions," in *Computer Communications*, pp. 1346–1354, 2015.

[3] A. Marotta, F. D ' Andreagiovanni, A. Kassler, and E. Zola, "On the energy cost of robustness for green virtual network function placement in 5g virtualized infrastructures," *Computer Networks*, 2017.

[4] K. Yang, H. Zhang, and P. Hong, "Energy-aware service function placement for service function chaining in data centers," in *Global Communications Conference*, pp. 1–6, 2017.

[5] W. Ma, C. Medina, and D. Pan, "Traffic-aware placement of nfv middleboxes," in *GLOBECOM 2015 - 2015 IEEE Global Communications Conference*, pp. 1–6, 2015.

[6] M. Casazza, P. Fouilhoux, M. Bouet, and S. Secci, "Securing virtual network function placement with high availability guarantees," pp. 1–9, 2017.

[7] RajJain, *The Art of computer systems performance analysis*. John Wiley & sons, 1991.

[8] D. P. Bertsekas, "Traffic behavior and queuing in a qos environment," *OPNETWORK 2005, Session 1813*, 2005.

[9] K. Park and W. Willinger, *Self-Similar Network Traffic and Performance Evaluation*. John Wiley Sons, Inc., 2000.

[10] A. V. Harten and A. Sleptchenko, "On markovian multi-class, multi-server queueing," *Queueing Systems*, vol. 43, no. 4, pp. 307–328, 2003.

[11] F. Gurski, "Efficient binary linear programming formulations for boolean functions," *Statistics Optimization Information Computing*, vol. 2, no. 4, pp. 274–279, 2014.

[12] A. Charnes and W. W. Cooper, "Programming with linear fractional functionals," *Naval Research logistics quarterly*, vol. 9, no. 3-4, pp. 181–186, 1962.

[13] H. D. Sherali and W. P. Adams, "A reformulation-linearization technique for solving discrete and continuous nonconvex problems," *Springer Berlin*, vol. 31, no. 8, pp. 790–790, 1999.

[14] D. Yue, G. Guill é n-Gos á lbez, and F. You, "Global optimization of largescale mixedinteger linear fractional programming problems: A reformulationlinearization method and process scheduling applications," *Aiche Journal*, vol. 59, no. 11, pp. 4255–4272, 2013.

[15] A. Miseviius, "A modified simulated annealing algorithm for the quadratic assignment problem," *Informatica*, vol. 14, no. 4, pp. 497–514, 2003.

[16] "Abilene core topology." https://itservices.stanford.edu/service/network/internet2/abilene.

[17] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 63–74, ACM, 2008.

[18] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "Simple-fying middlebox policy enforcement using sdn," in *ACM SIGCOMM computer communication review*, vol. 43, pp. 27–38, ACM, 2013.

Authorized licensed use limited to: Illinois Institute of Technology. Downloaded on March 11,2023 at 15:54:00 UTC from IEEE Xplore. Restrictions apply.