Ant Colony based Online Learning Algorithm for Service Function Chain Deployment

Yingling Mao, Xiaojun Shang, and Yuanyuan Yang

Department of Electrical and Computer Engineering, Stony Brook University, USA

IEEE INFOCOM 2023 - IEEE Conference on Computer Communications | 979-8-3503-3414-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/INFOCOM53939.2023.10229012

***Abstract*—Network Function Virtualization (NFV) emerges as a promising paradigm with the potential for cost-efficiency, manage-convenience, and flexibility, where the service function chain (SFC) deployment scheme is a crucial technology. In this paper, we propose an Ant Colony Optimization (ACO) meta-heuristic algorithm for the Online SFC Deployment, called ACO-OSD, with the objectives of jointly minimizing the server operation cost and network latency. As a meta-heuristic algorithm, ACO-OSD performs better than the state-of-art heuristic algorithms, specifically 42.88% lower total cost on average. To reduce the time cost of ACO-OSD, we design two acceleration mechanisms: the Next-Fit (NF) strategy and the many-to-one model between SFC deployment schemes and ant-tours. Besides, for the scenarios requiring real-time decisions, we propose a novel online learning framework based on the ACO-OSD algorithm, called prior-based learning real-time placement (PLRP). It realizes near real-time SFC deployment with the time complexity of $O(n)$, where n is the total number of VNFs of all newly arrived SFCs. It meanwhile maintains a performance advantage with 36.53% lower average total cost than the state-of-art heuristic algorithms. Finally, we perform extensive simulations to demonstrate the outstanding performance of ACO-OSD and PLRP compared with the benchmarks.**

I. INTRODUCTION

With the development of virtualization technology, network function virtualization [1] (NFV) emerges as a promising paradigm by migrating network functions, or middleboxes, from proprietary hardware appliances to common commercial servers. In an NFV system, a network service request is typically realized by chained-up virtual network functions (VNFs), also called service function chains (SFCs) [2]. NFV makes network services more cost-efficient, manage-convenient, and flexible. To realize its full potential, an efficient SFC deployment scheme is essential, which can help fully utilize server resources, save bandwidth, and reduce network latency. Thus, the SFC deployment scheme is a crucial technology for NFV.

For SFC deployment, two major objectives are operation cost reduction and network latency minimization. Specifically, the former is pursued by NFV providers seeking cost-effectiveness, while the latter is more emphasized by customers expecting a higher quality of service (QoS). Nevertheless, such two objectives are sometimes conflicted. For example, Fig. 1 plots two alternative solutions to deploy an SFC. Solution (a) benefits the NFV providers since they can shut down the idle server 3 to reduce the operation cost, but it increases the communication latency of the service compared with solution (b). Solution (b) improves the customers' QoS by cutting back the communication latency between two VNFs,

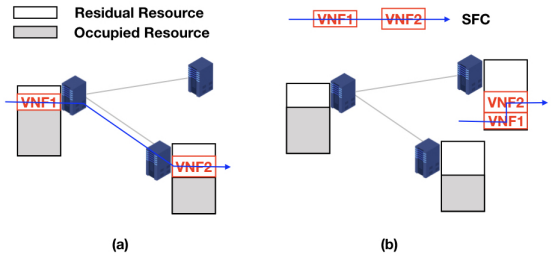


Fig. 1: An example of two SFC deployment schemes, where there are three servers with different residual resources and an SFC composed of two VNFs with different resource demands.

but it incurs more operation costs due to one more employed server compared with solution (a). Thus, in this work, we focus on a practical problem on SFC deployment: *How to achieve a win-win SFC deployment solution that jointly reduces the operation cost and minimizes the network latency?*

Such an SFC deployment problem with two objectives has been proven to be a combination of two NP-hard issues, i.e., VNF placement and flow routing [9], [10]. Due to this NP-hardness, most existing techniques are limited to the design of heuristic algorithms. To achieve better optimization, we focus on another popular choice to solve the NP-hard problem: the meta-heuristic algorithm. The meta-heuristic algorithm is typically a high-level problem-independent algorithmic framework. It has the advantage of better performance than heuristic algorithms since it can find near-optimal solutions by iteratively improving solutions.

Besides, most existing SFC deployment proposals usually work only for specific simplified models of the SFC deployment problem (Seen in Sec. II). As more practical considerations and constraints are introduced into the model, e.g., multiple resource types, queuing delays, online service requests, and their limited living time, existing SFC deployment strategies typically break. In our paper, we build a comprehensive model and target to make an online SFC deployment, with the objective of jointly reducing the operation cost and minimizing the network latency. And we select a popular-used meta-heuristic algorithm, called Ant Colony Optimization (ACO), which can successfully fit into our model to solve the online SFC deployment problem.

Moreover, we design two acceleration mechanisms to improve the conventional ACO for this particular problem, giving birth to a novel ACO-based Online SFC Deployment algorithm

(ACO-OSD). First, when fitting ACO into the online SFC deployment problem, we adopt a many-to-one model between the SFC deployment schemes and artificial ant-tours, which significantly speeds up our algorithm by cutting back the total number of possible ant-tours. Besides, we further facilitate our ACO-OSD algorithm by employing the Next-Fit (NF) strategy to obtain the shortest feasible ant-tour in a tour family and remove the possibilities of all other "not-best" ant-tours.

In addition, for the scenarios requiring real-time decisions, we propose an online learning framework based on ACO-OSD, called prior-based learning real-time placement (PLRP). It has two stages. One is the prior-based learning stage, where ACO-OSD runs based on the real-time network data and prior SFC data in advance of a time slot. The other is the real-time placement stage, where the NF strategy is adopted to deploy the newly arrived SFCs based on the best-learned routing path that the prior-based learning stage returns. In PLRP, only the real-time placement stage runs after the new service requests arrive, and its time complexity is $O(n)$, where n is the total number of VNFs of all newly arrived SFCs. Thus, PLRP can realize near real-time placement. Meanwhile, the best-learned routing path that the prior-based learning stage returns help PLRP maintain a comparable performance advantage to that of the ACO-OSD algorithm.

Our main contributions are listed as follows.

- We formulate a comprehensive model for online SFC deployment, which considers multiple computing resources, such as CPU and RAM, the queuing delay, the online service requests, and their limited living time.
- We successfully fit the ACO meta-heuristic algorithm in our comprehensive model and propose two acceleration mechanisms, generating the ACO-OSD algorithm. It performs better than the state-of-art heuristic algorithms, specifically 42.88% lower total cost on average.
- We propose a novel online learning framework based on ACO-OSD, called PLRP, which realizes near real-time placement and maintains a performance advantage with 36.53% lower average total cost than the state-of-art heuristic algorithms.
- We perform extensive simulations, demonstrating the superiority of our ACO-OSD and PLRP compared with two state-of-art heuristic benchmarks.

The remainder of this paper is organized as follows. Section II reviews the related works. Section III describes the system model and formulates the online SFC deployment problem. Section IV states our basic ideas and challenges. Afterward, Section V proposes the ACO-OSD algorithm for the online SFC deployment, while Section VI designs the PLRP online learning framework. Then, Section VII is the performance evaluation of ACO-OSD and PLRP. Finally, we conclude the paper in Section VIII.

II. RELATED WORK

With the development of NFV, the SFC deployment problem has become a research hot spot. When deploying SFCs, it is promising to optimize the operation cost and the network latency jointly. It is NP-hard. Thus, most existing works focus on proposing heuristic algorithms to solve it. Among them, many works, e.g., [3]–[6], have no provable performance guarantee. Only four existing works [7]–[10] provide theoretical performance bounds for the proposed algorithms on the SFC deployment problem with these two optimization objectives. We list them below. Shang et al. [7] put forward provable approximation algorithms, based on the rounding algorithm, with an approximation ratio of $O(log(M))$, where M is the number of servers. Jin et al. [8] design a two-stage VNF deployment scheme with a constrained depth-first search algorithm (CDFSA) and a path-based greedy algorithm (PGA), which gives a theoretically-proved worst-case performance bound by an implicit constant factor. Mao et al. [9] propose an SFC deployment algorithm for the hybrid edge-and-cloud environment with a provable constant approximation ratio. In [10], the proposed algorithms for the online SFC deployment achieve an approximation ratio of 2 on the operation cost and a ratio of $O(log(M))$ on the network latency.In all, the state-of-art result is a provable worst-case performance bound by a constant factor no less than 2. Limited to the complexity of the problem, it is hard to obtain further improvement on the worst-case theoretical bound. Thus, we pay more attention to pursuing better average performance rather than the worst-case bound. We consider meta-heuristic algorithms, which typically perform better than a single heuristic algorithm.

Besides, these existing heuristic algorithms, specifically the approximation algorithms, only work for the specific simplified model and have some drawbacks which limit their performance for real-world applications. For example, works [7]–[10] only consider one kind of computing resource, ignoring the influence of capacity limitation of other resources. And works [7], [9], [10] all ignore the queuing delay in the computation of network latency. Additionally, work [8] only deals with the offline case and can not adapt to the dynamic network, while work [10] considers the online model but assumes the arrived service is always living in the system, which is unrealistic. We consider all these factors (i.e., multiple resource types, queuing delay, online service requests, and their limited living time) and build a comprehensive model to solve the online SFC deployment problem.

Additionally, there are two existing works [23], [24] trying to fit the ACO framework into the SFC deployment problem. However, they both consider an offline model and can not adapt to the dynamic network due to the high time cost caused by the iteration framework. What's worse, they both ignore queuing delay in the computation of network latency. Our designed ACO-OSD and PLRP both fit the comprehensive model for online SFC deployment. In particular, PLRP can adapt to the dynamic network due to its near real-time placement manner.

III. PROBLEM FORMULATION

A. System Model

The notations used in this model are shown in Table I.

m	total number of Service Function Chains (SFCs)
M	number of physical servers in the network
n_i	number of VNFs in SFC i
N	number of all VNFs
$F_{i,j}$	VNF j in SFC i
$f_{i,j}^A$	needed A(= CPU or RAM) resource of VNF $F_{i,j}$
λ_i	the flow rate of SFC i
V_k	the k-th physical server
R_k^A	capacity of commercial server V_k's A (= CPU or RAM) resource
C_k	operation cost of commercial server V_k
μ_k	processing capacity of the router at node V_k
l'_k	average queuing delay at node V_k
$B_{p,q}$	bandwidth limit of link (V_p, V_q)
$w_{i,j}^{p,q}$	1 if output data flow of $F_{i,j}$ pass through link (V_p, V_q), o/w 0
$l_{p,q}$	link latency of link (V_p, V_q)
$x_{i,j}^k$	1 if VNF $F_{i,j}$ is placed on V_k, otherwise 0
y_k	1 if server V_k is occupied, otherwise 0

TABLE I: Notations

a) Physical Network: Let us consider a physical network represented by a directed graph $G = (V, E)$, where each node is a commercial server, $V = \{V_1, V_2, \cdots, V_M\}$ is the set of server nodes, $M = |V|$ is the number of commercial servers, and E is the set of communication channels connecting servers in V. For each pair of servers $V_p, V_q \in V$, if $(V_p, V_q) \in E$, it implies V_p and V_q are directly connected, i.e., there exists a physical communication channel connecting the server V_p and V_q. Denote the bandwidth and link latency of this channel as $B_{p,q}$, $l_{p,q}$. If $(V_p, V_q) \notin E$, V_p and V_q are NOT directly connected and $B_{p,q}$ and $l_{p,q}$ are marked as 0.

Each server V_k has various kinds of computing resources, such as CPU (central processing unit), GPU (graphics processing unit), RAM (random-access memory), etc. Here, we only consider two primary computing resources: CPU and RAM. Note that if needed, our formulation and solution can be easily generalized to that with three or more kinds of computing resources. Denote by R_k^{cpu} and R_k^{ram} the CPU and RAM capacities of server $V_k (1 \leq k \leq M)$. Besides, denote the processing capacity of the router at node V_k, i.e., the number of packets per second such a router can sustain, as μ_k.

Each server V_k has two states: on and off. Once the commercial server is on, the NFV provider should pay for the operation cost of a whole server, even if only part of the computing resources are employed. Denote the operation cost of server V_k is C_k. At a time slot t, we use a binary variable $s_k(t)$ to indicate the state of commercial server V_k: $s_k(t) = 1$ if and only if the commercial server is on.

b) SFCs: Suppose there are m service requests, arriving sequentially over the entire time span T. Each service request needs a corresponding SFC to realize it. We number the requests in the order of their arrivals and denote by SFC i the SFC that works for i-th service request. At each time slot t, the NFV system executes the following procedures: first removing timeout SFCs, updating the network states, receiving arriving requests, making SFC deployment decisions, and finally again updating the network states.

Assume the i-th request arrives at time slot t_i^s. Its reserved time to live (TTL) is t_i^l time slots. Then, we use a binary variable $e_i(t)$ to indicate whether i-th service request (or SFC i) is still in service in time slot t: $e_i(t) = 1$, if and only if $t_i^s \leq t \leq t_i^s + t_i^l$.

In these SFCs, VNFs are chained up in some specific order to realize the corresponding service requests. As for SFC i, suppose there are n_i chained VNFs, noted as $F_{i,1}, F_{i,2}, \cdots, F_{i,n_i}$ in chaining order. Denoted the needed CPU and RAM resource of VNF $F_{i,j}$ as $f_{i,j}^{cpu}$ and $f_{i,j}^{ram}$. The total number of VNFs is $N = \sum_{i=1}^{m} n_i$. There is a data flow between adjacent VNFs in an SFC. Denote the flow rate of SFC i as λ_i.

B. Problem Formulation

In the online SFC deployment problem, our task is to deploy the m sequential SFCs onto the physical network G with M commercial servers, i.e., place the N VNFs onto M servers and route the corresponding data flow in the network G.

*a) **VNF placement**:* In order to formulate this problem, we first define a *Boolean* variable $x_{i,j}^k$ as the decision variable of the VNF placement scheme: $x_{i,j}^k = 1$, if and only if VNF $F_{i,j}$ is placed on server V_k.

When placing VNFs onto servers, we should satisfy the capacity constraint of each commercial server. Thus,

$$\sum_{i=1}^{m}\sum_{j=1}^{n_i} x_{i,j}^k \cdot f_{i,j}^{cpu} \cdot e_i(t) \leq R_k^{cpu}, \quad \forall 1 \leq k \leq M, t \in T. \quad (1)$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n_i} x_{i,j}^k \cdot f_{i,j}^{ram} \cdot e_i(t) \leq R_k^{ram}, \quad \forall 1 \leq k \leq M, t \in T. \quad (2)$$

Since each VNF can not be split, which implies it is exactly placed on a commercial server, we have

$$\sum_{k=1}^{M} x_{i,j}^k = 1, \qquad \forall 1 \leq i \leq m, 1 \leq j \leq n_i. \quad (3)$$

At the beginning of each time slot, the NFV system will remove timeout SFCs and deploy the newly arrived SFCs. These operations all have the possibility of changing the state of commercial servers. For example, once all VNFs placed on server V_k are removed, NFV providers will shut down server V_k immediately to reduce the operation cost, i.e., letting $s_k(t) = 0$. Once any VNF is scheduled to some closed server V_k, NFV providers should open server V_k first to run this VNF on it, i.e., letting $s_k(t) = 1$. In our model, we ignore the time cost of removing timeout SFCs and deploying the newly arrived SFCs since it is typically very short compared to a time slot. Thus, the state of server V_k at time slot t that we mentioned in our paper refers to the final stable state in this time slot after the executions of removing operation and deploying operation, i.e.,

$$s_k(t) = \begin{cases} 1, & \sum_{i=1}^{m}\sum_{j=1}^{n_i} x_{i,j}^k \cdot e_i(t) > 0, \\ 0, & \sum_{i=1}^{m}\sum_{j=1}^{n_i} x_{i,j}^k \cdot e_i(t) = 0. \end{cases} \quad (4)$$

*b) **Flow Routing**:* Besides, we define another *Boolean* variable $w_{i,j}^{p,q}$ as the decision variable of the flow routing scheme: $w_{i,j}^{p,q} = 1$, if and only if data flow between VNF $F_{i,j}$ and $F_{i,j+1}$ pass through link (V_p, V_q).

According to *Flow Conservation Law*, as for any data flow between VNF $F_{i,j}$ and $F_{i,j+1}$, we have $\forall\ 1 \leq k \leq M$,

$$\sum_{p=1}^{M} w_{i,j}^{p,k} - \sum_{q=1}^{M} w_{i,j}^{k,q} = x_{i,j+1}^{k} - x_{i,j}^{k}. \tag{5}$$

The limitation of bandwidth asks $\forall\ 1 \leq p < q \leq M$,

$$\sum_{i=1}^{m} \sum_{j=1}^{n_i-1} (w_{i,j}^{p,q} + w_{i,j}^{q,p}) \cdot \lambda_i \cdot e_i(t) \leq B_{p,q}. \tag{6}$$

In our model, we consider two kinds of network latency, namely the queuing delay in the router at node V_k, and the transmission delay $l_{p,q}$ on the link (V_p, V_q). To analyze the queuing delay of users at node V_k, we model it as an $M/M/1$ queue. By applying Little's law, the average queuing delay at V_k at time slot t, noted as $l'_k(t)$, can be calculated as follows:

$$l'_k(t) = \frac{1}{\mu_k - \sum_{i=1}^{m}(x_{i,1}^{k} + \sum_{j=1}^{n_i-1}\sum_{p=1}^{M} w_{i,j}^{p,k}) \cdot \lambda_i \cdot e_i(t)}. \tag{7}$$

In order to employ the above formula, we need to satisfy

$$\sum_{i=1}^{m}(x_{i,1}^{k} + \sum_{j=1}^{n_i-1}\sum_{p=1}^{M} w_{i,j}^{p,k}) \cdot \lambda_i \cdot e_i(t) < \mu_k, \quad \forall 1 \leq k \leq M. \tag{8}$$

*c) **Formulated Optimization Problem**:* Our objective is to jointly minimize the operation cost and the network latency.

The total operation cost $\mathbb{C}$ for holding all opened servers can be computed by

$$\mathbb{C} = \sum_{t \in T} \sum_{k=1}^{M} C_k \cdot s_k(t).$$

The total network latency $\mathbb{L} = \mathbb{D}_t + \mathbb{D}_q$, where $\mathbb{D}_t$ is the total transmission delay and $\mathbb{D}_q$ is the total queuing delay. $\mathbb{D}_t$ and $\mathbb{D}_q$ can be computed by

$$\mathbb{D}_t = \sum_{t \in T} \sum_{i=1}^{m} \sum_{j=1}^{n_i-1} \sum_{p=1}^{M} \sum_{q=1}^{M} l_{p,q} \cdot w_{i,j}^{p,q} \cdot e_i(t),$$

$$\mathbb{D}_q = \sum_{t \in T} \sum_{i=1}^{m} (x_{i,1}^{k} + \sum_{j=1}^{n_i-1} \sum_{p=1}^{M} w_{i,j}^{p,k}) \cdot e_i(t) \cdot l'_k(t).$$

In all, the SFC deployment problem can be formulated as

$$\begin{aligned} \min \quad & \mathbb{W} = \alpha \cdot \mathbb{C} + \beta \cdot (\mathbb{D}_t + \mathbb{D}_q) \\ \text{s.t.} \quad & (1) - (8), \end{aligned}$$

where α, β are two weighting factors.

IV. Basic ideas and challenges

In our paper, we build a comprehensive model for the online SFC deployment problem. Our basic idea is to choose a suitable meta-heuristic algorithm to solve the optimization problem in our model, achieving better performance than the state-of-art heuristic algorithms.

We choose the meta-heuristic algorithm from the ACO algorithm family. The first ant colony optimization algorithm, called Ant System (AS), was proposed by Dorigo et al. in the 1990s [11]. It is inspired by the foraging behavior of the real ant. One of the main ideas is the indirect communication of multi-agents, called artificial ants, based on pheromone trails. The (artificial) pheromone trails are a kind of distributed numeric information which is modified by the ants to reflect their experience while solving a particular problem. ACO is a probabilistic technique for solving NP-hard combinatorial optimization problems, which can be reduced to finding good paths through graphs. Recently, the ACO meta-heuristic algorithm family has been proposed, which provides a unifying framework for solving numerous optimization tasks involving some sort of graph, e.g., vehicle routing and internet routing. SFC deployment problem can be seen as such kind of flow routing problem.

One key challenge is that the online SFC deployment problem is a new optimization problem, and the existing ant colony optimization algorithms can not perfectly fit the SFC deployment problem. Under the framework of ACO, a new local search approach is needed to be designed for the server resource, bandwidth, and router capacity limitations in the SFC deployment problem. The other challenge is how to speed up the designed ACO meta-heuristic algorithms to meet the requirement of fast decisions in the dynamic network.

In the following two sections, we propose an ACO meta-heuristic algorithm for online SFC deployment, called ACO-OSD, with two acceleration mechanisms: the many-to-one fit model for tour construction and the Next-Fit strategy. Next, we design a novel online learning framework based on ACO-OSD, called PLRP, for scenarios requiring real-time decisions.

V. An ACO meta-heuristic algorithm for online SFC deployment

Before designing an ACO meta-heuristic algorithm for online SFC deployment, we first clarify the inputs of each time slot:

- targeted newly arrived SFCs: $i_0, \cdots, i_s$ at time slot t and the needed CPU and RAM resource of their VNFs;
- realtime residual CPU resource of server V_k: $r_k^{cpu} = R_k^{cpu} - \sum_{i=1}^{i_0-1}\sum_{j=1}^{n_i} x_{i,j}^{k} \cdot f_{i,j}^{cpu} \cdot e_i(t)$;
- realtime residual RAM resource of server V_k: $r_k^{ram} = R_k^{ram} - \sum_{i=1}^{i_0-1}\sum_{j=1}^{n_i} x_{i,j}^{k} \cdot f_{i,j}^{cpu} \cdot e_i(t)$;
- realtime residual processing capacity of router V_k: $\mu'_k = \mu_k - \sum_{i=1}^{i_0-1}(x_{i,1}^{k} + \sum_{j=1}^{n_i-1}\sum_{p=1}^{M} w_{i,j}^{p,k}) \cdot \lambda_i \cdot e_i(t)$;
- realtime residual bandwidth of link (V_p, V_q): $b_{p,q} = B_{p,q} - \sum_{i=1}^{i_0-1}\sum_{j=1}^{n_i-1}(w_{i,j}^{p,q} + w_{i,j}^{q,p}) \cdot \lambda_i \cdot e_i(t)$;
- realtime number of flows flowing into node V_k: $count_k = \sum_{i=1}^{i_0-1}(x_{i,1}^{k} + \sum_{j=1}^{n_i-1}\sum_{p=1}^{M} w_{i,j}^{p,k}) \cdot e_i(t)$.

Let us look at the total cost increment caused by deploying newly arrived SFCs at time slot t, which is our optimization target at time t. It has three parts: the increment of operation cost, transmission delay, and queuing delay.

The operation cost increment is $\Delta\mathbb{C}(t) = \sum_{k=1}^{M} C_k \cdot (\sum_{i=i_0}^{i_s}\sum_{j=1}^{n_i} x_{i,j}^{k} > 0) \cdot (count_k == 0)$, where $(count_k == 0) = 1$, if $count_k == 0$ is $True$; otherwise,

$(count_k == 0) = 0$. Here $\sum_{j=1}^{n_i} x_{i,j}^k > 0$ means server V_k is employed by these newly arrived SFCs while $count_k = 0$ means V_k is totally idle before. That is to say, the operation cost only increases when some server is employed from the idle status. Note that if V_k has been employed by the existing old SFCs and these newly arrived SFCs again employ it, the deployment of these newly arrived SFCs on server V_k will not cause an operation cost increase. The transmission delay increment is just the transmission delay of these newly arrived SFCs, i.e., $\Delta\mathbb{D}_t(t) = \sum_{i=i_0}^{i_s} \sum_{j=1}^{n_i-1} \sum_{p=1}^{M} \sum_{q=1}^{M} l_{p,q} \cdot w_{i,j}^{p,q}$. Denote $count'_k = \sum_{i=i_0}^{i_s} (x_{i,1}^k + \sum_{j=1}^{n_i-1} \sum_{p=1}^{M} w_{i,j}^{p,k})$. The queuing delay increment has two parts: one is the queuing delay of these newly arrived SFCs, i.e., $\Delta\mathbb{D}_q^1(t) = \sum_{k=1}^{M} count'_k / \left[\mu'_k - \sum_{i=i_0}^{i_s} (x_{i,1}^k + \sum_{j=1}^{n_i-1} \sum_{p=1}^{M} w_{i,j}^{p,k}) \cdot \lambda_i\right]$; the other is the queuing delay increment of existing old SFCs, caused by that some new data flows occupy part of router process capacity, i.e., $\Delta\mathbb{D}_q^2(t) = \sum_{k=1}^{M} count_k / \left[\mu'_k - \sum_{i=i_0}^{i_s} (x_{i,1}^k + \sum_{j=1}^{n_i-1} \sum_{p=1}^{M} w_{i,j}^{p,k}) \cdot \lambda_i\right] - \sum_{k=1}^{M} count_k / \mu'_k$. In all, the cost increment at time t is

$$\Delta\mathbb{W}(t) = \alpha\Delta\mathbb{R}(t) + \beta\left[\Delta\mathbb{D}_t + \Delta\mathbb{D}_q^1 + \Delta\mathbb{D}_q^2\right](t). \quad (9)$$

A. Tour Construction: Many-to-One Fit Model

In the ACO meta-heuristic framework, the basic idea is that in each iteration, multiple artificial ants construct different tours according to state transition rules based on the pheromone trails. After all the ants have constructed tours, the pheromones are updated based on the different objective costs of ant-tours. When we fit the ACO meta-heuristic framework into the online SFC deployment problem, the first problem is how to build a relationship between the SFC deployment schemes and artificial ant-tours. The natural idea is to map them one to one. Specifically, if ant currently exists at the server node of VNF $F_{i,j}$, its next node in the tour is exactly the server node of the next chained VNF $F_{i,j+1}$. For example, Fig. 2 gives two SFC deployment schemes for deploying an SFC with 4 VNFs in a network with three server nodes. According to one-to-one mapping, the corresponding two artificial ant-tours are respectively $1-1-2-3$ and $1-2-2-3$.

There is another more cost-efficient mapping proposal. Here we define an ant-tour by considering the process of SFC data flowing into a different server as a move in the tour. In this case, we will create a many-to-one relationship between the SFC deployment schemes and artificial ant-tours. According to this idea, both SFC deployment schemes in Fig. 2 share the same artificial ant-tour $1-2-3$. Compared with the one-to-one mapping, this many-to-one model has fewer possible artificial ant-tours. Thus, it can work as an acceleration mechanism in ACO-OSD.

B. A Next-Fit Speed-up Strategy

To introduce the many-to-one model into the ACO framework for the online SFC problem, we need to give a map from the artificial ant-tour back to the SFC deployment scheme. Note that not all randomly produced artificial ant-tour is

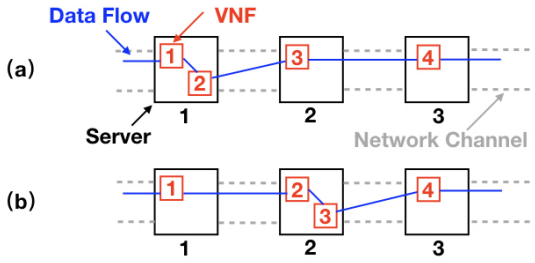


Fig. 2: Tour Constructions

feasible. Some ant-tours may have no SFC deployment scheme that maps to it because the servers along the tour do not have enough resources to place all VNFs. We call them infeasible ant-tours. Besides, since the network graph is not fully connected, only permitting the ant moves to the unvisited node, as the classical ACO does, is unfeasible.

That is to say, in our defined ant-tour, some server nodes may appear multiple times. Then, another problem appears: *How to decide the stop point of the tour construction ?* If we stop too early, the ant-tour will be infeasible. Below, we will show the Next-Fit strategy can solve these two problems.

Let us first consider a tour family T^f, where if $T_1, T_2 \in T^f$, then the start node of T_1 and T_2 are the same and T_1 is a part of T_2 or T_2 is a part of T_1. For example, $\{1, 1-2, 1-2-3, 1-2-3-4\}$ is a tour family.

If given a tour family and the order of possible employed server nodes is fixed by the longest tour, the NF strategy can return an SFC deployment scheme for an SFC, which maps to an artificial ant-tour belonging to this tour family. Specifically, start from the first VNF of this SFC and the first server node in the tours of this tour family. If this VNF fits inside the currently considered server, i.e., this server has enough residual available CPU and RAM resources for this VNF, it is placed on this server. Otherwise, the placement on the current server ends. Following the order of servers, find the next server with enough residual CPU and RAM resources for this VNF, place the VNF on it and consider this server as the current server. Repeat the same procedures on the next chained VNF until all VNFs are placed on the servers.

Theorem 1. *If given an ant-tour family T^f, whose longest tour is $T = [V_{a_1}, V_{a_2}, \cdots, V_{a_{|T|}}]$, the NF strategy returns an SFC deployment scheme for SFC i. This SFC deployment scheme is mapped to the shortest feasible tour in this tour family, which is also the tour with the minimal cost among all feasible tours.*

Proof. It can be proved by mathematical induction. First, it is easy to see the Next-Fit strategy returns the shortest feasible tour for deploying a sub-SFC with only VNF $F_{i,1}$. Assume NF returns a shortest feasible tour for deploying a sub-SFCs with VNF $F_{i,1}$ to VNF $F_{i,j}$ $(1 \leq j \leq j_0)$. We need to prove NF returns the shortest feasible tour for deploying sub-SFCs with VNF $F_{i,1}$ to VNF F_{i,j_0+1}, which can be proved by contradiction.

Assume the feasible tour given by NF for deploying a sub-SFCs with VNF $F_{i,1}$ to VNF F_{i,j_0+1}, noted as T_1, is not the shortest feasible tour. There exist a feasible tour T_2 among

the tour family T^f, which is shorter than T_1, i.e., $|T_2| < |T_1|$. Since T_2 is a feasible tour, there must exist an SFC deployment scheme for SFC i, which maps to the ant-tour T_2. Denote in this SFC deployment scheme for tour T_2, the last VNF placed on the server $V_{a_{|T_2|-1}}$ as F_{i,j_1} ($j_1 < j_0 + 1$). Denote F_{i,j_1} is placed on V_{a_k} by the NF strategy. Since $|T_2| < |T_1|$, then $|T_2| - 1 < k$, which contradicts with the assumption that NF strategy returns a shortest feasible tour for deploying a sub-SFCs with VNF $F_{i,1}$ to VNF F_{i,j_1} ($1 \leq j_1 \leq j_0$). □

Therefore, we can use the Next-Fit strategy to control the stop point of the tour construction and map from the ant-tour back to the SFC deployment. It speeds up our ACO-OSD by further removing all other possibilities, except the shortest feasible tour, in each tour family.

C. ACO Meta-heuristic Algorithm

Based on the above-mentioned many-to-one model and Next-Fit strategy, we design our ACO-OSD algorithm as below. In ACO-OSD, the ACO framework algorithm that we choose is Ant Colony System (ACS) [12], which is a popularly used ACO algorithm proposed for the Travelling Salesman Problem (TSP). ACS is based on an earlier algorithm, called Ant-Q [13], which combines AS, the first proposed ACO algorithm, with Q-learning, a specific reinforcement learning algorithm.

ACO-OSD differs in three main aspects from ACS, mainly in the tour construction. First, ACO-OSD extends the tour only when requiring more servers to deploy SFCs. The tour may not traverse all network nodes. Second, in steps of the tour extension, the feasible neighborhood of node V_k, Ω, is redefined by the set of κ nearest feasible nodes, rather than the neighborhood of node V_k in the network graph G. It is essentially a local search procedure added to the ACS framework. Third, every time of tour extension, we add the path from node V_k to node V_l, rather than node V_l, to the tour. It may cause some nodes repeatedly appear in the tour, which permits full use of the residual resource of some servers.

In the following, we present these modifications in more detail.

*1) **Tour Construction and SFC deployment**:* Step 1: Initially, each artificial ant a is put on some randomly chosen feasible node V_k. Note that the feasible node here refers to the node where the server has enough residual available CPU and RAM resources for the current considered VNF $F_{i,j}(i = i_0, j = 1)$ and the router has enough residual processing capacity for the data flow of SFC i_0, i.e., $r_k^{cpu} \geq f_{i,j}^{cpu}$, $r_k^{ram} \geq f_{i,j}^{ram}$, and $\mu'_k > \lambda_i$. Now, node V_k is added to the tour of ant a, and V_k is the current node.

Step 2: Employ the Next-Fit strategy to place all chained VNFs after the current considered VNF $F_{i,j}$ (including $F_{i,j}$) onto the current node V_k. There are two possible results: (1) if all chained VNFs after VNF $F_{i,j}$ (including $F_{i,j}$) have been successfully placed onto node V_k, go to Step 3; (2) if the placement stops at some VNF $F_{i,j'}$ since the server V_k has no enough residual available CPU and RAM resource to place VNF $F_{i,j'}$, go to Step 4.

Step 3: If $i < i_s$, update the current considered VNF by $i = i + 1, j = 1$ and repeat Step 2. If $i \geq i_s$, finish the tour construction.

Step 4: Update the current considered VNF by $j = j'$. If artificial ant a currently stops at the current node, i.e., the current node is the last node of the current ant-tour, do **tour extension** to update ant-tour and update the current node by its next node in the ant-tour. Otherwise, directly update the current node by its next node in the ant-tour. Then, repeat Step 2.

*2) **Tour Extension**:* If artificial ant a currently stops at node V_k but the VNFs of all newly arrived SFCs have not been totally placed, the tour should be extended from node V_k. Ant a applies the pseudo-random-proportional action choice rule to extend the tour. That is, with probability $1 - q_0$, ant a follows a probabilistic action choice rule. In particular, the probability with which ant a, currently at node V_k, chooses to go to node V_l at r-iteration of the algorithm is:

$$p_{kl}(r) = \frac{\tau_{kl}(r) \cdot \eta_{kl}^{\gamma}}{\sum_{o \in \Omega_k} \tau_{ko}(r) \cdot \eta_{ko}^{\gamma}}, \quad \text{if} \ \ l \in \Omega_k, \tag{10}$$

where $\tau_{kl}(r)$ is the pheromone value with an initialized value of 1 and updated via Eq. 11-12, η_{kl} is a heuristic value, γ ($\gamma > 0$) is a hyper-parameter which determines the relative influence of the pheromone trail versus the heuristic information, and Ω_k is the feasible neighborhood of node V_k. Unlike ACS, Ω_k here is redefined by the set of κ nearest feasible nodes. They can be found by the Dijkstra algorithm, with link latency as link weights of the network graph, under the bandwidth and router capacity limitation. It is a local search procedure that avoids possible flow blocks caused by the random walk. If $l \in \Omega_k$, the heuristic value of η_{kl} is defined by the length (the sum of link latency) of the path from V_k to V_l given by the Dijkstra algorithm.

With probability q_0, ant a follows a deterministic action choice rule. That is, ant a, currently at node V_k, chooses to go to node V_l that $\tau_{kl}(r) \cdot \eta_{kl}^{\gamma}$ is maximal and $l \in \Omega_k$. In this case, the best possible move under current learned knowledge is made.

Above all, the path from node V_k to node V_l is added to the tour of artificial ant a.

*3) **Global pheromone trail update**:* After all the ants have constructed their tours, we evaluate these tours by Eq. 9 and its mapped SFC deployment scheme. Afterward, the pheromone trails are updated globally. This is done by first lowering the pheromone strength on all arcs by a constant factor and then only allowing the artificial ant, which produces the global-best tour, to add pheromone on the arcs it has visited. Specifically, the global pheromone trail is updated following the below equation.

$$\tau_{kl}(r+1) = (1 - \rho) \cdot \tau_{kl}(r) + \rho \cdot \Delta\tau_{kl}^{gb}(r), \tag{11}$$

where ρ ($0 < \rho \leq 1$) is the global pheromone trail evaporation rate. It is used to avoid the unlimited accumulation of pheromone trails and enables the algorithm to "forget" previously bad decisions. If an arc is not chosen by the

global-best tour, its associated pheromone strength decreases exponentially. $\Delta\tau_{kl}^{gb}(r)$ is the amount of pheromone that the artificial ant, which produces the global-best tour, puts on the arcs it has visited. It is defined as follows:

$$\Delta\tau_{kl}^{gb}(r) = \begin{cases} 1, & \text{if arc } (k,l) \in \text{the global-best tour,} \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

*4) **Local pheromone trail update**:* In addition to the global updating rule, we also employ a local update rule, applied immediately after each tour construction:

$$\tau_{kl}(r) = (1-\xi)\cdot\tau_{kl}(r) + \xi\cdot\tau_0, \quad (13)$$

where local pheromone evaporation rate ξ $(0 < \xi < 1)$, and intensification value τ_0 are two Hyper-parameters. The effect of the local updating rule is to make an already chosen arc less desirable for the following ants. In this way, the exploration of not yet visited arcs is increased.

VI. Prior-based Learning Real-time Placement

Although we have tried our best to design a fast ACO meta-heuristic algorithm for the online SFC deployment problem, its time cost must be much more than that of the heuristic algorithms. It is caused by the iteration framework and can not be solved without framework change. It means ACO-OSD can not adapt to scenarios that require real-time decisions. Thus, we propose an online learning framework based on ACO-OSD, called PLRP, which realizes near real-time placement and meanwhile maintains the advantage of performance.

It is inspired by the separation of the training and testing stages in many popular machine learning algorithms, where although the training stage takes a lot of time, the testing stage is very fast, near real-time feedback. And in applications, we only care about the time cost of the test stage. Similarly, in our online SFC deployment problem, *can we do some preparations in advance?* Following this idea, our PLRP framework also has two stages: one is prior-based learning, which is done in advance; the other is the real-time placement stage, which is very fast, near real-time feedback. Specifically, it works as follows.

When we do the online SFC deployment decision for a new time slot t, there are two kinds of input (seen in Sec. V-C): the real-time SFC data and the real-time network data. The real-time network data can be known by a time slot in advance because it is totally decided by the deployment schemes of the SFCs that have arrived before, and their time to live. Take the simple case in Fig. 3 for example, where SFCs 1-3 arrive at the beginning of time slot t_1, SFCs 4-6 arrive at the beginning of time slot t_2. In this example, we target to find the deployment schemes for SFCs 4-6. Thus, this deployment decision will happen after the arrival of SFCs 4-6 at the beginning of time slot 2. The input of the real-time network data is decided by the deployment schemes of the SFCs 1-3 and their time to live, which can be known at the beginning of time slot t_1 when SFCs 1-3 have arrived and been deployed. Thus, combined with some prior SFC data, ACO-OSD can be executed right after SFCs 1-3 have been

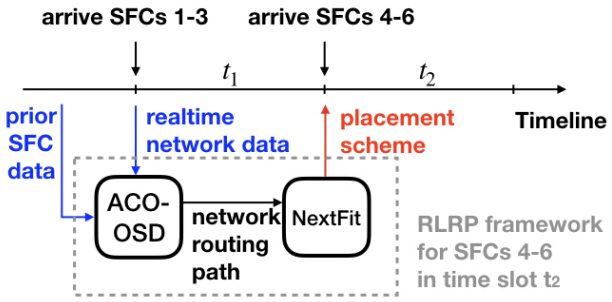


Fig. 3: Sketch map of PLRP framework for the newly arrived SFCs 4-6 in time slot t_2

deployed at the beginning of time slot t_1. In this plan, we have a whole time slot, which is typically much more than the time cost of ACO-OSD, to run the ACO-OSD algorithm. We call it the prior-based learning stage, and the output is the learned best network routing path (artificial ant-tour). Next, right after SFCs 4-6 arrive at the beginning of the time slot, we can do real-time SFC deployment based on the Next-Fit strategy and the learned network routing path. This stage is very fast with the time complexity of $O(n)$, where n is the total number of VNFs of all newly arrived SFCs. Thus, we call it the real-time placement stage.

Besides, let us talk about the selection of prior SFC data, where we assume there is one SFC with n_{prior} same-sized VNFs. Suppose the prior values of VNF needed CPU and RAM resources equal to the mean value of the required CPU and RAM resource of all possible employed VNFs. Typically, there are only 20 types of popularly-used VNFs. Besides, we set the prior data flow rate as the mean value of all data flow rates in the network. And the natural idea for n_{prior} is to let it equal the mean value of the service request number in each time slot times the mean value of the VNF number in each possible employed SFCs, noted as n_{mean}. However, if so, it may appear that the servers along the learned network routing path do not have enough resources to deploy all newly arrived SFCs. Such infeasible cases are because all prior values are based on the corresponding mean value, and the real values may be larger.

We can eliminate the infeasible cases by setting n_{prior} as a large number. But it is not the best choice. According to our simulation results shown in Section VII.C, a small feasible n_{prior} helps PLRP perform better. Thus, we eliminate infeasibility by setting multiple n_{prior} values, parallel running ACO-OSD based on these n_{prior} values, and obtaining multiple learned network routing paths with different lengths. Specifically, we set $n_{prior} \in \{n_{min}, n_{min} + \Delta_n, n_{min} + 2\Delta_n, \cdots, n_{min}\}$, where $n_{min}(n_{min} < n_{mean})$, $n_{max}(n_{max} > n_{mean})$ and Δ_n are hyper-parameters which affect the multiple learned network routing path lengths.

VII. Performance Evaluation

A. Simulation Setting

*a) **Network topology**:* We implement our simulations on 3 real network topologies of different sizes from the Internet

topology zoo [26]: (1) ARNES (34 nodes and 46 links), (2) DFN (58 nodes and 87 links), (3) ITCDeltacom (113 nodes and 160 links). Without special mention, we execute on DFN.

*b) **Network data setting**:* The CPU and RAM resource configured for each server node, i.e., R_k^{cpu} and R_k^{ram}, were randomly selected from $\{1, 2, 4, 6\}$ (with the unit of CPU cores) and $\{2, 4, 8, 16\}$ (with the unit of GB), respectively. Typically, the server with more computing resource needs more operation cost, thus, we set the operation cost for server V_k by $C_k = \frac{1}{2}(R_k^{cpu} + R_k^{ram})$. Besides, we configure each node's router with a processing capacity randomly chosen among $[50, 200]$ Mbps.

For each link (V_p, V_q), we randomly assign it a link latency, $l_{p,q}$, ranging from 0.05 ms to 0.2 ms and a bandwidth capacity, $B_{p,q}$, of 1300 Mbps (the bandwidth of Wireless 802.11ac).

*c) **SFC data setting**:* We generate SFCs by randomly picking 4 to 8 VNFs from 20 VNF types, and each kind of VNF instance requires $[0.1, 0.4]$ CPU cores and $[0.05, 0.2]$ GB of memory. The flow rate of each SFC, μ_i, ranges from 0.5 Mbps to 5 Mbps.

*d) **Online model setting**:* We consider a time span with 10 time slots, and at the beginning of each time slot, there are random 1 to 10 service requests arriving. The time to live (TTL) of each SFC is randomly set among $[1, 10]$.

*e) **Parameter setting**:* First, we set the two weights in the optimization objective of the SFC deployment by $\alpha = 1, \beta = 100$. In our ACO-OSD, we set 50 ants in each iteration and, in total 100 iterations. Besides, the hyper-parameters among ACO-OSD are set as follows: (1) ant action-choice proportion $q_0 = 0.3$; (2) relative weight parameter $\gamma = 1$; (3) the size of neighborhood $\kappa = 6$; (4) global pheromone evaporation rate $\rho = 0.5$; (5) local pheromone evaporation rate $\xi = 0.001$; (6) intensification value $\tau_0 = 1$. In PLRP, we set the hyper-parameter $\Delta_n = 6$.

*f) **Benchmarks**:* (1) **Next-Fit (NF) + Nearest Neighbour (NN) algorithm**: Work [10] employs such algorithms and proves NF maintains an approximation ratio of 2 on the operation cost while NN helps achieve a ratio of $O(log(M))$ on the network latency; (2) **Next-Fit (NF) + Double Spanning Tree (DST) algorithm**: Work [9] resorts to such algorithms, which has a provable constant approximation ratio in SFC deployment problem.

Note that for each outcome shown on the plots below, we use the average value of 100 groups of simulations to show moderate cases.

B. ACO-OSD Performance Evaluation in A Time Slot

In Fig. 4, the top two plots display the iteration process of ACO-OSD in a time slot with the input of a different number of newly arrived service requests and different network topology, where the y-axis value is the ratio of the cost of the current-best SFC deployment scheme to that of the best SFC deployment scheme after all iterations. Besides, the bottom two plots exhibit the time cost as the iterations increase under different cases, where the time cost is all proportional to the iterations.

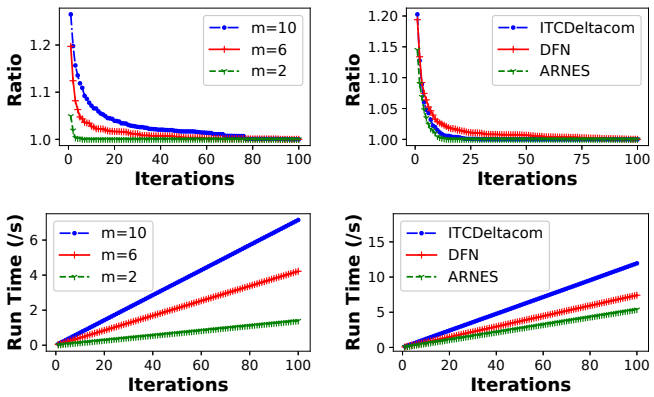


Fig. 4: The iteration process of ACO-OSD based on different numbers of service requests and different network topologies

In the left two plots of Fig. 4, we can see the more new service requests arrive in a time slot, the more slowly the iteration process converges, and the more time each iteration costs on average. But even for the maximal number of newly arrived requests in a time slot, i.e., $m = 10$, it drops to a near-convergent value (i.e., no matter drop after a fixed large number of iterations) among 100 iterations and costs less than 7 seconds in total. It implies ACO-OSD can work for online scenarios that can accept execution time on the order of seconds.

The right two plots show the influence of different topologies on the convergent process and the time cost. As for the time cost, the influence is totally determined by the number of nodes in the network topology. The more nodes there are in the network topology, the more time each iteration costs on average. And all of the time cost is on the order of seconds. But as for the convergent process, it is not the case. The topology structure also has an influence. For example, we can see DFN has fewer nodes than ITCDeltacom, but the iteration process on DFN converges more slowly than that on ITCDeltacom.

C. PLRP Performance Evaluation in A Time Slot

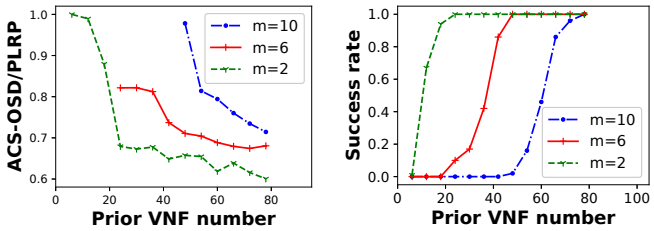


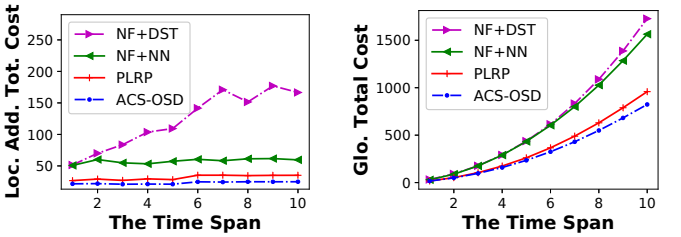
Fig. 5: The performance of PLRP with different prior VNF numbers under the number of service requests $m = 2, 6, 8$

In Fig. 5, we show the different performance of PLRP compared to ACO-OSD with the prior VNF number $n_{prior} \in \{6, 12, 18, 24, 30, 36, 42, 48, 54, 60, 66, 72, 78\}$. Since the performance of ACO-OSD will not be affected by the different settings of n_{prior} and PLRP is a proposed framework based on ACO-OSD, we use ACO-OSD as a baseline. The left plot

of Fig. 5, where the y-axis value is the ratio of the cost of ACO-OSD (baseline) to that of PLRP, demonstrates that in general, the less feasible prior VNF number is set, the better performance PLRP achieves. But the more new service requests arrive in a time slot, the larger the smallest feasible prior VNF number is. It is because the feasible learned routing path works better when its length is larger than but more close to the real needed length. But more new service requests require more resources, so the real needed length is larger. It can also be reflected by the right plot of Fig. 5.

The right plot exhibits the success ratios to deploy the different numbers of newly arrived SFCs (m) with the input of different prior VNF numbers. We can find as for all different m, the larger the prior VNF number is, the higher the success ratio is. But as for less m, the success ratio increases more early with the increase of the prior VNF number. Since we do not know the number of newly arrived SFCs m in advance, setting multiple different n_{prior} in PLRP is necessary and helps improve the performance of PLRP.

D. Performance Comparison over A Whole Time Span

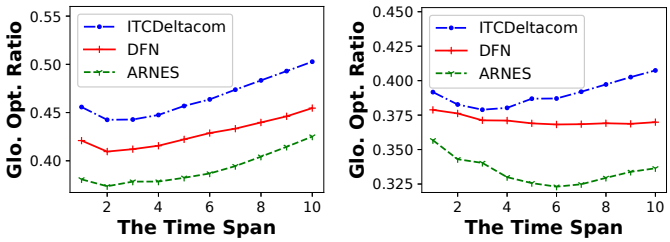


(a) Local added total cost: cost increment caused by the deployment of newly arrived SFCs in each time slot

(b) Global total cost: the total cost of all working SFCs in so-far all time slots

Fig. 6: Performance comparison of our ACO-OSD and PLRP with two benchmarks over a time span with 10 time slots

In an online problem, algorithms make optimization decisions in each time slot only based on the current information. The local performance on the optimization decisions in each time slot and the global performance of all decisions over the whole time span are two critical judgment criteria for online algorithms. Thus, we compare the performance of our ACO-OSD algorithm and PLRP framework with two state-of-art benchmarks on both local added total cost, the cost increment caused by the deployment of newly arrived SFCs in each time slot (in Fig. 6a), and the global total cost, the total cost of all working SFCs in so-far all time slots (in Fig. 6b). In Fig. 6, we can see our ACO-OSD algorithm and PLRP framework always maintain better performance than the two benchmarks, no matter in terms of local performance in each time slot or global performance over the whole time span. Besides, the performance of PLRP is always a bit worse than ACO-OSD. This is because of the error between prior SFC data and the real employed SFCs. PLRP essentially sacrifices partial performance benefits for realizing real-time placement. Luckily, Fig. 6 shows this part of the sacrifice is acceptable since the performance of PLRP is still always better than two benchmarks and is very close to ACO-OSD.



(a) ACO-OSD

(b) PLRP

Fig. 7: The performance improvement of our ACO-OSD and PLRP on different network topologies

Fig. 7 quantitatively presents the performance improvement of our ACO-OSD algorithm and PLRP framework compared with the two state-of-art benchmarks on different network topologies, where the y-axis value is $\frac{\min\{gtc(NF+DST),gtc(NF+DST)\}-gtc(algo)}{\min\{gtc(NF+DST),gtc(NF+DST)\}}$, where $algo$ points to our ACO-OSD or PLRP, $gtc(\cdot)$ is the global total cost of the target algorithm. In general, ACO-OSD achieves 42.88% lower global total cost on average, compared with that of two state-of-art benchmarks, while PLRP obtains 36.53% lower total cost on average. Besides, Fig. 7 reveals our ACO-OSD algorithm and PLRP framework have more performance improvement on the larger network, i.e., the network topologies with more nodes. This is because compared with the benchmarks, our algorithms learn more global network information by the ant colony pheromone.

VIII. Conclusion and Future Work

We build a comprehensive model for the online SFC deployment problem with the objective of minimizing both operations cost and network latency. To address this issue, we first design the ACO-OSD algorithm, which has two acceleration mechanisms: the many-to-one model between SFC deployment schemes and ant-tours and the Next-Fit strategy. Besides, as for the scenarios requiring real-time responses, we also propose a PLRP online learning framework based on ACO-OSD, which realizes near real-time placement with the time complexity of $O(n)$, where n is the total number of VNFs of all newly arrived SFCs. Finally, we perform extensive simulations on realistic network topologies, which demonstrates our ACO-OSD has 42.88% lower total cost on average, compared with the state-of-art benchmarks, while PLRP has 36.53% lower average total cost.

In the future, we will do more research on how to get better prior SFC data by predicting the future arrived SFCs, to improve the performance of PLRP further. Moreover, we will also try more complex but accurate learning algorithms, such as the combination of deep neural networks and reinforcement learning, to substitute ACO.

IX. Acknowledge

This research work was supported in part by the U.S. National Science Foundation under grant number CCF-1717731.

REFERENCES

[1] ETSI, and NFVISG, "Network Functions Virtualization-Introductory White Paper," *SDN and OpenFlow World Congress*, 2012.

[2] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," *2014 IEEE 3rd International Conference on the cloud Networking(CloudNet)*, pp.7–13, 2014.

[3] J. Pei, P. Hong, K. Xue, and D. Li, "Efficiently embedding service function chains with dynamic virtual network function placement in geo-distributed cloud system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, PP. 2179–2192, 2018.

[4] H. Ren, Z. Xu, W. Liang, Q. Xia, P. Zhou, O. F. Rana, A. Galis, and G. Wu, "Efficient algorithms for delay-aware NFV-enabled multicasting in mobile edge clouds with resource sharing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 9, pp. 2050–2066, 2020.

[5] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary. "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 98–106, 2015.

[6] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," *2015 IEEE 4th International Conference on the cloud Networking (CloudNet)*, pp. 171–177, 2015.

[7] X. Shang, Z. Liu, and Y. Yang, "Network Congestion-aware Online Service Function Chain Placement and Load Balancing," *Proc. of the 48th International Conference on Parallel Processing*, pp.1–10, 2019.

[8] P. Jin, X. Fei, et al, "Latency-aware VNF Chain Deployment with Efficient Resource Reuse at Network Edge,"*IEEE Conference on Computer Communications (INFOCOM)*, pp.267–276, 2020.

[9] Y. Mao, X. Shang, and Y. Yang, "Joint Resource Management and Flow Scheduling for SFC Deployment in Hybrid Edge-and-Cloud Network," *IEEE Conference on Computer Communications (INFOCOM)*, 2022.

[10] Y. Mao, X. Shang, and Y. Yang, "Provably Efficient Algorithms for Traffic-sensitive SFC Placement and Flow Routing," *IEEE Conference on Computer Communications (INFOCOM)*, 2022.

[11] M. Dorigo, V. Maniezzo, and A. Colorni, "The Ant System: Optimization by a Colony of Cooperating Agents," *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, vol. 26, no. 1, pp. 29-–42, 1996.

[12] M. Dorigo, and L.M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53-–66, 1997.

[13] L.M. Gambardella, and M. Dorigo, "Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem," *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 252-–260, 1995.

[14] Y. Sang, B. Ji, et al, "Provably efficient algorithms for joint placement and allocation of virtual network functions," *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pp.1–9, 2017.

[15] D. Li, P. Hong, and K. Xue, "Virtual network function placement considering resource optimization and SFC requests in cloud datacenter," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 7, pp.1664–1677, 2018.

[16] J. liu, H. Xu, G. Zhao, C. Qian, X. Fan, and L. Huang, "Incremental Server Deployment for Scalable NFV-enabled Networks,"*IEEE Conference on Computer Communications (INFOCOM)*, pp. 2361–2370, 2020.

[17] X. Shang, Y. Huang, Z. Liu, and Y. Yang, "Reducing the Service Function Chain Backup Cost over the Edge and Cloud by a Self-adapting Scheme,"*IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 2096–2105, 2020.

[18] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal VNF placement at the network edge,"*IEEE Conference on Computer Communications (INFOCOM)*, pp. 693–701, 2018.

[19] S. Yang, F. Li, S. Trajanovski, X. Chen, Y. Wang, and X. Fu, "Delay-aware virtual network function placement and routing in edge clouds,"*IEEE Transactions on Mobile Computing*, 2019.

[20] D. Zheng, C. Peng, X. Liao, L. Tian, G. Luo, and X. Cao, "Towards Latency Optimization in Hybrid Service Function Chain Composition and Embedding,"*IEEE Conference on Computer Communications (INFOCOM)*, pp. 1539–1548, 2020.

[21] V. Valls, G. Iosifidis, G. d. Mel, and L. Tassiulas, "Online Network Flow Optimization for Multi-Grade Service Chains,"*IEEE Conference on Computer Communications (INFOCOM)*, pp. 1329–1338, 2020.

[22] J. Martín-Peréz, F. Malandrino, et al, "OKpi: All-KPI Network Slicing Through Efficient Resource Allocation,"*IEEE Conference on Computer Communications (INFOCOM)*, pp. 804–813, 2020.

[23] A. Farshin, and S. Sharifian, "A modified knowledge-based ant colony algo- rithm for virtual machine placement and simultaneous routing of NFV in distributed cloud architecture,"*The Journal of Supercomputing*, vol. 75, no. 8, pp. 5520-–5550, 2019.

[24] M. Shokouhifar, "FH-ACO: Fuzzy heuristic-based ant colony optimization for joint virtual network function placement and routing,"*Applied Soft Computing*, vol. 107, pp 107401, 2021.

[25] D. S. Johnson, "Near-optimal bin packing algorithms," *PhD diss., Massachusetts Institute of Technology*, 1973.

[26] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo,"*IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[27] D. J. Rosenkrantz, E. S. Richard Edwin, and M. L. Philip, "Approximate algorithms for the traveling salesperson problem,"*15th Annual Symposium on Switching and Automata Theory (SWAT 1974)*, pp. 33–42, 1974.